UNIVERSITY OF CALIFORNIA

Los Angeles

A Generative Account of Latent Abstractions

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Sirui Xie

2024

ABSTRACT OF THE DISSERTATION

A Generative Account of Latent Abstractions

by

Sirui Xie
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Demetri Terzopoulos, Co-chair
Professor Song-Chun Zhu, Co-chair

Abstractions are fundamental to human intelligence, extending far beyond pattern recognition. They enable the distillation and organization of complex information into structured knowledge, facilitate the succinct communication of intricate ideas, and empower us to navigate complex decision-making scenarios with consistent value prediction. The ability to abstract is particularly fascinating because abstractions are not inherently present in raw data — they are latent variables underlying our observations. Despite the recent phenomenal advances in modeling data distributions, Generative Artificial Intelligence (GenAI) systems still lack robust principles for the autonomous emergence of latent abstractions.

This dissertation studies the problem of unsupervised latent abstraction learning, focusing on developing modeling, learning, and inference methods for *latent-variable generative models* across diverse high-dimensional data modalities. The core premise is that by incorporating algebraic, geometric, and statistical structures into the latent space and generator, we can cultivate representations of latent variables that explain observed data in alignment with human understanding.

The dissertation consists of four parts. The first three explore the generative constructs of latent abstractions for Category, Object, and Decision, respectively. Part I examines the basic structure of categories, emphasizing their symbol-vector duality. We

develop a latent-variable text model with a coupling of symbols and vectors in its representations. We investigate another representation that is both discrete and continuous — iconic symbols — in a visual communication game. Part II enriches the abstract structure by shifting focus to object-centric abstractions in visual data. We introduce a generative model that disentangles objects from backgrounds in the latent space. We then rethink the algebraic structures of object abstractions and propose a novel metric that measures compositionality as a more generic form than disentanglement. Part III incorporates situational context by introducing a sequential decision-making aspect with trajectory data. Here, latent abstractions manifest as actions and plans. We bridge the theories of decision-making and generative modeling, proving that the inference of latent decisions enhances consistency with the model's understanding while optimizing intrinsic values. Whereas these three parts adopt the paradigm of directly learning from raw data, Part IV introduces a dialectic discussion with an alternative paradigm, Knowledge Distillation. We demonstrate how to distill from and accelerate the state-of-the-art massive-scale data-space models by re-purposing our methods and techniques for latent-variable generative modeling.

Together, the contributions of this dissertation enable GenAI systems to overcome the critical bottlenecks of alignment, efficiency, and consistency in representation, inference, and decision-making.

The dissertation of Sirui Xie is approved.

Yingnian Wu

Yizhou Sun

Aditya Grover

Song-Chun Zhu, Committee Co-chair

Demetri Terzopoulos, Committee Co-chair

University of California, Los Angeles

2024

To my parents

TABLE OF CONTENTS

1 Introduction **1**

1.1 Research Objective 2

1.2 Dissertation Overview 2

I Category **8**

2 Learning Symbol-Vector Latent Space in Generative Text Modeling **9**

2.1 Introduction 9

2.2 Preliminaries: Symbol-Vector Coupling EBM 12

2.3 Latent Diffusion Energy-Based Model 13

2.4 Experiments 21

2.5 Discussion and Related Work 29

2.6 Summary 31

3 Emerging Iconic Symbols in a Visual Communication Game **32**

3.1 Introduction 32

3.2 Related Work 35

3.3 The Visual Communication Game 37

3.4 Agents 38

3.5 Learning to Communicate 41

3.6 Experiments 42

3.7 Limitation 51

3.8 Summary 52

II Object **53**

4 Disentangling Objects From Background With Region Competition **54**

4.1 Introduction . . . 54

4.2 Related Work . . . 56

4.3 Method . . . 59

4.4 Experiments . . . 67

4.5 Summary . . . 73

5 Measuring Object Compositionality in Latent Representations . . . **74**

5.1 Introduction . . . 74

5.2 Related Work . . . 79

5.3 Background . . . 81

5.4 Method . . . 85

5.5 Experiments . . . 89

5.6 Summary . . . 99

III Decision **101**

6 Learning Latent Decisions From State-Only Sequences . . . **102**

6.1 Introduction . . . 102

6.2 Non-Markov Decision Process . . . 104

6.3 Learning and Sampling . . . 106

6.4 Decision-Making as Inference . . . 110

6.5 Experiments . . . 113

6.6 Discussion . . . 121

6.7 Summary . . . 122

7 Planning as Inference of Latent Temporal Abstractions . . . 123

7.1 Introduction . . . 123

7.2 Background . . . 125

7.3 Latent Plan Transformer . . . 126

7.4 A Sequential Decision-Making Perspective . . . 131

7.5 Related Work . . . 135

7.6 Experiments . . . 136

7.7 Limitation . . . 141

7.8 Summary . . . 142

IV Knowledge Distillation 143

8 Distilling Data-Space Diffusion Models to Latent-Variable Models . 144

8.1 Introduction . . . 144

8.2 Preliminary . . . 146

8.3 Method . . . 149

8.4 Related work . . . 154

8.5 Experiments . . . 155

8.6 Summary . . . 162

9 Conclusions . . . 163

9.1 Summary of Contributions . . . 163

9.2 Future Work . . . 165

A Derivations and Experimental Details for Chapter 2 . . . 167

B Derivations and Experimental Details for Chapter 3 . . . . . **181**

C Derivations and Experimental Details for Chapter 4 . . . . . **188**

D Derivations and Experimental Details for Chapter 5 . . . . . **211**

E Derivations and Experimental Details for Chapter 6 . . . . . **218**

F Derivations and Experimental Details for Chapter 7 . . . . . **232**

G Derivations and Experimental Details for Chapter 8 . . . . . **236**

References **249**

LIST OF FIGURES

2.1 Diagram of the latent diffusion process 16

2.2 Evaluation on 2D synthetic data 22

2.3 Visualization of color-coded data points 23

3.1 An example of the visual communication game 33

3.2 Communication process 39

3.3 The validation accuracy of different game settings and the ablation baseline 45

3.4 The average communication steps under different settings and ablation baselines 45

3.5 The prediction accuracy when receivers are presented with sketches 46

3.6 Testing results of classifiers trained with sketches evolved under different settings 49

3.7 t-SNE of visual embeddings of unevolved sketches, and evolved sketches . . . 50

3.8 Sketch evolution of rabbit and giraffe 51

4.1 Schematic illustration of Deep Region Competition 61

4.2 Pixel re-assignment 63

4.3 Foreground extraction results for each dataset 69

5.1 An illustration of the parallelogram in COAT 76

5.2 An illustration of trivial compositionality vs object compositionality 77

5.3 Visualization of Object Algebra for Autoencoder and VAEs 92

5.4 Duplicated slots in Slot Attention 93

5.5 Invisible slots in Slot Attention 94

5.6 Greedy matching to remove invisible slots in Slot Attention 94

5.7 Visualization of Object Algebra for IODINE 97

6.1 Graphical model of policy and transition for MDP and nMDP 106
6.2 Results on cubic curve generation for LanMDP and baselines 115
6.3 Mapping a generated curve to a trajectory in the value landscape 117
6.4 Results in MuJoCo for LanMDP and baselines 119

7.1 Graphical model of the Latent Plan Transformer 128
7.2 Trajectory stitching in Maze2D-medium and Maze2D-large environments . . 140
7.3 t-SNE visualization of latent variable in Maze2D-medium 140

8.1 Image samples before and after MCMC correction 149
8.2 Images after 8-step Langevin updates with and without accumulated noise . 151
8.3 Training and evaluation curves 156
8.4 ImageNet samples from the distilled 1-step generator 158
8.5 Samples from the student model distilled from Stable Diffusion v1.5 161

A.1 Full evolution of SVEBM-IB and our models 178
A.2 Visualization of $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t)$ over t 180

B.1 t-SNE of visual embedding 182
B.2 Evolution of *rabbit* and *giraffe* under different settings 185
B.3 Evolution of *cow* and *deer* under different settings 186
B.4 Evolution of *horse* and *pig* under different settings 187

C.1 Examples of excluded images 189
C.2 Samples from TM-dSprite 190
C.3 An example situation when using masks 197
C.4 An example situation when thresholding and combining the output 197
C.5 Foreground extraction results on the Caltech-UCSD Birds dataset 198

C.6 Foreground extraction results on the Stanford Dogs dataset 199
C.7 Foreground extraction results on the Stanford Cars dataset 200
C.8 Foreground extraction results on the CLEVR6 and TM-dSprites datasets . . 201
C.9 Failure modes on the Caltech-UCSD Birds and TM-dSprites datasets 202
C.10 Results of GrabCut on the Caltech-UCSD Birds and TM-dSprites datasets . 203
C.11 Results of ReDO on the Caltech-UCSD Birds and TM-dSprites datasets . . 204
C.12 Results of IODINE on Caltech-UCSD Birds datasets 205
C.13 Results of IODINE on TM-dSprites datasets 206
C.14 Results of Slot-Attention on Caltech-UCSD Birds datasets 207
C.15 Results of Slot-Attention on the TM-dSprites datasets 208
C.16 Image histograms for foreground objects and background regions 208
C.17 Heatmaps of ground-truth masks for each dataset 209
C.18 Preliminary results on learning slot-based object representation 209
C.19 Failure modes of energy-based slot representation model 210

D.1 Example of the test corpus of COAT 212
D.2 Example of the test corpus of COAT 213
D.3 Example of the IID training data with colorful background 214
D.4 Example of the correlated training data with colorful background 214
D.5 COAT l2 for Autoencoder and VAEs 216
D.6 COAT acos for Autoencoder and VAEs 217

E.1 More results for cubic curve generation 228

G.1 Additional qualitative results for ImageNet 240

LIST OF TABLES

2.1 Results of language generation on PTB dataset . . . 24
2.2 Sentence completion on JerichoWorld dataset . . . 24
2.3 Results of interpretable text modeling on Daily Dialog . . . 25
2.4 Evaluation results on Stanford Multi-domain Dialog and Daily Dialog . . . 26
2.5 Samples of unsupervisedly discovered action categories and utterances . . . 26
2.6 Dialog cases generated by the LDEBM given the context . . . 27
2.7 Accuracy of sentence attribute control on Yelp . . . 28
2.8 Generated positive and negative reviews on Yelp . . . 28
2.9 Accuracy on AGNews . . . 29

3.1 Game settings and results . . . 43
3.2 Semantic correlation between target word vectors and visual features . . . 50

4.1 Foreground extraction results on training data measured in IoU and Dice . . . 68
4.2 Ablation study on Caltech-UCSD Birds . . . 71
4.3 Performance of DRC on training and held-out testing data . . . 72
4.4 Performance of DRC on unseen testing categories . . . 72

5.1 Models characteristics and performace on ARI and COAT . . . 90

6.1 Comparison between Markovian and non-Markovian policy in MuJoCo . . . 120
6.2 Computational overheads . . . 120

7.1 Evaluation results of offline OpenAI Gym MuJoCo tasks . . . 138
7.2 Evaluation results of Maze2D tasks . . . 139
7.3 Evaluation results of Antmaze tasks . . . 139

7.4 Evaluation results on Connect Four . . . 141

8.1 EMD-8 on ImageNet 64×64, 100k steps of training . . . 157

8.2 Evaluation results for class-conditional generation on ImageNet 64×64 . . . 159

8.3 Evaluation results for class-conditional generation on ImageNet 128×128 . . . 160

8.4 FID-30k for text-to-image generation in MSCOCO . . . 160

8.5 CLIP Scores in high CFG regime . . . 161

A.1 Network architecture for the LDEBM prior . . . 173

A.2 Hyperparameters of the LDEBM . . . 174

B.1 Categories used in the visual communication game . . . 181

C.1 Dimension of latent variables on each dataset . . . 190

C.2 Architecture of the generators, LEBMs and auxiliary classifiers . . . 192

E.1 Hyper-parameters in MuJoCo experiments . . . 231

F.1 Gym-Mujoco Environments LPT Model Parameters . . . 233

F.2 Maze2D Environments LPT Model Parameters . . . 233

F.3 Franka Kitchen Environments LPT Model Parameters . . . 234

F.4 Connect 4 LPT Model Parameters . . . 234

F.5 Ablation study on Gym and Connect Four . . . 234

F.6 Evaluation results of online Open AI Gym MuJoCo and Antmaze tasks . . . 235

G.1 Hyperparameters for EMD on ImageNet 64×64 . . . 238

G.2 Hyperparameters for EMD on ImageNet 128×128 . . . 239

G.3 Hyperparameters for EMD on Text-to-image generation . . . 239

G.4 More side-by-side comparisons on text-to-image . . . 242

G.5 More side-by-side comparisons on text-to-image 243
G.6 More side-by-side comparisons on text-to-image 244
G.7 More side-by-side comparisons on text-to-image 245
G.8 Additional qualitative results of EMD 246
G.9 Additional qualitative results of EMD 247
G.10 Additional qualitative results of EMD 248

ACKNOWLEDGMENTS

First and foremost, I would like to thank my PhD advisors, Professors Song-Chun Zhu, Ying Nian Wu, and Demetri Terzopoulos. Song-Chun's visionary guidance has significantly broadened my research horizons. Our deep and intriguing conversations have been a wellspring of inspiration. I am particularly grateful for his trust and support in several high-risk projects that I proposed early in my doctoral study. I hope this dissertation, to some extent, satisfies his relentless pursuit of philosophy. Ying Nian has been more of a best friend than an advisor to me. He is the one who always validates my feelings, restores my confidence, and brings peace and clarity to my mind. I feel extremely fortunate to be influenced by his taste of research and wisdom of life. Even after five years of mentorship, I still feel that there are countless things that I must learn from him. Dare I presume the honor of considering him my kindred spirit? Demetri is my role model as a scholar. I took his classes in my early years at UCLA and was deeply impressed by his curiosity and passion for science. To this day, I continue to learn from him the delicate balance of formality and intuition as well as sharpness and tenderness. This dissertation wouldn't be in its current form without him. And I deeply regret that I didn't work with him more due to the COVID-19 pandemic. Having these three professors as my advisors, I could not have asked for more. Their combined influence has shaped me profoundly as a researcher and as an individual.

I would also like to express my gratitude to Professors Yizhou Sun and Aditya Grover for serving on my thesis committee. I thank them for their flexibility and understanding during periods of change in my academic circumstances. I extend my appreciation to the UCLA Computer Science Department for its proactive approach and unwavering dedication to ensuring the continuity of my research, especially to Helen Tran, the Graduate Student Affairs Officer.

My experience at UCLA would have been incomplete without my peers at the Center for Vision, Cognition, Learning, and Autonomy. Feng Gao gave me a warm welcome during my campus visit and then became my first collaborator in the lab. I thank him

for the care and protection he provided to our research project. Yixin Zhu connected me with the resources inside and outside the lab. I thank him for introducing me to the colorful world of UCLA. Chi Zhang was my roommate during the pandemic. We have lots of unforgettable memories cooking and playing *overcooked* together. I thank him for demonstrating his brightness, resilience, and self-discipline. Baoxiong Jia is a nice and caring friend who always brings fun to the table. I thank him for the deep connections that largely resolved my loneliness during the quarantine time at Weyburn, and for catalyzing my collaborations with other peers in the lab. Shuwen Qiu is my mentee in developing research projects and my mentor in developing kindness, empathy, and self-awareness. I thank her for providing emotional support during the darkest days of my life. Xiaojian Ma and Peiyu Yu are close collaborators with whom I developed the blueprint of this dissertation. I thank them for bearing with me in my premature projects, and for their resourcefulness to navigate complex scenarios. They are now rising stars in their fields of expertise.

Many senior mentors have led, guided, and supported me through this journey. Chunxiao Liu, Xinjiang Wang, Zhanghui Kuang, and Wei Zhang at SenseTime Research introduced me to the world of AI and Machine Learning when I felt lost after graduating from HKUST. I thank them for their mentorship and trust. My appreciation also extends to the management-level support from Xiaoou Tang, Xiaogang Wang, Dahua Lin, Shihong Lao, and Liang Lin. Their endorsements propelled my intellectual endeavor. I miss them all, for I almost lost touch with them during the past 5 years. Ramakrishna Vedantam and Ari Morcos, who were my mentors at FAIR, pushed me out of my comfort zone and taught me a more intuitive way of thinking. Onkar Dabeer and Dongqing Zhang provided support, patience, and assistance during my short stay at AWS AI. Tao Zhu, Jiahui Yu, and Zhishuai Zhang introduced me to Google Brain/DeepMind. They kindly supported me when I was struggling with other things in life. I hope to have the chance to give back to them someday. Ruiqi Gao, Zhisheng Xiao, Ben Poole, Tim Salimans, Durk Kingma, Tingbo Hou, and Kevin Murphy gave me a fruitful internship at Google DeepMind/Research. I thank them for the stimulating discussions and the great

collaboration. I also got the chance to work with Stefano Ermon when I was finishing up. I enjoyed his trademark clarity in thinking and do hope that we can have further collaboration in the future.

I have also had the privilege of mentoring several junior students, with whom I've enjoyed working ever since my employment at SenseTime. Hehui Zheng, the most gifted mentee I have ever had, has become a long-term friend. Her thoughtful perspectives have always been a source of inspiration. I felt remorse when seeing her suffer with her soft robots, but she seemed to enjoy it. I am very fortunate to work with Ying Nian's student Deqian Kong, and thank him for always being warm and encouraging, as well as being a comrade in my intellectual quest both in research and in life. Aoyang Qin, Song-Chun's student at Tsinghua, contributed significantly to one of the pillar projects of this dissertation. I admire his attitude towards knowledge and life.

Pursuing my doctoral study during the COVID-19 pandemic has been an immense challenge, both physically and mentally. I am deeply grateful for the profound connections I have been fortunate to make with members of our human species, from ancient times to the present, whose insights and support blessed me through this difficult journey. I would especially like to express my gratitude to Zhuo Chen, Xinyi Cao, Ruoxi Jia, Xiaoxi Du, Xiangning Chen, Haotian Wu, Yingfei Wang, Yining Hong, and Siqing Tang for being my long-term no-filter friends; to Zhuang Zhou, Friedrich Nietzsche, and Ludwig Wittgenstein for foundational tenets drawn from their philosophies; to Hermann Hesse, Ang Lee, Hideaki Anno, and Satoshi Kon for their poetic and inspirational portrayals of psychological and spiritual phenomena; and to Wolfgang Amadeus Mozart, Milan Kundera, Krzysztof Kieślowski, and Hamaguchi Ryūsuke for the emotional sustenance provided by their masterpieces.

Finally, I would like to express my deepest gratitude to my parents, Jie Xie and Yinhua Xiong, as well as my grandparents Songying Wang and Jinxiang Xie, for nurturing my curiosity, for loving me unconditionally, and for gifting me the name 思锐 (Si Rui) — "sharp thinking" — a constant reminder of their hopes for my intellectual journey. This one is for them.

VITA

2012–2016	B.Eng., (Computer Science and Engineering), The Hong Kong University of Science and Technology
2019–2021	M.Sc., (Computer Science), UCLA
2022–	Ph.D. Candidate, (Computer Science), UCLA
2016–2019	Researcher, SenseTime Research
2019–2021	Graduate Student Researcher, Computer Science Department, UCLA
2021–2023	Teaching Assistant, Computer Science Department, UCLA
2021–2022	Research Scientist Intern, Meta FAIR Labs
2022	Applied Scientist Intern, Amazon AWS AI
2022–2024	Student Researcher, Google DeepMind

PUBLICATIONS

* Equal contribution and co-primary authorship.

Sirui Xie, Zhisheng Xiao, Durk Kingma, Tingbo Hou, Ying Nian Wu, Kevin Murphy, Tim Salimans, Ben Poole, Ruiqi Gao, "EM Distillation for One-step Diffusion Models," in submission to *NeurIPS 2024*.

Deqian Kong*, Dehong Xu*, Minglu Zhao*, Bo Pang, Jianwen Xie, Andrew Lizarraga, Yuhao Huang, **Sirui Xie***, Ying Nian Wu, "Latent Plan Transformer: Planning as Latent Variable Inference," in submission to *NeurIPS 2024*.

Aoyang Qin*, Feng Gao, Qing Li, Song-Chun Zhu, **Sirui Xie***, "Learning non-Markovian Decision-Making from State-only Sequences," in *Neural Information Processing Systems (NeurIPS)*, 2023.

Peiyu Yu, Yaxuan Zhu, **Sirui Xie**, Xiaojian Ma, Ruiqi Gao, Song-Chun Zhu, Ying Nian Wu, "Learning Energy-Based Prior Model with Diffusion-Amortized MCMC," in *Neural Information Processing Systems (NeurIPS)*, 2023

Shuwen Qiu*, **Sirui Xie***, Lifeng Fan, Tao Gao, Song-Chun Zhu, Yixin Zhu, "Emergent Graphical Conventions in a Visual Communication Game," in *Neural Information Processing Systems (NeurIPS)*, 2022.

Sirui Xie, Ari Morcos, Song-Chun Zhu, Ramakrishna Vedantam, "COAT: Measuring Object Compositionality in Emergent Representations," in the *Proceedings of the International Conference on Machine Learning*, 2022.

Peiyu Yu, **Sirui Xie**, Xiaojian Ma, Baoxiong Jia, Ruiqi Gao, Yixin Zhu, Song-Chun Zhu, Ying Nian Wu, "Latent Diffusion Energy-Based Model for Interpretable Text Modelling," in the *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

Peiyu Yu, **Sirui Xie**, Xiaojian Ma, Yixin Zhu, Ying Nian Wu, Song-Chun Zhu, "Unsupervised Foreground Extraction via Deep Region Competition," in *Neural Information Processing Systems (NeurIPS)*, 2021.

Sirui Xie, Xiaojian Ma, Peiyu Yu, Yixin Zhu, Ying Nian Wu, Song-Chun Zhu, "HALMA: Humanlike Abstraction Learning Meets Affordance in Rapid Problem Solving," *arXiv: 2102.11344*, 2020. Short version presented at the *ICLR 2021 Workshop on Generalization Beyond the Training Distribution in Brains and Machines*.

Chi Zhang*, **Sirui Xie***, Baoxiong Jia*, Ying Nian Wu, Song-Chun Zhu, Yixin Zhu, "Learning Algebraic Representation for Systematic Generalization in Abstract Reasoning," in the *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

Sirui Xie*, Shoukang Hu*, Xinjiang Wang, Chunxiao Liu, Jianping Shi, Xunying Liu, Dahua Lin, "Understanding the Wiring Evolution in Differentiable Neural Architecture Search" , in the *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

Junning Huang*, **Sirui Xie***, Jiankai Sun, Qiurui Ma, Chunxiao Liu, Dahua Lin, Bolei Zhou, "Learning a Decison Module by Imitating Drivers' Control Behaviors," in the *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

Shoukang Hu*, **Sirui Xie***, Hehui Zheng, Chunxiao Liu, Jianping Shi, Dahua Lin, "DSNAS: Direct Neural Architecture Search without Parameter Retraining," in the *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Peiwen Lin, Peng Sun, Guangliang Cheng, **Sirui Xie**, Xi Li, Jianping Shi, "Graph-guided Architecture Search for Real-time Semantic Segmentation," in the *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Sirui Xie, Hehui Zheng, Chunxiao Liu, Liang Lin, "SNAS: Stochastic Neural Architecture Search," in the *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Sirui Xie, Junning Huang, Chunxiao Liu, Lanxin Lei, Zheng Ma, Wei Zhang, Liang Lin, "NADPEx: An On-Policy Temporally Consistent Exploration Method for Deep Reinforcement Learning," in the *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

CHAPTER 1

Introduction

Abstractions play a fundamental role in human intelligence. They enable us to create mental models that capture the essential patterns of objects, ideas, or situations while filtering out irrelevant details. This cognitive ability enhances our capacity to make predictions and decisions across diverse domains, ultimately allowing us to understand and navigate the world more efficiently. Situated at the intersection of the fields of machine learning, cognitive science, developmental psychology, and neuroscience, a fundamental challenge for the development of artificial intelligence is abstraction learning and inference that supports the autonomous organization of complex information into knowledge that aligns with human understanding.

The study of information patterns from a generative perspective has a rich history across various disciplines. Among them, Grenander's *pattern theory* (Grenander, 1970; Grenander and Miller, 2007) is a unifying framework in which patterns are represented through probabilistic models on random variables with algebraic structures. This mathematical approach encompasses most modern probabilistic generative models, which have driven the impressive advancements in *Generative AI* (Brown et al., 2020; Team et al., 2023; Touvron et al., 2023; Ramesh et al., 2021; Saharia et al., 2022; Brooks et al., 2024). However, most of these recent successes model the probability distributions of observed data. To date, we still lack robust methodologies for extending pattern theoretic principles to model abstractions latent within the data.

1.1 Research Objective

The objective of the research presented in this dissertation is to develop abstraction learning and inference methods for latent-variable generative models. In particular, we study the key methodological aspects of latent abstractions:

- *Modeling:* How does one build generic priors into latent-variable models for the algebraic structures of various abstractions across diverse data modalities?
- *Learning:* What are the effective objectives for unsupervised latent abstraction learning in generative modeling, and how are they related?
- *Inference:* What are the advantages of explicit inference over latent abstractions in learning and decision-making, and how can it be implemented?

These aspects are concretely investigated in the generative constructs of three latent abstractions: Category, Object and Decision. Depending on the structures of the abstractions, each individual case study may focus on different aspects. This dissertation provides multifaceted answers to the above questions that open avenues of future research on unsupervised latent abstraction learning and inference in generative modeling.

1.2 Dissertation Overview

This dissertation consists of four parts. The first three are dedicated to developing the pattern theoretic approach in case studies of core abstractions in human cognition: *Category*, *Object*, and *Decision*. These topics are inspired by how developmental psychologists study the human learning of abstract concepts (Carey, 2000). In the final part, we show how these methods and techniques can be applied to an alternative paradigm that is currently attracting much interest the research community.

1.2.1 Category

Category is the most fundamental abstraction in human knowledge. The development of categorization abilities in humans begins in infancy (0–2 years of age) (Bloom, 2002). While the structures between categories (independent or correlated, hierarchical or compositional) are the usual perspectives for studying this abstraction, we are more interested in the (discrete) symbol versus (continuous) vector duality of its representation. This is inspired the well-known duality between particles and waves in physics, which can be interpreted as the impossibility of a strict discretization and the existence of a continuum where all categories reside. Part I of this dissertation investigates the emergence of symbol-vector representations for categories. It consists of the following two chapters, which provide distinctive views to the symbol-vector duality:

Chapter 2 is primarily based on our publication (Yu et al., 2022b) wherein we introduce a generative model with coupled vector and symbolic latent variables. This model is learned with Maximum Likelihood Estimation, augmented by an Information Bottleneck (IB). We further introduce geometric structures into the vector representation to encourage latent clustering. We show that with a dedicated design that introduces multiple levels of noises to the latent space to facilitate Markov Chain Monte Carlo (MCMC) sampling, this latent variable model with a specially structured prior can match the distribution of the observed text data, and create categorically interpretable latent abstractions in the meantime. This chapter demonstrates how to design latent abstraction models with the language and tools from generative modeling.

Chapter 3 is primarily based on our publication (Qiu et al., 2022) wherein we introduce a visual communication game resembling a repeated version of *pictionary*. The communication between the sender and the receiver is a metaphor for the encoder-decoder architecture of representation learning. Uniquely to this game, the representation of the latent abstractions is sequentially generated with a Markov Decision Process (MDP). By developing a novel learning algorithm with techniques from Reinforcement Learning (Sutton and Barto, 2018), we show how the modeling of this MDP implements the IB in a way

that is distinct from the conventional framework of generative modeling, as in Chapter 2. We demonstrate how we can define metrics for *symbolicity*, *iconicity*, and *semanticity* to assess the representational structure of the emergent abstractions for the symbol-vector duality. We discover some intriguing phenomena suggestive of the advantages of this sequentially modeled latent abstraction in alignment and generalization.

1.2.2 Object

Object is another core abstraction in human cognition. Experiments show that both adults and infants form latent abstractions of objects, which are usually referred to as 'object files' in psychology (Kahneman et al., 1992). To learn an object-centric abstraction bears overloaded meanings in computer vision and cognitive science, which may include acquiring structured representations of texture, shape, part-whole structure, functionality, etc. At the start of this doctoral study, there already existed abundant prior literature on this topic. Nevertheless, the following two chapters, which comprise Part II of the dissertation, are relevant and significant because one demonstrates a holistic prototype that answers all the above questions and the other evaluates and challenges existing models.

Chapter 4 is primarily based on our publication (Yu et al., 2021) wherein we introduce a specially structured Latent Energy-Based Model whose inference realizes a latent abstraction variant of the seminal image segmentation method, Region Competition (Zhu and Yuille, 1996). We show how to build generic statistical inductive biases in both the prior and the generator to encourage structures in the latent space. We apply the EM algorithm (Dempster et al., 1977) as a universal framework to achieve Maximum Likelihood Estimation. We further demonstrate the advantage of iterative inference over latent abstraction, following the Bayesian principles, with state-of-the-art performance and out-of-distribution generalization in natural image foreground disentanglement tasks.

Chapter 5 is primarily based on our publication (Xie et al., 2022) wherein we introduce a new metric to measure compositionality, an algebraic structure for object abstraction

that is more general than disentanglement. We demonstrate how to form a minimal hypothesis for representations that preserves the equivariance of object compositionality and how to realize it as a statistical test. By measure existing models against this metric, we reveal insights that were obscure in previous disentanglement or segmentation metrics. Surprisingly, existing models seem to lack an understanding of the absence or the unique identity of an object, which are intuitive properties of object abstractions.

1.2.3 Decision

Decision is the abstraction that underpins the core cognition of agency. A decision differentiates itself from a passive transition because there is an associated value that this decision is expected to optimize. The study of Heider and Simmel (1944) is one of the most well-known experiments that illustrate the latent decisions that humans can infer from videos of geometric shapes moving around. While the conventional modeling for sequential decision-making is based upon the optimization of the expected cumulative returns over a Markov Decision Process, we challenge it with the outlook of non-Markovian context induced by partial observability and the validity of extrinsic, human-crafted, step-wise rewards. In Part III, of the dissertation, comprising the following two chapters, we introduce new formalizations of decision-making with the language of generative sequence modeling to bypass the limitation of Temporal Difference (TD) learning (Sutton and Barto, 2018):

Chapter 6 is primarily based on our publication (Qin et al., 2023) wherein we design a generative model, Latent-action non-Markov Decision Process (LanMDP), for learning from demonstrations (state-only sequences). We start by developing Maximum Likelihood learning and associated inference algorithms from a purely generative standpoint and then derive an interpretation from the decision-making perspective. The connection is built by constructing a non-Markovian reward that instantiates a Bellman fixed point for the Maximum Likelihood Estimate. This constructive proof justifies the equivalence of a non-Markovian decision-making objective that maximizes the intrinsic value and the

autoregressive sequence modeling objective over the demonstrations. This equivalence provides profound insights for the consistency between the understanding of an agent's prior experience and the agent's intrinsic value.

Chapter 7 is primarily based on our publication (Kong et al., 2024) wherein we extend the investigation of latent generative modeling for decision-making into the data of trajectory-return pairs. The introduction of the extrinsic returns provides flexibility for task specification without involving human interventions into the intricate credit assignment task. We design a new generative model, Latent Plan Transformer, in which we introduce a temporal abstraction of plans. We show that the inference of this latent abstraction during training enforces temporal consistency, potentially resolving the breakdown of the equivalence in Chapter 6 when the Maximum Likelihood Estimation has not been optimized to consistency. We also develop a principled implementation for the exploration-exploitation trade-off with the inference of the latent plan. Thus, we offer an affirmative answer to the advantages of latent abstraction inference in both learning and planning. Due to the architectural similarity between LPT's trajectory generator and state-of-the-art Large Language Models (Brown et al., 2020; Team et al., 2023), this answer can be further generalized to modern GenAI systems for real-world reasoning and decision-making.

1.2.4 Knowledge Distillation

While all the abstraction learning methods that we introduce in the previous three parts of this dissertation directly deal with raw data, it is debatable whether such a learning-from-scratch paradigm is efficient, especially given the currently popular research trend focusing on foundational generative models (Bommasani et al., 2021). In the final part of the dissertation, we provide a dialectic discussion with knowledge distillation as an alternative paradigm that fits better with the ongoing research efforts.

Chapter 8 is primarily based on our publication (Xie et al., 2024) wherein we introduce an algorithm, EM Distillation, that distills knowledge from data-space generative

models to latent-variable models. Despite of the paradigmic shift, we demonstrate how the Maximum Likelihood learning and MCMC-based inference techniques developed in the earlier chapters can be seamlessly applied to promote empirical success at significantly larger scales. This success validates the generality of the methodological research independent from the learning paradigm, opening up new avenues for future research into unsupervised abstraction learning.

Part I

Category

CHAPTER 2

Learning Symbol-Vector Latent Space in Generative Text Modeling

2.1 Introduction

Categories are arguably the most fundamental abstractions in human cognition. Their external manifestations — symbols such as nouns, verbs, and adjectives — are universal across human languages. As the building blocks of human knowledge, these symbols have fueled modernization. However, an intriguing phenomenon associated with this powerful cognitive faculty is the frustration that often accompanies the joy of organizing things into categories, stemming from the inevitable ambiguity inherent in the thought process of categorization. Franz Kafka masterfully captured this subtle tension in his surrealist novels, *The Trial* and *The Castle*. This duality prompts us to question the origins of symbols and whether they are the sole representational form for categories. In this opening chapter of the technical body of the dissertation, we explore a generative model that not only learns the latent abstraction of categories, but also produces a representation that couples the dual forms of discrete symbols and continuous vectors. To ensure interpretability, we focus our initial investigation on text data.

Text modeling has achieved impressive progress with the fast development of neural generative models (Serban et al., 2016; Li et al., 2017a; Zhao et al., 2017; Gupta et al., 2018; Zhao et al., 2018a). It allows near human-level text generation quality and also leads to a wide range of real-world applications such as dialog system (Young et al., 2013) and machine translation (Brown et al., 1993). Although the quality of generation (*e.g.*, fluency and diversity) is the primary concern of most work, interpretability of the

generation process has drawn much attention recently. Among the existing frameworks, the Deep Latent Variable Model (DLVM) is especially suitable for the task, as the learned latent space can capture high-level structures with semantic meanings like topics (Wang et al., 2019) and dialog actions (Zhao et al., 2018b); such a latent space can further enable more interpretable text modeling, featuring unsupervised text attributes discovery (Wen et al., 2017), conditional and controllable text generation (Fang et al., 2019; Shi et al., 2020), and semi-supervised text classification (Pang and Wu, 2021).

In essence, the DLVM summarizes the observed sample (*e.g.*, a piece of text) into inferred latent variables. Earlier text-modeling methods with the DLVM mostly follow the formulation of the Variational Auto-Encoder (VAE) (Kingma and Welling, 2014a; Rezende et al., 2014; Bowman et al., 2016), which assumes a continuous latent space. More recently, Zhao et al. (2018b) explore the possibility of using a discrete latent space to capture dialog actions, andShi et al. (2020) propose to use the VAE with the mixture of Gaussians as the prior, demonstrating promising interpretability of dialog utterance generation. To further improve the expressivity of the latent space, Pang and Wu (2021) leverage the flexibility of *energy-based priors* (Pang et al., 2020a) and learn a structured latent space for interpretable text generation and classification. Specifically, they propose a symbol-vector coupling prior model. The continuous latent variables are coupled with discrete one-hot symbol variables, allowing better discrete structure induction without sacrificing the generation quality offered by the continuous latent space. However, similar to learning an Energy-Based Model (EBM) in data space, the learning of energy-based priors requires Markov Chain Monte Carlo (MCMC) sampling, whose quality can degenerate in practice (Grathwohl et al., 2019; Nijkamp et al., 2019, 2020a; Gao et al., 2020), especially on data with complex latent structures; it often leads to instability during training. As we demonstrate empirically in Section 2.4.1, this phenomenon is particularly concerning when adopting the variational learning scheme to update model parameters.

To remedy this MCMC sampling issue, we may take a look at the endeavor of EBM learning in general. Among the recent efforts, methods drawing inspiration from diffusion

probabilistic models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song and Ermon, 2020; Song et al., 2020b) have demonstrated superior results. In particular, Gao et al. (2020) propose a diffusion recovery likelihood method to learn and sample from a sequence of EBMs defined on increasingly noisy versions of a dataset and the models are trained by optimizing conditional likelihoods, which are more tractable than the marginal likelihood. This greatly mitigates the burden of sampling during training. A natural question thus emerges: *Can we leverage the methodology of diffusion models to address the learning issue of energy-based priors?*

In this chapter, we make the first attempt to address the learning issue of energy-based priors by leveraging diffusion models in the latent space, with a focus on interpretable text modeling. We first unveil the non-trivial symbiosis between latent-space EBMs and diffusion models. Specifically, we focus on the symbol-vector coupling prior and we construct a flexible process that restores the hidden structure in text data by noise-level-aware sampling from a learned sequence of conditional EBMs in the latent space. A variational learning framework is then derived from it. We further employ a geometric clustering-based regularization that complements the symbol-inducing information bottleneck to improve the quality of learned latent space. We refer to the resulting model as the Latent Diffusion Energy-Based Model (LDEBM). Compared to Gao et al. (2020) who deal with EBMs in the data space, the LDEBM is directly applicable to text data with or without labels; it extracts interpretable latent structures that benefit potential downstream tasks such as semi-supervised classification. Although there are methods that use diffusion models in the latent space, some of which have achieved very impressive image generation results (*e.g.*, (Vahdat et al., 2021)), few of them to our knowledge have explored (unsupervised) symbol induction in the latent space, especially on text data. In addition, our method can be trained from scratch and forms a well-structured latent space without pretraining, as required by concurrent work on image modeling, such as (Vahdat et al., 2021) and (Nie et al., 2021). In our experiments on generative modeling and interpretable text modeling, the LDEBM largely outperforms strong counterparts in terms of both the generation quality and interpretability of the learned latent space.

Contributions (1) We introduce a novel symbiosis of the latent space EBM and diffusion model in a variational learning framework; the model can be trained from scratch, is directly applicable to text data with or without labels, and shows superior sampling quality. (2) We develop a geometric clustering-based regularization jointly with the information bottleneck that tackles the mode-collapse problem in variational learning of the latent space EBM. (3) Our experiments demonstrate that the proposed model learns a well-structured latent space and delivers strong results on interpretable text modeling.

2.2 Preliminaries: Symbol-Vector Coupling EBM

We assume that for an observed high-dimensional sample $\mathbf{x} \in \mathbb{R}^D$, there exists $\mathbf{z} \in \mathbb{R}^d$ as its compact continuous latent variables. We assume that $\mathbf{y}$ is the symbolic one-hot vector indicating one of K categories that $\mathbf{z}$ belongs to. The complete-data distribution is $p_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z}, \mathbf{x}) = p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}) p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$, where $p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z})$ is the joint prior model with parameters $\boldsymbol{\alpha}$, and $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$ is the top-down generation model with parameters $\boldsymbol{\beta}$; henceforth, we use $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ to summarize the parameters. Given $\mathbf{z}$, $\mathbf{y}$ and $\mathbf{x}$ are independent; *i.e.*, $\mathbf{z}$ is sufficient for $\mathbf{y}$ in this model.

Pang and Wu (2021) propose to formulate the joint prior model

$$p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}) = \frac{1}{Z_{\boldsymbol{\alpha}}} \exp(\langle \mathbf{y}, f_{\boldsymbol{\alpha}}(\mathbf{z}) \rangle) p_0(\mathbf{z}), \tag{2.1}$$

as an EBM, where $p_0(\mathbf{z})$ is a reference distribution, assumed to be the non-informative prior (*e.g.*, isotropic Gaussian or uniform) of the conventional generation model, $f_{\boldsymbol{\alpha}}(\mathbf{z}) \in \mathbb{R}^K$ is parameterized by a small multi-layer perceptron, and $Z_{\boldsymbol{\alpha}}$ is the normalizing constant or partition function. The energy term $\langle \mathbf{y}, f_{\boldsymbol{\alpha}}(\mathbf{z}) \rangle$ in (2.1) forms an associative memory that couples the symbol $\mathbf{y}$ and the dense vector $\mathbf{z}$. Given $\mathbf{z}$,

$$p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}) \propto \exp(\langle \mathbf{y}, f_{\boldsymbol{\alpha}}(\mathbf{z}) \rangle) \tag{2.2}$$

becomes a softmax classifier, where $f_{\boldsymbol{\alpha}}(\mathbf{z})$ provides the logit scores for the K categories. Marginally, we have

$$p_{\boldsymbol{\alpha}}(\mathbf{z}) = \frac{1}{Z_{\boldsymbol{\alpha}}} \exp(F_{\boldsymbol{\alpha}}(\mathbf{z}))p_0(\mathbf{z}), \tag{2.3}$$

where the marginal energy term is in a log-sum-exponential form, $F_{\boldsymbol{\alpha}}(\mathbf{z}) = \log \sum_{\mathbf{y}} \exp(\langle \mathbf{y}, f_{\boldsymbol{\alpha}}(\mathbf{z}) \rangle)$. It is shown that the coupling between $\mathbf{z}$ and $\mathbf{y}$ enables a symbol-aware continuous vector computation during prior and posterior sampling, which helps to induce a structural latent space (Pang and Wu, 2021). Finally, the prior model $p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z})$ stands on a generator $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$. In text modeling, let $\mathbf{x} = (\mathbf{x}^{(t)}, t = 1, \ldots, T)$ be a sentence, where $\mathbf{x}^{(t)}$ is the t-th token. $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$ can be defined as a conditional autoregressive model, $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\beta}}(\mathbf{x}^{(t)}|\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t-1)}, \mathbf{z})$. The complete model $p_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z}, \mathbf{x})$ with the energy-based prior $p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z})$ and the generation model $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$ is termed as Symbol-Vector Coupling Energy-Based Model (SVEBM).

In principle, a SVEBM can be learned through maximizing the log-likelihood function, where the learning gradient is $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}[\nabla_{\boldsymbol{\theta}}(\log p_{\boldsymbol{\alpha}}(\mathbf{z}) + \log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}))]$. To estimate the expectation, one may sample from the prior $p_{\boldsymbol{\alpha}}(\mathbf{z})$ and the posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with Langevin dynamics (Welling and Teh, 2011). Since $f_{\boldsymbol{\alpha}}$ is a small network, prior sampling is particularly affordable. In comparison, the posterior sampling can be more expensive as it requires back-propagating through the generation network. One promising solution is to follow the variational learning scheme (Kingma and Welling, 2014a) that amortizes the posterior sampling from $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ by an inference network $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$; MCMC-based sampling can be used for prior samples.

2.3 Latent Diffusion Energy-Based Model

2.3.1 A Symbiosis Between SVEBM and Diffusion Model

Contrasting to the vanilla sampling process of the latent variables in SVEBM, LDEBM follows the philosophy of diffusion probabilistic models (Sohl-Dickstein et al., 2015); it assumes a sequence of perturbed samples, $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T$, to construct a flexible process

that restores the structure in data. First, we define the forward diffusion process that systematically and gradually destroys structure in a data distribution: $\mathbf{z}_0 \sim q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x}); \mathbf{z}_{t+1} = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t + \sigma_{t+1}\boldsymbol{\epsilon}_{t+1}$, where $t = 0, 1, \ldots, T-1$ and $\boldsymbol{\epsilon}_t$ is the zero-mean standard Gaussian noise. The scaling factor $\sqrt{1-\sigma_{t+1}^2}$ ensures that the sequence is a spherical interpolation between the posterior sample and the Gaussian white noise. The forward trajectory and the Markov transition between each perturbed samples $\mathbf{z}_1, \ldots, \mathbf{z}_T$ are thus

$$\begin{aligned} q_{\boldsymbol{\phi}}(\mathbf{z}_{0:T}|\mathbf{x}) &= q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x}) \prod_{t=0}^{T-1} q(\mathbf{z}_{t+1}|\mathbf{z}_t); \\ q(\mathbf{z}_{t+1}|\mathbf{z}_t) &= \mathcal{N}(\mathbf{z}_{t+1}; \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t, \sigma_{t+1}^2\mathbf{I}). \end{aligned} \tag{2.4}$$

Our goal is to learn the generative distribution that describes the same trajectory but in reverse. Inspired by Gao et al. (2020), we start by constructing a sequence of *marginal* EBMs at each diffusion step in the latent space. The *conditional* EBMs aiming at recovering $\mathbf{z}_0$ from noisy inputs then follows as (see the derivation in Section A.1.1):

$$p_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1}) = \frac{1}{\tilde{Z}_{\boldsymbol{\alpha},t}(\mathbf{z}_{t+1})} \exp\left(F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t) - \frac{1}{2\sigma_{t+1}^2}||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2\right), \tag{2.5}$$

where $t = 0, 1, \ldots, T-2$. We denote $\tilde{\mathbf{z}}_t = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t$ for brevity. $F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t)$ is the neural network that parameterizes the energy function at each diffusion step, and $\tilde{Z}_{\boldsymbol{\alpha},t}(\mathbf{z}_{t+1}) = \int \exp\left(F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t) - \frac{1}{2\sigma_{t+1}^2}||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2\right) d\tilde{\mathbf{z}}_t$ is the partition function of each conditional EBM. For $t = T-1$, $p_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1}) = \frac{1}{\tilde{Z}_{\boldsymbol{\alpha},t}} \exp\left(F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t) - \frac{1}{2\sigma_{t+1}^2}||\tilde{\mathbf{z}}_t||^2\right)$ since the diffused samples at time step T should be close to Gaussian white noise; the distribution of $\tilde{\mathbf{z}}_{T-1}$ can thus be exponentially tilting of a zero-mean Gaussian distribution.

(2.5) shares the idea of denoising generative modeling (Bengio et al., 2013), where a denoising autoencoder is trained by maximizing the conditional probabilities of the observed samples given their noisy versions. Compared to the vanilla definition (see (2.3)), the noise-level-aware quadratic term constrains the energy landscape to be localized around the noisy sample; this makes the latent space much less multi-modal and easier to sample from. To be specific, Gao et al. (2020) show that $p_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})$ is approximately

a single-mode Gaussian distribution when σ is sufficiently small; it greatly reduces the burden of MCMC sampling. After sampling $\tilde{\mathbf{z}}_t$ from the model, we can easily obtain $\mathbf{z}_t = \tilde{\mathbf{z}}_t / \sqrt{1 - \sigma_{t+1}^2}$.

Next, we show that the forward and reverse process in the latent space can be naturally integrated into the variational learning scheme to amortize the time-consuming posterior sampling. Similar to VAE, the ELBO in SVEBM is

$$\begin{aligned} \text{ELBO}_{\boldsymbol{\theta},\boldsymbol{\phi}} &= \log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\alpha}}(\mathbf{z})), \end{aligned} \tag{2.6}$$

where D_{KL} denotes the Kullback-Leibler divergence. Since we now consider the full trajectory of the perturbed samples, in LDEBM we may optimize

$$\text{ELBO}_{\text{Diff},\boldsymbol{\theta},\boldsymbol{\phi}} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})} \left[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x}), q(\mathbf{z}_{1:T}|\mathbf{z}_0)} \left[\log \frac{p_{\boldsymbol{\alpha}}(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\right], \tag{2.7}$$

which is a valid ELBO by applying Jensen's inequality to (2.6). The joint training of inference, prior and generation models can be largely reduced to finding the agreement of the forward and reverse Markov transitions defined by $q_{\boldsymbol{\phi}}$ and $p_{\boldsymbol{\theta}}$, respectively. Refer to Section A.1.2 for more detailed derivations and discussions.

Finally, we show how to introduce the symbolic one-hot vector $\mathbf{y}$ into our formulation. We assume a complete data distribution that considers the full trajectory of the perturbed latent variables, $p_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z}_{0:T}, \mathbf{x})$. Among several possibilities for coupling the symbolic vector y with the latent variables, two major options arise: We can couple the symbol with the whole trajectory, *i.e.*, $p_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z}_{0:T}, \mathbf{x}) = p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}_{0:T}) p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_{0:T})$; or we can couple the symbol with only the clean posterior sample $\mathbf{z}_0$, *i.e.*, $p_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z}_{0:T}, \mathbf{x}) = p(\mathbf{z}_T) p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1) \prod_{t=1}^{T-1} p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1}) p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0)$. We prefer the latter one, since it is sufficient to model the reverse Markovian transition, while enabling a simpler and more efficient training scheme following Ho et al. (2020) (see Section 2.3.4). Of note, coupling only $\mathbf{z}_0$ to $\mathbf{y}$ means that we condition only the final reverse diffusion step $[\mathbf{z}_0|\mathbf{z}_1]$ on $\mathbf{y}$ when performing controllable generation. This could be a bit counter-intuitive as no label in-

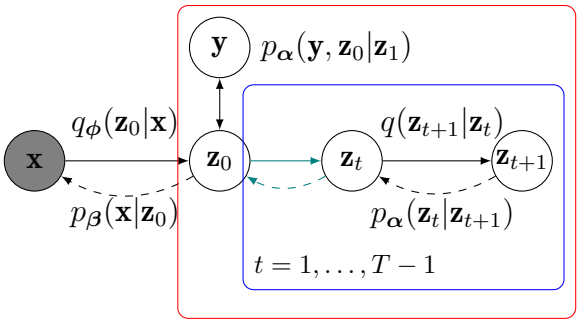


Figure 2.1: Diagram of the latent diffusion process. We construct the forward and reverse diffusion processes in the latent space. The symbolic one-hot vector is coupled with the initial latent vector $\mathbf{z}_0$. The latent and diffused latent variables are highlighted by the red and blue plates, respectively. The cyan arrows indicate that $\mathbf{z}_0$ is connected with only $\mathbf{z}_1$. We learn a sequence of EBMs to model the reverse diffusion process $p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1})$.

formation is injected in previous reverse steps. Theoretically, $\mathbf{y}$ and $\mathbf{z}_{1:T}$ are independent given $\mathbf{z}_0$ in our formulation; however, we empirically observe that $\mathbf{y}$ and $\mathbf{z}_t$ for $t > 0$ are nearly independent even marginally, after we integrating out $\mathbf{z}_{0:t-1}$ in our model. In other words, $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t),\ t > 0$ are in general non-informative since adding noise in the latent space could be much more corrupting than adding noise in the data space. The model learns to enjoy the less multi-modal energy landscape in previous reverse steps; it then seeks the given mode only in the most informative final reverse step. Specifically, we achieve this coupling by similarly defining $p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1)$ as in (2.1) and using the log-sum-exponential form for learning as in (2.3).

Refer to Figure 2.1 for a diagram of our model and to Section A.1.3 and Section A.2.3 for more details and discussions.

2.3.2 Information Bottleneck

To learn the symbolic vector $\mathbf{y}$, we may consider adopting the Information Bottleneck (IB) principle (Tishby et al., 2000), an appealing approach for inducing symbolic representations. In this section, we re-interpret the above ELBO as a cooperative learning objective, defined as the divergence between two joint distributions; we then show how this formulation helps to incorporate the IB-based regularization into LDEBM in a principled manner.

As shown by Han et al. (2019), the variational learning scheme can be regarded as performing alternating projection between two joint distributions, $Q_{\boldsymbol{\phi}}$ and $P_{\boldsymbol{\theta}}$. In our modeling, we have: $Q_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z}_{0:T}) = q_{\text{data}}(\mathbf{x}) q_{\boldsymbol{\phi}}(\mathbf{z}_{0:T}|x)$, and $P_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}_{0:T}) = p(\mathbf{z}_T) \prod_{t=0}^{T-1} p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1}) p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0)$; we use $q_{\text{data}}(\mathbf{x})$ to denote the data distribution of $\mathbf{x}$ for notation consistency. Maximizing $\mathbb{E}_{q_{\text{data}}(\mathbf{x})}[\text{ELBO}_{\text{Diff},\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})]$ over $(\boldsymbol{\theta}, \boldsymbol{\phi})$ is equivalent to minimizing the following divergence:

$$D_{\text{KL}}(Q_{\boldsymbol{\phi}} \| P_{\boldsymbol{\theta}}) = D_{\text{KL}}(q_{\text{data}}(\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{x})) + \mathbb{E}_{q_{\text{data}}(\mathbf{x})}[D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}_{0:T}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z}_{0:T}|\mathbf{x}))], \tag{2.8}$$

since $\mathcal{H}(\mathbf{x}) = -\mathbb{E}_{q_{\text{data}}(\mathbf{x})}[\log q_{\text{data}}(\mathbf{x})]$, *i.e.*, the entropy of data distribution is fixed. Minimizing the KL-divergence $\min_{\boldsymbol{\theta}} \min_{\boldsymbol{\phi}} D_{\text{KL}}(Q_{\boldsymbol{\phi}} \| P_{\boldsymbol{\theta}})$ defines a cooperative game, with the dynamics that $q_{\boldsymbol{\phi}}$ and $p_{\boldsymbol{\theta}}$ run towards each other.

Since the initial posterior sample $\mathbf{z}_0$ is coupled with the symbolic vector $\mathbf{y}$, it should be the most informative latent variable for inducing the discrete symbol. We can therefore plug in (2.8) with a mutual information term between $\mathbf{z}_0$ and $\mathbf{y}$: $\mathcal{I}(\mathbf{z}_0, \mathbf{y}) = \mathcal{H}(\mathbf{y}) - \mathcal{H}(\mathbf{y}|\mathbf{z}_0)$, which essentially incorporates the IB as we show below. Given the distribution $Q_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z}_{0:T})$, we can first define the marginal distribution of $\mathbf{z}_0$ as the aggregated posterior by integrating out $\mathbf{z}_{1:T}$: $q_{\boldsymbol{\phi}}(\mathbf{z}_0) = \mathbb{E}_{q_{\text{data}}(\mathbf{x})}[q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})]$. The entropy of $\mathbf{z}_0$ and conditional entropy of $\mathbf{z}_0$ on $\mathbf{x}$ then follow as $\mathcal{H}(\mathbf{z}_0)$ and $\mathcal{H}(\mathbf{z}_0|\mathbf{x})$, respectively. Taken together, the KL-Divergence with $\lambda \mathcal{I}(\mathbf{z}_0, \mathbf{y})$ can therefore be parsed as

$$\mathcal{L} = D_{\text{KL}}(Q_{\boldsymbol{\phi}} \| P_{\boldsymbol{\theta}}) - \lambda \mathcal{I}(\mathbf{z}_0, \mathbf{y}) = \mathcal{C} + \mathcal{L}_{\text{RC}} + \mathcal{L}_{\text{EBM}} + \mathcal{L}_{\text{IB}}, \tag{2.9}$$

where $\mathcal{C} = -\mathcal{H}(\mathbf{x}) + \sum_{t=0}^{T-1} \mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t)$ does not involve learnable parameters, $\mathcal{L}_{\text{RC}} = -\mathbb{E}_{Q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0)]$ is the reconstruction loss, $\mathcal{L}_{\text{EBM}} = D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}_0) \| p_{\boldsymbol{\alpha}}(\mathbf{z}_{0:T}))$ corresponds with learning latent space models, and $\mathcal{L}_{\text{IB}} = \mathcal{I}(\mathbf{x}, \mathbf{z}_0) - \lambda \mathcal{I}(\mathbf{z}_0, \mathbf{y})$ is the IB, where $\mathcal{I}(\mathbf{x}, \mathbf{z}_0) = \mathcal{H}(\mathbf{z}_0) - \mathcal{H}(\mathbf{z}_0|\mathbf{x})$ is the mutual information between $\mathbf{x}$ and $\mathbf{z}_0$ under $Q_{\boldsymbol{\phi}}$; $\lambda \geq 0$ controls the expressivity of $\mathbf{z}_0$ to $\mathbf{y}$. Refer to Section A.1.4 for more details.

2.3.3 Geometric Clustering Anchors the Modes

As shown in the previous subsection, IB provides an elegant solution for inducing the symbolic vector $\mathbf{y}$. In this section, we further introduce an approach that facilitates the emergence of $\mathbf{y}$ from a geometric perspective. To induce a latent space with interpretable structures, ideally, the location of data points in the latent space encodes their semantic meaning, *i.e.*, it indicates the semantic class; semantically similar points should be placed closer and produce the same symbolic vector $\mathbf{y}$. This resembles geometric clustering algorithms: Labels of data points are assigned based on their geometric (typically Euclidean) distance from each other. Below, we show how to realize this intuition in LDEBM.

Let us consider the joint distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$. We can decompose its log-likelihood into $\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \log p_{\boldsymbol{\theta}}(\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ as in Grathwohl et al. (2019), where $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ is substituted with the ELBO derived in Section 2.3.1. $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ is the classification model in the latent space: $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \approx \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}[p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0)]$. $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0)$ is the softmax classifier of $\mathbf{y}$ based on $\mathbf{z}_0$ similarly as in (2.2), detailed in Section A.1.3. Therefore, we can encode the semantic information from the label $\mathbf{y}$ into $\mathbf{z}_0$ through learning the classifier $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0)$. In case there is full or partial access to the ground-truth semantic class labels, we could directly utilize these labels to supervise the classifier, jointly with the existing ELBO objective. When no label is provided, we generate pseudo label $\hat{\mathbf{y}}$ by clustering $\mathbf{z}_0$, which optimizes $\mathbb{E}_{\mathbf{y}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ instead; $\mathbb{E}_{\mathbf{y}}$ is defined by the clustering algorithm. It is akin to the EM algorithm, where geometric clustering serves as a hard-decision E-step to help induce $\mathbf{y}$. In practice, we employ K-means to cluster $\mathbf{z}_0$. In Section 2.4.1, we empirically show that this strategy learns a better latent space and significantly alleviates the mode-collapse problem.

2.3.4 Algorithms and Implementation

Training and Sampling Algorithms For learning the prior model, we have for each $t = 0, 1, \ldots, T-1$,

$$\nabla_{\boldsymbol{\alpha}} \mathrm{ELBO}_t = \mathbb{E}_{q_{\boldsymbol{\phi}}(\tilde{\mathbf{z}}_t, \mathbf{z}_0|\mathbf{x})}[\nabla_{\boldsymbol{\alpha}} F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t)] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_{t+1}, \mathbf{z}_0|\mathbf{x}), p_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})}[\nabla_{\boldsymbol{\alpha}} F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t)]. \tag{2.10}$$

Let $\psi = \{\boldsymbol{\beta}, \boldsymbol{\phi}\}$ collect the parameters of the inference (encoder) and generation (decoder) models:

$$\begin{aligned} \nabla_\psi \mathrm{ELBO} &= \nabla_\psi \, \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})] \\ &- \nabla_{\boldsymbol{\phi}} \, \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_{0:T}|\mathbf{x})} \left[\log p(\mathbf{z}_T) + \sum_{t=0}^{T-1} \log p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1}) \right]. \end{aligned} \tag{2.11}$$

Recall that we denote $\tilde{\mathbf{z}}_t = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t$. $\mathbb{E}_{p_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})}$ is approximated by MCMC samples from the prior. $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}$ is approximated by samples from the inference network. We also add the gradient from $\mathcal{I}(\mathbf{z}_0, \mathbf{y})$, denoted as $\nabla \mathcal{I}$, to (2.10) and (2.11) during training to incorporate IB. See Section A.1.5 for the detailed derivations.

Note that the expectation in (2.10) requires MCMC sampling (*e.g.*, Langevin dynamics (Welling and Teh, 2011)) of the prior model. For a target distribution $\pi(\tilde{\mathbf{z}})$, the dynamics iterates $\tilde{\mathbf{z}}^{k+1} = \tilde{\mathbf{z}}^k + \frac{s^2}{2}\nabla_{\tilde{\mathbf{z}}} \log \pi(\tilde{\mathbf{z}}^k) + s\boldsymbol{\epsilon}^k$, where k indexes the iteration of the dynamics, s is a small step size, and $\boldsymbol{\epsilon}^k \sim \mathcal{N}(0, \mathbf{I})$ is the Gaussian noise. In this chapter, we follow the heuristics in (Gao et al., 2020) and set the step size $s_t = b\sigma_t c_t$, where $b < 1$ is a tuned hyperparameter, and $c_t = \sqrt{\prod_{i=1}^t \sigma_i/\sigma_1}$ is a scaling factor. Let t indexes the diffusion step; K steps of Langevin dynamics thus iterates

$$\tilde{\mathbf{z}}_t^{k+1} = \tilde{\mathbf{z}}_t^k + \frac{b^2\sigma_t^2 c_t^2}{2}\left(\nabla_{\tilde{\mathbf{z}}} F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t^k, t) - \frac{1}{\sigma_t^2}(\tilde{\mathbf{z}}_t^k - \mathbf{z}_{t+1})\right) + b\sigma_t c_t \boldsymbol{\epsilon}^k. \tag{2.12}$$

In principle, training the model amounts to minimizing the ELBO in (2.7), which requires a summation over all the diffusion steps; it involves sampling a full forward trajectory. To make the training simpler and more efficient, following Ho et al. (2020), we randomly choose one diffusion step from the summation to optimize at each training

Algorithm 1 Learning algorithm of LDEBM

input: initial parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\phi})$, learning rate η, observed unlabeled examples $\{\mathbf{x}^{(i)}\}_{i=1}^{M}$, observed labeled examples $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=M+1}^{M+N}$ (alternative, needed in controllable generation or semi-supervised learning).
repeat
  posterior sampling: For each $\mathbf{x}^{(i)}$, sample $\mathbf{z}_0^{(i)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x}^{(i)})$ using inference network.
  prior sampling: For each $\mathbf{z}_0^{(i)}$, sample diffusion step t from Unif$(\{0, \ldots, T-1\})$, and the perturbed pair $(\tilde{\mathbf{z}}_t^{(i)}, \mathbf{z}_{t+1}^{(i)})$ following (2.4). Set $\tilde{\mathbf{z}}_t^{(i)}$ as the positive sample $\tilde{\mathbf{z}}_t^{(i)+}$. Initialize the MCMC using $\mathbf{z}_{t+1}^{(i)}$ and update by (2.12) for K steps to obtain $\tilde{\mathbf{z}}_t^{(i)-}$.
  learning prior model: Update $\boldsymbol{\alpha}$ with $\eta(\sum_i [\nabla_{\boldsymbol{\alpha}} F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t^{(i)+}, t) - \nabla_{\boldsymbol{\alpha}} F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t^{(i)-}, t)] - \nabla_{\boldsymbol{\alpha}} \mathcal{I})$.
  learning inference and generation models: Update $\boldsymbol{\beta}$ and $\boldsymbol{\phi}$ with (2.11) and $\nabla_{\boldsymbol{\phi}} \mathcal{I}$.
  if labeled data $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ is available **then**
    update $\gamma = (\boldsymbol{\alpha}, \boldsymbol{\phi})$ **using** $\mathbf{y}^{(i)}$**:** Learning gradient $\eta \sum_i \nabla_\gamma \log p_{\boldsymbol{\alpha}_t}(\mathbf{y}^{(i)}|\mathbf{z}_0^{(i)})$ is provided by ground-truth label.
  else if only unlabeled data is available **then**
    update $\gamma = (\boldsymbol{\alpha}, \boldsymbol{\phi})$ **using pseudo-label** $\hat{\mathbf{y}}^{(i)}$**:** Geometric clustering generates $\hat{\mathbf{y}}^{(i)}$ for each $\mathbf{x}^{(i)}$. $\eta \sum_i \nabla_\gamma \log p_{\boldsymbol{\alpha}_t}(\hat{\mathbf{y}}^{(i)}|\mathbf{z}_0^{(i)})$, *i.e.*, the gradient comes from pseudo-label generated by geometric clustering.
  end if
until converged.

iteration. After training, we initialize the reverse trajectory from Gaussian white noise. The synthesized sample at each step serves to initialize an MCMC that samples from the model of the previous step.

The learning and synthesizing algorithms are summarized in Algorithm 1 and Algorithm 2, respectively.

Implementation For the K-means algorithm, we use the implementation of Johnson et al. (2019), which explicitly deals with the empty clusters and trivial parameter-

Algorithm 2 Synthesizing algorithm for LDEBM

input: $\mathbf{z}_T \sim \mathcal{N}(0, \mathbf{I})$
output: $\mathbf{z}_0$
for $t = T-1$ **to** $t = 0$ **do**
  Initialize $\tilde{\mathbf{z}}_t = \mathbf{z}_{t+1}$.
  for $k = 1$ **to** K **do**
    Update $\tilde{\mathbf{z}}_t$ using (2.12).
  end for
  $\mathbf{z}_t = \tilde{\mathbf{z}}_t / \sqrt{1 - \sigma_{t+1}^2}$
end for

ization problems. To emphasize that the proposed model shows better capability of modeling latent space, we use the same encoder and decoder as Pang and Wu (2021) for all the experiments. We use a shared network $F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t)$ for each $t = 0, 1, \ldots, T-1$; $T = 6$; t is encoded by sinusoidal position embedding as in (Ho et al., 2020), and we set σ_t^2 to increase linearly. For Langevin dynamics, we use $K = 50$ and $b^2 = 0.002$ throughout the experiments. See Section A.2.1 for network architecture and further training details.

2.4 Experiments

Through a series of experiments, we empirically examine the capability of LDEBM for generative modeling and interpretability on text modeling tasks. Refer to Section A.2.2 for additional experimental settings and baselines.

2.4.1 Generative Modeling

2D Synthetic Data We first perform experiments of our model on 2D synthetic datasets as a sanity check to validate our assumptions; results are displayed in Figure 2.2. The gap between LDEBM and SVEBM is very clear. As mentioned in Section 2.1, for more complex datasets (*e.g.*, datasets with more modes or more complex data structure), SVEBM struggles to capture regularities in the data; the model is prone to collapse, which features an exploding KL-term and poor performance on generation. We provide more results that show the full evolution of these models during training with more discussions in Section A.2.3. In contrast, LDEBM without geometric clustering already overcomes this problem, performing relatively well in terms of modeling both *posterior* $\mathbf{x}$ and *prior* $\mathbf{x}$. Although LDEBM without geometric clustering faithfully reconstructs the data and shows significant improvement on generation quality, the generated distribution can be slightly distorted, and some modes are missing. The problem is clearer in the latent space: Mode-collapse occurs in the *prior* $\mathbf{z}$ distribution, where the latent structure is broken. LDEBM with geometric clustering maintains the number of modes as in the data distribution and induces a highly-structural latent space, echoing our intuition in

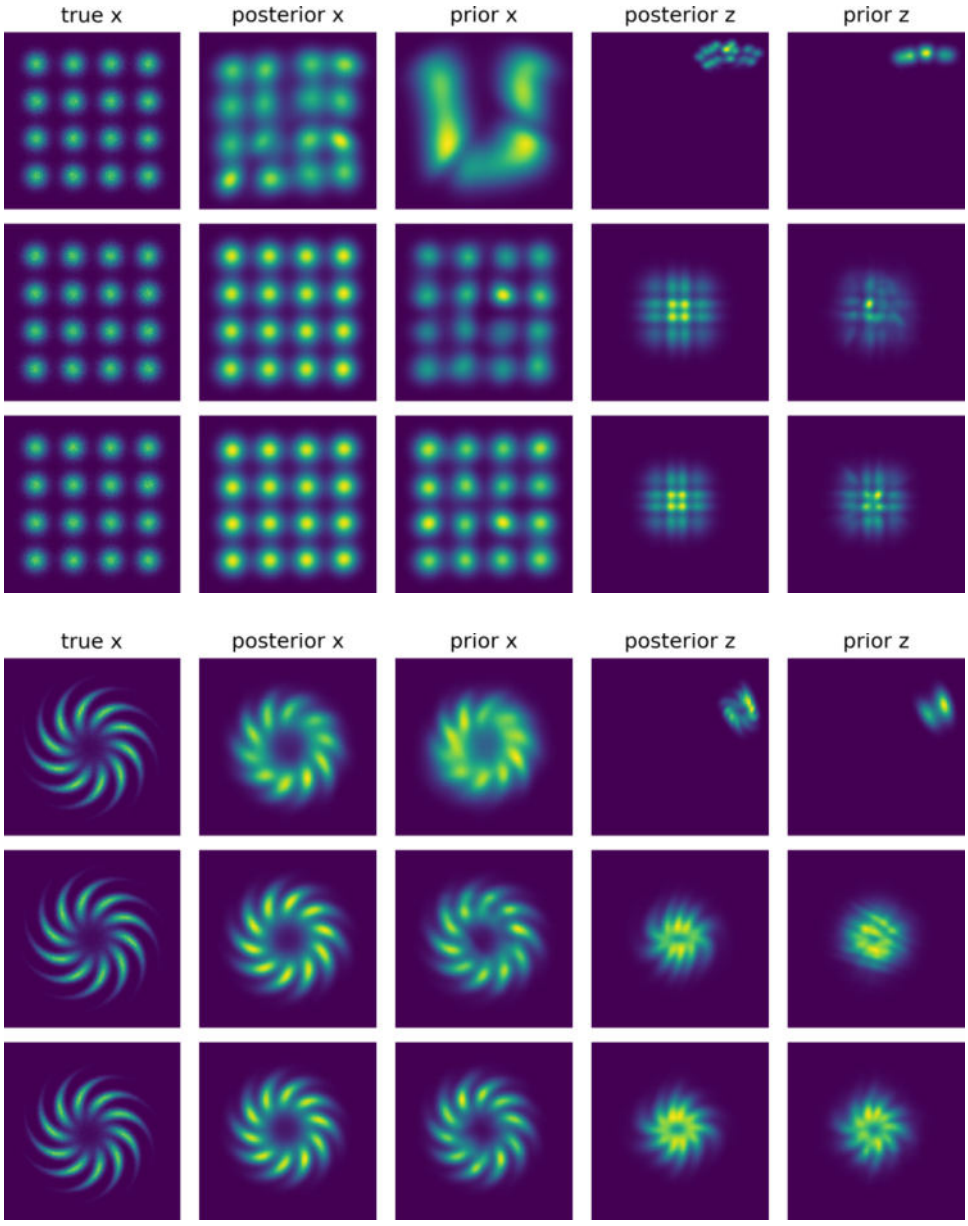


Figure 2.2: Evaluation on 2D synthetic data. A mixture of 16 Gaussians (upper panel) and a 10-arm pinwheel-shaped distribution (lower panel). In each panel, the top, middle, and bottom row display densities learned by SVEBM-IB, our model w/o geometric clustering, and our full model, respectively. In each row, from left to right, it displays the data distribution and the Kernel Density Estimations (KDEs) of: $\mathbf{x}$ generated by amortized posterior $\mathbf{z}$ samples, $\mathbf{x}$ by MCMC sampled prior $\mathbf{z}$ samples, posterior $\mathbf{z}$ samples, and prior $\mathbf{z}$ samples.

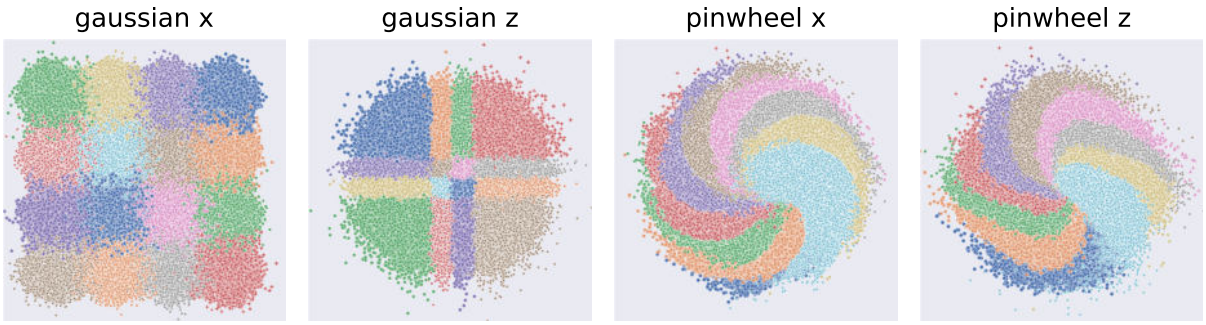


Figure 2.3: Visualization of color-coded data points. We visualize data points and the corresponding inferred latent variables of two 2D synthetic datasets (*gaussian* and *pinwheel*). Data points with different labels are assigned with different colors.

Section 2.3.3. Figure 2.3 shows the structural similarity between the data distribution and the learned latent distribution.

Language Generation Following previous state-of-the-art competitors (Zhao et al., 2018b; Shi et al., 2020; Pang and Wu, 2021), we evaluate the quality of generation on a real-world text dataset, Penn Treebanks (PTB) (Marcus et al., 1993) as pre-processed by Mikolov et al. (2010). We report four metrics of the generation performance: Reverse Perplexity (rPPL) (Zhao et al., 2018a), BELU (Papineni et al., 2002), Word-Level KL Divergence (wKL), and Negative Log-Likelihood (NLL); Table 2.1 summarizes results.

The proposed model, either w/ or w/o geometric clustering, demonstrates the best performance on reconstruction (highest BLEU) and fitting capacity (lowest NLL) than all baseline models. Moreover, the higher expressivity of our models enables the generation of high-quality sentences. The lowest rPPL indicates that our models can further improve over these strong baselines on fluency and diversity of generated text; the lowest wKL indicates that the word distribution of the generated sentences is the most consistent with that of the original data.

Sentence Completion Further, we show that the trained model enables text completion on a masked JerichoWorld dataset (Ammanabrolu and Riedl, 2021). We perform conditional sampling in the latent space to complete the masked sentences; see additional details in Section A.2.3 and Table 2.2.

Model	**rPPL**$^{\downarrow}$	**BLEU**$^{\uparrow}$	**wKL**$^{\downarrow}$	**NLL**$^{\downarrow}$
Test Set	-	100.0	0.14	-
RNN-LM	-	-	-	101.21
AE	730.81	10.88	0.58	-
VAE	686.18	3.12	0.50	100.85
DAE	797.17	3.93	0.58	-
DVAE	744.07	1.56	0.55	101.07
DI-VAE	310.29	4.53	0.24	108.90
semi-VAE	494.52	2.71	0.43	100.67
semi-VAE $+\mathcal{I}$	260.28	5.08	0.20	107.30
GM-VAE	983.50	2.34	0.72	99.44
GM-VAE $+\mathcal{I}$	287.07	6.26	0.25	103.16
DGM-VAE	257.68	8.17	0.19	104.26
DGM-VAE $+\mathcal{I}$	247.37	8.67	0.18	105.73
SVEBM	180.71	9.54	0.17	95.02
SVEBM-IB	177.59	9.47	0.16	94.68
Ours w/o GC	<u>168.32</u>	<u>11.12</u>	<u>0.07</u>	**79.84**
Ours	**164.57**	**11.16**	**0.06**	<u>82.38</u>

Table 2.1: Results of language generation on PTB dataset. The best and second-best performances are marked in bold numbers and underlines, respectively; tables henceforth follows this format.

Input	... A bathroom lies to the south, while a door to the east leads to the living room. On the bed are a driver's license, some keys and a wallet On the end table is a telephone.
Pred.	... A bathroom lies to the south, while a door to the east leads to the living room. On the bed is a wallet. On the end table are a telephone and some keys.
Input	... All around you the crowd is in a state of pandemonium. The paths of least resistance are up, down and west.
Pred.	... All around you the crowd is in a state of pandemonium. The paths of least resistance are down and east.

Table 2.2: Sentence completion on JerichoWorld dataset. The gray words in the input sentences are masked with `<unk>` token.

Model	**MI**$^{\uparrow}$	**BLEU**$^{\uparrow}$	**Act.**$^{\uparrow}$	**Emo.**$^{\uparrow}$
DI-VAE	1.20	3.05	0.18	0.09
semi-VAE	0.03	4.06	0.02	0.08
semi-VAE $+\mathcal{I}$	1.21	3.69	0.21	0.14
GM-VAE	0.00	2.03	0.08	0.02
GM-VAE $+\mathcal{I}$	1.41	2.96	0.19	0.09
DGM-VAE	0.53	7.63	0.11	0.09
DGM-VAE $+\mathcal{I}$	1.32	7.39	0.23	0.16
SVEBM	0.01	11.16	0.03	0.01
SVEBM-IB	2.42	10.04	0.59	0.56
Ours w/o GC	<u>2.44</u>	<u>16.72</u>	<u>0.65</u>	<u>0.63</u>
Ours	**3.94**	**28.75**	**0.74**	**0.74**

Table 2.3: Results of interpretable text modeling on Daily Dialog (DD). We use mutual information (MI), BLEU, and homogeneity with actions and emotions for evaluation.

2.4.2 Interpretable Text Modeling

In this section, we move on to evaluate our model on the interpretability of text modeling.

Unsupervised Text Attributes Discovery First, we examine the efficacy of our model on the unsupervised text attributes discovery task. We assess the model on the DD dataset (Li et al., 2017b), a chat-oriented dataset of 13,118 daily conversations with human-annotated dialog action and emotion labels for the utterances. The interpretability is evaluated through the ability to unsupervisedly capture the utterance attributes of DD. We flatten the dialogues for text modeling and use $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ to infer the utterance label. In particular, we take the argmax of the classification head as the inferred label. Following Zhao et al. (2018b), we recruit homogeneity to evaluate the consistency between ground-truth action and emotion labels and those inferred from our model. Table 2.3 displays the results of our model and baselines. It shows that the proposed model outperform other baselines in reconstruction by a large margin and give a much better homogeneity on both the dialog action and emotion. The superior performance of LDEBM equipped with latent space geometric clustering again verifies our intuition in Section 2.3.3.

Data	Model	**BLEU**$^{\uparrow}$	**Avg.**$^{\uparrow}$	**Extr.**$^{\uparrow}$	**Grdy.**$^{\uparrow}$
Stanford Multi-Domain Dialog (SMD)	DI-VAE	7.06	76.17	43.98	60.92
	DGM $+\mathcal{I}$	10.16	78.93	48.14	64.87
	SVE-IB	**12.01**	**80.88**	51.35	67.12
	w/o GC	11.44	80.16	51.26	66.51
	Ours	11.51	**80.88**	**51.57**	**67.13**
DD	DGM $+\mathcal{I}$	2.19	74.73	45.85	64.28
	SVE-IB	2.23	77.37	43.32	63.99
	Ours	**3.72**	**78.89**	**46.19**	**65.87**

Table 2.4: Dialog evaluation results on SMD and DD. Models are assessed using four metrics: BLEU, average, extrema, and greedy word embedding based similarity.

Action	Request-weather
Utterance	I need to know if it is going to be foggy in Fresno today and tomorrow car.
	Manhattan, please. Will it be cloudy on Monday?
	I need current weather data about New York, specifically information about the temperature.
Action	Request-city
Utterance	In what city are you interested?
	What city would you like to know the weather about?
	Okay, what city should I look in?

Table 2.5: Samples of unsupervisedly discovered action categories and corresponding utterances on SMD.

SMD	
Ctx.	*User:* What gas stations are here? *Sys:* There is a Chevron.
Ref.	That's good! Please pick the quickest route to get there and avoid all heavy traffic!
Pred.	(Req.-address) What is the address? (Req.-route) Please set the quickest route to go.
DD	
Ctx.	*A:* Hi. Have you got a personal computer? *B:* Certainly. What ' s the matter? *A:* I wonder if you often trade with others on the internet.
Ref.	Sure. I often buy things or do business through it without going out to the physical stores.
Pred.	Yes, but I think it is a little different way.

Table 2.6: Dialog cases generated by LDEBM given the context. On SMD, we provide the same context but with different $\mathbf{y}$ values to generate each response; actions indicated by $\mathbf{y}$ are listed in parentheses. On DD, LDEBM can well capture the dialog topic; we provide the ground-truth response in each case for reference.

Conditional Response Generation Next, we evaluate our model on dialog generation with SMD (Eric et al., 2017) and DD datasets. We evaluate the quality of generated responses using BELU and three word-embedding-based topic similarity metrics (Shi et al., 2020): embedding average (Mitchell and Lapata, 2008), embedding extrema (Forgues et al., 2014), and embedding greedy (Rus and Lintean, 2012). Table 2.4 shows that LDEBM has competitive performance compared with SVEBM-IB on SMD and outperforms the strong baselines on all metrics on DD; see qualitative examples in Table 2.5 and Table 2.6.

Sentence Sentiment Control Finally, we inspect the capability of our model for controllable generation on Yelp reviews, pre-processed by (Li et al., 2018). The Yelp dataset is of larger scale, containing 180,000 negative reviews and 270,000 positive ones. For a controllable generation process, the symbolic vector $\mathbf{y}$ is provided to guide the sampling in latent space; see details in Section A.2.3. Following Pang and Wu (2021), we train the model with sentiment supervision and use the same pre-trained classifier

Model	**Overall**$^{\uparrow}$	**Positive**$^{\uparrow}$	**Negative**$^{\uparrow}$
DGM-VAE $+\mathcal{I}$	64.7%	95.3%	34.0%
CGAN	76.8%	94.9%	58.6%
SVEBM-IB	<u>90.1%</u>	<u>95.1%</u>	<u>85.2%</u>
Ours	**99.0%**	**98.8%**	**99.1%**

Table 2.7: Accuracy of sentence attribute control on Yelp

Positive	The food here was very tasty and our server was very attentive.
	I was very satisfied for my birthday party!
	Definitely the best Philly Cheesesteaks I've ever been.
	They are the best customer service ever!
Negative	Ugh the staff is so incompetent and rude.
	It just can't make it worse.
	Avoid this company at all costs.
	Just ruined the experience with a horrible attitude on it.

Table 2.8: Generated positive and negative reviews on Yelp

to determine the sentiment of the generated sentence. The pre-trained classifier has an accuracy of 98.5% on the testing data and thus can accurately evaluate the sentiment of given sentences. The quantitative and qualitative results are summarized in Table 2.7 and Table 2.8, respectively. LDEBM generates positive and negative reviews with a nearly saturate accuracy, significantly outperforming all the baselines.

2.4.3 Semi-Supervised Classification

In this experiment, we switch from neural sequence models used in previous experiments to neural document models (Miao et al., 2016; Card et al., 2018); we show our model can be similarly extended to semi-supervised settings as in (Pang and Wu, 2021) and benefit from the better learned latent space. Our model is evaluated on AGNews (Zhang et al., 2015), a popular benchmark for text classification with 127,600 documents from 4

Model	200	500	2500	10000
Glove-ID	70.4	78.0	84.1	87.1
Glove-OD	68.8	78.8	85.3	88.0
VAMPIRE	82.9	84.5	85.8	87.7
Hard EM	83.9	84.6	85.1	86.9
CatVAE	84.6	85.7	86.3	87.5
SVEBM	84.5	84.7	86.0	88.1
SVEBM-IB	86.4	87.4	87.9	88.6
Ours	**87.4**	**88.1**	**89.2**	**90.1**

Table 2.9: Accuracy on AGNews

classes. Table 2.9 shows that LDEBM performs the best when having only partial access to ground-truth data labels; it further validates the proposed formation for learning a well-structured latent space.

2.5 Discussion and Related Work

Text Modeling VAE has been one of the most prominent latent variable models for generative modeling (Kingma and Welling, 2014a; Rezende et al., 2014). It is first applied to text modeling by Bowman et al. (2016), followed by a wide range of work attacking challenging text generation problems using the shared framework of VAE (Serban et al., 2016, 2017; Wen et al., 2017; Zhao et al., 2017, 2018b; Fang et al., 2019; Zhang et al., 2016; Li et al., 2017a; Gupta et al., 2018). In parallel, extensive efforts have been made to address issues like posterior collapse (Bowman et al., 2016; Higgins et al., 2016; Zhao et al., 2017, 2018a; He et al., 2018; Li et al., 2019a; Fu et al., 2019) and mode-collapse (Shi et al., 2020) in training VAE to further improve the language modeling performance and text generation quality.

The interpretability of the generation process is naturally brought up as the generation quality achieves impressive progress. Recently, Zhao et al. (2018b), Shi et al. (2020), and Pang and Wu (2021) have explored interpretable text generation with deliberately designed latent spaces. Zhao et al. (2018b) use a discrete latent space to capture dialog

actions; Shi et al. (2020) adopt a mixture of Gaussians as the VAE prior. To further improve the expressivity of latent space, Pang and Wu (2021) propose a symbol-vector coupling energy-based prior to learn a structured latent space. Our formulation inherits the advantages from (Pang and Wu, 2021) by choosing an appropriate symbol-vector coupling scheme and principally incorporating the IB. We further develop a geometric clustering-based regularization that complements the IB; it alleviates the mode-collapse problem in variational learning of the latent space model.

Energy-Based Model EBMs (Xie et al., 2016; Nijkamp et al., 2019, 2020a; Han et al., 2020) have drawn growing interest in generative modeling. As an interesting branch, Pang et al. (2020a) learn an EBM in the latent space as a prior model for continuous latent variables; it greatly improves the expressivity over non-informative priors and demonstrates strong performance on downstream tasks, *e.g.*, image segmentation, molecule generation, and trajectory prediction (Yu et al., 2021; Pang et al., 2020b, 2021; Jing et al., 2019). However, both EBM and latent space EBM require MCMC sampling to learn the model. The degenerate sampling quality in practice can lead to poor generation quality and instability in training (Grathwohl et al., 2019; Du et al., 2021). We leverage diffusion models as a cure for the vanilla latent space EBM in this chapter; the proposed model shows reliable sampling quality in practice.

Diffusion Model Diffusion models (Ho et al., 2020; Gao et al., 2020), introduced by Sohl-Dickstein et al. (2015), learn from a sequence of noise-perturbed versions of the data. From such perturbed data, one can learn the conditional model to invert the diffusion process and generate high-quality samples given noisy inputs. On another front, Song and Ermon (2019, 2020) and Song et al. (2020b) extend the denoising score matching method (Vincent, 2011), modeling the diffusion process with continuous time step. Our formulation moves the model to the latent space in a variational framework with two benefits: (a) learning in a lower-dimensional space enables faster sampling and better convergence, and (b) learning the diffusion model in a continuous latent space avoids the discreteness of text data, which hinders the direct application of vanilla diffusion models to text modeling (Austin et al., 2021).

Similar to our work, Wehenkel and Louppe (2021), Sinha et al. (2021), Nie et al. (2021), and Vahdat et al. (2021) have proposed to learn a diffusion model in the latent space. Specifically, Wehenkel and Louppe (2021) empirically demonstrate that a diffusion prior can perform better than the non-informative Gaussian prior when jointly trained with a VAE. Sinha et al. (2021) combine contrastive learning with diffusion models in the latent space of VAEs for controllable generation. Nie et al. (2021) and Vahdat et al. (2021) extend the idea of Song et al. (2020b) in the latent space: Nie et al. (2021) perform controllable image generation by training a latent energy-based attribute classifier on a pre-trained generator; Vahdat et al. (2021) train score-based denoising diffusion models in the latent space of a powerful VAE (Vahdat and Kautz, 2020). Both methods have achieved very impressive image generation results. However, the listed methods are generally limited to image generation with tailored or pre-trained encoders and decoders. In contrast, our method is a general improvement for the sampling quality of latent space EBM; it is not restricted to a certain data type. Moreover, the proposed model can be trained from scratch to form a well-structured latent space, in contrast to those of Vahdat et al. (2021) and Nie et al. (2021), which require a pre-learned latent space.

2.6 Summary

In this chapter, we presented the LDEBM, a latent variable generative model that couples symbol and vector representations in the latent space. As a novel symbiosis between EBM and a diffusion model, the LDEBM achieves reliable generation quality. More importantly, it learns from scratch a structured latent space that aligns categorically well with human understanding. It prototypes the duality between symbolic and vector representations for latent abstractions. Although the architecture of the generator in the LDEBM is relatively naive, the empirical success of the principled learning and inference methods for latent attraction inspires our later attempts in latent-variable autoregressive models, which we will introduce in Chapter 7.

CHAPTER 3

Emerging Iconic Symbols in a Visual Communication Game

3.1 Introduction

Building upon the prototypical latent-variable model for the symbol-vector duality in categorization introduced in the previous chapter, we now expand our investigation to address the limitations imposed by the naive one-hot representation of symbols. This chapter explores a more sophisticated approach inspired by pictorial sign systems: employing sequentially generated sketches as an alternative form for representing the same duality. The unique nature of these sketches allows us to examine how the principles of the Information Bottleneck (Tishby et al., 2000) can be flexibly incorporated into a sequence generative model over the latent variables. We also extend from latent-variable generative modeling into the general representation learning problem within an encoding-decoding architecture. Metaphorically, this setup can be viewed as a communication problem between a sender and a receiver.

The communication problem naturally arises when traveling in a foreign country where one does not speak the native language, which necessitates exploring non-linguistic means of communication, such as drawings. Due to its *iconic* nature (*i.e.*, perceptual resemblance to, or natural association with, the referent), drawings serve as a powerful tool to communicate concepts transcending language barriers (Fay et al., 2014). In fact, humans started to use drawings to convey messages dating back 40,000–60,000 years (Hoffmann et al., 2018; Hawkins et al., 2019). Some cognitive science studies hypothesize a transition from sketch-based communication before the formation of sign

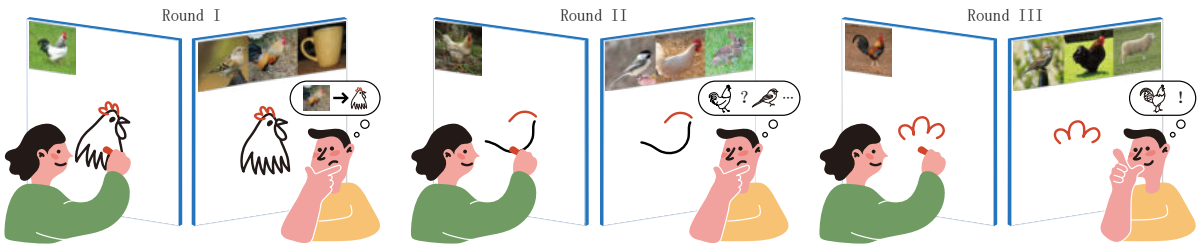


Figure 3.1: An example of the visual communication game. In an iterative sketch communication game (Hawkins et al., 2019), players first need to ground sketches to referents. The drawer (Alice) gradually simplifies the drawing but keeps the most salient parts of the target concept (rooster crown). This evolution process enables the viewer (Bob) to promptly distinguish the target (rooster) from distractors (bird, cup, rabbit, and sheep).

systems and provide evidence that iconic signs can gradually become *symbolic* through repeated communication (Fay et al., 2014). In contrast to *icons*, *symbols* are special forms bearing arbitrary relations to the referents. Figure 3.1 describes a typical scenario of such phenomena: Alice (in green) uses a sketch to communicate the concept "rooster" to Bob (in yellow). Initially, they need to ground the sketch to the referent. Later, details of the visual concept, such as strokes of the head and body, are gradually abstracted away, leaving only the most salient part, the crown. The iconicity in the communicated sketch drops while the symbolicity rises.

While models of emerging communication protocols have attracted attention (Lazaridou et al., 2017; Cao et al., 2018; Evtimova et al., 2018; Havrylov and Titov, 2017; Lazaridou et al., 2018; Lazaridou and Baroni, 2020; Mordatch and Abbeel, 2018; Ren et al., 2020; Eccles et al., 2019; Graesser et al., 2019), the initial and primary communication medium is presumed and limited to be symbolic rather than iconic. By simulating a multi-agent *referential game*, prior work seeks the *environmental driving forces* behind the emergence of effective communications. In a typical setup of referential games, two agents play similar roles as in the above Alice-Bob example but share a primitive set of arbitrary tokens (*i.e.*, the vocabulary). Using these tokens, an agent (the sender) attempts to communicate a message to another agent (the receiver). A communication convention emerges when two agents successfully communicate by associating visual concepts in the images with tokens in the pre-selected vocabulary. Though this line of work has probed

into certain linguistic properties of the established communication conventions (Lazaridou et al., 2017, 2018; Ren et al., 2020), some intriguing questions remain open: How do agents make a trade-off between iconicity and symbolicity to emerge iconic symbols?

In this chapter, we present the very first step of modeling and simulating the evolution process of *graphical conventions* (Hawkins et al., 2019), a two-participant communication convention whose medium is drawings in an abstract form. Specifically, our contributions are threefold:

First, we model a multi-agent visual communication game and propose a learning framework wherein the sender and the receiver evolve jointly. This visual communication game is an alternating sequential decision-making process, in which the sender generates a sequence of strokes step by step, terminated by the receiver. In contrast to discretized tokens in prior work, strokes can be continuously parametrized (Ha and Eck, 2018; Huang et al., 2019) such that the derivatives of learning objectives can be more effectively back-propagated through communication channels (Foerster et al., 2016). We further derive a novel training surrogate for multi-agent reinforcement learning based on a joint value function and the eligibility trace method (Sutton and Barto, 2018). This differs from the REINFORCE-based surrogate in prior work on early-terminable sequential communication (Cao et al., 2018; Evtimova et al., 2018) and the TD learning method in the literature of multi-agent turn-taking games, wherein each agent has its own value function and has to model the value of others (Wen et al., 2019). In experiments, we empirically demonstrate that our integration of function approximation and Monte Carlo sampling facilitates the agents' awareness of the correlation between complex and simple sketches, thereby enabling a smooth abstraction process.

Second, we define essential properties in studying evolved sketches. Specifically, we define *iconicity* (Fay et al., 2014) as the drawings exhibiting high visual resemblance to the corresponding images, such that they are proximal to the latter when measured on the high-level embedding of a general-purpose visual system; we define *symbolicity* (Fay et al., 2018) as these drawings being consistently separable in the high-level visual embedding, which facilitates new communication participants to easily distinguish between categories

without grounding them to referents; and we define *semanticity* (Harispe et al., 2015) as the topography of the high-level embedding space of the drawings being strongly correlated to that of images, such that semantically similar instances and categories lie close to each other in the embedding space.

Third, we present a suite of quantitative and qualitative methods to evaluate the emergent graphical conventions based on the above carefully defined *iconicity*, *symbolicity*, and *semanticity*. This is necessary because a high communication rate does not imply good representations (Bouchacourt and Baroni, 2018). The graphical nature of the communication medium mandates us to repurpose representation learning metrics rather than adopt linguistic metrics in emergent symbolic communication. We evaluate the contribution of different environmental drivers, early decision, sender's update, and sequential communication, to the three properties of the emergent conventions. Critically, the empirical results assessed on our metrics align well with our prediction based on the findings of human graphical conventions (Fay et al., 2010; Hawkins et al., 2019), which justifies our environment, model, and evaluation. One of these setups emerges conventions where the three properties are consistent with our expectation of a sign system. Particularly, we find two inspiring phenomena: (i) Evolved sketches from semantically similar classes are perceptually more similar to each other than those falling into different superclasses. (ii) To communicate concepts not included in their established conventions, evolved agents can return to more iconic communications than humans. These phenomena reveal the potential of more sophisticated probabilistic models over latent abstractions of categories.

3.2 Related Work

Learning to Sketch Ha and Eck (2018) begin the endeavor of teaching modern neural models to sketch stroke by stroke. However, generating meaningful stroke sequences directly from various categories of natural images is still in the early phase (Song et al., 2018; Wang et al., 2021; Zou et al., 2018). To assure the interestingness of

the category-level sketch communication, we design a stage-wise agent that transfers a natural image into a pixel-level sketch (Kampelmuhler and Pinz, 2020) and draws the sketch on the canvas stroke by stroke with a policy (Huang et al., 2019). To pre-train our neural agents to sketch, we select Sketchy (Sangkloy et al., 2016) from many datasets (Yu et al., 2016; Ha and Eck, 2018; Eitz et al., 2012; Sangkloy et al., 2016) for its fine-grained photo-sketch correspondence, rich stroke-level annotations, and well-organized categorization structures.

Communication Games While learning to sketch is formulated as a single-agent task with explicit supervision, our focus is on how sketches would evolve when utilized as the communication medium between *two cooperative agents*. Their cooperation is always formulated as a communication game, recently adopted to study phenomena in human-robot teaming (Yuan et al., 2022) and natural languages, such as symbolic language acquisition (Graesser et al., 2019) and the emergence of compositionality (Ren et al., 2020). Some notable works (Lazaridou et al., 2017, 2018; Evtimova et al., 2018; Havrylov and Titov, 2017) have devised interesting metrics, such as *purity* (Lazaridou et al., 2017) and *topographic similarity* (Lazaridou et al., 2018). In comparison, our work is unique due to the distinctive communication medium, continuously parametrized sketches. Although a concurrent work (Mihai and Hare, 2021) also enables the agents to sketch in a communication game, it focuses primarily on drawing interpretable sketches without abstracting them into graphical symbols along the communication. We position our work as an alternative to emergent symbolic communication, since the emergent graphical symbols may better represent the continuum of semantics, as encoded in the vector representation of tokens (Mikolov et al., 2013a). Methodologically, we devise new evaluation metrics for sketch modality, assessing *iconicity*, *symbolicity*, and *semanticity* in the evolved sketches.

Emergent Graphical Conventions Evolving from sketches to graphical conventions / symbols is an active field in cognitive science under the banner of "emergent sign systems." Fay et al. (2010) show that pair interaction can form their local conventions when they play the Pictionary game. Using natural images instead of texts as the prompt, Hawkins et al. (2019) show that visual properties of the images also influence

the formed graphical conventions besides partners' shared interaction history; *i.e.*, the evolved sketches highlight visually salient parts. Nevertheless, only a few computational models exist apart from these human behavior studies. Fan et al. (2020) describe a model for selecting complex or simple sketches considering the communication context. Bhunia et al. (2020) and Muhammad et al. (2018) consider stroke selection and reordering to simplify the sketch. In contrast to sketch or stroke selection, we model embodied agents who can draw and recognize sketches.

3.3 The Visual Communication Game

Our visual communication game is formulated as a tuple

$$(\mathcal{I}, \mathcal{C}, \mathcal{A}_S, \mathcal{A}_R, G, r, \gamma),$$

where $\mathcal{I}$ is the image set to be presented to the sender S and the receiver R. These images contain a single object in the foreground, and hence the image space $\mathcal{I}$ can be partitioned into N classes according to the object categories. In each game round, the sender is presented with one image $\mathbf{I}_S$, and the receiver is presented with M images $\{\mathbf{I}_R^1, \ldots, \mathbf{I}_R^M\}$. Prior work shows that category-level context can force the agent to coordinate on more abstract properties and help draw sketches to be more recognizable as the type of object (Lazaridou et al., 2017; Mihai and Hare, 2021). We make the observations of S and R disjoint (*i.e.*, $\mathbf{I}_S \notin \{\mathbf{I}_R^1, \ldots, \mathbf{I}_R^M\}$) but with a target image $\mathbf{I}_R^*$ in the same class as $\mathbf{I}_S$. We refer to the M images that the receiver can see as the *context*. Neither the receiver or the sender would see the image(s) presented to their partner; they can only communicate this information by drawing sketches on the canvas space $\mathcal{C}$, observable to both players. As shown in Figure 3.2, $\mathbf{C}_0$ is initialized to be blank at the beginning of each round. Only the sender can draw on the canvas with actions chosen from $\mathcal{A}_S$. The action at each time step consists of 5 strokes, which are continuous vectors in $\mathbb{R}^6$. We constrain each dimension to be in $(0, 1)$ due to the limited space of the canvas. The canvas is updated from $\mathbf{C}_t$ to $\mathbf{C}_{t+1}$ by the renderer G after each step of the sender's sketching. The

receiver, after observing the updated canvas, would choose among the M images or wait for the next step from the sender; these $M+1$ possible actions constitute $\mathcal{A}_R$. A game round terminates when the receiver gives up waiting and chooses one from the images. After the termination, the sender and the receiver will receive a shared reward or penalty, depending on if the receiver makes the right choice:

$$r : \mathcal{I} \times \mathcal{I} \to \{-1, 1\}.$$

This reward/penalty is temporally decayed with a decay factor γ. That is, if the receiver decides to choose from the images at step t, this cooperating pair will receive either γ^t or $-\gamma^t$. Hence, even though the players do not receive an explicit penalty for long conversations, there is an implicit penalty/reward for delayed positive/negative answers. No reward will be assigned if the receiver chooses to wait. The next round starts after the reward/penalty assignment.

3.4 Agents

The two agents involved in the visual communication game are modeled with two decision policies, π_S and π_R, for the sender and the receiver, respectively. These policies are stochastic mapping from the agents' observation space to the action space:

$$\pi_S : \mathcal{I} \times \mathcal{C} \to \mathcal{P}(\mathcal{A}_S), \quad \pi_R : \mathcal{I}^M \times \mathcal{C} \to \mathcal{P}(\mathcal{A}_R), \tag{3.1}$$

where $\mathcal{P}(\mathcal{A})$ is a distribution over the support set $\mathcal{A}$. As shown in Figure 3.2, at each time step $t \in \{0, \ldots, T\}$, the sender first emits the stroke parameters for the next five strokes $\mathbf{a}_{St} \sim \pi_S(\mathbf{I}_S, \mathbf{C}_t)$. These strokes are applied to the canvas by a differentiable renderer, $\mathbf{C}_{t+1} = G(\mathbf{C}_t, \mathbf{a}_{St})$. Next, the updated canvas is transmitted to the receiver. The receiver decides whether to terminate the game by making its prediction (*i.e.*, $\mathbf{a}_{Rt} \in \{1, \ldots, M\}$) or wait for another round (*i.e.*, $\mathbf{a}_{Rt} = M + 1$); its decision is sampled from π_R. If a prediction is made, it is used to select the image $I_R^{\mathbf{a}_R}$ from $\{\mathbf{I}_R^1, \ldots, \mathbf{I}_R^M\}$ and score this

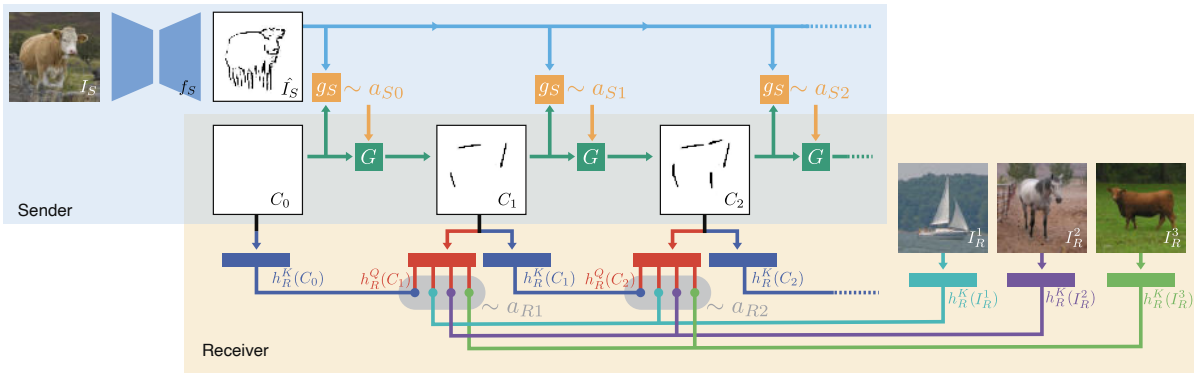


Figure 3.2: Communication process. In our visual communication game, a sender S and a receiver R only share the observation of the canvas $\mathbf{C}$. The sender converts the natural image $\mathbf{I}_S$ to a pixel-level sketch $\hat{\mathbf{I}}_S$. At each step, the sender first draws five strokes $\mathbf{a}_S$ through the renderer G, which updates the canvas to $\mathbf{C}_{t+1}$. Next, the receiver uses the updated canvas $\mathbf{C}_{t+1}$ to query from the context images $\{\mathbf{I}_R^1, \ldots, \mathbf{I}_R^M\}$ and the last canvas $\mathbf{C}_t$, deciding the action $\mathbf{a}_R$ at this step. The game continues if the receiver chooses to wait.

game with $r(\mathbf{I}_S, \mathbf{I}_R^{\mathbf{a}_R})$. Otherwise, this routine repeats in the next step $t \leftarrow t+1$.

3.4.1 Sender

Prior to playing the visual communication game, the sender should be able to (i) extract edges from natural images (Xie and Tu, 2015) and (ii) draw sketches that closely resemble the configurations of salient edges (Li et al., 2019b), just as humans do (Sayim and Cavanagh, 2011). To endow the sender with these capabilities, we design a stage-wise architecture $h_S = g_S \circ f_S$. Specifically, $\mathbf{I}_S$ is first converted to a target sketch $\hat{\mathbf{I}}_S$ using a visual module f_S (Kampelmuhler and Pinz, 2020), capturing the salient edge information in the natural image; we directly adopt the pre-trained model from the referred work. Next, $\hat{\mathbf{I}}_S$ is concatenated with the current canvas $\mathbf{C}_t$ and fed to the sketching module g_S, whose architecture is built upon the one by Huang et al. (2019). This sketching module outputs five vectors in the form $(x_0, y_0, x_1, y_1, x_2, y_2)$, which parametrizes the curve of one stroke. The policy is parametrized as a Gaussian distribution during training:

$$\pi_S = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}^2), \quad \boldsymbol{\mu}_t = h_S(\mathbf{I}_S, \mathbf{C}_t), \quad \boldsymbol{\sigma}^2 = c \cdot \mathbf{I}, \tag{3.2}$$

where $\mathbf{I}$ is the identity matrix, and c is a constant hyperparameter. During the testing, we set $c = 0$.

These stroke parameters $\mathbf{a}_{St}$ are fed into a pre-trained renderer G (Huang et al., 2019) to update the canvas, $\mathbf{C}_{t+1} = G(\mathbf{C}_t, \mathbf{a}_{St})$. This renderer is fully differentiable, enabling end-to-end model-based training (Hafner et al., 2019) of the sketching module g_S. We pre-train g_S on Sketchy (Sangkloy et al., 2016); refer to the supplementary video for the results.

3.4.2 Receiver

The receiver, similar to the sender, should also carry some rudimentary visual capability to this game. Unlike the low-level vision needed for the sender, the requirement for the receiver is high-level visual recognition. Therefore, we adopt a pre-trained VGG16 (Simonyan and Zisserman, 2015) as the visual module $f_R : \mathcal{I} \to \mathbb{R}^{4096}$ of the receiver, following a similar practice in recent literature (Lazaridou et al., 2017; Havrylov and Titov, 2017). The output of this visual module is a vector, and further transformed by two separate linear layers, g_R^K and g_R^Q, into visual embeddings, $h_R^K(\mathbf{I})$ and $h_R^Q(\mathbf{I})$. That is, $h_R^K = g_R^K \circ f_R$, $h_R^Q = g_R^Q \circ f_R$.

When observing both the context $\{\mathbf{I}_R^1, \ldots, \mathbf{I}_R^M\}$ and the canvas $\mathbf{C}_t$, the receiver first embeds each of them with h_R. Next, it makes the decision based on the similarity between the current canvas and each option in the context. The decision module is thus realized by a Boltzmann distribution based on resemblance:

$$\pi_R(\mathbf{a}_{Rt}|\mathbf{I}_R^1, \ldots, \mathbf{I}_R^M, \mathbf{C}_{t-1}, \mathbf{C}_t) = \frac{\exp(h_R^Q(\mathbf{C}_t) \cdot h_R^K(\mathbf{I}_R^{\mathbf{a}_{Rt}}))}{\sum_{m=1}^{M+1} \exp(h_R^Q(\mathbf{C}_t) \cdot h_R^K(\mathbf{I}_R^m))}, \tag{3.3}$$

where $\mathbf{I}_R^{M+1} = \mathbf{C}_{t-1}$. Of note, although a similar policy was proposed before (Lazaridou et al., 2018; Havrylov and Titov, 2017), our π_R is distinct as it is endowed with an external memory of $\mathbf{C}_{t-1}$. Intuitively, if the receiver finds the current canvas $\mathbf{C}_t$ closer to the last canvas $\mathbf{C}_{t-1}$ in the embedding space than all M options in the context, it will choose to

emit $\mathbf{a}_{Rt} = M+1$ and delay the decision to the next step; a prediction can only be made when the receiver finds the current canvas is informative enough. As a result, the sender would draw informative strokes as early as possible to avoid the implicit penalty in the decayed reward.

3.5 Learning to Communicate

Policies of the sender and the receiver are trained jointly to maximize the objective

$$\pi_S^*, \pi_R^* = \arg\max_{\pi_S, \pi_R} \mathbb{E}_{\tau \sim (\pi_S, \pi_R)} \left[\sum\nolimits_t \gamma^t r_t \right], \tag{3.4}$$

where $\tau = \{\mathbf{C}_0, \mathbf{a}_{S0}, \mathbf{C}_1, \mathbf{a}_{R1}, \mathbf{a}_{S1}, \dots\}$ is the simulated episodic trajectory. As well known in reinforcement learning, the analytical expectation in (3.4) is intractable to calculate along the trajectory τ. We devise value functions $\mathcal{V}(\mathbf{X}_t)$ and $V_\lambda(\mathbf{X}_t)$ for an optimization surrogate:

$$\mathcal{V}(\mathbf{X}_t) = \mathbb{E}_{\pi_S(\mathbf{a}_{St}|\mathbf{I}_S, \mathbf{C}_{t-1}), \pi_R(\mathbf{a}_{Rt}|\hat{X}_t)} \left[(r_t + \gamma \delta(\mathbf{a}_{Rt}) V_\lambda(\mathbf{X}_{t+1}) \right], \tag{3.5}$$

where $\delta(\mathbf{a}_{Rt})$ is the Dirac delta function that returns 1 when the action is *wait* and 0 otherwise. $\mathbf{X}_t = \text{cat}([\mathbf{I}_S], \hat{X}_t)$, where $\hat{\mathbf{X}}_t = [\mathbf{I}_R^1, \dots, \mathbf{I}_R^M, \mathbf{C}_{t-1}, \mathbf{C}_t]$. The expectation $\mathbb{E}_{\pi_S(\mathbf{a}_{St}|\mathbf{I}_S, \mathbf{C}_{t-1})}[\cdot]$ is approximated with the point estimate, as in the reparametrization in VAE (Kingma and Welling, 2014b). The expectation $\mathbb{E}_{\pi_R(\mathbf{a}_{Rt}|\hat{\mathbf{X}}_t)}[\cdot]$ can be calculated analytically because π_{Rt} is a categorical distribution. The connection between $\mathbb{E}_{\pi_S}$ and $\mathbb{E}_{\pi_R}$ is one of our contributions. Specifically, $\mathbf{C}_t$ in $r\hat{m}X_t$ in $\pi_{Rt}(\mathbf{a}_{Rt}|\hat{\mathbf{X}}_t)$ is generated by the differentiable renderer G with the actions $\mathbf{a}_{St}$ from the sender policy $\pi_S(\mathbf{a}_{St}|\mathbf{I}_S, \mathbf{C}_{t-1})$. Hence, we can have both $\partial\mathcal{V}/\partial\pi_{Rt}$ and $\partial\mathcal{V}/\partial\pi_{St}$ based on (3.5). This results in a novel multi-agent variant of the general policy gradient (Sutton et al., 2000; Silver et al., 2014).

$V_\lambda(\mathbf{X}_t)$ in (3.5) is an eligibility trace approximation (Sutton and Barto, 2018) of the ground-truth value function. Intuitively, a value estimate with eligibility trace V_λ mixes the bootstrapped Monte Carlo estimate $V_N^k(\mathbf{X}_t) = \mathbb{E}_{\pi_S, \pi_R}[\sum_{n=t}^{h-1} \gamma^{n-t} r_n + \gamma^{h-t} \delta(\mathbf{a}_{Rh}) \upsilon_{\boldsymbol{\phi}}(\mathbf{X}_h)]$ at different roll-out lengths k, with $h = min(t+k, T_{choice})$ being the maximal timestep.

Algorithm 3 Training Algorithm

Input: Initialize neural network parameters $\boldsymbol{\theta}$, ρ, $\boldsymbol{\phi}$ for π_S, π_R, and $v_{\boldsymbol{\phi}}$, respectively.
for game round $l = 1, \ldots, L$ **do**
  for time step $t = 0, \ldots, T$ **do**
    $\mathbf{a}_{St} \sim \pi_S(\mathbf{a}_{St}|\mathbf{C}_t, \mathbf{I}_S)$, $\mathbf{C}_{t+1} = G(\mathbf{C}_t, \mathbf{a}_{St})$, $\mathbf{a}_{Rt+1} \sim \pi_R(\mathbf{a}_{Rt+1}|\mathbf{C}_t, \mathbf{C}_{t+1}, \mathbf{I}_R^1, \ldots, \mathbf{I}_R^M)$
    if $\mathbf{a}_{Rt+1}$ is not wait **then**
      $T_{\text{choice}} = t$
    end if
  end for
  Compute $\{V_\lambda(\mathbf{X}_t)\}_{t=1}^T$ via (3.6)
  Compute $\{\mathcal{V}(\mathbf{X}_t)\}_{t=1}^T$ via (3.5)
  Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\alpha}_S \nabla_{\boldsymbol{\theta}} \sum_t \mathcal{V}(\mathbf{X}_t)$
  Update $\rho \leftarrow \rho + \boldsymbol{\alpha}_R \nabla_\rho \sum_t \mathcal{V}(\mathbf{X}_t)$
  Update $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \boldsymbol{\alpha}_v \nabla_{\boldsymbol{\phi}} \sum_t \frac{1}{2} ||v_{\boldsymbol{\phi}}(\mathbf{X}_t) - V_\lambda(\mathbf{X}_t)||^2$
end for

T_{choice} is the timestamp when the receiver stops waiting. The articulation of such termination also makes our eligibility trace deviate from the general derivation with infinite horizon. We derive an episodic version as

$$V_\lambda(\mathbf{X}_t) = \begin{cases} (1-\lambda) \sum_{n=1}^{H-1} \lambda^{n-1} V_N^n(\mathbf{X}_t) + \lambda^{H-1} V_N^H(\mathbf{X}_t) & \text{if } t \leq T_{choice} \\ v_{\boldsymbol{\phi}}(\mathbf{X}_t) & \text{otherwise,} \end{cases} \tag{3.6}$$

where $H = T_{choice} - t + 1$. Refer to Section B.3 for a detailed derivation. Finally, $v_{\boldsymbol{\phi}}(\mathbf{X}_t)$ is trained by regressing the value returns:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}} \mathbb{E}_{\pi_S, \pi_R} \left[\sum\nolimits_t \frac{1}{2} \left\| v_{\boldsymbol{\phi}}(\mathbf{X}_t) - V_\lambda(\mathbf{X}_t) \right\|^2 \right]. \tag{3.7}$$

Algorithm 3 summarizes the training algorithm.

3.6 Experiments

3.6.1 Settings

Images We used the Sketchy dataset (Sangkloy et al., 2016) as the image source. Due to the limited performance of the sketching module on open-domain image-to-sketch

Game Settings					Communication Accuracy (%) ± SD (avg. step)		
early decision	update sender	max/one step	description	setting names	seen	unseen instance	unseen class
yes	yes	max	our experimental setting	complete	98.07 ± 0.01 (1.03)	70.37 ± 0.04 (2.36)	39.40 ± 0.05 (3.76)
no	yes	max	control setting for early decision	max-step	86.27 ± 0.03 (7.00)	67.93 ± 0.02 (7.00)	38.40 ± 0.04 (7.00)
yes	no	max	control setting for evolving sender	sender-fixed	99.60 ± 0.01 (2.41)	71.80 ± 0.02 (3.83)	45.40 ± 0.02 (4.75)
yes	yes	one	control setting for sequential game	one-step	22.87 ± 0.23 (1.00)	14.07 ± 0.15 (1.00)	9.60 ± 0.09 (1.00)
no	no	max	baseline for all settings above	retrieve	99.47 ± 0.01 (7.00)	76.80 ± 0.02 (7.00)	48.00 ± 0.02 (7.00)

Table 3.1: Game settings and results. The first three columns represent the configurations of the environmental drivers. Setting names and descriptions specify our purposes for intervention. The last three columns show success rates and conversation length in testing games. Games marked with "seen" are validation games with the same image set as training. Games marked with "unseen" are testing games with unseen images.

sequential generation, we select 40 categories (10 images per category, see Section B.1) with satisfactory sketching behaviors.

Environmental Drivers With the visual communication game and the learning agents at hand, we investigate the factors in emergent graphical conventions with controlled experiments. Table 3.1 lists all designed settings. Specifically, we consider the following:

- *Can receiver make early decisions?* The hypothesis is that the receiver's decision before exhausting the allowed communication steps may inform the sender of the marginal benefit of each stroke and incentivize it to prioritize the most informative strokes. The corresponding control setting is *max-step*, wherein the receiver can only make the choice after the sender finishes its drawing at the maximum step. This factor is on in other settings.

- *Can the sender change its way of drawing?* The hypothesis is that the mutual adaptation of the sender and the receiver may lead to better abstraction in the evolved sketches. Particularly, the sender may develop new ways of drawing in the evolution. The corresponding control setting is *sender-fixed*, wherein we freeze the parameters of the sender such that the receiver has to adapt to its partner. This factor is on in other settings.

- *Is the game sequential, and can the receiver observe more complex drawings?* The hypothesis is that the modeling of a sequential decision-making game and the evolu-

tion from more complex sketches may regularize the arbitrariness, which is unique for graphical conventions. The corresponding control setting is *one-step*: There only exists one step of sketching, thus no sequential decision-making at all. This factor is on in other settings.

Training, Validation, and Generalization We train the sender/receiver with batched forward and back-propagation, with a batch size of 64 and maximum roll-out step $T = 7$. We update using Adam (Kingma and Ba, 2015) with the learning rate 0.0001 for a total of 30k iterations. We train each model on a single Nvidia RTX A6000; one experiment takes 20 hours. Except for the *sender-fixed* setting, there is a warm-up phase in the first 2000 iterations for the sender where we linearly increase the learning rate to 0.0001. After the warm-up phase, the learning rate of both agents will be decreased exponentially by $0.99^{\frac{i-2000}{1000}}$, where i is the number of training iterations. In all settings, we set $M = 3$, $\gamma = 0.85$. Between iterations, we randomly sample another 10 batches for validation. We use 30 categories (10 images per category) for training and held out 10 images per category for the unseen-instance test; another 10 categories are for the unseen-class test. Each image is communicated as the target, resulting in 300+100 pairs in the test set. At each iteration, the categories and instances are sampled randomly to constitute the context. Results henceforth are statistics of 5 random seeds.

3.6.2 Results

3.6.2.1 Communication Efficacy and Sketch Abstraction

We record both the success rate and the communication steps over the training iterations. The communication is considered successful when the receiver makes the correct prediction at the end of the game. In Figure 3.3, agents in all settings except *one-step* converge to a success rate above 80%. Among them, the communicating pairs in the *complete* setting and the *sender-fixed* setting evolve to achieve a comparable success rate with the *retrieve* baseline. Interestingly, these two pairs also emerge a phenomenon resembling the natural observation in human studies, named *systematic reduction* (Lewis, 1969): The

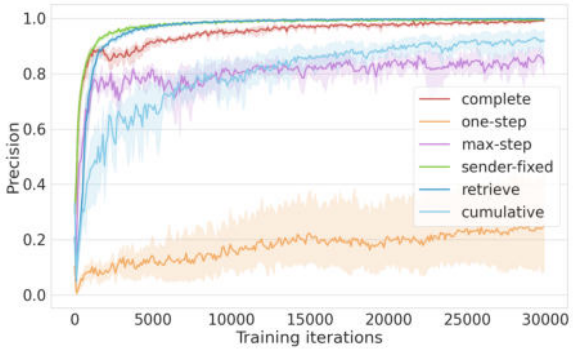


Figure 3.3: The validation accuracy of different game settings and the ablation baseline.

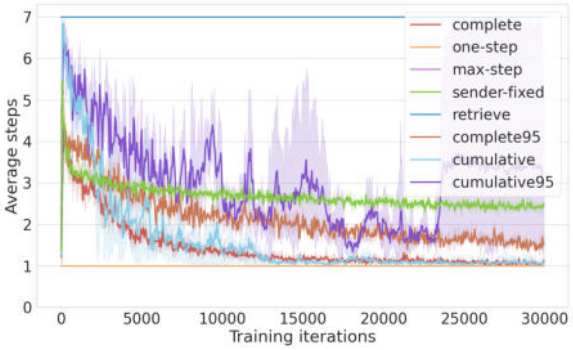


Figure 3.4: The average communication steps under different settings and ablation baselines. γ is 0.85 by default, and 0.95 in complete95 and cumulative95.

average steps first increase and then gradually decrease as in Figure 3.4. Contrasting *complete* and *sender-fixed*, we can see: (i) The emergent conventions in the former is much simpler than the latter (less steps in Figure 3.4), which implies the contribution of mutual adaptation in sketch abstraction. (ii) The success rate in Figure 3.3 in the former converges a bit more slowly, which is reasonable since the senders can explore and change the way of drawing. In comparison, if the receiver cannot make early decisions, it has no intention to relate sketches (*i.e.*, $\mathbf{C}_{t-1}$ and $\mathbf{C}_t$) at consecutive timesteps. The sender is thus *unaware* of each stroke's marginal information gain, which in return makes their learning harder. This might explain the relatively low success rate in the *max-step* setting. The failure of the *one-step* pairs reveals the irreplaceable roles of sequential decision-making and observing complex sketches in emergent graphical communication.

To further inspect how our proposed modeling and learning on sequential decision-

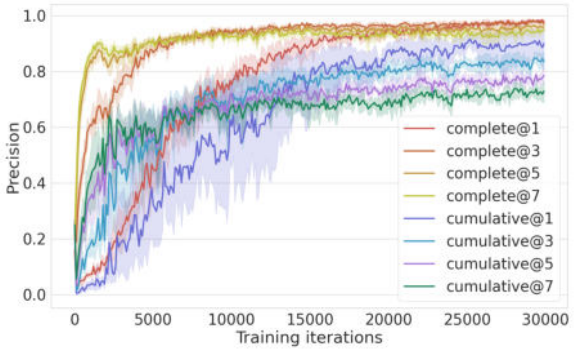


Figure 3.5: The prediction accuracy when receivers are presented with sketches drawn by corresponding senders at 1, 3, 5, and 7 steps, respectively.

making facilitate the desired evolution in the sketches, we conduct an ablation study by comparing our proposed learning surrogate (3.5) and a vanilla policy gradient baseline, REINFORCE (Williams, 1992) with Monte Carlo cumulative rewards

$$\mathbb{E}_{\pi_S,\pi_R}\left[\sum_t\left[\nabla \log \pi_R(\mathbf{a}_{Rt}|\hat{\mathbf{X}}_t)\sum_{n=t}^{T}\gamma^{n-t}r_n\right]\right].$$

Our comparison spans three axes. First, the REINFORCE baseline converges much more slowly than the proposed surrogate; see *cumulative* vs *complete* in Figure 3.3. Second, we check the robustness under the variation of decay factor γ. As shown in Figure 3.4, while the proposed method shows stable convergence in the communication steps under $\gamma = 0.85$ and 0.95, the REINFORCE baseline exhibits high-variance behavior under certain setup ($\gamma = 0.95$). Third, we check if agents' early terminations are *caused* by their *awareness* of the indistinguishable performance in longer and shorter episodes. Given a *precondition* that the longer the episodes are, the earlier the success rate increases, it should be the increase in the average performance of shorter episodes that *causes* the average timesteps to decrease. Taking 1-step and 3-step communication for example, in the *complete* setting, we shall see the success rate of the 3-step to achieve high earlier, which is then caught up but not exceeded by the 1-step. The not exceeding condition is a crucial cue to validate that the communicating partners were *actively* pursuing the Pareto front (Kim and De Weck, 2005) / efficiency bound (Zaslavsky et al., 2018) of

accuracy and complexity. This is exactly what our proposed method emerges as shown in Figure 3.5. In contrast, in the REINFORCE baseline, under the same decay factor, the performance of 1-step surpasses 3-step communication. It seems as if the incapability of *learning* long episodes *caused* the agents to *avoid* them.

Together, all our results on success rate and communication steps are consistent with predictions based on our hypotheses, which justifies our environments and models. However, a high success rate does not necessarily imply high convention quality. Another essential property for conventions is *stability* (Lewis, 1969): There should exist common patterns in the repeated usage of conventions to convey similar concepts. We take the viewpoint of representation learning and concretize the vague *stability* with three formally defined properties: *iconicity*, *symbolicity*, and *semanticity*. Below, we introduce our experiments to measure these properties systematically.

3.6.2.2 Iconicity: Generalizing to Unseen Images

We define *iconicity* as the drawings exhibiting a high visual resemblance with the corresponding images, such that they are proximal to the latter when measured on the high-level embedding of a general-purpose visual system. To quantitatively measure the visual similarities, we need to compute the distance between the sketch and image in the visual embedding space (*e.g.*, with cosine similarity). However, since we do not know how the embeddings distribute in their space, this unnormalized measure is prone to model-specific biases. Fortunately, the receiver's policy takes the form of a softmax of the cosine similarity between the embedding of the sketch and a context set with a target image and some randomly sampled distractors, which naturally approximates the normalization we want. Therefore, communication accuracy can serve as a visual similarity measure, and its *generalizability* to unseen images can pinpoint the emergent preservation of *iconicity*. Intuitively, in open-domain communication, agents would see novel scenes with known or unknown concepts —unseen instances of seen classes and unseen classes, respectively. They should still be able to communicate with established conven-

tions or with unconventionalized iconic drawings. A successful generalization to unseen classes (*i.e.*, zero-shot generalization) is more difficult than unseen instances; partners may increase the conversation length and communicate with more complex sketches.

Table 3.1 reports the success rates and average timesteps in our generalization tests. The *retrieve* setting is the baseline since there is no evolution at all and the sketches should resemble the original images the most (*i.e.*, possessing the highest *iconicity*). Unsurprisingly, its generalization performance upper-bounds all other settings. Among the experimental and controlled settings, the *complete*, the *max-step*, and the *sender-fixed* agents generalize relatively well in unseen instances (70.37±0.04, 67.93±0.02, 71.80±0.02) and generalize above chance ($39.40 \pm 0.05, 38.40 \pm 0.04, 45.40 \pm 0.02 > 25.00$) in unseen classes. Interestingly, *complete* and *sender-fixed* communicating partners intelligently turn to longer episodes for better generalization, better than the *max-step* agents. This finding implies the partners may turn to more iconic communication when there is no established conventions/symbols, just as we humans do. Strikingly, the *max-step* conventions seem to loose more *iconicity*, possibly due to confusion on marginal information gains. The *one-step* drawings seem to lack *iconicity*.

3.6.2.3 Symbolicity: Separating Evolved Sketches

Next, we measure *symbolicity* to evaluate the graphical conventionalization. We define *symbolicity* as the drawings being consistently separable in the high-level visual embedding, which facilitates new communication partners to easily distinguish between categories without grounding them to referents. Based on this definition, a more *symbolic* drawing should be more *easily separable* into their corresponding categories. To measure such *separability*, we use a pre-trained VGG16 as the new learner and finetune the last fully connected layer to classify evolved sketches into the 30 categories. Technically, we first get the 300 final canvases from the communication game, 10 for each category. Among them, we use 70% for training and 30% for testing.

The bar plot in Figure 3.6 shows the classification results. Since agents in the *one-*

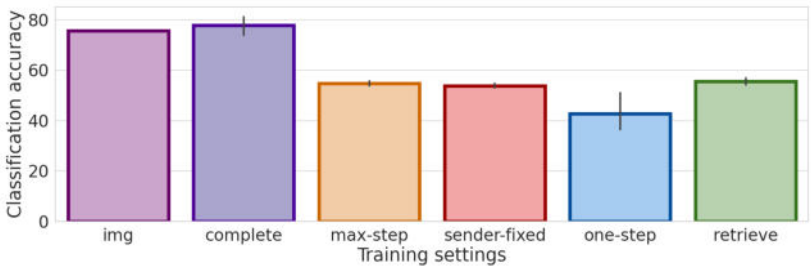


Figure 3.6: Testing results of classifiers trained with sketches evolved under different settings. *img* denotes images, and *retrieve* denotes unevolved sketches.

step setting do not play the game successfully, they may not form a consistent way to communicate. Agents in the *complete* setting achieve the highest accuracy, higher even compared with the result of the original images. This finding indicates that agents in the *complete* setting agree on a graphical convention that consistently highlights some features across all training instances in each category, which are also distinguishable between categories. Comparing the *max-step* with the *complete* setting, we know that early decision is a crucial factor for more *symbolic* conventions. Comparison between the *sender-fixed* setting and the *complete* setting suggests that the sender's evolution also contributes to high *symbolicity*.

3.6.2.4 Semanticity: Correlating Category Embedding

The last desired property is that the evolved sketches can preserve the perceptual metric in images (Zhang et al., 2018). We define *semanticity* as the topography of the high-level visual embedding space of drawings being strongly correlated to that of images, such that semantically similar instances and categories lie close to each other in the embedding space. To examine such *topographic correlation*, we project category names to the feature space using word2vec (Mikolov et al., 2013a) as featureA, and project the evolved sketches to the feature space using the trained VGG in Section 3.6.2.3 as featureB. We compute the correlation of the distances between all the possible pairs of featureB and the corresponding pairs of featureA as the semanticity measure. The results in Table 3.2 show that semanticity can be better retained in the *complete* setting compared with the

setting	correlation
complete	0.43 ± 0.02
max-step	0.41 ± 0.13
sender-fixed	0.36 ± 0.06
one-step	0.31 ± 0.08
retrieve	0.55 ± 0.00

Table 3.2: Semantic correlation between the word vector space of the communicated target and the feature vector space of the trained VGG.

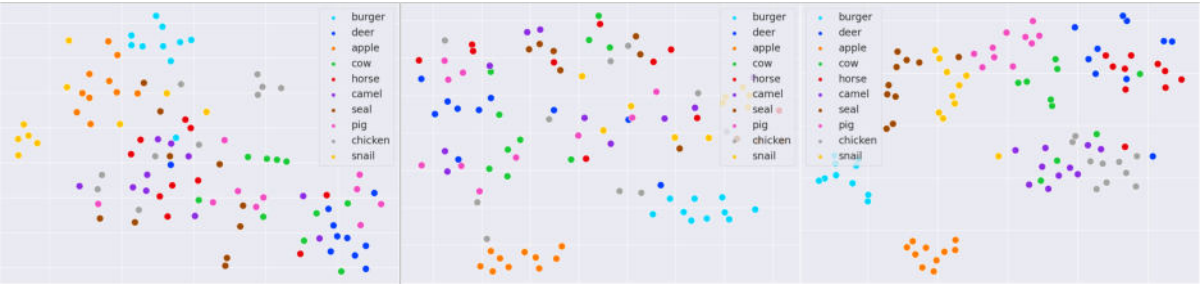


Figure 3.7: t-SNE of visual embedding of the original images (left), unevolved sketches in the *retrieve* setting (middle), and evolved sketches in the *complete* setting (right).

retrieve baseline. We further visualize the embeddings of the sketches in a 2D space via t-SNE (Van der Maaten and Hinton, 2008). Figure 3.7 show the visualization of the original images and the sketches in the *retrieve* and *complete* settings; refer to Section B.2 for results of other settings. Images/drawings from the same category are marked in the same color. As shown, boundaries between categories are clearer in the evolved drawings in the *complete* setting than in *retrieve* or original images; but semantically similar categories are still close to each other. For example, cow, deer, horse, and camel are proximal to each other, while burger and apple are far from them. These results highlight another uniqueness of visual communication over its symbolic counterpart: The similarity in the visual cues in the conventions may hint the *semantic* correlation between the referents.

3.6.2.5 Visualizing Sketch Evolution

To better understand the nature of the emerged conventions, we inspect the intermediate sketches in the evolution processes. Specifically, we visualize the process under the

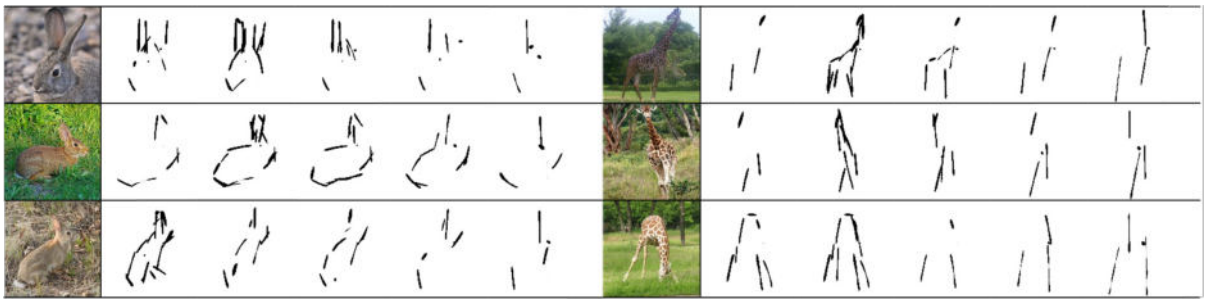


Figure 3.8: Sketch evolution of rabbit and giraffe.

complete setting. Figure 3.8 shows three instances in two categories. For each example, drawings from the left to the right show the change of the final-step canvas. Sketches' complexity gradually decreases after an initial increase, echoing the trend of reduction described in Section 3.6.2.1. For rabbits, in the beginning, the strokes may depict instances from different perspectives; through iterations, they converge to highlight the rabbit's long ear. As for the giraffe, the agents gradually learn to emphasize the long neck. In the third example, although the giraffe lowers its head, we can still see an exaggerated vertical stroke for the neck, similar to the first example where the giraffe's head is raised. These examples show how the sender gradually unifies drawings of the same category: After evolution, the sender is inclined to use the first several strokes to depict the most salient parts of visual concepts.

3.7 Limitation

The key limitation of this work is the two-stage pre-trained senders, for an ideal sender would not need to convert the images to pixel-level sketches before it starts sketching. Additionally, the learned sketches may align better with human understanding with an improved pre-trained sketching module.

3.8 Summary

In this chapter, we studied a sequence generative model over sketches as a latent representation for categories. This generative model was presented as a visual communication game whose repetition drives the evolution of graphical conventions and achieves representation learning. The sender and the receiver of this game are the encoder and the decoder of the learned representations. The learning objective of the encoder results in a sequence distribution over latent variables. We defined three properties, *iconicity*, *symbolicty*, and *semanticity* to measure the abstract structures of categories. Our controlled ablation studies justify the necessity of the components and other design choices. Notably, during out-of-distribution generalization, this model also manifests an understanding of uncertainty with the variation in the sequence lengths.

As a closing remark for Part I, the models we introduced for *Category* in these two chapters demonstrate how to employ different methods in generative modeling to learn latent abstractions whose representations are interpretable as symbols, interpolatable as vectors, generalizable in distribution shifts, and aligned with human intuitions.

Part II

Object

CHAPTER 4

Disentangling Objects From Background With Deep Region Competition

4.1 Introduction

In Part I, we explored the abstraction of *Category* — a broad, abstract classification that groups similar items based on shared characteristics or properties. However, numerous scenarios exist where the concept of a specific instance plays a more crucial role. In these circumstances, we turn to the abstraction of *Object*. Object-Oriented Programming (OOP) provides the most prominent example of such abstraction, where objects serve as fundamental units that encapsulate attributes and behaviors. In this chapter, we consider the problem setup of foreground extraction. We will delve into a latent-variable generative model that realizes this concept of "object encapsulation" in a statistical and computational context.

Foreground extraction, being a special case of generic image segmentation, aims for a binary partition of the given image with specific semantic meaning; *i.e.*, a foreground that typically contains identifiable objects and the possibly less structural remaining regions as the background. There is a rich literature on explicitly modeling and representing a given image as foreground and background (or more general visual regions), such that a generic inference algorithm can produce plausible segmentations ideally for any images without or with little supervision (Zhu and Yuille, 1996; Shi and Malik, 2000; Tu and Zhu, 2002; Boykov and Jolly, 2001; Rother et al., 2004; Cheng et al., 2014; Jiang et al., 2013; Zhu et al., 2014). However, such methods are essentially pixel-space modeling since they rely on low-level visual features (*e.g.*, edges, color, and texture).

There exists a rich literature on latent-variable models (Greff et al., 2016; Eslami et al., 2016; Greff et al., 2017; van Steenkiste et al., 2018; Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020; Engelcke et al., 2020; Lin et al., 2020) that treat the background as a "virtual object" and disentangle it along with all foreground objects through independence-inducing slots. Such independence is modeled with a permutation-invariant encoder, or a fully factorized prior of the latent slot variables and their shared generators. Although these methods exhibit strong performance on synthetic multi-object datasets with simple backgrounds and foreground shapes, they may fail on complex real-world data or even synthetic datasets with more challenging backgrounds (Greff et al., 2019; Locatello et al., 2020). In addition, few unsupervised learning methods have provided explicit identification of foreground objects and background regions. While they can generate valid segmentation masks, most of these methods do not specify which output corresponds to the foreground objects. These deficiencies necessitate rethinking the problem of unsupervised foreground extraction. We propose to confront the challenges in formulating (1) a generic inductive bias for modeling foreground and background regions that can be baked into neural generators, and (2) an effective inference algorithm based on a principled criterion for foreground-background partition.

Inspired by Region Competition (Zhu and Yuille, 1996), a seminal approach that combines optimization-based inference (Kass et al., 1988; Cohen, 1991; Adams and Bischof, 1994) and probabilistic visual modeling (Zhu et al., 1998; Guo et al., 2007) by minimizing a generalized Bayes criterion (Leclerc, 1989), we propose to solve the foreground extraction problem by reconciling the energy-based prior (Pang et al., 2020a) with generative image modeling in the form of Mixture of Experts (MoE) (Jacobs et al., 1991; Jordan and Jacobs, 1994). To generically describe background regions, we further introduce the learned pixel re-assignment as the essential inductive bias to capture their regularities. Enabled by our modeling, we propose to find the foreground-background partition through Expectation-Maximization (EM). Our algorithm effectively exploits the interaction between the mixture components during the partitioning process, echoing the intuition described in Region Competition (Zhu and Yuille, 1996). We therefore

coin our method Deep Region Competition (DRC).

In summary, our contributions as follows:

1. We provide a latent-variable model that reconciles energy-based prior with generative image modeling in the form of MoE. With this modeling, the foreground-background partition, as well as their latent abstraction process, can naturally be produced through EM. We further introduce an inductive bias, *pixel re-assignment*, to facilitate foreground-background disentanglement.

2. In experiments, we demonstrate that DRC exhibits more competitive performance on complex real-world data and challenging multi-object scenes compared to prior methods. Furthermore, we empirically show that using learned pixel re-assignment as the inductive bias helps to provide explicit identification of foreground and background regions.

3. We find that DRC can potentially generalize to novel foreground objects even from categories unseen during training, which may provide some inspiration for the study of out-of-distribution (OOD) generalization in more general unsupervised disentanglement.

4.2 Related Work

A typical line of methods frames unsupervised or weakly supervised foreground segmentation within a generative modeling context. Several methods build upon generative adversarial networks (GAN) (Goodfellow et al., 2014) to perform foreground segmentation. LR-GAN (Yang et al., 2017) learns to generate background regions and foreground objects separately and recursively, which simultaneously produces the foreground objects mask. ReDO (ReDrawing of Objects) (Chen et al., 2019a) proposes a GAN-based object segmentation model, based on the assumption that replacing the foreground object in the image with a generated one does not alter the distribution of the training data, given that the foreground object is correctly discovered. Similarly, SEIGAN (Ostyakov et al.,

2018) learns to extract foreground objects by recombining the foreground objects with the generated background regions. FineGAN (Singh et al., 2019) hierarchically generates images (*i.e.*, first specifying the object shape and then the object texture) to disentangle the background and foreground object. Benny and Wolf (2020) further hypothesize that a method solving an ensemble of unsupervised tasks altogether improves the model performance compared with the one that solves each individually. Therefore, they train a complex GAN-based model (OneGAN) to solve several tasks simultaneously, including foreground segmentation. Although LR-GAN and FineGAN do produce masks as part of their generative process, they cannot segment a given image. Despite SEIGAN and OneGAN achieving decent performance on foreground-background segmentation, these methods require a set of clean background images as additional inputs for weak supervision. ReDO captures the foreground objectness with possibly oversimplified assumptions, limiting its application to datasets with diverse object shapes.

On another front, disentangling latent-variable models (Greff et al., 2016; Eslami et al., 2016; Greff et al., 2017; van Steenkiste et al., 2018; Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020; Engelcke et al., 2020; Lin et al., 2020), sharing the idea of scene decomposition stemming from DRAW (Gregor et al., 2015), learn to represent foreground objects and background regions in terms of a collection of latent variables with the same representational format. These methods typically exploit the spatial mixture model for generative modeling. Specifically, IODINE (Greff et al., 2019) proposes a slot-based object representation method and models the latent space using iterative amortized inference (Marino et al., 2018). Slot-Attention (Locatello et al., 2020), as a step forward, effectively incorporates the attention mechanism into the slot-based object representation for flexible foreground object binding. Both methods use fully shared parameters among individual mixture components to entail permutation invariance of the learned multi-object representation. Alternative models such as MONet (Burgess et al., 2019) and GENESIS (Engelcke et al., 2020) use multiple encode-decode steps for scene decomposition and foreground object extraction. Although these methods exhibit strong performance on synthetic multi-object datasets with simple background and fore-

ground shapes, they may fail when dealing with complex real-world data or even synthetic datasets with more challenging background (Greff et al., 2019; Locatello et al., 2020).

More closely related to the classical methods, another line of work focuses on utilizing image features extracted by deep neural networks or designing energy functions based on data-driven methods to define the desired property of foreground objects. Pham et al. (2018) and Silberman et al. (2012) obtain impressive results when depth images are accessible in addition to conventional RGB images, while such methods are not directly applicable for data with RGB images alone. W-Net (Xia and Kulis, 2017) extracts image features via a deep auto-encoder jointly trained by minimizing reconstruction error and normalized cut. The learned features are further processed by CRF smoothing to perform hierarchical segmentation. Kanezaki (2018) proposes to employ a neural network as part of the partitioning criterion (inspired by Ulyanov et al. (2020)) to minimize the chosen intra-region pixel distance for segmentation directly. Ji et al. (2019) propose to use Invariant Information Clustering as the objective for segmentation, where the network is trained to be part of the learned distance. As an interesting extension, one may also consider adapting methods that automatically discover object structures (Lorenz et al., 2019) to foreground extraction. Though being pioneering work in image segmentation, the aforementioned methods are generally bottle-necked by the selected post-processing segmentation algorithm or require extra transformations to produce meaningful foreground segmentation masks. In their seminal work, Yang et al. (2019, 2021) propose an information-theoretical principle and adversarial contextual model for unsupervised segmentation and detection by partitioning images into maximally independent sets, with the objective of minimizing the predictability of one set by the other sets. Additional efforts have also been devoted to weakly supervised foreground segmentation using image classification labels (Papandreou et al., 2015; Pathak et al., 2015; Huang et al., 2018), bounding boxes (Dai et al., 2015; Khoreva et al., 2017), or saliency maps (Oh et al., 2017; Zeng et al., 2019; Voynov et al., 2021).

4.3 Method

Foreground extraction performs a binary partition for the image $\mathbf{I}$ to extract the foreground region. Without explicit supervision, we propose to use learned pixel re-assignment as a generic inductive bias for background modeling, upon which we derive an EM-like partitioning algorithm. Compared with prior methods, our algorithm can handle images with more complex foreground shapes and background patterns, while providing explicit identification of foreground and background regions.

4.3.1 Preliminaries

Adopting the language of EM algorithm, we assume that for the observed sample $\mathbf{x} \in \mathbb{R}^D$, there exists $\mathbf{z} \in \mathbb{R}^d$ as its latent variables. The complete-data distribution is

$$p_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{x}) = p_{\boldsymbol{\alpha}}(\mathbf{z}) p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}), \tag{4.1}$$

where $p_{\boldsymbol{\alpha}}(\mathbf{z})$ is the prior model with parameters $\boldsymbol{\alpha}$, $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$ is the top-down generative model with parameters $\boldsymbol{\beta}$, and $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$.

The prior model $p_{\boldsymbol{\alpha}}(\mathbf{z})$ can be formulated as an energy-based model, which we refer to as the Latent-space Energy-Based Model (LEBM) (Pang et al., 2020a) throughout the chapter:

$$p_{\boldsymbol{\alpha}}(\mathbf{z}) = \frac{1}{Z_{\boldsymbol{\alpha}}} \exp\left(f_{\boldsymbol{\alpha}}(\mathbf{z})\right) p_0(\mathbf{z}), \tag{4.2}$$

where $f_{\boldsymbol{\alpha}}(\mathbf{z})$ can be parameterized by a neural network, $Z_{\boldsymbol{\alpha}}$ is the partition function, and $p_0(\mathbf{z})$ is a reference distribution, assumed to be isotropic Gaussian prior commonly used for the generative model. The prior model in (4.2) can be interpreted as an energy-based correction or exponential tilting of the original prior distribution p_0.

The LEBM can be learned by Maximum Likelihood Estimation (MLE). Given a training sample $\mathbf{x}$, the learning gradient for $\boldsymbol{\alpha}$ is derived as (Pang et al., 2020a)

$$\delta_{\boldsymbol{\alpha}}(\mathbf{x}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{z})\right] - \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{z})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{z})\right]. \tag{4.3}$$

In practice, the above expectations can be approximated by Monte-Carlo average, which requires sampling from $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ and $p_{\boldsymbol{\alpha}}(\mathbf{z})$. This step can be done with stochastic gradient-based methods, such as Langevin dynamics (Welling and Teh, 2011) or Hamiltonian Monte Carlo (Brooks et al., 2011).

An extension to LEBM is to further couple the vector representation $\mathbf{z}$ with a symbolic representation $\mathbf{y}$ (Pang and Wu, 2021). Formally, $\mathbf{y}$ is a K-dimensional one-hot vector, where K is the number of possible $\mathbf{z}$ categories. Such symbol-vector duality can provide extra entries for auxiliary supervision; we will detail it in Section 4.3.4.

4.3.2 Generative Image Modeling

Mixture of Experts for Image Generation Inspired by the regional homogenity assumption proposed by Zhu and Yuille (1996), we use separate priors and generative models for foreground and background regions, indexed as $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k,\ k = 1, 2$, respectively; see Figure 4.1. This design leads to the form of MoE (Jacobs et al., 1991; Jordan and Jacobs, 1994) for image modeling, as shown below.

Let us start by considering only the i-th pixel of the observed image $\mathbf{x}$, denoted as $\mathbf{x}_i$. We use a binary one-hot random variable $\mathbf{w}_i$ to indicate whether the i-th pixel belongs to the foreground region. Formally, we have $\mathbf{w}_i = [w_{i1}, w_{i2}]$, $w_{ik} \in \{0, 1\}$ and $\sum_{k=1}^{2} w_{ik} = 1$. Let $w_{i1} = 1$ indicate that the i-th pixel $\mathbf{x}_i$ belongs to the foreground, and $w_{i2} = 1$ indicate the opposite.

We assume that the distribution of $\mathbf{w}_i$ is prior-dependent. Specifically, the mixture parameter $\pi_{ik},\ k = 1, 2$, is defined as the output of a gating function $\pi_{ik} = p_{\boldsymbol{\beta}}(w_{ik} = 1|\mathbf{z}) = \text{Softmax}(l_{ik})$; $l_{ik} = h_{\boldsymbol{\beta}_k}(\mathbf{z}_k),\ k = 1, 2$ are the logit scores given by the foreground and background generative models respectively; $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2\}, \mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2\}$. Taken together, the joint distribution of $\mathbf{w}_i$ is

$$p_{\boldsymbol{\beta}}(\mathbf{w}_i|\mathbf{z}) = \prod_{k=1}^{2} \pi_{ik}^{w_{ik}}. \tag{4.4}$$

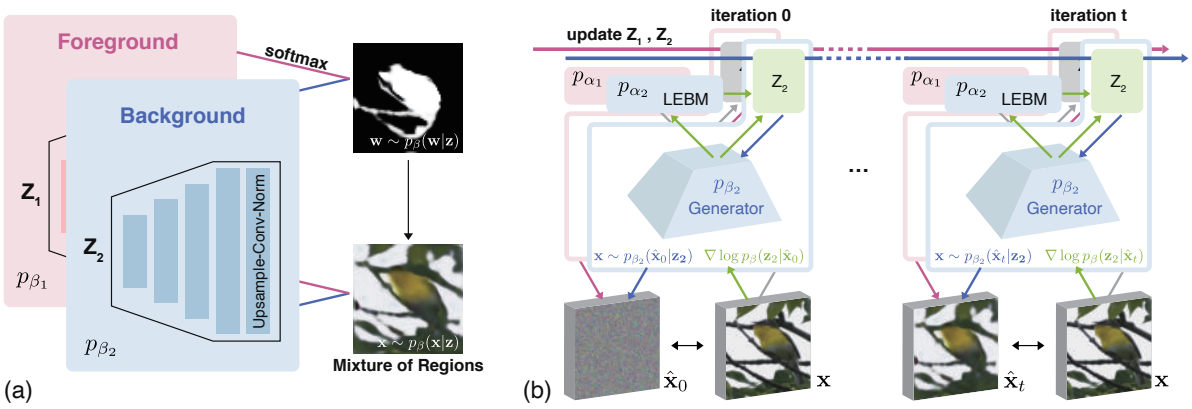


Figure 4.1: Schematic illustration of Deep Region Competition. (a) The model generates foreground and background regions using sampled latent variables $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2\}$. $p_{\boldsymbol{\beta}_k}$, $k = 1, 2$ represents the generator for each region. Of note, the pixel re-assignment function is absorbed in the background generator; see Section 4.3.2 for details. (b) DRC samples the latent variables $\mathbf{z}$ in an iterative manner. Let $\mathbf{x}$ denote the observed image; we use $\hat{\mathbf{x}}_t$, $t = 0, 1, \ldots$ to represent the image generated by $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$ at the t-th sampling step. DRC has a two-step workflow for learning unsupervised foreground extractors that resembles the E- and M-step in the classic EM algorithm. In the E-step, it employs gradient-based MCMC sampling to infer the latent variables $\mathbf{z}$ as shown in (b). Of note, only the latent variables $\mathbf{z}$ are updated in this step. In the M-step, the sampled latent variables $\mathbf{z}$ are fed into the model for image generation as shown in (a), where the generators are updated to minimize the reconstruction error.

The learned distribution of foreground and background contents are

$$p_{\boldsymbol{\beta}}(\mathbf{x}_i|w_{ik} = 1, \mathbf{z}_k) = p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k) \sim \mathcal{N}(g_{\boldsymbol{\beta}_k}(\mathbf{z}_k), \sigma^2\mathbf{I}), \ k = 1, 2, \tag{4.5}$$

where we assume that the generative model for region content, $p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)$, $k = 1, 2$, follows a Gaussian distribution parameterized by the generator network $g_{\boldsymbol{\beta}_k}$. As in VAE, σ takes an assumed value. We follow the common practice and use a shared generator for parameterizing π_{ik} and $p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)$. We use separate branches only at the output layer to generate logits and contents.

Generating $\mathbf{x}_i$ based on $\mathbf{w}_i$'s distribution involves two steps: (1) sample $\mathbf{w}_i$ from the distribution $p_{\boldsymbol{\beta}}(\mathbf{w}_i|\mathbf{z})$, and (2) choose either the foreground model (*i.e.*, $p_{\boldsymbol{\beta}_1}(\mathbf{x}_i|\mathbf{z}_1)$) or the background model (*i.e.*, $p_{\boldsymbol{\beta}_2}(\mathbf{x}_i|\mathbf{z}_2)$) to generate $\mathbf{x}_i$ based on the sampled $\mathbf{w}_i$. As such, this

distribution of $\mathbf{x}_i$ is a MoE,

$$p_{\boldsymbol{\beta}}(\mathbf{x}_i|\mathbf{z}) = \sum_{k=1}^{2} p_{\boldsymbol{\beta}}(w_{ik}=1|\mathbf{z}) p_{\boldsymbol{\beta}}(\mathbf{x}_i|w_{ik}=1,\mathbf{z}_k) = \sum_{k=1}^{2} \pi_{ik} p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k), \tag{4.6}$$

wherein the posterior responsibility of w_{ik} is

$$\gamma_{ik} = p(w_{ik}=1|\mathbf{x}_i,\mathbf{z}) = \frac{\pi_{ik} p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)}{\sum_{m=1}^{2} \pi_{im} p_{\boldsymbol{\beta}_m}(\mathbf{x}_i|\mathbf{z}_m)}, \ k=1,2. \tag{4.7}$$

Using a fully-factorized joint distribution of $\mathbf{x}$, we have $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^{D} \sum_{k=1}^{2} \pi_{ik} p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)$ as the generative modeling of $\mathbf{x} \in \mathbb{R}^D$.

Learning Pixel Reassignment for Background Modeling We use pixel re-assignment in the background generative model as the essential inductive bias for modeling the background region. This is partially inspired by the concepts of "texture" and "texton" by Julez (Guo et al., 2007; Julesz, 1981), where the textural part of an image may contain fewer structural elements in preattentive vision, which coincides with our intuitive observation of the background regions.

We use a separate pair of energy-based prior model $\boldsymbol{\alpha}_{\texttt{pix}}$ and generative model $\boldsymbol{\beta}_{\texttt{pix}}$ to learn the re-assignment. For simplicity, we absorb $\boldsymbol{\alpha}_{\texttt{pix}}$ and $\boldsymbol{\beta}_{\texttt{pix}}$ in the models for background modeling, *i.e.*, $\boldsymbol{\alpha}_2$ and $\boldsymbol{\beta}_2$, respectively. In practice, the re-assignment follows the output of $\boldsymbol{\beta}_{\texttt{pix}}$, a shuffling grid with the same size of the image $\mathbf{x}$. Its values indicate the re-assigned pixel coordinates; see Figure 4.2. We find that shuffling the background pixels using the learned re-assignment facilitates the model to capture the regularities of the background regions. Specifically, the proposed model with this essential inductive bias learns to constantly give the correct mask assignment, whereas most previous fully unsupervised methods do not provide explicit identification of the foreground and background regions; see discussion in Section 4.4.1 for more details.

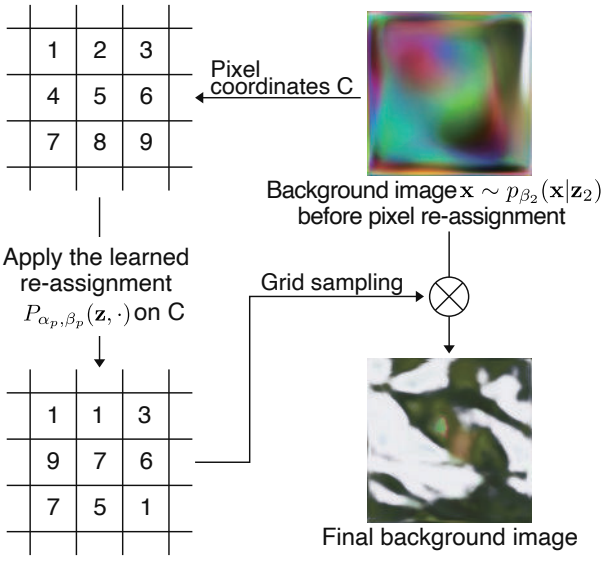


Figure 4.2: Schematic illustration of pixel re-assignment. The output of $\boldsymbol{\beta}_p$ can be viewed as a learned re-assignment of the original background pixels that follows the mapped grid $P_{\boldsymbol{\alpha}_p,\boldsymbol{\beta}_p}(\mathbf{z}, C)$. Note that the re-assignment function $P_{\boldsymbol{\alpha}_p,\boldsymbol{\beta}_p}(\mathbf{z},\cdot)$ might not be injective. The final background image is generated via grid sampling.

4.3.3 Deep Region Competition: From Generative Modeling to Foreground Extraction

The complete-data distribution from the image modeling is

$$\begin{aligned} p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z},\mathbf{w}) &= p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{w},\mathbf{z})p_{\boldsymbol{\beta}}(\mathbf{w}|\mathbf{z})p_{\boldsymbol{\alpha}}(\mathbf{z}) \\ &= \left(\prod_{i=1}^{D}\prod_{k=1}^{2} p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)^{w_{ik}}\right)\left(\prod_{i=1}^{D}\prod_{k=1}^{2}\pi_{ik}^{w_{ik}}\right)p_{\boldsymbol{\alpha}}(\mathbf{z}) \\ &= p_{\boldsymbol{\alpha}}(\mathbf{z})\prod_{i=1}^{D}\prod_{k=1}^{2}\left(\pi_{ik}p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)\right)^{w_{ik}}, \end{aligned} \tag{4.8}$$

where $p_{\boldsymbol{\alpha}}(\mathbf{z}) = p_{\boldsymbol{\alpha}_1}(\mathbf{z}_1)p_{\boldsymbol{\alpha}_2}(\mathbf{z}_2)$ is the prior model given by LEBMs. $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2\}$, and $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$. $\mathbf{w}$ is the vector of $(\mathbf{w}_i), i = 1, \ldots, D$, whose joint distribution is assumed to be fully-factorized.

Next, we derive the complete-data log-likelihood as our learning objective:

$$\mathcal{L}(\boldsymbol{\theta}) = \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z},\mathbf{w}) = \log p_{\boldsymbol{\alpha}}(\mathbf{z}) + \sum_{i=1}^{D}\sum_{k=1}^{2} w_{ik}\left(\log \pi_{ik} + \log p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)\right). \tag{4.9}$$

Of note, $\mathbf{w}$ and $\mathbf{z}$ are unobserved variables in the modeling, which makes it impossible to learn the model directly through MLE. To calculate the gradients of $\boldsymbol{\theta}$, we instead optimize $\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x}),\mathbf{w}\sim p(\mathbf{w}|\mathbf{x},\mathbf{z})}[\mathcal{L}(\boldsymbol{\theta})]$ based on the fact that underlies the EM algorithm:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \int_{\mathbf{w}} p_{\boldsymbol{\theta}}(\mathbf{w}|\mathbf{z},\mathbf{x}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z},\mathbf{w}) d\mathbf{w} \\ &= \mathbb{E}_{\mathbf{z}\sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}),\mathbf{w}\sim p_{\boldsymbol{\theta}}(\mathbf{w}|\mathbf{x},\mathbf{z})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z},\mathbf{w})].\end{aligned} \tag{4.10}$$

Therefore, the derived surrogate learning objective becomes

$$\max_{\boldsymbol{\theta}} \ \mathbb{E}_{\mathbf{z}\sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})} \left[\mathcal{J}(\boldsymbol{\theta})\right], \ \text{s.t.} \ \forall i, \sum_{k=1}^{2} \pi_{ik} = 1, \tag{4.11}$$

$$\mathcal{J}(\boldsymbol{\theta}) = \underbrace{\log p_{\boldsymbol{\alpha}}(\mathbf{z})}_{\text{objective for LEBM}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2} \gamma_{ik} \log \pi_{ik}}_{\text{foreground-background partitioning}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2} \gamma_{ik} \log p_{\boldsymbol{\theta}_k}(\mathbf{x}_i|\mathbf{z}_k)}_{\text{objective for image generation}}, \tag{4.12}$$

where $\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{w}\sim p_{\boldsymbol{\theta}}(\mathbf{w}|\mathbf{x},\mathbf{z})}\left[\mathcal{L}(\boldsymbol{\theta})\right]$ is the conditional expectation of $\mathbf{w}$, which can be calculated in closed form; see the supplementary material for additional details.

(4.11) has an intuitive interpretation. We can decompose the learning objective into three components as in (4.12). In particular, the second term $\sum_{i=1}^{D}\sum_{k=1}^{2} \gamma_{ik} \log \pi_{ik}$ has a similar form to the cross-entropy loss commonly used for supervised segmentation task, where the posterior responsibility γ_{ik} serves as the target distribution. It is as if the foreground and background generative models compete with each other to fit the distribution of each pixel $\mathbf{x}_i$. If the pixel value at $\mathbf{x}_i$ fits better to the distribution of foreground, $p_{\boldsymbol{\beta}_1}(\mathbf{x}_i|\mathbf{z}_1)$, than to that of background, $p_{\boldsymbol{\beta}_2}(\mathbf{x}_i|\mathbf{z}_2)$, the model tends to assign that pixel to the foreground region (see (4.7)), and vice versa. This mechanism is similar to the process derived by Zhu and Yuille (1996), which is the reason why we coin our method Deep Region Competition (DRC).

Prior to our proposal, several methods (Zhu and Yuille, 1996; Greff et al., 2019; Locatello et al., 2020) also employ mixture models and competition among the components

to perform unsupervised foreground or image segmentation. The original Region Competition (Zhu and Yuille, 1996) combines several families of image modeling with Bayesian inference but is limited by the expressiveness of the pre-specified probability distributions. More recent methods, including IODINE (Greff et al., 2019) and Slot-attention (Locatello et al., 2020), learn amortized inference networks for latent variables and induce the independence of foreground and background representations using an identical generator. Our method combines the best of the two worlds, reconciling the expressiveness of learned generators with the regularity of generic texture modeling under the framework of LEBM.

To optimize the learning objective in (4.11), we approximate the expectation by sampling from the prior $p_{\boldsymbol{\alpha}}(\mathbf{z})$ and posterior model $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \propto p_{\boldsymbol{\alpha}}(\mathbf{z})p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$, followed by calculating the Monte Carlo average. We use Langevin dynamics (Welling and Teh, 2011) to draw persistent MCMC samples, which iterates

$$\mathbf{z}_{t+1} = \mathbf{z}_t + s\nabla_{\mathbf{z}} \log Q(\mathbf{z}_t) + \sqrt{2s}\boldsymbol{\epsilon}_t, \tag{4.13}$$

where t is the Langevin dynamics's time step, s the step size, and $\boldsymbol{\epsilon}_t$ the Gaussian noise. $Q(\mathbf{z})$ is the target distribution, being either $p_{\boldsymbol{\alpha}}(\mathbf{z})$ or $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. $\nabla_{\mathbf{z}} \log Q(\mathbf{z}_t)$ is efficiently computed via automatic differentiation in modern learning libraries (Paszke et al., 2019). We summarize the above process in Algorithm 4.

During inference, we initialize the latent variables $\mathbf{z}$ for MCMC sampling from Gaussian white noise and run only the posterior sampling step to obtain $\mathbf{z}_+$. The inferred mask and region images are then given by the outputs of generative models $p_{\boldsymbol{\beta}_k}(\mathbf{w}|\mathbf{z}_+)$ and $p_{\boldsymbol{\beta}_k}(\mathbf{x}|\mathbf{z}_+)$, $k = 1, 2$, respectively.

4.3.4 Technical Details

Pseudo Label for Additional Regularization Although the proposed DRC explicitly models the interaction between the regions, it is still possible that the model converges to a trivial extractor, which treats the entire image as the foreground or back-

Algorithm 4 Learning models of DRC via EM.

Require: Learning iterations T, initial parameters for LEBMs $\boldsymbol{\alpha}^{(0)} = \{\boldsymbol{\alpha}_1^{(0)}, \boldsymbol{\alpha}_2^{(0)}\}$ and generators $\boldsymbol{\beta}^{(0)} = \{\boldsymbol{\beta}_1^{(0)}, \boldsymbol{\beta}_2^{(0)}\}$, $\boldsymbol{\theta}^{(0)} = \{\boldsymbol{\alpha}^{(0)}, \boldsymbol{\beta}^{(0)}\}$, learning rate $\eta_{\boldsymbol{\alpha}}$ for LEBMs, $\eta_{\boldsymbol{\beta}}$ for foreground and background generators, observed examples $\{\mathbf{x}^{(i)}\}_{i=1}^N$, batch size M, and initial latent variables $\{\mathbf{z}_-^{(i)} = \{\mathbf{z}_{1-}^{(i)}, \mathbf{z}_{2-}^{(i)}\} \sim p_0(\mathbf{z})\}_{i=1}^N$ and $\{\mathbf{z}_+^{(i)} = \{\mathbf{z}_{1+}^{(i)}, \mathbf{z}_{2+}^{(i)}\} \sim p_0(\mathbf{z})\}_{i=1}^N$.

Ensure: $\boldsymbol{\theta}^{(T)} = \{\boldsymbol{\alpha}_1^{(T)}, \boldsymbol{\beta}_1^{(T)}, \boldsymbol{\alpha}_2^{(T)}, \boldsymbol{\beta}_2^{(T)}\}$.

1: **for** $t = 0$ **to** $T - 1$ **do**
2: Sample a minibatch of data $\{\mathbf{x}^{(i)}\}_{i=1}^M$
3: **Prior sampling for learning LEBMs:** For each $\mathbf{x}^{(i)}$, update $\mathbf{z}_-^{(i)}$ using (4.13), with target distribution $\pi(\mathbf{z}) = p_{\boldsymbol{\alpha}^{(t)}}(\mathbf{z})$
4: **Posterior sampling for foreground and background generation:** For each $\mathbf{x}^{(i)}$, update $\mathbf{z}_+^{(i)}$ using (4.13), with target distribution $Q(\mathbf{z}) = p_{\boldsymbol{\theta}^{(t)}}(\mathbf{z}|\mathbf{x})$
5: **Update LEBMs:** $\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \eta_{\boldsymbol{\alpha}} \frac{1}{m} \sum_{i=1}^m [\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}^{(t)}}(\mathbf{z}_+^{(i)}) - \nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}^{(t)}}(\mathbf{z}_-^{(i)})]$
6: **Update foreground and background generators:** $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \eta_{\boldsymbol{\beta}} \frac{1}{m} \sum_{i=1}^m \nabla_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}^{(t)}}(\mathbf{x}^{(i)}|\mathbf{z}_+^{(i)})$
7: **end for**

ground region, leaving the other region null. We exploit the symbolic vector $\mathbf{y}$ emitted by the LEBM (see Section 4.3.1) for additional regularization. The strategy is similar to the mutual information maximization used in InfoGAN (Chen et al., 2016). Specifically, we use the symbolic vector $\mathbf{y}$ inferred from $\mathbf{z}$ as the pseudo-class label for $\mathbf{z}$ and train an auxiliary classifier jointly with the above models; it ensures that the generated regions $\mathbf{x}_k$ contain similar symbolic information for $\mathbf{z}_k$. Intuitively, this loss prevents the regions from converging to null since the symbolic representation $\mathbf{y}_k$ would never be well retrieved if that did happen.

Implementation We adopt a similar architecture for the generator as in DCGAN (Radford et al., 2016) throughout the experiments and only change the dimension of the latent variables $\mathbf{z}$ for different datasets. The generator consists of a fully connected layer followed by five stacked upsample-conv-norm layers. We replace the batch-norm layers (Ioffe and Szegedy, 2015) with instance-norm (Ulyanov et al., 2016) in the architecture. The energy-term in LEBM is parameterized by a 3-layered MLP. We adopt orthogonal initialization (Saxe et al., 2014) commonly used in generative models to initialize the networks and orthogonal regularization (Brock et al., 2016) to facilitate training. In ad-

dition, we observe performance improvement when adding Total-Variation norm (Rudin et al., 1992) for the background generative model. More details, along with specifics of the implementation used in our experiments, are provided in the supplementary material.

4.4 Experiments

We design experiments to answer three questions: (1) How does the proposed method compare to previous state-of-the-art competitors? (2) How do the proposed components contribute to the model performance? (3) Does the proposed method exhibit generalization on images containing unseen instances (*i.e.*, same category but not the same instance) and even objects from novel categories?

To answer these questions, we evaluate our method on five challenging datasets in two groups: (1) Caltech-UCSD Birds-200-2011 (Birds) (Welinder et al., 2010), Stanford Dogs (Dogs) (Khosla et al., 2011), and Stanford Cars (Cars) (Krause et al., 2013) datasets; (2) CLEVR6 (Johnson et al., 2017) and Textured Multi-dSprites (TM-dSprites) (Matthey et al., 2017) datasets. The first group of datasets covers complex real-world domains, whereas the second group features environments of the multi-object foreground with challenging spatial configurations or confounding backgrounds. As to be shown, the proposed method is generic to various kinds of input and produces more competitive foreground-background partition results than prior methods.

4.4.1 Results on Foreground Extraction

Single Object in the Wild In the first group of datasets, there is typically a single object in the foreground, varying in shapes, texture, and lighting conditions. Unsupervised foreground extraction on these datasets requires much more sophisticated visual cues than colors and shapes. Birds dataset consists of 11,788 images of 200 classes of birds annotated with high-quality segmentation masks, Dogs dataset consists of 20,580 images of 120 classes annotated with bounding boxes, and Cars dataset consists of 16,185 images of 196 classes. The latter two datasets are primarily made for fine-grained cat-

Model	Single Object						Multi-Object			
	Birds		Dogs		Cars		CLEVR6		TM-dSprites	
	IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice
W-Net*	24.8	38.9	47.7	62.1	52.8	67.6	-	-	-	-
GrabCut	30.2	42.7	58.3	70.9	61.3	73.1	19.0	30.5	61.9	71.0
ReDO§	46.5	60.2	55.7	70.3	52.5	68.6	18.6	31.0	9.4	17.2
OneGAN*†	55.5	69.2	71.0	81.7	71.2	82.6	-	-	-	-
IODINE§	30.9	44.6	54.4	67.0	51.7	67.3	19.9	32.4	7.3	12.8
Slot-Attn.§	35.6	51.5	38.6	55.3	41.3	58.3	83.6	90.7	7.3	13.5
Ours	**56.4**	**70.9**	**71.7**	**83.2**	**72.4**	**83.7**	**84.7**	**91.5**	**78.8**	**87.5**

Table 4.1: Foreground extraction results on training data measured in IoU and Dice. Higher is better in all scores. *Results of W-Net and OneGAN are provided by Benny and Wolf (2020). Of note, results of these two models on Dogs and Cars datasets may **not** be directly comparable to other listed methods, as the data used for training and evaluation could be different. We include these results as a rough reference since no official implementation or pretrained model are publicly available. § indicates unfair baseline results obtained using extra ground-truth information, *i.e.*, we choose the best-matching scores from the permutation of foreground and background masks. †OneGAN is a strong **weakly supervised** baseline, which requires clean background images to provide additional supervision. We include this model as a potential upper bound of the fully unsupervised methods.

egorization. To evaluate foreground extraction, we follow the practice in (Benny and Wolf, 2020), and approximate ground-truth masks for the images with Mask R-CNN (He et al., 2017), pre-trained on the MS COCO (Lin et al., 2014) dataset with a ResNet-101 (He et al., 2016) backend. The pre-trained model is acquired from the detectron2 toolkit (Wu et al., 2019). This results in 5,024 dog images and 12,322 car images with a clear foreground-background setup and corresponding masks.

On datasets featuring a single foreground object, we use the 2-slot version of IODINE and Slot-attention. Since ReDO, IODINE, and Slot-Attention do not distinguish foreground and background in output regions, we choose the best-matching scores from the permutation of foreground and background masks as in (Chen et al., 2019a). We observe that the proposed method and Grabcut are the only two methods that provide explicit identification of foreground objects and background regions. While the Grabcut algorithm actually requires a predefined bounding box as input that specifies the foreground

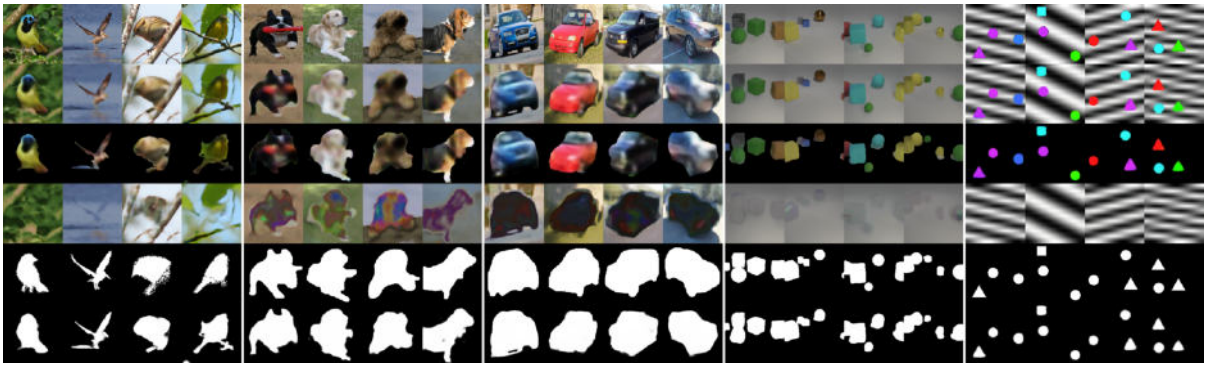


Figure 4.3: Foreground extraction results for each dataset; zoom in for better visibility. From top to bottom: (i) observed images, (ii) generated images, (iii) masked generated foregrounds, (iv) generated backgrounds, (v) ground-truth foreground masks, and (vi) inferred foreground masks. More samples and results of baselines can be found in the supplementary material.

region, our method, thanks to the learned pixel re-assignment (see Section 4.3.2), can achieve this in a fully unsupervised manner. Results in Table 4.1 show that our method outperforms all the unsupervised baselines by a large margin, exhibiting comparable performance even to the weakly supervised baseline that requires additional background information as inputs (Benny and Wolf, 2020). We provide samples of foreground extraction results as well as generated background and foreground regions in Figure 4.3. Note that our final goal is not to synthesize appealing images but to learn foreground extractors in a fully unsupervised manner. As the limitation of our method, DRC generates foreground and background regions less realistic than those generated by state-of-the-art GANs, which hints a possible direction for future work. More detailed discussions of the limitation can be found in supplementary material.

Multi-Object Scenes The second group of datasets contains images with possibly simpler foreground objects but more challenging scene configurations or background parts. Visual scenes in the CLEVR6 dataset contain various objects and often with partial occlusions and truncations. Following the evaluation protocol in IODINE and Slot-attention, we use the first 70K samples from CLEVR (Johnson et al., 2017) and filter the samples for scenes with at most 6 objects for training and evaluation, *i.e.*, CLEVR6. The TM-dSprites dataset is a variant of Multi-dSprites (Matthey et al., 2017)

but has strongly confounding backgrounds borrowed from Textured MNIST (Greff et al., 2016). We generate 20K samples for the experiments. Like Greff et al. (2019) and Locatello et al. (2020), we evaluate on a subset containing 1K samples for testing. Note that IODINE and Slot-attention are designed for segmenting complex multi-object scenes using slot-based object representations. Ideally, the output of these models consists of masks for each individual object, while the background is viewed as a virtual "object" as well. In practice, however, it is possible that the model distributes the background over all the slots as mentioned in (Locatello et al., 2020). We therefore propose two corresponding approaches (see the supplementary material for more details) to convert the output object masks into a foreground-background partition and report the best results of these two options for IODINE and Slot-attention in Table 4.1.

On the CLEVR6 dataset, we use the publicly available pretrained model for IODINE, which achieves a reasonable ARI (excluding background pixels) of 94.4 on the testing data, close to the testing results in (Greff et al., 2019). We observe that IODINE distributes the background over all the slots for some of the testing samples, resulting in much lower IoU and Dice scores. We re-train the Slot-attention model using the official implementation on CLEVR6, as no pretrained model is publicly available. The re-trained model achieves a foreground ARI of 98.0 on the 1K testing samples, which we consider as a sign of valid re-implementation. Results in Table 4.1 demonstrate that the proposed method can effectively process images of challenging multi-object scenes. To be specific, our method demonstrates competitive performance on the CLEVR6 dataset compared with the SOTA object discovery method. Moreover, as shown empirically in Figure 4.3, the proposed method can handle the strongly confounding background introduced by Greff et al. (2016), whereas previous methods are distracted by the background and mostly fail to capture the foreground objects.

Model	IoU	Dice
amortized inference*	-	-
w/o pix. re-assign.	21.8	35.3
w/o pseudo label	48.7	64.2
w/o TV-norm reg.	53.0	68.1
w/o ortho. reg.	54.3	69.2
short-run chain†	52.5	67.7
Full model	56.4	70.9

Table 4.2: Ablation study on Birds. *We replace the LEBM with encoders to perform amortized inference for the latent variables $\mathbf{z}$ within a variational framework as in VAE (Kingma and Welling, 2014a). †We explore the possibility of using short-run MCMC (Nijkamp et al., 2019) instead of persistent chain sampling.

4.4.2 Ablation Study

We provide ablation studies on the Birds dataset to inspect the contribution of each proposed component in our model. As shown in Table 4.2, we observe that replacing the LEBMs in the foreground and background models with amortized inference networks significantly harms the performance of the proposed method. In particular, the modified model fails to generate any meaningful results (indicated as "-" in Table 4.2). We conjecture that LEBM benefits from the low-dimensionality of the latent space (Pang et al., 2020a) and therefore enjoys more efficient learning. However, the inference networks need to learn an extra mapping from the high-dimensional image space to the latent space and require more elaborate architecture and tuning for convergence. Furthermore, we observe that the model that does not learn pixel re-assignment for background can still generate meaningful images but only vaguely captures masks for foreground extraction.

4.4.3 Generalizable Foreground Extraction

Extracting Novel Foreground Objects From Training Categories We show results on generalizing to novel objects from the training classes. To evaluate our method, we split the Birds dataset following Chen et al. (2019a), resulting in 10K training images and 1K testing images. On Dogs and Cars datasets, we split the dataset based on the original train-test split (Khosla et al., 2011; Krause et al., 2013). This split gives

Model	Birds		Dogs		Cars	
	IoU	Dice	IoU	Dice	IoU	Dice
	Tr.|Te.	Tr.|Te.	Tr.|Te.	Tr.|Te.	Tr.|Te.	Tr.|Te.
GrabCut*	30.2|30.3	42.7|42.8	58.3|57.9	70.8|70.5	60.9|61.6	72.7|73.5
ReDO	46.8|47.1	61.4|61.7	54.3|52.8	69.2|67.9	52.6|52.5	68.7|68.6
Ours	**54.8|54.6**	**69.5|69.4**	**71.6|72.3**	**83.2|83.6**	**71.9|70.8**	**83.3|82.5**

Table 4.3: Performance of DRC on training and held-out testing data. *Note that GrabCut is a deterministic method that does not require training. We report the results of GrabCut (Rother et al., 2004) on these splits only for reference. Tr. indicates the performance on training data, and Te. indicates the performance on testing data.

Test	Train	GrabCut		ReDO		Ours	
		IoU	Dice	IoU	Dice	IoU	Dice
Birds	Birds*	30.3	42.8	47.1	61.7	54.6	69.4
	Dogs			22.2	35.3	**41.3**	**57.4**
	Cars			16.4	27.7	39.2	55.3
Dogs	Dogs*	57.9	70.5	52.8	67.9	72.3	83.6
	Cars			44.5	61.2	**67.8**	**80.4**
	Birds			44.0	60.3	53.6	69.1
Cars	Cars*	61.6	73.5	52.5	68.6	70.8	82.5
	Dogs			51.6	67.1	**68.6**	**81.0**
	Birds			41.8	58.6	45.1	61.7

Table 4.4: Performance of DRC on unseen testing categories. *We include the testing results of models trained with data from the same categories for reference.

3,286 dog images and 6,218 car images for training, and 1,738 dog images and 6,104 car images for testing, respectively. As summarized in Table 4.3, our method shows superior performances compared with baselines; the performance gap between training and testing is constantly small over all datasets.

Extracting Novel Foreground Objects From Unseen Categories To investigate how well our method generalizes to categories unseen during training, we evaluate the models trained in real-world single object datasets on the held-out testing data from different categories. We use the same training and testing splits on these datasets as in the previous experiments. Table 4.4 shows that our method outperforms the baselines on

the Birds dataset when the model has trained on Dogs or Cars dataset, which have quite different distributions in foreground object shapes. Competitors like ReDO also exhibit such out-of-distribution generalization but only to a limited extent. Similar results are observed when using Dogs or Cars as the testing set. We can see that when the model is trained on Dogs and evaluated on Cars or vice versa, it still maintains comparable performances w.r.t those are trained&tested on the same class. We hypothesize that these two datasets have similar distributions in foreground objects and background regions. In the light of this, we can further entail that the distribution of Dogs is most similar to that of Cars and less similar to that of Birds according to the results, which is consistent with our intuitive observation of the data. We provide a preliminary analysis of the statistics of these datasets in the supplementary material.

4.5 Summary

In this chapter, we presented DRC, a latent variable generative model that is specially structured for disentangling and abstracting objects from backgrounds. We identified the lack of independence and symmetry between objects and backgrounds and introduced learned pixel re-assignment as an inductive bias to capture the background regularities. Experiments demonstrate that DRC exhibits more competitive performance on complex real-world data and challenging multi-object scenes. We also showed empirically that learned pixel re-assignment helps to provide explicit identification for foreground and background regions. Moreover, we found that DRC can potentially generalize to novel foreground objects even from categories unseen during training. Extending the conclusion of Part I in terms of the advantage of latent abstraction inference, this chapter demonstrated the representational and inferential benefits of additional generic statistical modeling in latent-variable models.

CHAPTER 5

Measuring Object Compositionality in Latent Representations

5.1 Introduction

The previous chapter examined latent object abstraction in images featuring a single iconic object; however, a universal observation about objects is that they rarely exist in isolation — shoes typically appear in pairs and a table often accompanies a chair. Consequently, the assumption of independence between latent object slots, although principled, does not align with human intuition. In this chapter, we shift our focus from the statistical structure to the algebraic structure of object abstractions. Specifically, we expand our scope from the single-object foreground extraction problem to a multi-object scene understanding problem. This transition allows us to explore the intricate relationships and dependencies between objects in complex environments.

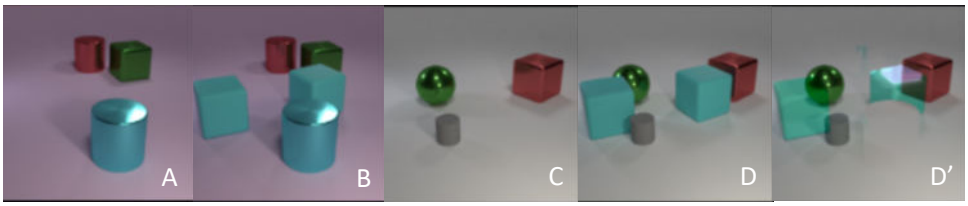
Much work in representation learning has recently focused on the property of "objectness", where the goal is to form abstract, low dimensional representations $\mathbf{z} \in \mathbb{R}^D$ of input scenes $\mathbf{x} \in \mathbb{R}^N$ ($D << N$) that parse the scene into its constituent objects such that these object primitives can be recomposed together to parse entirely novel scenes(Burgess et al., 2019; Bapst et al., 2019; Goyal et al., 2020; Greff et al., 2019, 2020; Kosiorek et al., 2018; Singh et al., 2021; van Steenkiste et al., 2018; Eslami et al., 2016; Huang and Murphy, 2015). Such representations can support reasoning and higher level tasks such as counting (Chattopadhyay et al., 2017; Antol et al., 2015), abstraction (Higgins et al., 2016; Vedantam et al., 2021), puzzle solving (Barrett et al., 2018), and reinforcement learning (Bapst et al., 2019). More generally, objectness is a promising inductive bias for

generalization to out-of-distribution scenes (Locatello et al., 2020; Chattopadhyay et al., 2017).

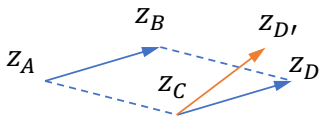
One can formalize the notion of object-centric representations or abstractions from the viewpoint of compositionality (Mikolov et al., 2013b) or disentanglement (Higgins et al., 2016). For compositionality, we aim to learn representations featuring low-level primitives that can be composed via simple vector operations to represent more complex inputs. Critically, such a representation should be learnable from a small set of primitive combinations and should enable those vector operations to generalize to any combination of primitives (Lake and Baroni, 2018; Radford et al., 2016; Mitchell and Lapata, 2008). Disentanglement takes a stricter view in which we not only aim to infer and compose different factors of variation, but also require that these factors are partitioned into distinct dimensions z_i or K "slots", namely, $\mathbf{z}_k \in \mathbb{R}^{\frac{D}{K}}$ of the representation space $\mathcal{Z}$, in order to recover ground-truth, statistically independent factors of variation that generate the observations. However, representations may be compositional without being disentangled. For example, applying a change of basis through a rotation to a disentangled representation will yield a representation that is still compositional but no longer disentangled. Nevertheless, the goal of uncovering true factors of variation in a data driven manner remains conceptually appealing, and there has been a lot of interest in learning disentangled, object-centric representations (Locatello et al., 2020; Greff et al., 2019; Singh et al., 2021).

Slot-Based Approaches Slot-based approaches to objectness aim to disentangle K statistically independent sub-spaces or slots (Greff et al., 2019; Locatello et al., 2020) $\mathbf{z}_k \in \mathbb{R}^{\frac{D}{K}}$ such that each slot specializes to capture all relevant ground truth properties of one of the objects in the observed scene. Evaluation of such models is typically done using clustering-based metrics by decoding each of the slots back to the pixel space with a generator.

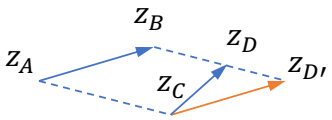
Slot-Free approaches Although slot-based approaches to measuring objectness are relatively simple to define and evaluate, the strict requirement of slot-based delineation in the representations limits its applicability to architectures without slots. This



(a) COAT test case



(b) Compositional representation

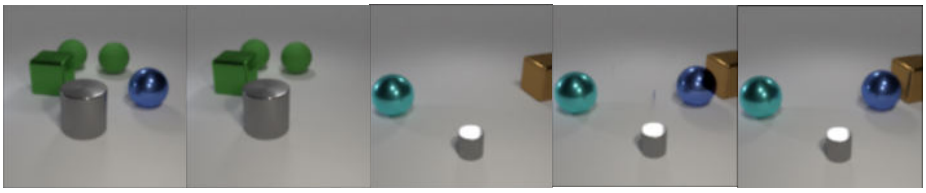


(c) Non-compositional representation

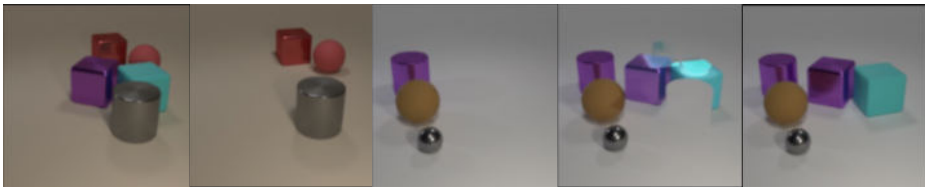
Figure 5.1: An illustration of the *parallelogram* in COAT. The transformation from scene A to scene B is to add two blue rubber cubes, so is the one from C to D. Hence in the representation space, we would expect the translation vector $\mathbf{z}_B - \mathbf{z}_A$ to be identical to $\mathbf{z}_D - \mathbf{z}_C$. Further, we would expect the parallelogram not to hold for A, B, C and hard negative D'. Here D' is the pixel-level hard negative, resulted from $B - A + C$ in the pixel space.

downside is particularly relevant as most commonly utilized models for visual recognition such as residual networks (He et al., 2016) or newer models such as vision transformers (Dosovitskiy, 2020) lack such slot-based structure. This challenge has led to a gap between the generic architectures used for visual recognition and slot-based approaches used for learning object-centric representations. In this chapter, we bridge this gap by measuring compositionality in the emergent representations, providing a unified treatment of slot-based and slot-free approaches. We propose a new metric, the Compositional Object Algebra Test (COAT), and associated testing corpus based on the CLEVR (Johnson et al., 2017) domain for measuring object-centric compositionality.

If a representation is compositional with respect to objects, we would expect that the change in a scene's representation when the same input space transformation is applied is consistent across scenes. For example, consider four images A, B, C, and D, where A:B::C:D (Figure 5.1). If two blue objects are added to A to yield B and the same blue objects are added to C (at the same locations) to yield D, then we would expect some analogical structure to be present in the corresponding representations as well. Taking a geometric perspective, if this is true, we would expect $\overrightarrow{AB}$ to be parallel to $\overrightarrow{CD}$.



(a) Weakly Occluded Scene



(b) Strongly Occluded Scene

Figure 5.2: An illustration of *trivial compositionality* vs *object compositionality* and the importance of occlusion for measuring the latter. In (a), compositionality is trivial since an algebra $B - A + C$ in the pixel representation can obtain a D' that is almost the same as the D resulted from the transformation in the semantic space. In (b), compositionality is non-trivial and requires object-centric abstraction. An algebra $B - A + C$ in the pixel space results in a D' that is obviously different from the ground-truth D.

Equivalently, if one can compose D from A, B, and C through translation operations in the latent space, to check their equivariance, it should form a parallelogram. Such emergent properties have previously been studied in the context of word representations (and have indeed emerged without any explicit slot-like structure or specific inductive bias in some cases (Mikolov et al., 2013b)). We demonstrate that one can evaluate object-centric representations against similar geometric structures, by showing how to translate the geometric desiderata into a concretely measurable device.

Interestingly, when measuring transformations across closely related scenes, even a representation such as $q(\mathbf{x}) = \mathbf{x}$ can be trivially compositional. For example, Figure 5.2a portrays a situation in which compositionality is trivially achieved in the pixel space representation. This occurs because there is very little occlusion in this scene and, as such, the representation of objects in pixel-space is inherently independent since the pixels

corresponding to each object are non-overlapping. Thus, when constructing a metric for compositionality, one needs to carefully measure to what extent an approach captures more abstract, non-trivial compositionality. We do so by populating the testing corpus with images like Figure 5.2b with stronger occlusion such that the representations of objects is intertwined even in pixel space, and use $q(\mathbf{x}) = \mathbf{x}$ as a null, trivially compositional representation to contextualize our metric. Any learned representation, in order to exhibit non-trivial compositionality, has to be able to reject the null hypothesis that it is no better than the trivial model in order to receive a COAT score.

In terms of the metric itself, our key technical challenges are centered around how to best measure the extent to which a representation shows an (approximate) parallelogram structure. For instance, some representations might be very closely concentrated in some part of the output feature space, while others may be more spread out. Any comparisons of parallelogram structure must take such global statistics into account in order to be relevant over the course of training a particular model and across different models spanning different design choices (*e.g.*, generative *vs.*contrastive, slot-based *vs.*slot free, *etc.*). Moreover, a representation may short-circuit the parallelogram test by discarding information or not forming object abstraction at all. We address these desiderata by normalizing our metric, correcting for chance and a careful design of hard negative examples to capture compositionality in object-centric representations.

Overall, our contributions are as follows:

- We propose a novel measure, COAT for evaluating compositionality in emergent representations (see Section 5.3.3 for a discussion of the benefits of measuring compositionality over disentanglement).
- We demonstrate the importance of comparing to pixel-space as a null representation when evaluating compositionality (Section 5.4).
- We evaluate a suite of emergent representations, spanning models with various assumptions and inductive biases (Section 5.5).

- As an intriguing negative result, we demonstrate that representations learned by state-of-the-art models for disentangling objects are not as compositional as one might expect, especially with respect to pixel-space compositionality, hinting at the need for further modeling improvements (Table 5.1).

5.2 Related Work

5.2.1 Generative Models of Multi-Object Scenes

Generative modeling of multi-object scenes is a long-standing problem in computer vision (Zhu and Yuille, 1996; Tu and Zhu, 2002). While the previous work focuses on segmenting the pixel space to match the ground-truth, more recent interest has shifted to representation learning — hypothesizing that structural constraints in the representations will facilitate transfer to various visual reasoning (Barrett et al., 2018) or planning tasks (Schölkopf et al., 2021). MONet (Burgess et al., 2019), IODINE (Greff et al., 2019) and Slot Attention (Locatello et al., 2020), among other recent works (Greff et al., 2017; Kim and Mnih, 2018; Engelcke et al., 2020), demonstrate that one can learn disentangled, object-specific slots in latent representations on perceptually simple datasets like CLEVR (Johnson et al., 2017), although progress needs to be made to generalize to more perceptually challenging domains proposed by Karazija et al. (2021). In contrast, our focus is not on perceptual difficulty, but in simply measuring the extent to which compositionality is captured by existing CLEVR-domain models[1].

5.2.2 Disentanglement Metrics

Most of the slot-based object-centric models discussed above decode the slots back to the pixel space, and utilize clustering based approaches for comparing segmentations, such as the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) for evaluation. ARI aims to measure the similarity between the pixel mask for an object predicted by a generative

[1]With a minor modification, namely that we allow different colors for the background

model given a slot with the ground truth mask for that object in the original image, and has a number of desirable properties: it is bounded, normalized and corrected for chance, but it only indirectly measures the representations because a mapping back to the image space is always needed (Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020). In contrast, our metric is decoder-free, does not assume a slot-based structure and measures compositionality instead of disentanglement, broadening its applicability to a much wider range of models and settings. We anticipate this to facilitate more modeling avenues for research on object-centric representation learning.

As opposed to objects, there is another line of work which focuses on disentanglement at the attribute level, and which typically models scenes with single objects (though not always). To evaluate how well factors of variation such as pose, size, shape *etc.*are uncovered, these works adopt approaches like fitting a linear classifier or performing majority vote classifiers on each latent dimension (Higgins et al., 2016; Kim and Mnih, 2018; Chen et al., 2019b), the mean distance between the classification errors of the two latent dimensions that are most predictable (Kumar et al., 2018), mutual information between the representation and the ground truth (Chen et al., 2016), and a dimension-wise entropy reflecting the usefulness of the dimension to predict a single factor to variation (Eastwood and Williams, 2018). In contrast to these methods, our metric does not need a generative model, nor a partitioning of the latent space in terms of individual factors of variation in the latent space[2], nor annotations of all the ground truth factors of variation. Moreover, previous work focuses on attribute-level disentanglement while we are concerned with object-centric, compositional representation learning.

5.2.3 Qualitative Analogical Structures

Our metric comes with an analogy test corpus which is inspired by the pioneering work of Mikolov et al. (2013b); Radford et al. (2016), where they show the learned word embeddings enable simple linear algebra for analogical reasoning. Such a paradigm was

[2]Note this is more challenging for slots in object-centric learning, as opposed to attribute disentanglement where the size of a slot is usually one.

later adopted by Eslami et al. (2018), Ha and Eck (2018) and Achlioptas et al. (2018) for static images, sketches, and 3D point clouds respectively. While these works largely provide qualitative analysis/ visualizations of representations, we aim to quantify the extent to which there exists such an analogical structure in the representations.

5.2.4 Quantifying Compositionality

Andreas (2019) propose a learning based method to identify the extent of compositional structure in any generic, emergent representation, by learning an approximated composition function. While their approach is very generic and broadly applicable compared to ours, our work makes a number of more specific innovations to appropriately measure compositional structure for object-centric representations. Firstly, we note that the metric proposed by Andreas (2019) does not account for pixel-level hard negatives, which is a potential shortcut test constructed by applying the compositional operator in the raw input space as opposed to the latent representation space (which was not a concern for their applications). Further, our metric corrects for representation collapse and by adjusting for chance, which the metric of Andreas (2019) does not account for. That metric may give a high score to a collapsed representation, say $q(\mathbf{x}) = \mathbf{0}$. Finally, Keysers et al. (2019) measure compositional generalization in sequence to sequence tasks by constructing various evaluation tasks with different difficulties for evaluation. In contrast, our evaluation is towards objectness in learned, continuous valued representations rather than the difficulty in compositional generalization entailed by a sequence to sequence task.

5.3 Background

In this section, we make connections between disentanglement representation learning (Locatello et al., 2020; Greff et al., 2019), causal representation learning (Schölkopf et al., 2021), compositionality (Lake and Baroni, 2018), and equivariance (Jayaraman and Grauman, 2015) — notions which underpin both the models we evaluate, as well as

the properties we choose to measure in our COAT metric.

5.3.1 Disentanglement

Attribute-level disentanglement. For attribute-level disentanglement approaches (Higgins et al., 2016), the goal is to recover the ground truth factors of variation used to generate the dataset (*e.g.*, the size, color orientation and shape of 2-D shapes (Higgins et al., 2016)). This is achieved by learning representations using variational autoencoders with a factorized prior $p(\mathbf{z}) = \prod_i p(z_i)$ (Higgins et al., 2016) which indirectly encourages the aggregate posterior $\int_{\mathbf{x}} q(\mathbf{z}|\mathbf{x})p(\mathbf{x})dx$ that generates the representations to be factorized as well (Hoffmann et al., 2018) or by encouraging such behavior more explicitly (Chen et al., 2019b).

Object-Centric disentanglement A parallel line of work focuses on object-centric disentanglement (Greff et al., 2019; Locatello et al., 2020), where instead of attempting to fit all the information corresponding to an entire object into a single, independent latent dimension, one attempts to learn K independent sub-spaces or 'slots', $i.e. \mathbf{z} = [\mathbf{z}_0, \cdots, \mathbf{z}_K]$, with the hope of specializing each object to a slot in the input scene. In addition to this independence assumption used in the latent space, such work also often utilizes functional constraints, which are inspired by the independent causal mechanism (ICM) principle (Schölkopf et al., 2021). Essentially, they start with (1) a pixel-space representation $\mathbf{u}_i \in \mathbb{R}^N$ to disentangle each object separately in the pixel observations $\mathbf{x} \in \mathbb{R}^N$ and (2) a vector of latent abstract causal variables $\mathbf{z}_i \in \mathbb{R}^{\frac{D}{K}}$, with a functional constraint on the decoder, namely that $\mathbf{u}_i = f_i(\mathbf{z}_i, \epsilon)$, where ϵ is a source of noise. The $\mathbf{u}_i$ are then combined with a function g which is often hand-coded to be a mixture of gaussians (Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020) (where the weight of each gaussian is a pixel-level mask, and the predicted means are the pixel values) or a more complex generative process (Eslami et al., 2018; Huang and Murphy, 2015). Learning often proceeds by fitting g, f and an image encoder q jointly, where q maps input images $\mathbf{x}$ to latent representations $\mathbf{z}$. Intuitively, the functional constraint means that the mechanics of

image formation are the same, regardless of the object in question. In addition, the popular Slot Attention (Locatello et al., 2020) also has a strong inductive bias based on self-attention on the encoder which aids emergence of effective slot representations.

Object- *vs.*Attribute- disentanglement Conceptually, popular attribute-level disentanglement models such as β-VAE (Higgins et al., 2016) or β-TC-VAE (Kim and Mnih, 2018) can be thought of as learning statistically independent slots of size 1, with a relaxation of the functional constraint that "object-level" disentanglement models have. As such, it is not *a priori* obvious that so called "attribute-level" disentanglement models would fail when trained on multi-object scenes to uncover object-centric representations. To our knowledge, our work provides the first evaluation to quantitatively measure objectness in such "attribute-level" models, along with an extensive study of the extent to which "object-level" disentanglement models are compositional.

5.3.2 Compositionality

Compositional representations of scenes that enable the flexible addition or removal of objects to/from a scene and the ability to reason with these objects are potentially useful for a large number of applications including counting (Chattopadhyay et al., 2017), reasoning (Barratt and Sharma, 2018), and out of distribution generalization (Lake and Baroni, 2018; Gordon et al., 2019). Beyond these benefits, a compositional representation is conceptually easier to measure, compared to a disentangled representation which often needs access to the ground truth factors of variation, which are not always available in a lot of real world applications. Finally, compositional representations can also aid interpretability (Andreas, 2019).

Disentanglement implies (approximate) compositionality. Moreover, disentanglement is closely related to compositionality. Consider a continuous valued disentangled representation $\mathbf{z}$ where each scalar z_i represents a factor of variation. Let $\mathbf{z}_A$ be the representation of image A and $\mathbf{z}_C$ be the representation of image C. Let z_0 be the dimension in the disentangled feature that corresponds to the size of an object, for in-

stance. Increasing z_0 by a value ϵ should increase the size of the object for both image A as well as C, yielding images B and D respectively. Thus, a translation by ϵ in this case gives us compositionality in terms of size in the input domain. Strictly speaking, one might have to translate by a different amount ϵ and say, δ to obtain the same increase in shape for A and C, in general but the resultant difference vectors are still parallel to each other (Figure 5.1). In our work, we measure both notions of compositionality, by checking for an exact parallelogram structure as well as simply checking if the vectors are parallel (Section 5.4). While we explain attribute level disentanglement for ease of explanation, a similar argument can be made for "slot" based disentanglement with functional constraints (see Section D.1.1 for an informal proof). Moreover, any full rank linear transformation of a disentangled representation (such as a rotation) will still be compositional (while retaining all the information in the original representation) but not disentangled (see Section D.2.1). Thus, exact disentanglement implies perfect compositionality with respect to the ground-truth factors of variation.

Connection to Equivariance. The notion of compositionality we discuss here is an instantiation of group equivariance which has been well studied in the representation learning literature (Jayaraman and Grauman, 2015; Gordon et al., 2019). Specifically, our measure of compositionality is more formally expressed as to translation equivariance in the representation space. Since a slot-based, disentangled representation is compositional with repsect to translations (as discussed above), we measure compositionality with respect to the same translation operation for slot-free models, essentially testing to what extent a slot-free model behaves compositionally as a slot-based model ideally would. However, in principle, one could evaluate with other operations in the latent space or even learn them (Andreas, 2019). Nevertheless, we believe our contributions such as choice of hard negatives, proper normalization *etc.*will prove to be useful regardless of these orthogonal design choices.

5.3.3 Benefits of Measuring Compositionality Over Disentanglement

Overall, measuring compositionality instead of disentanglement for object-centric representation learning has the following benefits:

- Directly evaluates representations (which is what will be used for downstream tasks) without requiring a decoder that maps the representations back to pixel space.
- Allows one to evaluate objectness in representations when slot-based structure is not present.
- Related to the above, one can potentially handle scenes with highly variable numbers of objects in a distributed representation since one does not need to pre-specify the number of object-specific slots.
- While evaluation of slot-based disentanglement requires annotation of all the objects in the scene and their properties, measuring compositionality only requires us to know the relationship between two scenes which is much easier to annotate or obtain from say, videos. More specifically, annotating changes in scenes (in videos) is easier than densely annotating all objects (in images), and temporally close frames could be useful hard negatives.

5.4 Method

Desiderata. While measuring equivariance is necessary for a useful compositional representation, it is not sufficient in practice. We impose the following additional desiderata to identify useful, compositional representations:

1. **Avoiding Shortcuts:** It is possible to "shortcut", for example the analogy test in Figure 5.1 by ignoring the color of the blue objects being added and simply learning a translation operator for "adding two objects" instead of "adding two blue objects". Such a representation would still be compositional, but potentially

not useful for downstream tasks. As discussed in Section 5.1, another shortcut is when one performs no better than say a null model $q(\mathbf{x}) = \mathbf{x}$ in terms of capturing abstract compositional structure. If the model exhibits parallelogram structure, but doesn't capture the relevant information regarding the scene, our metric should penalize the model.

2. **Consistent Blank Slots:** When evaluating a model with say K slots, it is important to have a consistent representation of scenes with any number $k < K$ objects. Specifically, for a scene with k slots, there are $K-k$ "blank slots". One would desire a consistent representation of such slots across different input scenes, otherwise it would be hard to preserve the consistency in the consequence of applying the same vector operation.

3. **Calibration:** Finally, it is a non-trivial question in cases where the parallelogram structure is not exactly followed (which will almost always be the case in practice) how to quantify the *degree* to which A, B, C, D (Figure 5.1) follow the parallelogram structure in order to obtain a measure that is calibrated to give sensible measurements both across training checkpoints and across models.

5.4.1 COAT Measure

The COAT measure comprises three parts: 1) A corpus with carefully designed analogy tests, 2) A binary pass/fail hypothesis test to detect shortcuts to compositionality, 3) a normalized measure reported for the cases where a model passes step 2. We discuss each of these elements below.

Analogy Test. We create a corpus with a set of 1600 analogy / compositionality tests[3], each containing images A, B, C, D (Figure 5.1). Following the observations in Section 5.1 we utilize images with sufficient occlusion (Figure 5.2) where the null represen-

[3]These 1600 test cases are obtained by rejection sampling from 100,000 scenes under a strong occlusion criterion. Moreover, in progressively increasing the test volume n, we find that the COAT score as well as hard-negative tests appear to be stable with respect to the number of samples (n) used to compute them at $n = 1000$.

tation $q(\mathbf{x}) = \mathbf{x}$ is not sufficient to capture compositionality. Finally, in our analogy test we assume that B is related to A, and D is related to C via a given transformation $\texttt{add}(obj_0, obj_1, obj_2, \dots)$ (without loss of generality) which adds, for example, two blue objects to a base scene A and C respectively (Figure 5.1). Denoting $\mathbf{z}_A$ as the representation for scene A, we would like the corresponding representations to satisfy $\mathbf{z}_B - \mathbf{z}_A + \mathbf{z}_C = \mathbf{z}_D$. The degree to which this is satisfied is measured through a loss function $\mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_D)$. We either measure approximate parallelogram structure or the degree to which $\overrightarrow{BA}$ and $\overrightarrow{DC}$ are parallel. For the former, we use $\mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_D) = ||\mathbf{z}_B - \mathbf{z}_A + \mathbf{z}_C - \mathbf{z}_D||_2$; for the latter we use $\mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_D) = acos(\mathbf{z}_B - \mathbf{z}_A, \mathbf{z}_D - \mathbf{z}_C)$. Check Section D.2 for example test cases.

Shortcut Detection. As discussed above, a model might appear equivariant but not be compositional in a manner useful for downstream tasks. To test for this, we employ hard negatives (denoted $D' \neq D$), and compute the losses $\mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_D)$ and $\mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_{D'})$ for a one-sample proportion hypothesis test. Our null hypothesis H_0 is that there is no difference between the two values, *i.e.*, $\texttt{Pr}[\mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_D) < \mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_{D'})] = 0.5$, with the alternative hypothesis being that the hard negatives incur a higher loss, namely, $\texttt{Pr}[\mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_D) < \mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_{D'})] > 0.5$. We use the standard test statistic for the one-sample proportion test with a significance level=0.005 to reject or fail to reject the null hypothesis.

Concretely, we utilize the following hard negatives (D'):

- Object-level: in D' there is one object different from or dropped from D. This tests that the representation captures properties of all objects in the scene, and not just say a subset of objects that are used for the transofrmation Figure 5.1.
- Attribute-level: D' has one object with one attribute (color, material, shape or size) different from D. This tests that the representation is sensitive to the properties of the object, and not just the location (for example).
- Pixel-level: $D' = B - A + C$ in the pixel space. This validates that the evaluated representation is better than a trivial representation $q(\mathbf{x}) = \mathbf{x}$.

If a model fails these tests, COAT will not provide an accurate evaluation of the model's compositionality. Therefore, we only apply the COAT metric to models which can pass the above test. Empirically, we find the hard negative tests for attributes like shape, material and pixel-level representations are most difficult to pass.

Normalization and Correction for Chance.

When comparing across different models, we need to calibrate the metric to ensure that model-specific biases such as differences in the concentration of features do not confound our estimates of whether the structure we desire approximately exists. Denoting by $\hat{D}$ a random image from the minibatch of examples, we use the following normalization and chance correction:

$$\texttt{COAT} = 1 - \frac{\mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_D)}{\mathbb{E}_{\hat{D}}\left[\mathcal{L}(\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_{\hat{D}}\right]}. \tag{5.1}$$

The key idea being that we compute the average loss incurred by a random datapoint to calibrate the extent to which one claims that the desired structure is present. In practice, we use the minibatch size of 64 for calculating $\mathbb{E}_{\hat{D}}[\cdot]$. In terms of the two concrete losses for "perfectness", $L2-$COAT and for "parallelness", $acos-$COAT, we hypothesize for downstream tasks such as counting "perfectness" might be more important (since magnitude of vectors is important) but in other reasoning tasks relaxing it to "parallelness" might be sufficient.

5.4.2 Sanity Checks and Baselines

As a sanity check, we apply the COAT metric to (1) a disentangled vector representation that concatenates the symbolic attributes of all objects and (2) the vector representation of a random full-rank linear projection from (1). Note that (1) is a disentangled representation while (2) is entangled, but they are both compositional by our definition (see Section D.2.1), so they should both obtain a perfect score of 1.0 in our metric. Indeed, this is validated by our experiments. In contrast, disentanglement metric such as training a linear regressor along with a Hungarian matching towards the set of object attribute vectors (Locatello et al., 2020) would rate (1) highly, but not (2), despite the presence

of compositional structure in a different basis set.

We next provide a baseline in terms of the COAT score for the trivial representation $q(\mathbf{x}) = \mathbf{x}$ (*e.g.*, taking pixel-space as our representation). Overall the score for the baseline is 75.47% in terms of L_2 and 36.28% for *acos*, demonstrating that pixel-space is already somewhat compositional. Note that here we do not use the pixel-baseline image as a hard negative for a hypothesis test, but as a trivial representation to compare COAT scores for models against.

5.5 Experiments

Our goal is to identify key modeling choices for object-centric compositionality (Section 5.3) and evaluate them in a unified manner (Section 5.4). From a conceptual standpoint, several factors of variation across models stand out:

1. Whether one has a "slot structure" in the representation in conjunction with functional independence in the decoder (`Slot Struct`)

2. Whether there is an independence prior on the latents or some other mechanism to enforce disentangling via a factorized aggregate posterior distribution (*e.g.*, β-TC-VAE (Chen et al., 2019b)) (`Factorized Prior`)

3. Whether the model is generative or contrastive to isolate if one requires the use of a generative model for learning object-centric representations (`Training Paradigm`)

4. Whether we train the models on an IID dataset or a highly correlated training dataset (`Train Set`) (see Section D.3 for more details).

Table 5.1 (left side) breaks down several models of interest along these axes, *i.e.*β-TC-VAE (Chen et al., 2019b), vanilla Auto-Encoder (Kramer, 1991) which we implement as a β-TC-VAE with $\beta = 0$, Slot Attention (Locatello et al., 2020), IODINE (Greff et al., 2019), and MoCo v2 (He et al., 2020; Karazija et al., 2021) in a matrix of these factors. All

	Slot Struct	Factorized Prior	Training Paradigm	Train Set	ARI (%)	HN L_2	L_2-COAT (%)	HN acos	*acos*-COAT (%)
Pixel baseline (w/ occlusion)	N/A	N/A	N/A	N/A	N/A	N/A	75.47	N/A	36.28
Pixel baseline (w/o occlusion)	N/A	N/A	N/A	N/A	N/A	N/A	97.18	N/A	73.17
Auto-encoder(w/ occlusion)	No	No	Gen	IID	N/A	Fail	-	Fail	-
β-TC-VAE (w/ occlusion)	No	Yes	Gen	IID	N/A	Fail	-	Fail	-
Slot attn (w/ occlusion)	Yes	No	Gen	IID	95.53 ± 1.84	Pass	48.55 ± 14.11	Pass	21.53 ± 10.73
Slot attn* (w/ occlusion)	Yes	No	Gen	IID	95.53 ± 1.84	Pass	60.70 ± 15.55	Pass	31.18 ± 8.01
Slot attn*†(w/ occlusion)	Yes	No	Gen	IID	95.53 ± 1.84	Pass	77.07 ± 0.72	Pass	43.12 ± 0.78
Slot attn*†(w/o occlusion)	Yes	No	Gen	IID	95.53 ± 1.84	Pass	83.84 ± 6.23	Pass	47.45 ± 4.34
Slot attn*†(w/ occlusion)	Yes	No	Gen	CORR	69.12 ± 9.34	Pass	64.82 ± 9.20	Pass	31.95 ± 7.21
IODINE (w/ occlusion)	Yes	Yes	Gen	IID	92.21 ± 0.15	Pass	47.52 ± 0.29	Pass	16.33 ± 0.33
IODINE (w/ occlusion)	Yes	Yes	Gen	CORR	40.08 ± 8.90	Fail	-	Pass	9.16 ± 1.08
MoCo v2 (w/ occlusion)	No	N/A	Con	IID	N/A	Fail	-	Pass	14.05 ± 1.25

Table 5.1: Models, their inductive biases, their training paradigms, their training sets, and their performance on ARI and COAT. "HN" is the Hard Negative Test; models need to pass all hard negative tests to obtain a COAT score, otherwise it is indicated with "-". Since representations directly obtained from slot attention do not perform well on the COAT metric, we also tried some post-processing: * indicates duplication removal, †indicates removing "invisible slots" with zero mask weights. (w/o occlusion) and (w/ occlusion) indicate non-occluded (Figure 5.2a) and strongly occluded (Figure 5.2b) test cases. Statistics are Mean and SEM summarized over 5 random seeds.

models are trained with the default architectures and hyperparameters except that in β-TC-VAE we use latent dimension 256, and use the same encoder for MoCo. We modified CLEVR to include multiple background colors, which forces models to represent the background explicitly for compositional evaluation (see Section D.3 for some examples and an ablation study against the COAT measure).

5.5.1 Autoencoder and Beta-TC-VAE

We generally found that a β-TCVAE as well as a vanilla autoencoder ($\beta = 0$) fail to learn object-centric representations. We sweep over $\beta \in \{0, 1, 2, 3\}$ and generally found a trend that with $\beta = 1, 2$ the models performs the best on the most challenging pixel-level hard negative test, albeit not passing it. Thus, without the "slot" structure or functional independence assumptions it appears that vanilla autoencoders and β-TC-VAE style models do not achieve object-centric compositionality.

A reasonable causal prediction for the vanilla autoencoder is that it neither forms abstraction of objects nor represents them compositionally due to the absence of any

inductive bias. In particular, we may expect its performance to be upper-bounded by the pixel representation. In our experiment, we found an interesting phenomenon that although their L_2/acos metric does not generally surpasses the pixel representation, it is relatively high in the first several epochs (Figure D.5a). But the model certainly does not form object-centric abstraction because it fails the pixel-level hard negative test — the model regards the result of $B - A + C$ as a nearer neighbor than D in the representation space.

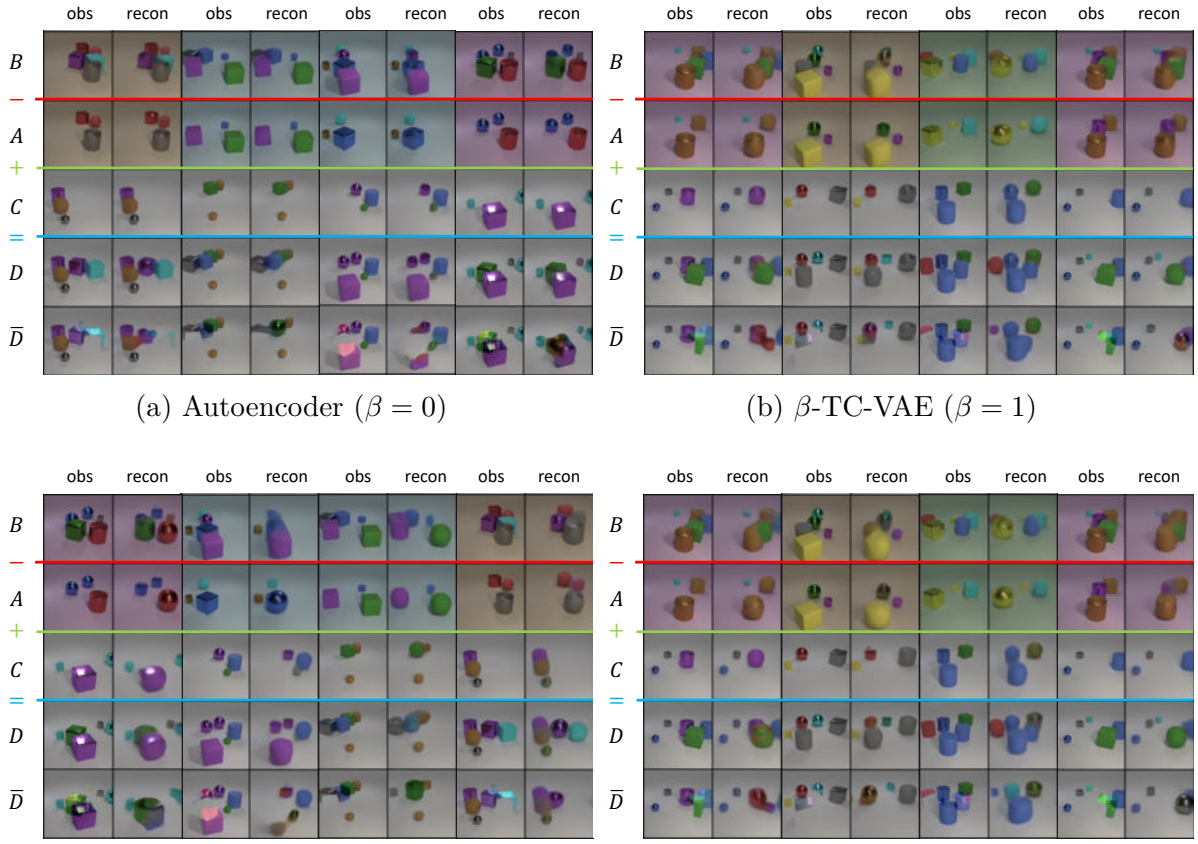
Is a regularization of posteriors towards a factorized prior sufficient for reversing the failure of object-centric abstraction? We apply our metric to β-TC-VAEs, a variant of β-VAE that decompose the KL regularizer to isolate the control of a total Correlation term, with $\beta \in \{1.0, 2.0, 3.0, 4.0\}$. As shown in app, even though these models still fail the pixel-level hard negative test, they show strong inclination towards passing it with larger β. But it is also worth noticing that this inclination can be easily invalidated by the failures on attribute-level hard negative test when $\beta > 3.0$.

Figure 5.3 shows how the learned decoders reconstruct the observation of A, B, C, D and how they generate $\bar{D}$ from $\mathbf{z}_B - \mathbf{z}_A + \mathbf{z}_C$. Comparing $\bar{D}$ against D, we can see lots of artifacts, which qualitatively illustrate the failure of forming object-centric abstractions and compositional representations.

5.5.2 Slot Attention

Next, we study the slot attention model that has a slot structure in the latent space, functional independence in the decoder and a strong inductive bias on the encoder q which infers the slots. This model in previous evaluations using ARI has achieved really strong results and thus given that it is likely "disentangled" one would expect it to also be "compositional" (Section 5.3).

Is slot attention compositional? We align the slots for the same objects across A, B and C, D respectively to facilitate the application of the COAT metric, we perform greedy matching (see Section D.4 for more details). Overall, while slot attention passes the hard-



(a) Autoencoder ($\beta = 0$)

(b) β-TC-VAE ($\beta = 1$)

(c) β-TC-VAE ($\beta = 2$)

(d) β-TC-VAE ($\beta = 3$)

Figure 5.3: Visualization of Object Algebra for Autoencoder and VAEs. Odd columns are obserations A, B, C, D and pixel-level hard negative D'. Even columns are reconstructed A, B, C, D and decoded $\bar{D}$ from $z_B - z_A + z_C$. Interestingly, when $\beta = 0$, the decoded $z_B - z_A + z_C$ looks similar to the pixel-level hard negatives, while $\beta > 0$ gives more natural images in the decoded $z_B - z_A + z_C$.

negative statistical test (Table 5.1), in terms of the COAT score, the vanilla representation from slot-attention does not outperform the trivial pixel-level representation, achieving an L_2-COAT score of 48.55 ± 14.11 as opposed to a pixel-level score of 75.47.

What causes the poor performance of slot-attention? One of the key bottlenecks is that slot attention often assigns two different slots to the same object (Figure 5.4). This redundancy in the latent representation causes difficulties with a proper compositional structure emerging in the latent space. We hypothesize this redundancy is caused by the lack of a sparsity prior on the latent space (Locatello et al., 2020) compared to other models such as (Greff et al., 2019). To account for this when computing COAT, we detect

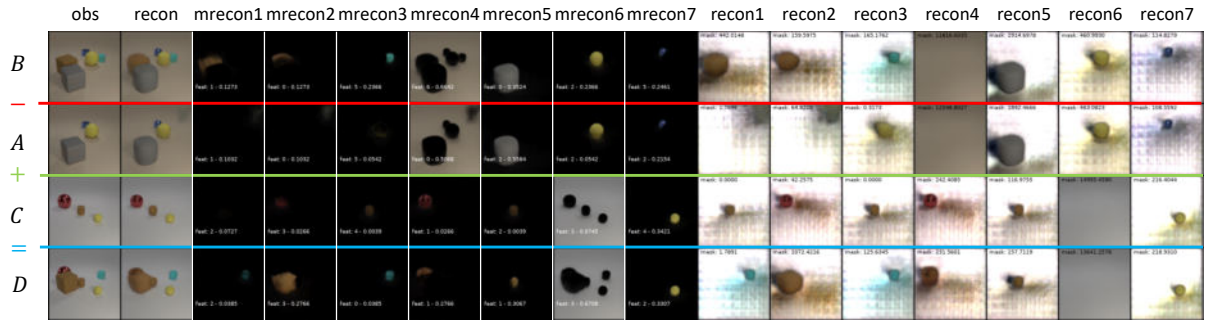


Figure 5.4: Duplicated slots in Slot Attention. Visualization of Slot Attention's observation, combined reconstruction, masked reconstruction and reconstruction of each slot by columns and A, B, C, D by rows. These are after a greedy matching. White captions are index and the cosine similarity to the nearest slots. Black captions are the mask mass.

duplicates by measuring the cosine similarity between different pairs of slots $\mathbf{z}_i \in \mathbb{R}^{\frac{D}{K}}$ and $\mathbf{z}_j \in \mathbb{R}^{\frac{D}{K}}$ and replacing all duplicates with the mean of all other slots. This improves the L_2-COAT score to 60.70 ± 15.15 (Table 5.1), which is still worse than the pixel baseline. This result also indicates that the ARI metric (Locatello et al., 2020) is not sensitive to redundancy in the representations. In essence, it measures recall but not precision of the latent factors, while COAT tests for precision as well as recall.

Another issue is that slot attention does not have a consistent representation of "blank slots" which contain no objects, but instead has a more general notion of "invisible slots" which do not contribute to the reconstruction / generation because they are masked out, but still have some content in them (Figure 5.5). This makes blank slots difficult to detect and standardize without access to masks from the generative decoder. However, utilizing the decoder in this manner does elicit performance on the L_2-COAT metric that surpasses the pixel level baseline (77.02 ± 0.72 *vs.*75.47). Figure 5.6 provides a visualization of the matching, where we can see the imperfect score may be caused by missing objects. This indicates that on it's own, the slot attention model does not exhibit object-centric compositionality without access to the weights of the particular decoder that has been learned in a model run. While this is not an issue for generative modeling, these redundancies and external dependency on the generator might hurt performance of on downstream transfer learning or out-of-distribution generalization tasks where one

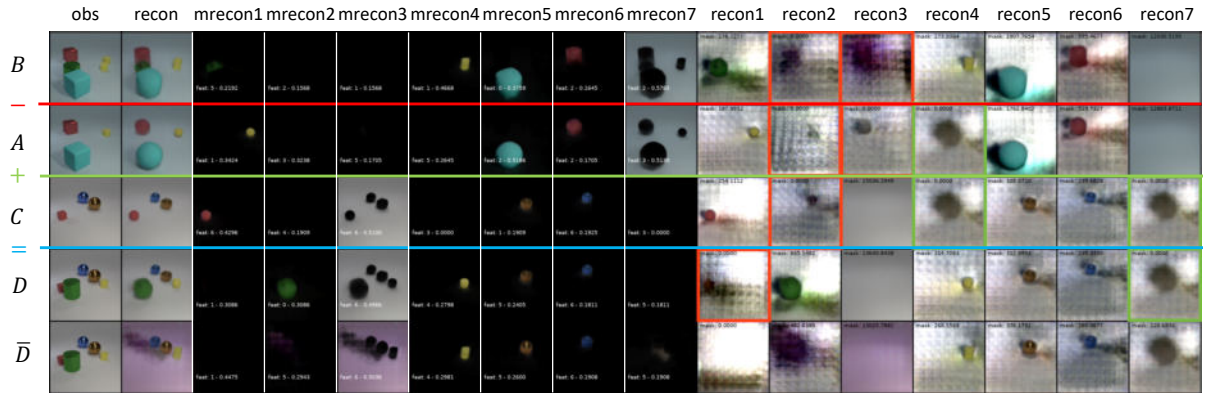


Figure 5.5: Invisible slots in Slot Attention. Visualization of Slot Attention's observation, combined reconstruction, masked reconstruction and reconstruction of each slot by columns and A, B, C, D, decoded $z_A - z_B + z_C$ by rows. These are greedy matching after removing duplicated. White captions are index and cosine similarity to the nearest slots. Black captions are the mask mass. Red boxes highlight "invisible slots", whose mask weights are zero, but apparently have different unmasked reconstructions. This inconsistency may cause the matching to fail. Green boxes highlight "pseudo blank slots", which are designed to be consistent.

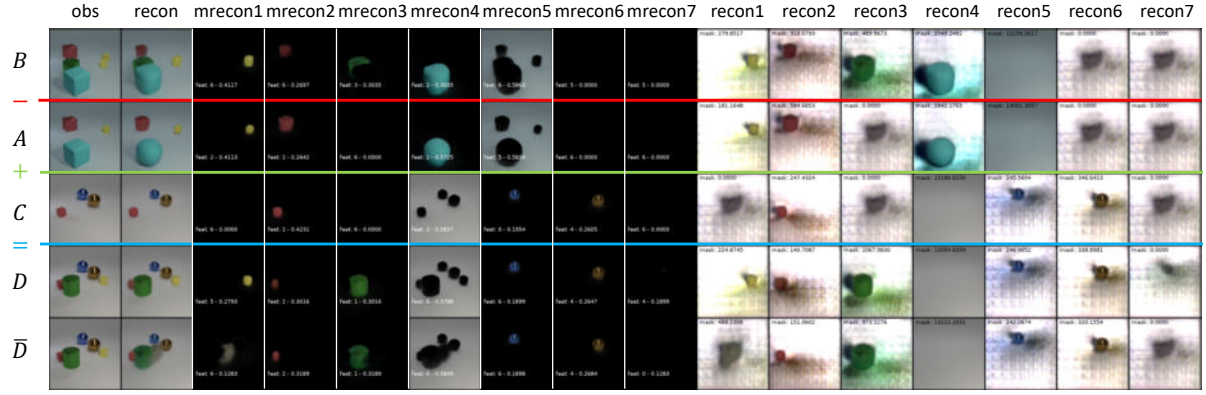


Figure 5.6: Greedy matching to remove invisible slots in Slot Attention. Visualization of Slot Attention's observation, combined reconstruction, masked reconstruction and reconstruction of each slot by columns and A, B, C, D, decoded $\mathbf{z}_A - \mathbf{z}_B + \mathbf{z}_C$ by rows. These are greedy matching after removing duplicated and invisible slots. White captions are index and cosine similarity to the nearest slots. Black captions are the mask mass. The matching is almost perfect, but we can still see the discrepency between 4th and 5th row due to (1) missing a yellow cube (2) uncertainty about the occluded green cylinder in A.

usually does not have privileged access to a generator. Together, our results demonstrate that despite the strong inductive biases present in slot attention models, they do not exhibit improved object-centric compositionality relative to the raw pixel representation, demonstrating both the importance of comparison with the raw pixel baseline for contextualizing compositionality measures and highlighting potential for modeling improvements.

Slot attention is an encoding architecture that iteratively assigns pixels to slots with self-attention. It has gained popularity shortly after its debut, possibly due to its simplicity in inference and success in segmenting the pixel space. If disentanglement is about each subspace being registered with at most one object, the learned representation is certainly disentangled. Then, intuitively, we would also expect them to be compositional by our definition. We apply our metric to slot attention to see if this prediction is true.

One technical issue we have to resolve before we apply our metric is that subspaces are permutation invariant, so we would want to obtain a permutation with the best matching of slots. To this end, we implement a greedy matching that minimizes $||\mathbf{z}_B - \mathbf{z}_A + \mathbf{z}_C - \mathbf{z}_D||_2$ by greedily picking one slot from each image per step without replacement. Here L_2 distance is a natural fit for this greedy matching because it can be computed slot by slot.

Surprisingly, the representation in slot attention obtains a score no higher than the pixel representations, see Table 5.1. A visualization of the reconstruction and masked reconstruction of each slot (Figure 5.4) shows that duplicated slots seem to be the main bottleneck. To better assist slot attention to acquire compositionality by our definition, we implement a post hoc deduplication on its slot representation. Specifically, we use cosine similarity between slots to detect duplicated slots, randomly select one from the duplicates to keep, and replace the others with the mean of all slots as a pseudo "blank slot". Feature similarity is picked over attention similarity because (1) the former is more model-agnostic and (2) the latter correlates poorly with generated masks from our observation.

However, even if we removed duplicated slots, the model still cannot surpass the

pixel representation. After a deeper look into the reconstruction of each slot Figure 5.5, we found this model cannot consistently represent "blank slots". Specifically, there are some emergent "blank slots" with zero-weight masks from the mixture decoder. These slots contain information about arbitrary objects represented in other slots, which are often different in different images. They are not duplicates of others because their cosine similarity to the nearest slots is indistinguishable from the cosine similarity between visible slots. From now on we refer to them as "invisible slots" to (1) highlight their nature of not being "blank" but only "invisible" and (2) differentiate from the pseudo "blank slots" that we manually select. The arbitrary nature of these "invisible slots" make them difficult to detect solely from the latent representation. This indicates that *compositional transformations* are invalidated due to the lack of consistency in "blank slots".

To causally test if these "invisible slots" are the bottleneck of slot attention's performance on our metric, we turn to the decoded masks to detect visibility. Note that the use of mask information should be regarded as cheating in our metric because we aim to directly inspect the latent representation. Here we only leverage this extra information as a certain form of oracle since these masks are independently evaluated to be of high quality according to ARI. After the removal of invisible slots, the slot attention model finally surpasses the pixel representation baseline (Table 5.1). A reasonable claim from this experiment is that it is not that the latent representation in slot attention has non-decodable information of visibility, but rather that the representation of this information does not meet our structural requirements. Figure 5.6 shows the reconstruction of A, B, C, D and the generation of D' from $\mathbf{z}_A - \mathbf{z}_B + \mathbf{z}_C$. We can see that the generation is reasonable, but still not perfect due to some incorrect inference in occluded scenes. Such a failure can be expected because i.i.d. training set does not contain many samples with such strong occlusion.

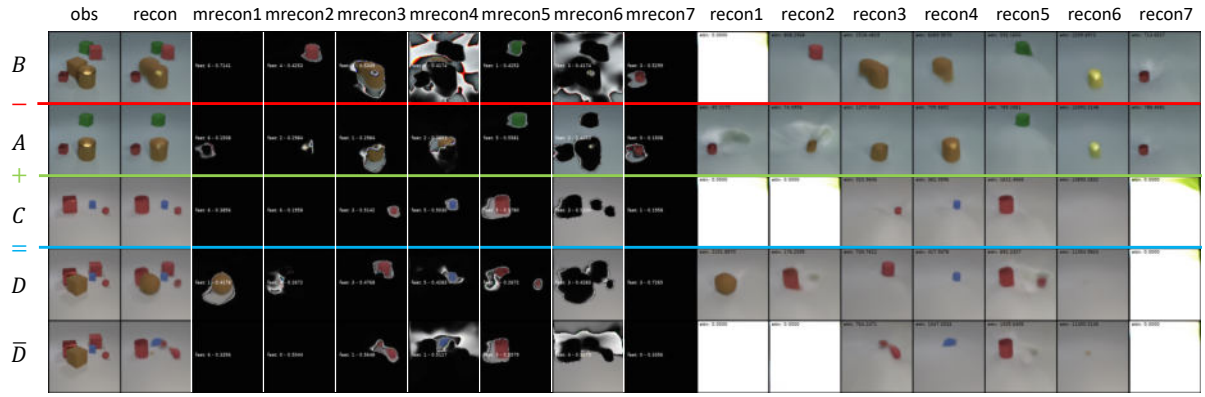


Figure 5.7: Visualization of IODINE's observation, combined reconstruction, masked reconstruction and reconstruction of each slot by columns and A, B, C, D, decoded $z_A - z_B + z_C$ by rows. The emergent "blank slots" are consistent — they are almost white in the unmasked reconstructions — so the matching is good in general. However, the objectness in slots is not consistent. It seems every slot has some background content in the masked reconstruction. Some occluded objects are not disentangled. Some objects are over-segmented into multiple slots and these slots cannot be detected with cosine similarity as duplicates. All these lead to the discrepency between D and decoded $z_A - z_B + z_C$, which may explain the unsatisfying performance of IODINE on our metric.

5.5.3 IODINE

In contrast to slot attention, IODINE (Greff et al., 2019) is a full Bayesian generative model with not only independence constraints in the variational encoder and decoder but also a factorized prior (key difference from slot attention being that the inductive bias in the encoder for computing the slots is more explicitly designed in IODINE instead of using a generic self-attention mechanism). Given the explicit prior, we found that IODINE more consistently represents "blank slots" (Figure 5.7); however, while it passes all hard-negative tests (Table 5.1), it is not able to disentangle foreground objects from background properly (as also noted in the original paper (Greff et al., 2019) and a more recent work (Yu et al., 2021)), which sets it back in terms of the L_2-COAT score (47.52 $\pm$ 0.29). This is more or less equivalent to the vanilla slot attention model 48.55 $\pm$ 14.11 and again substantially lower than the pixel baseline of 75.47.

5.5.4 Visualization of IODINE

Given the large efforts required to make slot attention compositional on our metric, we would expect prior modeling as in IODINE can help resolve the issue of "duplicated slots" and "blank slots". However, it is also worth noticing that in the original paper, the authors mentioned two flaws of IODINE: (1) it segments images without identifying backgrounds, (2) it cannot handle gray-scale images. So in our experiments, we also want to test the prediction that our colored background can help it identify background slots.

As shown in Table 5.1, IODINE does not obtain high score even when the foreground ARI is sufficiently high. Figure 5.7 shows that for some random seeds the model can indeed get the whole background out reasonably well. But these "background slots" often contain a foreground object. There are also random seeds with which each slot still has an arbitrary segments of background. In spite of this failure, we can also see the model does learn consistent "blank slots", thanks to the prior modeling. And as expected, the learned representation can pass all our hard negative tests, among which shape and material seem to be the hardest.

5.5.5 MoCo v2 With ConvNet

Next, we evaluate the MoCo v2 (He et al., 2020) self-supervised learning model trained on our IID set using the same architecture used for the β-TC-VAE. Over the course of training, the model appears to pass the hard-negative test in terms of the *acos*-COAT score (Table 5.1) (which the β-TC-VAE models failed to do) – indicating some initial promise. However, the model is still substantially worse than the corresponding raw pixel baseline (14.05 ± 1.25 *vs*.36.28).

We modify the data augmentation scheme in the original MoCo v2 by (1) removing the jittering in hue and saturation because we don't want the color attribute to collapse in the representation, (2) removing the random horizontal flip and setting the lower bound of random crop ratio to 0.5, this ensures that the two images being compared have almost the same set of objects with identical layout. We train the model to achieve

70%+ accuracy in instance discrimination.

In contrast to autoencoder where the model generally always fails the pixel hard negative test, the self-supervised training signal can encourage the model to form object-centric abstraction gradually. In the acos metric, the model finally passes the pixel hard negative test. Unfortunately, at the time the model passes the pixel hard negative test, it normalized metric also drops to relatively low. This leaves an open question if self-supervised learning may acquire representations with object-centric abstractions and object-level compositionality.

5.5.6 Influence of the Correlated Training Set

Next, we induce correlations between objects in a given scene, to understand how patterns in the data impact different models' ability to learn object-centric compositional representations. Specifically, we generate a set of training images with extremely correlated and cluttered objects that have the same color and material (see Section D.3.2). For slot-attention, even when we utilize the generative model to detect "invisible slots", we still observe a drop in the performance (64.82 ± 9.20 *vs.*77.07 ± 0.72), indicating that the model is not as robust as one might have hoped. However, slot attention is still better than IODINE which fails the pixel-level hard negative test in this setting.[4] This again indicates that the inductive bias used in slot attention based on positional embeddings and self-attention is more robust than those in the encoder of IODINE.

5.6 Summary

In this chapter we presented a new metric, COAT, for measuring object-level compositionality in latent representations. Our metric comprises two parts: (1) a hypothesis that examines the null hypothesis that the evaluated representations do not capture more compositionality than carefully crafted, trivial baselines (*e.g.*, pixel representation), and

[4] Locatello et al. (2020) show a similar comparison to IODINE using the ARI metric in grayscale images.

(2) a quantitative measure that assesses the degree of object compositionality present in the representations with respect to translations in the vector space. We applied COAT to a diverse range of latent object abstraction learning models, encompassing a broad spectrum of modeling assumptions. Somewhat surprisingly, the results showed that state-of-the-art approaches for object-centric disentanglement demonstrated significant limitations. Specifically, these models showed a lack of understanding in the concept of object absence and the unique identity of individual objects. These findings highlight important gaps in current object abstraction learning approaches, suggesting directions for future research.

Part III

Decision

CHAPTER 6

Learning Latent Non-Markovian Decisions From State-Only Sequences

6.1 Introduction

Having explored the symbol-vector duality for *Category* in Part I and the statistical and algebraic structures for *Object* in Part II, we are now well-equipped to investigate the latent abstraction of *Decision* — the fundamental concept underlying the core cognition of agency. While the statistical and algebraic structures of decisions have been extensively studied under the formalization of Markov Decision Process (MDP) and Bellman fixed points, this chapter calls for a re-examination within the context of latent abstractions. Specifically, we pose the following question: *If we formalize "understanding" as learning a generative model that is (1) consistent with observations in the data space, and (2) aligned with human intuition in the latent space, how would this concept of understanding relate to the notion of "value" in decision-making?* By the end of this chapter, we will arrive at an answer that reveals a fundamental connection between the theories of generative modeling and decision-making. To begin with, let's consider the problem setup of imitation from demonstrations.

Imitation of others is a prevalent phenomenon in humans and many other animals, where individuals learn by observing and mimicking the actions of others. An intriguing aspect of this process is the brain's ability to extract motor signals from sensory input. This remarkable capability is facilitated by *mirror neurons* (Di Pellegrino et al., 1992; Rizzolatti et al., 2001), which respond to observations as if the imitator is performing the actions themselves. In conventional imitation learning (Hayes and Demiris, 1994;

Ziebart et al., 2008) and offline reinforcement learning (Levine et al., 2020), action labels have served as proxies for mirror neurons. But it is important to recognize that they are actually productions of human interventions. Given the recent advancements in AI, this is an opportune time to explore imitation learning in a more naturalistic setting.

While the setting of state-only demonstrations is not common, there are certain exceptions. For example, Inverse Reinforcement Learning (IRL) initially formulated the problem as state visitation matching (Ng and Russell, 2000), where demonstrations consist solely of state sequences. Subsequently, this state-only setting was rebranded as Imitation Learning from Observations (ILfO), which introduced the generalized formulation of matching marginal state distributions (Torabi et al., 2018b; Sun et al., 2019). These methods typically rely on Temporal Difference (TD) learning techniques (Sutton and Barto, 2018) under the Markov assumption, and may degrade arbitrarily without the assumption (Singh et al., 1994). One consequence of this assumption, previously believed to be advantageous, is that sequences with different state orders are treated as equivalent. However, the success of general sequence modeling (Brown et al., 2020) has challenged this belief, leading to deep reflections. Notable progress since then includes an analysis of the expressivity of Markov rewards (Abel et al., 2021) and a series of sequence models tailored for decision-making problems (Janner et al., 2021; Chen et al., 2021; Janner et al., 2022; Ajay et al., 2023). Aligning with this evolving trend, we extend the state-only imitation learning problem to encompass non-Markovian domains.

In this chapter, we propose a generative model based on the non-Markov Decision Process (nMDP), in which states are fully observable and actions are latent. Unlike existing monolithic sequence models, we factorize the joint state-action distribution into policy and causal transition according to the standard MDP. To further extend to the non-Markovian domain, we condition the policy on sequential contexts. The density families of policy and transition are consistent with conventional IRL (Ziebart et al., 2008). We refer to this model as the Latent-action non-Markov Decision Process (LanMDP). Because the actions are latent variables following Boltzmann distribution, the present model is closely related to the LEBM (Pang et al., 2020a). To learn the latent policy by

MLE, we need to sample from the prior and the posterior. We sample the prior using short-run MCMC (Nijkamp et al., 2019), and the posterior using importance sampling. Specifically, the proposed importance sampling sidesteps back-propagation through time in posterior MCMC with a single-step lookahead of the Markov transition. The transition is learned from self-interaction.

Once the LanMDP is learned, it can be used for policy execution and planning through prior and posterior sampling, or in other words, *policy as prior, planning as posterior inference* (Attias, 2003; Botvinick and Toussaint, 2012). In our analysis, we derive an objective of the non-Markovian decision-making problem induced from the MLE. We show that the prior sampling at each step can indeed lead to optimal expected returns. Almost surprisingly, we find that the entire family of maximum entropy reinforcement learning (Ziebart et al., 2008; Ziebart, 2010; Fox et al., 2016; Haarnoja et al., 2017, 2018; Levine, 2018) naturally emerges from the algebraic structures in the MLE of latent policies. This formulation avoids the peculiarities of maximizing state transition entropy in prior arts (Ziebart, 2010; Levine, 2018). We also show that when a target goal state is in-distribution, the posterior sampling is optimizing a conditional variant of the objective, realizing model-based planning.

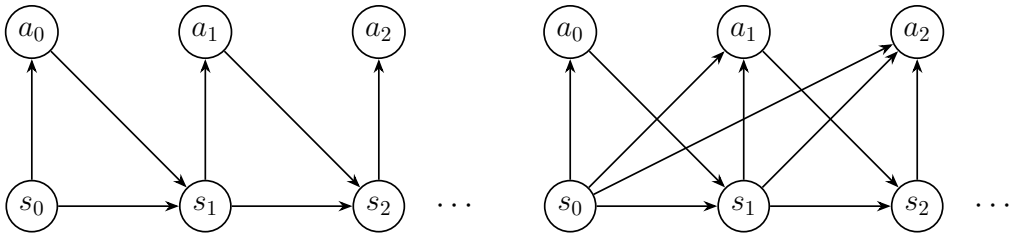
In our experiments, we validate the necessity and efficacy of our model in learning to sequentially plan cubic curves, and illustrate an *over-imitation* phenomenon (Horner and Whiten, 2005; Lyons et al., 2007) when the learned model is repurposed for goal-reaching. We also test the proposed modeling, learning, and computing method in MuJoCo, a domain with higher-dimensional state and action spaces, and achieve performance competitive to existing methods, even those that learn with action labels.

6.2 Non-Markov Decision Process

The most well-known sequence model of a decision-making process is Markov Decision Process. A MDP is a tuple $\boldsymbol{M} = \langle S, A, Tr, R, \rho, T \rangle$ that contains a set S of states, a set A of actions, a transition $Tr : S \times A \mapsto \Pi(S)$ that returns for every state $\mathbf{s}_t$ and action $\mathbf{a}_t$

a distribution over the next state $\mathbf{s}_{t+1}$; a reward function $R : S \times A \mapsto \mathbb{R}$ that specifies the real-valued reward received by the agent when taking action $\mathbf{a}_t$ in state $\mathbf{s}_t$; an initial state distribution $\rho : \Pi(S)$; and a horizon H that is the maximum number of actions/steps the agent can execute in one episode. A solution to an MDP is a policy that maps states to actions, $\pi : S \mapsto \Pi(A)$. The value of policy π, $V^\pi(\mathbf{s}) = \mathbb{E}_{T,\pi}[\sum_{t=0}^{H} R(\mathbf{s}_t)|\mathbf{s}_0 = \mathbf{s}]$ is the expected cumulative reward (*i.e.* return) when executing with this policy starting from state $\mathbf{s}$. The state-action value of policy π is $Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = R(\mathbf{s}_t, \mathbf{a}_t) + \mathbb{E}_{Tr(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}[V^\pi(\mathbf{s}_{t+1})]$. The optimal policy π^* can maximize either $E_{\rho(s_0)}[V^\pi(\mathbf{s}_0)]$, or the same objective plus the policy entropy (Todorov, 2006; Ziebart et al., 2008; Haarnoja et al., 2017). The Markovian assumption supports the convergence of a series of TD-learning methods (Sutton and Barto, 2018), whose reliability in non-Markovian domains is still an open problem (Allen et al., 2024).

A non-Markov Decision Process is also a tuple $\boldsymbol{M} = \langle S, A, Tr, R, \rho, T \rangle$. It generalizes MDP by allowing for non-Markovian transitions and rewards (Brafman and De Giacomo, 2019). Notably, assuming Markovian transition and non-Markovian reward is usually sufficient since a state space with non-Markovian transition can be represented with its Markov abstraction (Ronca et al., 2022). Markov abstraction can be done either by treating the original space as observations generated from the latent belief state in a Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998), or by projecting historic contexts into an embedding space for sequence pattern detection (Hutter, 2009; Toro Icarte et al., 2019; Brafman and De Giacomo, 2019). Presumably, it is statistically more interesting in deep learning to focus our attention on non-Markovian domains where the temporal dependencies in transition and reward differ. Therefore, without loss of generality, we assume that the state transition is Markovian $Tr : S \times A \mapsto \Pi(S)$, while the reward is not (Icarte et al., 2018; Abel et al., 2021), *i.e.* $R : S^+ \mapsto \mathbb{R}$, with S^+ denotes the set of all finite non-empty state sequences with length smaller than T. Obviously, the policy should also be non-Markovian $\pi : S^+ \mapsto \Pi(A)$. See Figure 6.1 for a probabilistic graphical model of the generation process of state sequences from a policy.



(a) MDP (b) nMDP

Figure 6.1: Graphical model of policy and transition in standard Markov Decision Process and non-Markov Decision Process.

6.3 Learning and Sampling

6.3.1 Latent-Action nMDP

A complete trajectory is denoted by

$$\boldsymbol{\tau} = \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{a}_{T-1}, \mathbf{s}_T\}. \tag{6.1}$$

The joint distribution of state and action sequences can be factorized according to the causal assumptions in nMDP:

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) &= p(\mathbf{s}_0)p_{\boldsymbol{\alpha}}(\mathbf{a}_0|\mathbf{s}_0)p_{\boldsymbol{\beta}}(\mathbf{s}_1|\mathbf{s}_0,\mathbf{a}_0)\cdots p_{\boldsymbol{\alpha}}(\mathbf{a}_{T-1}|\mathbf{s}_{0:T-1})p_{\boldsymbol{\beta}}(\mathbf{s}_T|\mathbf{s}_{T-1},\mathbf{a}_{T-1}) \\ &= p(\mathbf{s}_0)\prod_{t=0}^{T-1} p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t), \end{aligned} \tag{6.2}$$

where $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t-1})$ is the policy model with parameter $\boldsymbol{\alpha}$, $p_{\boldsymbol{\beta}}(\mathbf{s}_t|\mathbf{s}_{t-1},\mathbf{a}_{t-1})$ is the transition model with parameter $\boldsymbol{\beta}$, both of which are parameterized with neural networks, $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$. $p(\mathbf{s}_0)$ is the initial state distribution, which can be sampled as a black box.

The density families of policy and transition are consistent with the conventional setting of IRL (Ziebart et al., 2008), where the transition describes the predictable change in state as a single-mode Gaussian, $\mathbf{s}_{t+1} \sim \mathcal{N}(g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t), \sigma^2)$, and the policy accounts for bounded rationality as a Boltzmann distribution with state-action value as the unnormalized energy

$$p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) = \frac{1}{Z(\boldsymbol{\alpha}, \mathbf{s}_{0:t})}\exp\left(f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right), \tag{6.3}$$

where $f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})$ is the negative energy, $Z(\boldsymbol{\alpha}, \mathbf{s}_{0:t}) = \int \exp(f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t}))d\mathbf{a}_t$ is the normalizing constant given the contexts $\mathbf{s}_{0:t}$. We discuss a general push-forward transition in Section E.1.2.

Since we can only observe state sequences, the aforementioned generative model can be understood as a sequential variant of LEBM (Pang et al., 2020a), where the transition serves as the generator and the policy is a history-conditioned latent prior. The marginal distribution of state sequences and the posterior distribution of action sequences are

$$p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T}) = \int p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T}, \mathbf{a}_{0:T-1})d\mathbf{a}_{0:T-1}, \quad p_{\boldsymbol{\theta}}(\mathbf{a}_{0:T-1}|\mathbf{s}_{0:T}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T}, \mathbf{a}_{0:T-1})}{p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T})}. \tag{6.4}$$

6.3.2 Maximum Likelihood Learning

We need to estimate $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$. Suppose we observe *offline* training examples: $\{\boldsymbol{\xi}^i\}, i = 1, 2, \cdots, n, \quad \boldsymbol{\xi}^i = [\mathbf{s}_0^i, \mathbf{s}_1^i, \ldots, \mathbf{s}_T^i]$. The log-likelihood function is

$$L_{off}(\boldsymbol{\theta}) = \sum\nolimits_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\boldsymbol{\xi}^i). \tag{6.5}$$

Denote posterior distribution of action sequence $p_{\boldsymbol{\theta}}(\mathbf{a}_{0:T-1}|\mathbf{s}_{0:T})$ as $p_{\boldsymbol{\theta}}(A|S)$ for convenience where A and S means the complete action and state sequences in a trajectory. The full derivation of the learning method can be found in Section E.1.1, which results in the following gradient:

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\xi) = \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}[\sum\nolimits_{t=0}^{T-1}(\underbrace{\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}_{\text{policy/prior}}, \underbrace{\nabla_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}_{\text{transition}})]. \tag{6.6}$$

Due to the normalizing constant $Z(\boldsymbol{\alpha}, \mathbf{s}_{0:t})$ in the energy-based prior $p_{\boldsymbol{\alpha}}$, the gradient for the policy term involves both posterior and prior samples:

$$\delta_{\boldsymbol{\alpha},t}(S) = \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}\left[\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})\right] = \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right] - \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right], \tag{6.7}$$

where $\delta_{\boldsymbol{\alpha},t}(S)$ denotes the expected gradient of policy term for time step t. Intuition can be gained from the perspective of adversarial training (Finn et al., 2016; Ho and Ermon, 2016): On one hand, the model utilizes action samples from the posterior $p_{\boldsymbol{\theta}}(A|S)$ as pseudo-labels to supervise the unnormalized prior at each step. On the other hand, it discourages action samples directly sampled from the prior. The model converges when prior samples and posterior samples are indistinguishable.

To ensure the transition model's validity, it needs to be grounded in real-world dynamics Tr when jointly learned with the policy. Otherwise, the latent actions that the agent discovers may not align with our understanding. Throughout the training process, we allow the agent to interact with the environment with its actions, periodically collect *on-policy* data $\{(\mathbf{s}_t^i, \mathbf{a}_t^i, \mathbf{s}_{t+1}^i)\}$, $i = 1, 2, \cdots, m$, $t = 1, 2, \cdots, T$ with $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$ and update the transition with a *composite likelihood* (Varin et al., 2011)

$$L_{comp}(\boldsymbol{\beta}) = L_{off}(\boldsymbol{\theta}) + L_{on}(\boldsymbol{\beta}), \quad L_{on}(\boldsymbol{\beta}) = \sum\nolimits_{i=1}^{m} \sum\nolimits_{t=1}^{T} \log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}^i|\mathbf{s}_t^i, \mathbf{a}_t^i). \tag{6.8}$$

6.3.3 Prior and Posterior Sampling

The maximum likelihood estimation requires samples from the prior and the posterior distributions of actions. It would not be a problem if the action space is quantized. However, since we target general latent action learning, we proceed to introduce sampling techniques for continuous actions.

When sampling from a continuous energy space, short-run Langevin dynamics (Nijkamp et al., 2019) can be an efficient choice. For a target distribution $\pi(\mathbf{a})$, Langevin dynamics iterates $\mathbf{a}_{k+1} = \mathbf{a}_k + s\nabla_{\mathbf{a}_k} \log \pi(\mathbf{a}_k) + \sqrt{2s}\boldsymbol{\epsilon}_k$, where k indexes the number of iteration, s is a small step size, and $\boldsymbol{\epsilon}_k$ is the Gaussian white noise. $\pi(a)$ can be either the prior $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$ or the posterior $p_{\boldsymbol{\theta}}(A|S)$. One property of Langevin dynamics that is particularly amenable for EBM is that we can get rid of the normalizing constant. So for each t the iterative update for prior samples is

$$\mathbf{a}_{t,k+1} = \mathbf{a}_{t,k} + s\nabla_{\mathbf{a}_{t,k}} f_{\boldsymbol{\alpha}}(\mathbf{a}_{t,k}.\mathbf{s}_{0:t}) + \sqrt{2s}\boldsymbol{\epsilon}_k. \tag{6.9}$$

Given a state sequence $\mathbf{s}_{0:T}$ from the demonstrations, the posterior samples at each time step $\mathbf{a}_t$ come from the conditional distribution $p(\mathbf{a}_t|\mathbf{s}_{0:T})$. Notice that with Markov transition, we can derive

$$p_{\boldsymbol{\theta}}(\mathbf{a}_{0:T-1}|\mathbf{s}_{0:T}) = \prod_{t=0}^{T-1} p_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_{0:T}) = \prod_{t=0}^{T-1} p_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_{0:t+1}). \tag{6.10}$$

(6.10) reveals that given the previous and the next subsequent state, the posterior can be sampled at each step independently. So the posterior iterative update is

$$\mathbf{a}_{t,k+1} = \mathbf{a}_{t,k} + s\nabla_{\mathbf{a}_{t,k}}(\underbrace{\log p_{\boldsymbol{\alpha}}(\mathbf{a}_{t,k}|\mathbf{s}_{0:t})}_{\text{policy/prior}} + \underbrace{\log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_{t,k})}_{\text{transition}}) + \sqrt{2s}\boldsymbol{\epsilon}_k. \tag{6.11}$$

Intuitively, action samples at each step are updated by back-propagation from its prior energy and a single-step lookahead. While gradients from the transition term are analogous to the inverse dynamics in Behavior Cloning from Observations (BCO) (Torabi et al., 2018a), it may lead to poor training performance due to non-injectiveness in forward dynamics (Zhu et al., 2020).

We develop an alternative posterior sampling method with importance sampling to overcome this challenge. Leveraging the learned transition, we have

$$p_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_{0:t+1}) = \frac{p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}{\mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right]} p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}). \tag{6.12}$$

Let $c(\mathbf{a}_t; \mathbf{s}_{0:t+1}) = \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right]$, posterior sampling from $p_{\boldsymbol{\theta}}(\mathbf{a}_{0:T-1}|\mathbf{s}_{0:T})$ can be realized by adjusting importance weights of independent samples from the prior $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$, in which the estimation of weights involves another prior sampling. In this way, we avoid back-propagating through non-injective dynamics and save some computation overhead.

To train the policy, (6.7) can now be rewritten as

$$\delta_{\boldsymbol{\alpha},t}(S) = \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[\frac{p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}{c(\mathbf{a}_t; \mathbf{s}_{0:t+1})} \nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right] - \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right]. \tag{6.13}$$

Algorithm 5 LanMDP with importance sampling

Input: Learning iterations N, learning rate for energy-based policy $\eta_{\boldsymbol{\alpha}}$, learning rate for transition model $\eta_{\boldsymbol{\beta}}$, initial parameters $\boldsymbol{\theta}_0 = (\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0)$, expert demonstrations $\{\mathbf{s}_{0:H}\}$, context length L, batch size m, number of prior sampling steps K and step sizes s.
Output: $\boldsymbol{\theta}_N = (\boldsymbol{\alpha}_N, \boldsymbol{\beta}_N)$.
Reorganize $\{\mathbf{s}_{0:H}\}$ to to state sequenec segments $(\mathbf{s}_{t-L+1}, \cdots, \mathbf{s}_{t+1})$ with length $L+1$.
Use energy-based policy with $\boldsymbol{\alpha}_0$ collect transitions to fill in the replay buffer.
Use transitions in replay buffer to pre-train transition model $\boldsymbol{\beta}_0$.
for $t = 0$ **to** $N-1$ **do**
  Demo sampling Sample observed examples $(\mathbf{s}_{t-L+1}, \cdots, \mathbf{s}_{t+1})_{i=1}^m$.
  Prior sampling: Sample $\{\hat{\mathbf{a}}_t\}_{i=1}^m$ using (6.9) with K_0 iterations and stepsize s_0.
  Policy learning: Update $\boldsymbol{\alpha}_t$ to $\boldsymbol{\alpha}_{t+1}$ by (6.13) with learning rate $\eta_{\boldsymbol{\alpha}}$.
  Transition learning: Update replay buffer with trajectories from current policy model $\boldsymbol{\alpha}_{t+1}$, then update $\boldsymbol{\beta}_t$ to $\boldsymbol{\beta}_{t+1}$ by (6.8) with learning rate $\eta_{\boldsymbol{\beta}}$.
end for

We present the learning algorithm in Algorithm 5.

6.4 Decision-Making as Inference

In Section 6.3, we present our method within the framework of probabilistic inference, providing a self-contained description. However, from a decision-making perspective, the learned policy may appear arbitrary. In this section, we establish a connection between probabilistic inference and decision-making, contributing a novel analysis that incorporates the latent action setting, the non-Markovian assumption, and maximum likelihood learning. This analysis is inspired by, but distinct from, previous studies on the relationship between these two fields (Todorov, 2008; Ziebart, 2010; Toussaint, 2009; Kappen et al., 2012; Levine, 2018).

6.4.1 Policy Execution With Prior Sampling

Let the ground-truth distribution of demonstrations be $p^*(\mathbf{s}_{0:T})$, and the learned marginal distributions of state sequences be $p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T})$. (6.5) in Section 6.3.2 is an empirical estimate of

$$\mathbb{E}_{p^*(\mathbf{s}_{0:T})}[\log p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T})] = \mathbb{E}_{p^*(\mathbf{s}_0)}\left[\log p^*(\mathbf{s}_0) + \mathbb{E}_{p^*(\mathbf{s}_{1:T}|\mathbf{s}_0)}[\log p_{\boldsymbol{\theta}}(\mathbf{s}_{1:T}|\mathbf{s}_0)]\right]. \tag{6.14}$$

We can show that a sequential decision-making problem can be constructed to maximize the same objective. Our main result is summarized as Theorem 1.

Theorem 1. *Assuming the Markovian transition $p_{\boldsymbol{\beta}^*}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is known, the ground-truth conditional state distribution $p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})$ for demonstration sequences is accessible, we can construct a sequential decision-making problem, based on a reward function $r_{\boldsymbol{\alpha}}(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) := \log \int p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) p_{\boldsymbol{\beta}^*}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) d\mathbf{a}_t$ for an arbitrary energy-based policy $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$. Its objective is*

$$\sum\nolimits_{t=0}^{T} \mathbb{E}_{p^*(\mathbf{s}_{0:t})}[V^{p_{\boldsymbol{\alpha}}}(\mathbf{s}_{0:t})] = \mathbb{E}_{p^*(\mathbf{s}_{0:T})}\left[\sum\nolimits_{t=0}^{T} \sum\nolimits_{k=t}^{T} r_{\boldsymbol{\alpha}}(\mathbf{s}_{k+1}; \mathbf{s}_{0:k})\right],$$

where $V^{p_{\boldsymbol{\alpha}}}(\mathbf{s}_{0:t}) := E_{p^*(\mathbf{s}_{t+1:T}|\mathbf{s}_{0:t})}[\sum_{k=t}^{T} r_{\boldsymbol{\alpha}}(\mathbf{s}_{k+1}; \mathbf{s}_{0:k})]$ is the value function for $p_{\boldsymbol{\alpha}}$. This objective yields the same optimal policy as the Maximum Likelihood Estimation $\mathbb{E}_{p^*(\mathbf{s}_{0:T})}[\log p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T})]$.

If we further define a reward function $r_{\boldsymbol{\alpha}}(\mathbf{s}_{t+1}, \mathbf{a}_t, \mathbf{s}_{0:t}) := r_{\boldsymbol{\alpha}}(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) + \log p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$ to construct a Q function for $p_{\boldsymbol{\alpha}}$

$$Q^{p_{\boldsymbol{\alpha}}}(\mathbf{a}_t; \mathbf{s}_{0:t}) := \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})}\left[r_{\boldsymbol{\alpha}}(\mathbf{s}_{t+1}, \mathbf{a}_t, \mathbf{s}_{0:t}) + V^{p_{\boldsymbol{\alpha}}}(\mathbf{s}_{0:t+1})\right].$$

The expected return of $Q^{p_{\boldsymbol{\alpha}}}(\mathbf{a}_t; \mathbf{s}_{0:t})$ forms an alternative objective

$$\mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}[Q^{p_{\boldsymbol{\alpha}}}(\mathbf{a}_t; \mathbf{s}_{0:t})] = V^{p_{\boldsymbol{\alpha}}}(\mathbf{s}_{0:t}) - \mathcal{H}_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) - \sum\nolimits_{k=t+1}^{T-1} \mathbb{E}_{p^*(\mathbf{s}_{t+1:k}|\mathbf{s}_{0:t})}[\mathcal{H}_{\boldsymbol{\alpha}}(\mathbf{a}_k|\mathbf{s}_{0:k})]$$

that yields the same optimal policy, for which the optimal $Q^*(\mathbf{a}_t; \mathbf{s}_{0:t})$ can be the energy function.

Only under certain conditions, this sequential decision-making problem is solvable through non-Markovian extensions of the maximum entropy reinforcement learning algorithms.

Proof. See Section E.2. □

This theorem offers profound insights. By starting with the hypothesis of latent actions and MLE, and then considering known transition and accessible ground-truth

conditional state distribution, we witness the *automatic emergence* of the entire family of maximum entropy (inverse) RL. This includes prominent algorithms such as soft policy iteration (Ziebart, 2010), soft Q learning (Haarnoja et al., 2017) and soft Actor-Critic (SAC) (Haarnoja et al., 2018). The theorem demonstrates two crucial points: (1) A consistent Maximum Likelihood Estimate for auto-regressive generative modeling is equivalent to a Bellman fixed point of a non-Markov Decision Process. (2) Maximizing intrinsic value is synonymous with achieving consistent understanding. This result bridges the gap between generative modeling and decision-making processes, suggesting a fundamental unity between understanding a behavioral system (through generative modeling) and optimizing values of actions within it (through reinforcement learning).

6.4.2 Model-Based Planning With Posterior Sampling

Lastly, with the learned model, we can do posterior sampling given any complete or incomplete state sequences. The computation involved is analogous to model-based planning. In Section 6.3.3, we introduce posterior sampling with short-run MCMC and importance sampling when we have the target next state, which generalizes all cases where the targets of immediate subsequent states are given. Here we introduce the complementary case, where the goal state $\mathbf{s}_T$ is given as the target.

The posterior of actions given the sequential context $\mathbf{s}_{0:t}$ and a target goal state $\mathbf{s}_T$ is

$$\begin{aligned} & p_{\boldsymbol{\theta}}(\mathbf{a}_{t:T}|\mathbf{s}_{0:t}, \mathbf{s}_T) \propto p_{\boldsymbol{\theta}}(\mathbf{a}_{t:T}, \mathbf{s}_T|\mathbf{s}_{0:t}) \\ = & \int \prod_{k=0}^{T-t-1} \left[p_{\boldsymbol{\beta}}(\mathbf{s}_{t+k+1}|\mathbf{a}_{t+k}, \mathbf{s}_{t+k}) p_{\boldsymbol{\alpha}}(\mathbf{a}_{t+k}|\mathbf{s}_{0:t+k})\right] p_{\boldsymbol{\beta}}(\mathbf{s}_T|\mathbf{a}_{T-1}, \mathbf{s}_{T-1}) d\mathbf{s}_{t+1:T-1}, \end{aligned} \tag{6.15}$$

in which all Gaussian expectations $\mathbb{E}_{p_{\boldsymbol{\beta}}}[\cdot]$ can be approximated with the mean (Kingma and Welling, 2014a). Therefore, $\mathbf{a}_{t:T}$ can be sampled via short-run MCMC with $\nabla_{\mathbf{a}_{t:T}} \log p_{\boldsymbol{\theta}}(\mathbf{a}_{t:T}, \mathbf{s}_T|\mathbf{s}_{0:t})$ back propagated through time. The learned prior can be used to initialize these samples and facilitate the MCMC mixing.

6.5 Experiments

6.5.1 Cubic Curve Generation

To demonstrate the necessity of non-Markovian value and test the efficacy of the proposed model, we designed a motivating experiment. Path planning is a prototypical decision-making problem, in which actions are taken in a 2D space, with the x-y coordinates as states. To simplify the problem without loss of generality, we can further assume x_t to change with constant speed h, such that the action is Δy_t. Obviously, the transition model $(x_{t+1}, y_{t+1}) = (x_t + h, y_t + \Delta y_t)$ is Markovian.

Path planning can have various objectives. Imagining you are a passenger of an autonomous driving vehicle. You would not only care about whether the vehicle reaches the goal without collision but also how comfortable you feel. To obtain comforting smoothness and curvature, consider y is constrained to be a cubic polynomial $F(x) = ax^3 + bx^2 + cx + d$ of x, where (a, b, c, d) are polynomial coefficients. Then the policy for this decision-making problem is non-Markovian.

To see that, suppose we are at (x_t, y_t) at this moment, and the next state should be $(x_t + h, F(x_t + h))$. With Taylor expansion, we know $F(x_t + h) \approx F(x_t) + F'(x_t)h + \frac{F''(x_t)}{2!}h^2 + \frac{F'''(x_t)}{3!}h^3$, so we can have a representation for the policy, $\pi(\Delta y_t | x_t, y_t) = F'(x_t)h + \frac{F''(x_t)}{2!}h^2 + \frac{F'''(x_t)}{3!}h^3$. However, our representation of state only gives us (x_t, y_t), so we will need to estimate those derivatives. This can be done with the finite difference method if we happen to remember the previous states (x_{t-1}, y_{t-1}), ..., (x_{t-3}, y_{t-3}). Taking the highest order derivative for example, $F'''(x_t) = (y_t - 3y_{t-1} + 3y_{t-2} - y_{t-3})/h^3$. It is thus apparent that the policy would not be possibly represented if we are Markovian or don't remember sufficiently many prior states.

This representation of policy is what models should learn through imitation. However, they should not know the polynomial structure a priori. Given a sufficient number of demonstrations with different combinations of polynomial coefficients, models are expected to discover this rule by themselves. This experiment is a minimum vi-

able prototype for general non-Markovian decision-making. It can be easily extended to higher-order and higher-dimensional state sequences.

Setting We employ multi-layer perception (MLP) for this experiment. Demonstrations can be generated by rejection sampling. We constrain the demonstration trajectories to the $(x, y) \in (-1, 1) \times (-1, 1)$ area, and randomly select y and y' at $x = -1$ and $x = 1$. Curves with third-order coefficients less than 1 are rejected. Otherwise, the models may be confused in learning the cubic characteristics.

Non-Markovian dependency and latent energy-based policy are two prominent features of the proposed model. To test the causal role of non-Markovianness, we experiment with context length $\{1, 2, 4, 6\}$. Context length refers to how many prior states the policy is conditioned on. When it is 1, the policy is Markovian. From our analysis above, we know that context length 4 should be the ground truth, which helps categorize context lengths 2 and 6 into insufficient and excessive expressivity. With these four context lengths, we also train Behavior Cloning (BC) models as the control group. In a deterministic environment, there should not be a difference between BC and BCO, as the latter basically employs inverse dynamics to recover action labels. For our model, this simple transition can either be learned or implanted. Empirically, we don't notice a significant difference.

Performance is evaluated both qualitatively and quantitatively. As a 2D planning task, a visualization of the planned curves says a thousand words. In our experiment, we take $h = 0.1$, so the planned paths are rather discretized. We use mean squared error to fit a cubic polynomial and use the residual error as a metric. When calculating the residual error, we exclude those with a third-order coefficient is less than 0.5. Actually, the acceptance rate itself is also a viable metric. It is the number of accepted trajectories divided by the total number of testing trajectories. It is complementary to the residual error because it directly measures the understanding of cubic polynomials.

Results Figure 6.2(a-c) show paths generated with LanMDP after training for 3000 steps. They have context lengths 1, 2, 4 respectively. Compared with demonstrations

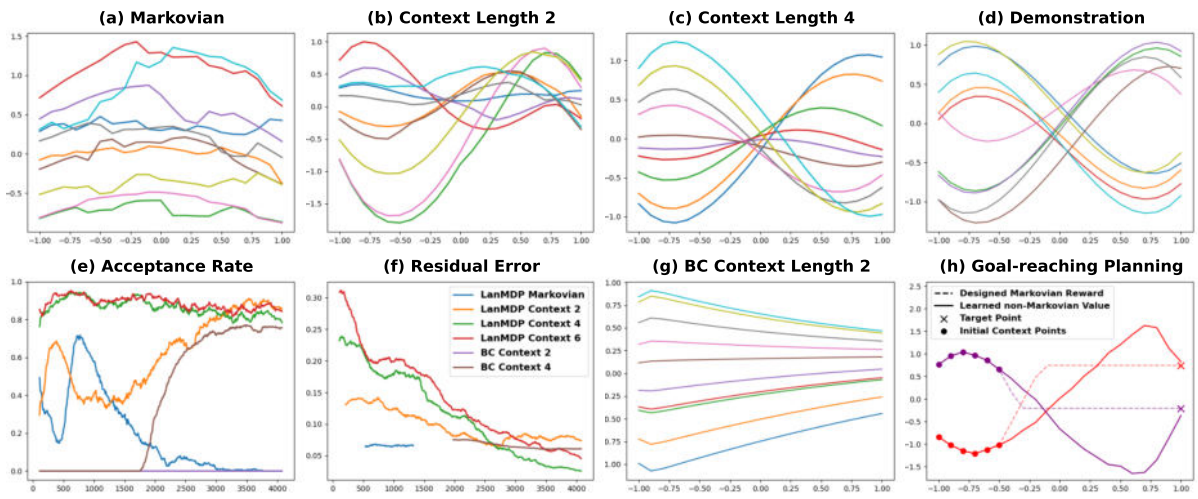


Figure 6.2: Results for cubic curve generation. (a-c) show curves generated at training step 3000 with context lengths 1, 2, 4. Starting points are randomly selected, and all following are sampled from the policy model. Only models with context length 4 learn the cubic characteristic. (d) shows curves from demonstrations. (e) and (f) present the smoothed acceptance rate and fitting residual of trajectories from policies with context lengths 1, 2, 4, 6. The x-axis is the training steps. (e)(f) are better to be viewed together because residual errors will only be calculated if the acceptance rate is above a threshold. For context length 1, the acceptance rate is always zero for BC, so it is not plotted here. (g) shows curves planned by BC with context length 2. It can be compared with (b). Interestingly, LanMDP with context length 2 demonstrates certain cubic characteristics when trained sufficiently long, while the BC counterpart only plans straight lines. (h) is the result of goal-reaching planning, where the dashed line comes from a hand-designed Markov reward, the solid line from the trained LanMDP.

in Figure 6.2(d), only paths from the policy with context length 4 exhibit cubic characteristics. The Markovian policy totally fails this task. But it still generates curves, rather than straight lines from Markovian BC (see Figure E.1). The policy with context length 2 can plan cubic-like curves at times. But some of its generated paths are very different from demonstrations. To investigate this interesting phenomenon, we plot the training curves in Figure 6.2(e)(f). While LanMDP policies with sufficient and excessive expressivity achieve high acceptance rates at the very beginning of the training, policies with Markovian and insufficient expressivity struggle to generate expected curves at the same time. Remarkably, as training goes by, the policy with context length 2, which can only approximate the ground-truth action in the first order, gradually improves in

acceptance rate and residual error. This observation is consistent with Figure 6.2(b).

Continuing our investigation, we plot curves generated by its BC counterparts in Figure 6.2(g) but only see straight lines like the Markovian BC. Therefore, we conjecture that the LanMDP policy with length context 2 leverages its energy-based multi-modality to capture the uncertainty induced by marginalizing part of the necessary contexts. The second-order error in Taylor expansion is possibly remedied by this, especially after long-run training. The Markovian LanMDP policy, however, fails to unlock such potential because it cannot even figure out the first-order derivative.

There are some other note-worthy observations. (i) Excessive expressivity does not impair performance, it just requires more training. As shown in Figure 6.2(e)(f), at the end of training, LanMDP policies with context length 6 perform as well as ones with context length 4. This demonstrates LanMDP's potential in inducing proper state abstraction from sequential contexts. TD learning, however, has been shown to be incapable of such abstraction in a prior work (Ferrer-Mestres et al., 2020). (ii) BC policies with sufficient contexts do not perform as well as LanMDP, as shown in Figure 6.2(e)(f). We conjecture that this might be attributed to the larger compounding error in BC. To shield the influence of compounding errors, we design an experiment where we measure the residual error of the next state after filling the historical contexts in the learned LanMDP context 4 and BC context 4 with expert states, rather than sampled states. The errors are both around 0.0004 for LanMDP and BC, closing the gap in Figure 6.2(f). The implication seems to be LanMDP is more robust to compounding errors than BC.

To verify our analysis in Section 6.4, we visualize the non-Markovian value function defined in Theorem 1 in Figure 6.3. We train a neural network to approximate the non-Markovian value function constructed with the learned policy and transition following Theorem 1, and then visualize the landscape by projecting all history-augmented states to a 2D space with a top view (left) and a front view (middle). Starting from a random initial state, decisions are sequentially made according to the learned policy, leaving a curve in the original state space (right) and a trajectory on the value landscape. It is evident that the non-Markovian value increases monotonically along the trajectory. The

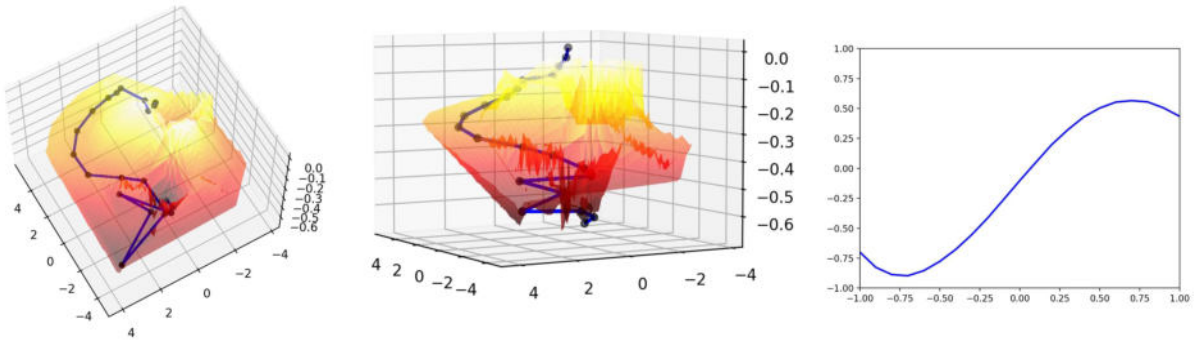


Figure 6.3: Mapping a generated curve to a trajectory in the value landscape. We train a neural network to approximate the non-Markovian value function constructed with the learned policy and transition following Theorem 1, and then visualize the landscape by projecting all history-augmented states to a 2D space with a top view (left) and a front view (middle). Starting from a random initial state, decisions are sequentially made according to the learned policy, leaving a curve in the original state space (right) and a trajectory on the value landscape. Non-Markovian value increases monotonically along the trajectory.

value increases monotonically when the policy generates the cubic curve step by step. In an animation we included on the project homepage[1], we further show that the action sampling at each state yields the highest value in reachable next states.

At last, we study repurposing the learned sequence model for goal-reaching. This is inspired by a surprising phenomenon, over-imitation, from psychology. Over-imitation occurs when imitators copy actions unnecessary for goal-reaching. In a seminal study (Horner and Whiten, 2005), 3- to 4-year-old children and young chimpanzees were presented with a puzzle box containing a hidden treat. An experimenter demonstrated a goal-reaching sequence with both causally necessary and unnecessary actions. When the box was opaque, both chimpanzees and children tended to copy all actions. However, when a transparent box was used such that the causal mechanisms became apparent, chimpanzees omitted unnecessary actions, while human children imitated them. As shown in Figure 6.2(h), planning with the learned non-Markovian value indeed leads to casually unnecessary states, consistent with the demonstrations. Planning with designed Markov rewards produces causally shortest paths.

[1] https://sites.google.com/view/non-markovian-decision-making

6.5.2 MuJoCo Control Tasks

We also report the empirical results of our model and baseline models on MuJoCo control tasks: Cartpole-v1, Reacher-v2, Swimmer-v3, Hopper-v2 and Walker2d-v2. We train an expert for each task using PPO (Schulman et al., 2017). They are then used to generate 10 trajectories for each task as demonstrations. Actions are deleted in the state-only setting.

Setting We conduct a comparative analysis of LanMDP against several established imitation learning baselines including BC (Ross and Bagnell, 2010), BCO (Torabi et al., 2018a), GAIL (Ho and Ermon, 2016), GAIFO (Torabi et al., 2018b), and OPOLO (Zhu et al., 2020). Note that BC and GAIL have access to action labels, positioning them as the control group. The experimental group includes state-only methods such as LanMDP, BCO, GAIFO, and OPOLO. The expert is the idealized baseline. For all tasks, we adopt the MLP architecture for both transition and policy. The input and output dimensions are adapted to the state and action spaces in different tasks, and so are short-run sampling steps. Sequential contexts are extracted from stored episodic memory. The number of neurons in the input and hidden layer in the policy MLP varies according to the context length. We use replay buffers to store the self-interaction experiences for training the transition model offline. See Section E.4 for detailed information on network architectures and hyper-parameters.

Results Results for context length 1 are illustrated through learning curves and a bar plot in Figure 6.4. These learning curves are the average progress across 5 seeds. Scores in the bar plot are normalized relative to the expert score. Our model demonstrates significantly steeper learning curves compared to the state-only GAIFO baselines, especially in Cartpole and Walker2d. This illustrates the remarkable data efficiency of model-based methods. Additionally, LanMDP consistently matches or surpasses the performance of BC and GAIL, despite the latter having access to action labels. In comparison to the expert, LanMDP only lags behind in the most complex Walker2d task. However, it still maintains a noticeable margin over other state-only baselines.

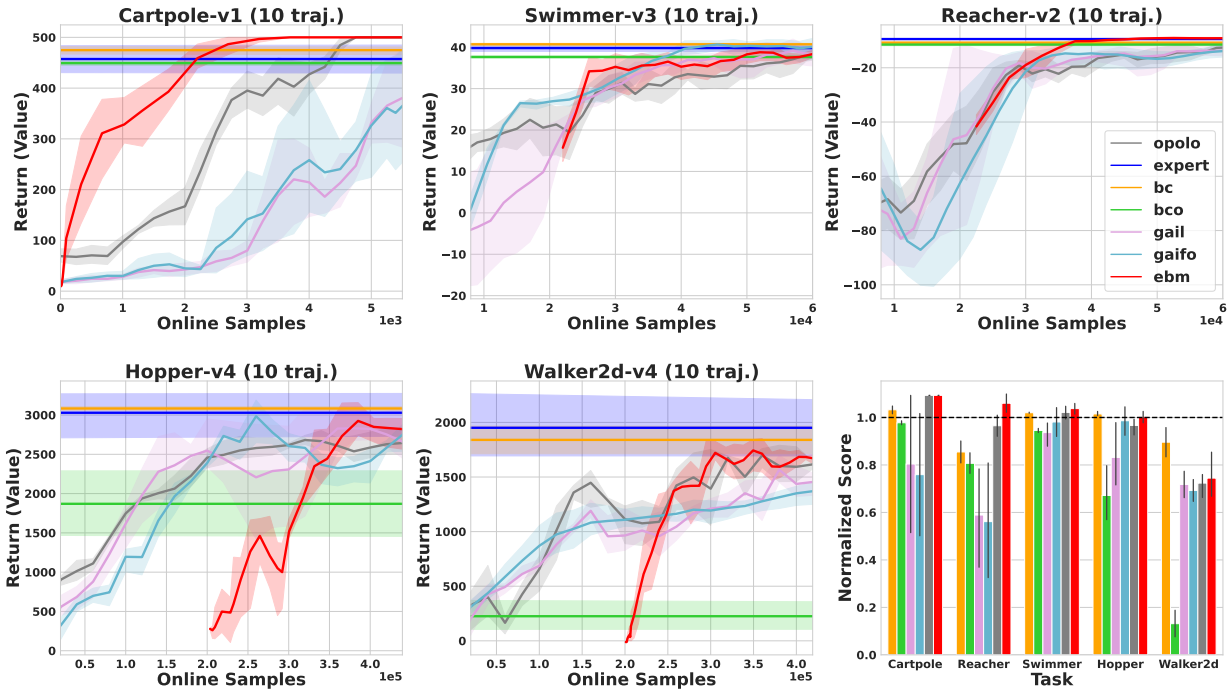


Figure 6.4: Results in MuJoCo for our LanMDP (red), BC (orange), BCO (green), GAIL (purple), GAIFO (cyan), OPOLO (gray), expert (blue). The learning curves are obtained by averaging progress over 5 seeds. We only plot curves for interactive learning methods. The scores of all other methods are plotted as horizontal lines. LanMDP does not have performance scores in the first K steps because this data is collected with random policy to fill the replay buffer, which is then used to train the transition model. $K = 0$ for Cartpole, $2e4$ for Reacher and Swimmer, $2e5$ for Hopper and Walker2d. We include these steps for fair comparisons. LanMDP outperforms existing state-only methods and matches BC, the best-performing state-action counterpart. The bar plot presents the scores from the best-performing policy during the training process, averaged across 5 seeds and normalized with the expert mean. The score in Reacher is offset by a constant before the division of the expert mean to align with the positive scores in all other tasks. The expert mean is plotted as a horizontal line. Our model clearly stands out in state-only methods, while matching and even outperforming those with action labels. Its scores only lag behind the expert mean in the most complex task. Better viewed in color.

Task	context 3	context 2	context 1	BC
CartPole	**500.00±0.00**	**500.00±0.00**	**500.00±0.00**	474.80±18.87
Reacher	-10.91±0.73	-9.70±0.64	-9.00±0.87	**-8.76±0.12**
Swimmer	**42.67±4.66**	**43.52±4.31**	41.22±2.67	38.64±1.76
Hopper	3051.16±111.78	3053.91±176.5	3045.27±240.45	**3083.32±156.61**
Walker2d	1703.02±228.86	**1811.77±369.54**	1753.46±193.69	**1839.94±376.87**

Table 6.1: Comparison between Markovian and non-Markovian policy in MuJoCo. Context length is the number of prior sequential states that the policy depends on, with the current one included. Recall that these MuJoCo tasks are inherently Markovian, thanks to highly specified state features. Nevertheless, non-Markovian policies perform on par with Markovian ones and BC, despite having higher expressivity than sufficient. The best and the second-best results are highlighted. Results are averaged over 5 random seeds.

Task	a dim	s dim	Architecture	10 steps	50 steps
Reacher	2	11	MLP(150;4)	0.0108/0.0076/0.0014	0.0480/0.0350/0.0014
Swimmer	2	8	MLP(150;4)	0.0100/0.0071/0.0014	0.0463/0.0340/0.0014
Hopper	3	12	MLP(512;4)	0.0268/0.0170/0.0074	0.1403/0.0836/0.0073
Walker2d	6	18	MLP(512;4)	0.0282/0.0184/0.0077	0.1487/0.0899/0.0076

Table 6.2: Computational overheads for posterior sampling, importance sampling, and gradient descent (in seconds) in one training step. MLP(n,m) means that we implement the policy model as an MLP with m layers and n hidden neurons each layer. Results are averaged over 10 epochs. The number of MCMC steps is set to 10, 50 respectively. Replacing posterior sampling with importance sampling improves training efficiency.

Results for longer context lengths, *i.e.* the non-Markovian setting, are reported in Table 6.1, in which the highest return across the training process is listed. Originally invented for studying differentiable dynamics, MuJoCo offers state features that are inherently Markovian. Though a MDP is sufficiently expressive, learning a more generalized nMDP does not impair the performance. Sometimes it can even improve a little bit. Due to the limit of time, the maximum context length is only 3. Within the investigated regime, our result is consistent with that reported by Janner et al. (2021). We leave the experiments with longer memory and more sophisticated neural networks to future research.

Table 6.2 is a study of the computational overhead for the sampling techniques involved. The short-run MCMC for posterior inference takes longer than a single step of

gradient descent. Replacing it with the proposed importance sampling improves training efficiency by a large margin.

6.6 Discussion

Related Work in Imitation Learning Earliest works in imitation learning utilized BC (Hayes and Demiris, 1994; Amit and Matari, 2002). When the training data is limited, temporal drifting in trajectories (Atkeson and Schaal, 1997; Argall et al., 2009) may occur, which led to the development of IRL (Ng and Russell, 2000; Abbeel and Ng, 2004; Ratliff et al., 2006; Ziebart et al., 2008; Finn et al., 2016; Ho and Ermon, 2016). In recent years, the availability of abundant sequence/video data is not the primary concern, but rather the difficulty in obtaining action labels. There has since been increasing attention in ILfO (Kidambi et al., 2021; Liu et al., 2018; Torabi et al., 2018b; Zhu et al., 2020; Sun et al., 2019), a setting similar to ours. Distinguished from existing ILfO solutions, our model probabilistically describes the entire trajectory. In particular, the energy-based model (Zhu et al., 1998; LeCun et al., 2006) in the latent policy space (Pang et al., 2020a) has been relatively unexplored. Additionally, the capability for model-based planning is also a novel contribution.

Limitation and Potential Impact The proposed model factorizes the joint distribution of state-action sequences into a time-invariant causal transition and a latent policy modulated by sequential contexts. While this model requires sampling methods, and can be non-negligible for higher-dimensional actions, it is worth noting that action quantization, as employed in transformer-based models (Janner et al., 2021; Chen et al., 2021), has the potential to reduce the computation overhead. In our experiments, a measure of the diversity of behavior is omitted, similar to other works in the literature of reinforcement learning. However, it deserves further investigation since multi-modal density matching is a crucial metric in generative modeling. Importantly, our training objective and analysis are independent of specific modeling and sampling techniques, as long as the state transition remains time-invariant. Given the ability of neural networks

to learn approximate invariance through data augmentation (He et al., 2020; Chen et al., 2020; Laskin et al., 2020; Sinha et al., 2022), we anticipate that our work will inspire novel training and inference techniques for monolithic sequential decision-making models (Janner et al., 2021; Chen et al., 2021; Janner et al., 2022; Ajay et al., 2023).

6.7 Summary

In this chapter, we presented a generative model, LanMDP, in which an auto-regressive energy-based prior of latent actions functions as a policy that interacts with the state transition generator. This model learns by EM-style maximum likelihood estimation. To showcase the problem setup of non-Markovian dependency, we introduced a dedicated experiment, cubic curve generation. LanMDP successfully discovers the non-Markovian contexts and decently expresses its uncertainty when its context window isn't sufficiently long. It demonstrated the robust performance across the MuJoCo suite. More importantly, we revealed the algebraic structure of a Bellman fixed point intrinsic to the presented latent-variable auto-regressive sequence modeling. This finding implies that in terms of the intrinsic value associated with the Bellman fixed point, the pursuit of consistent understanding in a complex behavioral system naturally leads to optimal decision-making strategies, and vice versa.

CHAPTER 7

Planning as Inference of Latent Temporal Abstractions

7.1 Introduction

In the previous chapter, we derived the connection between non-Markov decision-making and auto-regressive generative modeling — they are equivalent at the convergence of their learning objectives. In this chapter, we investigate their relationships without the consistency assumptions in Maximum Likelihood Estimates and Bellman fixed points. The latent abstraction that enables us to build up this new connection is a temporal abstraction of the entire trajectory — a plan. Intuitively, planning becomes essential in tasks without immediate rewards. While existing methods alleviate this challenge with hand-crafted step-wise rewards, we will see how the inference of the learned latent abstractions offers an alternative solution. The problem setup we consider is an extreme case of offline RL (Levine et al., 2020), which is also a generalization of the learning from demonstrations problem that we studied in the previous chapter. Let us start with an introduction of the state-of-the-art generative modeling methods for this problem.

The Decision Transformer (DT) (Chen et al., 2021) and some concurrent work (Janner et al., 2021) have popularized the research agenda of decision-making via generative modeling. The general idea is to consider decision-making as a generative process that takes in a representation of the task objective (*e.g.*, the rewards or returns of a trajectory) and outputs a representation of the trajectory. Intuitively, a purposeful decision-making process should shift the trajectory distribution towards regimes with higher returns. In the classical decision-making literature, this is achieved by two interweaving processes:

policy evaluation and policy improvement (Sutton and Barto, 2018). Policy evaluation promotes consistency in the estimated correlations between the trajectories and the returns. In DT, this is realized by the maximum likelihood estimation (MLE) of the joint distribution of sequences consisting of states, actions, and return-to-gos (RTG). Policy improvement shifts the distribution to improve the status quo expectation of the returns. In DT, this is naturally entailed since the policy is a distribution of actions conditioned on step-wise RTGs.

In this chapter, we are interested in the problem of *planning*. Among various ways to identify *planning* as a special class of decision-making problems, we pay particular attention to its data specification and inductive biases. As designing step-wise rewards requires significant effort and domain expertise, we focus on the problem of learning from trajectory-return pairs, where a trajectory is a sequence of states and actions, and the return is its total rewards. This design choice forces the agents to predict into the long-term future and figure out step-wise credits by themselves. A competitive Temporal Difference (TD) learning baseline, CQL (Kumar et al., 2020), was reported to be fragile under this data specification (Chen et al., 2021).

Our design of inductive biases reflects our intuition of a *plan*. While a policy is a factor of the trajectory distribution, a *plan* is an abstraction lifted from the space of trajectories. As a plan is always made in advance of receiving returns, it implies *significance*, *persistence*, and *contingency*. An agent should plan for more significant returns. It should be persistent in its plan even if the return is assigned in hindsight. It should also be adaptable to the environment's changes during the execution of the plan. We formulate this hierarchy of decision-making with a top-down latent variable model. The latent variable we introduce is effectively a *plan*, for it decouples the trajectory generation from the expected improvement of returns. The autoregressive policy always consults this temporally extended latent variable to be persistent in the plan. The top-down structure enables the agent to disentangle the variations in its plan from the environment's contingencies.

In this chapter, we introduce the Latent Plan Transformer (LPT), a novel genera-

tive model featuring a latent vector modeled by a neural transformation of Gaussian white noise, a Transformer-based policy conditioned on this latent vector and a return estimation model. LPT is learned by maximum likelihood estimation (MLE). Given an expected return, posterior inference of the latent vector in LPT is an explicit process for iterative refinement of the *plan*. The inferred latent variable replaces RTG in the conditioning of the auto-regressive policy, providing richer information about the anticipated future. We further develop a mode-seeking sampling scheme that strongly enforces the temporal consistency for long-range planning, which is particularly effective in *stitch* trajectory; *i.e.*, to compose parts of sub-optimal trajectories to reach far beyond (Fu et al., 2020). LPT demonstrates competitive performance in Gym-Mujoco locomotion, Franka kitchen, goal-reaching tasks in maze2d and antmaze, and a contingent planning task, Connect Four. These empirical results support that latent variable inference can enable and improve planning in the absence of step-wise rewards.

7.2 Background

A sequential decision-making problem can be formulated with a decision process $\langle S, A, T, Tr, r, \rho \rangle$ that contains a set S of states and a set A of actions. Horizon T is the maximum number of steps the agent can execute before the termination of the sequence. We further employ S^+ to denote the set of all non-empty state sequences within the horizon and A^+ for action sequences likewise. $Tr : S^+ \times A^+ \mapsto \Pi(S)$ is the transition that returns a distribution over the next state. $r : S^+ \times A^+ \mapsto \mathbb{R}$ specifies the real-valued reward at each step. $\rho : \Pi(S)$ is the initial state distribution that is always uncontrollable to the agent. The agent's decisions follow a policy $\pi : S^+ \times A^+ \mapsto \Pi(A)$. In each episode, the agent interacts with the transition model to generate a trajectory $\boldsymbol{\tau} = (\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots, \mathbf{s}_T, \mathbf{a}_T)$.

The objective of sequential decision-making is typically formulated as the expected trajectory return $y = \sum_{t=0}^{T} r_t$, $Q = \mathbb{E}_{p(\tau)}[y]$. Conventional RL algorithms solve for a policy $\pi(\mathbf{a}_t|\mathbf{s}_t, *)$, where the conditioning $*$ denotes the optimal expected return. DT generalizes this policy to $\pi(\mathbf{a}_t|\mathbf{s}_{\leq t}, \mathbf{a}_{<t}, RTG_{\leq t})$, by fitting the joint distribution $p(\mathbf{s}_1, \mathbf{a}_1, RTG_1, \ldots, \mathbf{s}_T, \mathbf{a}_T, RTG_T)$

with a Transformer. RTG_t is the return-to-go from step t to the horizon T, $RTG_t = \sum_{k=t}^{T} r(\mathbf{s}_{\leq k}, \mathbf{a}_{\leq k})$. It is a useful indication of future rewards, especially when rewards are dense and informative.

However, RTG becomes less reliable when rewards are sparse or have non-trivial relations with the return. Distributing the return to each step is a credit assignment problem. Consider an example of an ideal credit assignment mechanism: When students receive partial credits for their incomplete answers, it's more fair to give points equal to the full marks minus the expected points for all possible ways to finish the answer, rather than assuming students have no knowledge of the remaining parts. This credit assignment mechanism can be formalized as, $RTG_t^Q = \sum_{k=t}^{K} r(\mathbf{s}_{\leq k}, \mathbf{a}_{\leq k}) + \mathbb{E}[Q(\mathbf{s}_{\leq K}, \mathbf{a}_{\leq K})]$. Here Q can be estimated using deep TD learning with multi-step returns. Yamagata et al. (2023) instantiate a Markovian version and demonstrate improvement in trajectory *stiching*.

Whatever credit assignment we use, be it RTG or RTG^Q, the purpose is to explicitly model the statistical association between trajectory steps and final returns. This effort is believed to be necessary because of the exponential complexity of the trajectory space. This belief, however, can be re-examined given the success of sequence modeling. We explore an alternative design choice by directly associating the latent vector that generates the trajectory with the return.

7.3 Latent Plan Transformer

7.3.1 Model

Given a trajectory $\boldsymbol{\tau}$, $\mathbf{z} \in \mathbb{R}^d$ is the latent vector to represent the variable-length trajectory. $y \in \mathbb{R}$ is the return of the trajectory. The joint distribution of the trajectory and its return is defined as $p(\boldsymbol{\tau}, y)$.

The latent trajectory variable $\mathbf{z}$, conceptualized as a plan, is posited to decouple the autoregressive policy and return estimation. From a statistical standpoint, with $\mathbf{z}$ given, we assume that τ and y are conditionally independent, positioning $\mathbf{z}$ as the information

bottleneck. Under this assumption, the Latent Plan Transformer (LPT) can be defined as,

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y, \mathbf{z}) = p_{\boldsymbol{\alpha}}(\mathbf{z}) p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z}) p_{\boldsymbol{\gamma}}(y|\mathbf{z}), \tag{7.1}$$

where $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$. LPT approximates the data distribution $p_{\text{data}}(\boldsymbol{\tau}, y)$ using the marginal distribution $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y) = \int p_{\boldsymbol{\theta}}(\tau, y, \mathbf{z}) dz$. It also establishes a generation process,

$$\mathbf{z} \sim p_{\boldsymbol{\alpha}}(\mathbf{z}), \quad [\boldsymbol{\tau}|\mathbf{z}] \sim p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z}), \quad [y|\mathbf{z}] \sim p_{\boldsymbol{\gamma}}(y|\mathbf{z}). \tag{7.2}$$

The prior model $p_{\boldsymbol{\alpha}}(\mathbf{z})$ is an implicit generator, defined as a learnable neural transformation of an isotropic Gaussian, $\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)$ and $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. $U_{\boldsymbol{\alpha}}(\cdot)$ is an expressive neural network, such as the UNet (Ronneberger et al., 2015). This approach is inspired by, yet contrasts with that of Pang et al. (2020a), wherein the latent space prior is modeled as an Energy-based Model (EBM). While EBM offers explicit unnormalized density, its sampling process is complex. Conversely, our model provides an implicit density with simpler sampling.

The trajectory generator $p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z})$ is a conditional autoregressive model with finite context K, $p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\beta}}(\boldsymbol{\tau}_{(t)}|\boldsymbol{\tau}_{(t-K)}, \ldots, \boldsymbol{\tau}_{(t-1)}, \mathbf{z})$ where $\boldsymbol{\tau}_{(t)} = (\mathbf{s}_t, \mathbf{a}_t)$. It can be parameterized by a causal Transformer with parameter $\boldsymbol{\beta}$, similar to Decision Transformer (Chen et al., 2021). Specifically, the latent variable $\mathbf{z}$ is included in trajectory generation using cross-attention, as shown in Figure 7.1 and controls each step of the autoregressive trajectory generation as $p_{\boldsymbol{\beta}}(\mathbf{a}_t|\mathbf{s}_{t-K:t}, \mathbf{a}_{t-K:t-1}, \mathbf{z})$. The action is assumed to follow a single-mode Gaussian distribution, i.e. $\mathbf{a}_t \sim \mathcal{N}(g_{\boldsymbol{\beta}}(\mathbf{s}_{t-K:t}, \mathbf{a}_{t-K:t-1}, \mathbf{z}), \mathbf{I}_{|A|})$.

The return predictor is a non-linear regression on the latent variable $\mathbf{z}$, modeled as $p_{\boldsymbol{\gamma}}(y|\mathbf{z}) = \mathcal{N}(r_{\boldsymbol{\gamma}}(\mathbf{z}), \sigma^2)$. It directly predicts the final return from the latent trajectory variable. The function $r_{\boldsymbol{\gamma}}(\mathbf{z})$ is a small multi-layer perceptron (MLP) that estimates y based on $\mathbf{z}$. The variance σ^2, is treated as the hyper-parameter in our setting.

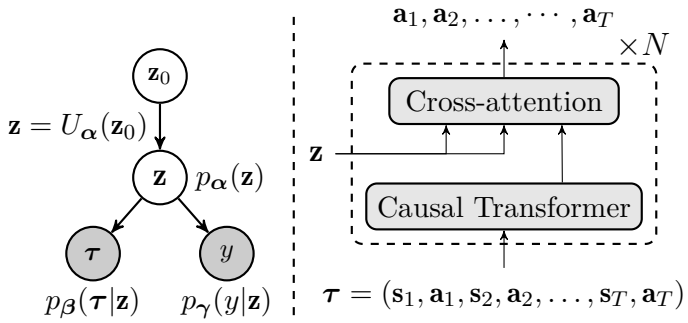


Figure 7.1: *Left*: Graphical model of the LPT. $\mathbf{z} \in \mathbb{R}^d$ is the latent vector. The prior distribution of $\mathbf{z}$ is a neural transformation of $\mathbf{z}_0$, i.e., $\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)$, $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Given $\mathbf{z}$, $\boldsymbol{\tau}$ and y are independent. $p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z})$ is the trajectory generator. $p_{\boldsymbol{\gamma}}(y|\mathbf{z})$ is the return predictor. *Right*: Illustration of trajectory generator $p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z})$.

7.3.2 Offline Learning

With a set of offline training examples $\{(\boldsymbol{\tau}_i, y_i)\}_{i=1}^n$, we aim to learn Latent Plan Transformer (LPT) through maximum likelihood estimation (MLE). The log-likelihood function is defined as $L(\boldsymbol{\theta}) = \sum_{i=1}^n \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, y_i)$. The joint probability of the trajectory and final return is

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y) = \int p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) p_\gamma(y|\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) p_0(\mathbf{z}_0) d\mathbf{z}_0, \tag{7.3}$$

where $p_0(\mathbf{z}_0) = \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. The learning gradient of log-likelihood can be calculated according to

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_0|\tau, y)}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) + \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\gamma}}(y|U_{\boldsymbol{\alpha}}(\mathbf{z}_0))]. \tag{7.4}$$

The full derivation of the learning method is in Section F.1. Let $\delta_{\boldsymbol{\alpha}}, \delta_{\boldsymbol{\beta}}, \delta_\gamma$ represent the expected gradients of $L(\boldsymbol{\theta})$ with respect to the model parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma$, respectively. The learning gradients for each component are formulated as follows.

For the prior model $p_{\boldsymbol{\alpha}}(\mathbf{z})$,

$$\delta_{\boldsymbol{\alpha}}(\boldsymbol{\tau}, y) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_0|\tau, y)}[\nabla_{\boldsymbol{\alpha}}(\log p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z} = U_{\boldsymbol{\alpha}}(z_0)) + \nabla_{\boldsymbol{\alpha}} \log p_\gamma(y|\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0))].$$

For the trajectory generator,

$$\delta_{\boldsymbol{\beta}}(\boldsymbol{\tau}, y) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_0|\boldsymbol{\tau},y)}[\nabla_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0))],$$

For the return predictor,

$$\delta_{\gamma}(\boldsymbol{\tau}, y) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_0|\boldsymbol{\tau},y)}[\nabla_{\boldsymbol{\gamma}} \log p_{\boldsymbol{\gamma}}(y|\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0))].$$

Estimating these expectations requires Markov Chain Monte Carlo (MCMC) sampling of the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}_0|\boldsymbol{\tau}, y)$. We use the Langevin dynamics (Neal, 2011) for MCMC sampling, iterating as follows for a target distribution $\pi(\mathbf{z})$:

$$\mathbf{z}^{k+1} = \mathbf{z}^k + s\nabla_{\mathbf{z}} \log \pi(\mathbf{z}^k) + \sqrt{2s}\boldsymbol{\epsilon}^k, \tag{7.5}$$

where k indexes the time step of the Langevin dynamics, s is the step size, and $\boldsymbol{\epsilon}^k \sim \mathcal{N}(0, \mathbf{I}_d)$ is the Gaussian white noise. Here, $\pi(\mathbf{z})$ is instantiated as the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}_0|\boldsymbol{\tau}, y)$. With $\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)$, we have $p_{\boldsymbol{\theta}}(\mathbf{z}_0|\boldsymbol{\tau}, y) \propto p_0(\mathbf{z}_0)p_{\gamma}(y|\mathbf{z})p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z})$ and the gradient is

$$\nabla_{\mathbf{z}_0} \log p_{\boldsymbol{\theta}}(\mathbf{z}_0|\boldsymbol{\tau}, y) = \nabla_{\mathbf{z}_0} \underbrace{\log p_0(\mathbf{z}_0)}_{\text{prior}} + \nabla_{\mathbf{z}_0} \underbrace{\log p_{\boldsymbol{\gamma}}(y|\mathbf{z})}_{\text{return prediction}} + \underbrace{\sum_{t=1}^{T} \nabla_{\mathbf{z}_0} \log p_{\boldsymbol{\beta}}(\boldsymbol{\tau}_{(t)}|\boldsymbol{\tau}_{(t-K:t-1)}, \mathbf{z})}_{\text{aggregating finite-context sub-trajectories}}.$$

This demonstrates that the posterior inference of $\mathbf{z}$ is an explicit process of optimizing a plan given its likelihood. In the presence of a finite context, $p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z})$ defines sub-trajectories with a maximum length of K. The latent variable $\mathbf{z}$ serves as an abstraction that integrates information from both the final return and K sub-trajectories using gradients.

The sampling process starts by initializing $\mathbf{z}_0^{k=0}$ from a standard normal distribution $\mathcal{N}(0, \mathbf{I}_d)$. We then apply N steps of Langevin dynamics (*e.g.*, $N = 15$) to approximate the posterior distribution, making our learning algorithm an approximate MLE. For a theoretical understanding of this noise-initialized finite-step MCMC, see (Pang et al.,

Algorithm 6 Offline learning of LPT

Input: Learning iterations T, initial parameters $\boldsymbol{\theta}_0 = (\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0, \boldsymbol{\gamma}_0)$, offline training samples $\mathcal{D} = \{\boldsymbol{\tau}_i, y_i\}_{i=1}^n$, posterior sampling step size s, the number of steps N, and the learning rate η_0, η_1, η_2.
Output: $\boldsymbol{\theta}_T$
for $t = 1$ **to** T **do**
1.**Posterior sampling**: For each $(\boldsymbol{\tau}_i, y_i)$, sample $\mathbf{z}_0 \sim p_{\boldsymbol{\theta}_t}(\mathbf{z}_0|\boldsymbol{\tau}_i, y_i)$ using (7.5), where the target distribution π is $p_{\boldsymbol{\theta}_t}(\mathbf{z}_0|\boldsymbol{\tau}_i, y_i)$.
2.**Learn prior model** $p_{\boldsymbol{\alpha}}(\mathbf{z})$, **trajectory generator** $p_{\boldsymbol{\beta}}(\boldsymbol{\tau}|\mathbf{z})$ and **return predictor** $p_\gamma(y|\mathbf{z})$:
$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta_0 \frac{1}{n} \sum_i \delta_{\boldsymbol{\alpha}}(\tau_i, y_i)$,
$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \eta_1 \frac{1}{n} \sum_i \delta_{\boldsymbol{\beta}}(\tau_i, y_i)$,
$\boldsymbol{\gamma}_{t+1} = \boldsymbol{\gamma}_t + \eta_2 \frac{1}{n} \sum_i \delta_\gamma(\tau_i, y_i)$ in Section 7.3.2.
end for

2020a; Nijkamp et al., 2020b). However, for large horizons (*e.g.*, T=1000), this method becomes slow and memory-intensive. To mitigate this, we adopt the persistent Markov Chain (PMC) (Tieleman, 2008; Han et al., 2017), which amortizes sampling across training iterations. During training, $\mathbf{z}_0^{k=0}$ is initialized from the previous iteration and the number of updates is reduced to $N = 2$ steps. The algorithm for offline learning can be found in Algorithm 6. See Section F.2 for training and architecture details.

7.3.3 Planning as Inference

The MLE learning of LPT gives us an agent that can plan. During testing, we first infer the latent $\mathbf{z}_0$ given the desired return y using Bayes' rule,

$$\mathbf{z}_0 \sim p_{\boldsymbol{\theta}}(\mathbf{z}_0|y) \propto p_0(\mathbf{z}_0) p_\gamma(y|\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)). \tag{7.6}$$

This posterior sampling is achieved using Langevin dynamics similar to the training process. Specifically, we replace the target distribution in (7.5) with $p_{\boldsymbol{\theta}}(\mathbf{z}_0|y)$ and run MCMC for a fixed number of steps. Sampling from $p_{\boldsymbol{\theta}}(\mathbf{z}_0|y)$ eliminates the need for expensive back-propagation through the trajectory generator $p_{\boldsymbol{\beta}}(\tau|\mathbf{z})$.

This posterior sampling of $p(\mathbf{z}_0|y)$ is an explicit process that iteratively refines the latent plan $\mathbf{z}$, increasing its likelihood given the desired final return. It aligns with our intu-

ition that planning is an inference process. This inferred $\mathbf{z}$, fixed ahead of the policy execution, effectively serves as a plan. At each step, the agent consults this plan to generate actions conditioned on the current state and recent history, $\mathbf{a}_t \sim p_{\boldsymbol{\beta}}(\mathbf{a}_t|\mathbf{s}_{t-K:t-1}, \mathbf{a}_{t-K:t-1}, \mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0))$.

Once a decision is made, the environment's (possibly non-Markovian) transition $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{a}_t, \mathbf{s}_t)$ emits the next state. This sequential decision-making process iterates the sampling of $\mathbf{s}_t$ and $\mathbf{a}_t$ until termination at the horizon.

Exploitation-Inclined Inference (EI) Inspired by the classifier guidance (CG) (Dhariwal and Nichol, 2021; Ho and Salimans, 2022) in conditional diffusion models, we introduce a guidance weight w to the original posterior in (7.6)

$$\tilde{p}_{\boldsymbol{\theta}}(\mathbf{z}_0|y) \propto p_0(\mathbf{z}_0)p_\gamma(y|\mathbf{z})^w, \mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0), \tag{7.7}$$

which has the score $\nabla_{\mathbf{z}_0} \log \tilde{p}_{\boldsymbol{\theta}}(\mathbf{z}_0|y) = \nabla_{\mathbf{z}_0} \log p_0(\mathbf{z}_0) + w\nabla_{\mathbf{z}_0} \log p_\gamma(y|\mathbf{z})$. This guidance weight w controls the interpolation between exploration and exploitation. When $w = 1$, the sampled plans collectively represent the posterior density and account for Bayesian uncertainty, resulting in a provably efficient exploration scheme (Osband and Van Roy, 2017). When $w > 1$, the sampled plans are more concentrated around the modes of the posterior distribution, which are plans more likely to the agent. The larger the value of w, the more confident the agent becomes, and the stronger the inclination towards exploitation.

The algorithm for planning as inference can be found in Algorithm 7.

7.4 A Sequential Decision-Making Perspective

We approach the sequential decision-making problem with techniques from generative modeling. In particular, our data specification of trajectory-return pairs omits step-wise rewards, based on the belief that the step-wise reward function is only a proxy of the trajectory return. However, step-wise rewards are indispensable input to classical

Algorithm 7 Planning as inference in LPT

Input: Expected return y, a trained model on offline dataset $\boldsymbol{\theta}$, posterior sampling step size s and the number of steps N, Horizon T and an evaluation environment.
Output: $\boldsymbol{\tau}$
//Posterior sampling
if Exploitation-inclined Inference (EI) **then**
Sample $\mathbf{z}_0 \sim \tilde{p}_{\boldsymbol{\theta}}(\mathbf{z}_0|y)$ as in (7.7) using (7.5) where the target distribution is replaced by $\tilde{p}_{\boldsymbol{\theta}}(\mathbf{z}_0|y) \propto p_0(\mathbf{z}_0)p_\gamma(y|\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0))^w$ and let $\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)$.
else
Sample $\mathbf{z}_0 \sim p_{\boldsymbol{\theta}}(\mathbf{z}_0|y)$ as in (7.6) using (7.5) where the target distribution is replaced by $p_{\boldsymbol{\theta}}(\mathbf{z}_0|y) \propto p_0(\mathbf{z}_0)p_\gamma(y|\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0))$ and let $\mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)$.
end if
//Sample trajectory
while current time step $t \leq T$ **do**
Sample $\mathbf{a}_t$ using trajectory generator as $\mathbf{a}_t \sim p_{\boldsymbol{\beta}}(\mathbf{a}_t|\mathbf{s}_{t-K:t-1}, \mathbf{a}_{t-K:t-1}, \mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0))$. Once a decision is made, the environment's (possibly non-Markovian) transition $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{a}_t, \mathbf{s}_t)$ emits the next state.
end while

decision-making algorithms. Accumulating the rewards from the current step to the future gives us the RTG, which naturally hints the future progress of the trajectory. How is temporal consistency enforced in our model without the assistance of the RTGs?

Without loss of generality, consider the trajectory distribution conditioned on a single return value y. The MLE objective is equivalent to minimizing the KL divergence between the data distribution and model distribution, $D_{\mathrm{KL}}(p^y_{\mathcal{D}}(\boldsymbol{\tau})\|p^y_{\boldsymbol{\theta}}(\boldsymbol{\tau}))$. Here, $p_{\mathcal{D}}$ denotes the data distribution and $p_{\boldsymbol{\theta}}$ denotes the model distribution. MLE upon autoregressive modeling imposes additional inductive biases by transforming the objective to $D_{\mathrm{KL}}(p^y_{\mathcal{D},\mathrm{AR}}(\boldsymbol{\tau})\|p^y_{\boldsymbol{\theta},\mathrm{AR}}(\boldsymbol{\tau}))$, which is reduced to next-token prediction for behavior cloning and transition model estimation:

$$\sum_{t=1}^{T} \underbrace{D_{\mathrm{KL}}(p^y_{\mathcal{D}}(\mathbf{a}_t|\mathbf{s}_{1:t},\mathbf{a}_{1:t-1})\|p^y_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_{1:t},\mathbf{a}_{1:t-1}))}_{\text{behavior cloning}} + \sum_{t=1}^{T} \underbrace{D_{\mathrm{KL}}(p^y_{\mathcal{D}}(\mathbf{s}_{t+1}|\mathbf{s}_{1:t},\mathbf{a}_{1:t})\|p^y_{\boldsymbol{\theta}}(\mathbf{s}_{t+1}|\mathbf{s}_{1:t},\mathbf{a}_{1:t}))}_{\text{transition model estimation}}.$$

However, behavior cloning is believed to suffer from drifting errors since it ignores *covariate shifts* in future steps (Ross and Bagnell, 2010). This concern is unique to sequential decision-making, as the agent cannot control the next state from a stochastic environ-

ment, like generating the next text token.

This temporal consistency issue could be alleviated by additionally modeling the sequence of RTG. Denote $\boldsymbol{\rho} = (RTG_0, RTG_1, \dots, RTG_T)$. Modeling the joint distribution is to minimize

$$\begin{aligned} &D_{\mathrm{KL}}(p^y_{\mathcal{D}}(\boldsymbol{\tau}, \boldsymbol{\rho}) \| p^y_{\boldsymbol{\theta}}(\boldsymbol{\tau}, \boldsymbol{\rho})) = D_{\mathrm{KL}}(p^y_{\mathcal{D}}(\boldsymbol{\tau}) \| p^y_{\boldsymbol{\theta}}(\boldsymbol{\tau})) + D_{\mathrm{KL}}(p^y_{\mathcal{D}}(\boldsymbol{\rho}|\boldsymbol{\tau}) \| p^y_{\boldsymbol{\theta}}(\boldsymbol{\rho}|\boldsymbol{\tau})) \\ =&D_{\mathrm{KL}}(p^y_{\mathcal{D},\mathrm{AR}}(\boldsymbol{\tau}) \| p^y_{\boldsymbol{\theta},\mathrm{AR}}(\boldsymbol{\tau})) + \mathbb{E}_{p^y_{\mathcal{D}}(\boldsymbol{\tau})}[\sum\nolimits_{t=1}^{T} \underbrace{D_{\mathrm{KL}}(p^y_{\mathcal{D}}(RTG_t|\boldsymbol{\tau}) \| p^y_{\boldsymbol{\theta}}(RTG_t|\boldsymbol{\tau}))}_{\text{RTG prediction}}]. \end{aligned} \tag{7.8}$$

Note that the *RTG prediction* term is conditioned on the entire trajectory, including the future steps. Minimizing this additional KL divergence correlates predicted RTGs with hindsight trajectory-to-go.

Our modeling of the latent trajectory variable $\mathbf{z}$ provides an alternative solution to the temporal consistency issue. The gradient in (7.4) is minimizing the KL divergence

$$\begin{aligned} &D_{\mathrm{KL}}(p^y_{\mathcal{D}}(\boldsymbol{\tau}, \mathbf{z}) \| p^y_{\boldsymbol{\theta}}(\boldsymbol{\tau}, \mathbf{z})) = D_{\mathrm{KL}}(p^y_{\mathcal{D}}(\boldsymbol{\tau}) \| p^y_{\boldsymbol{\theta}}(\boldsymbol{\tau})) + D_{\mathrm{KL}}(p^y_{\bar{\boldsymbol{\theta}}}(\mathbf{z}|\boldsymbol{\tau}) \| p^y_{\boldsymbol{\theta}}(\mathbf{z}|\boldsymbol{\tau})) \\ =&D_{\mathrm{KL}}(p^y_{\mathcal{D},\mathrm{AR}}(\boldsymbol{\tau}) \| p^y_{\boldsymbol{\theta},\mathrm{AR}}(\boldsymbol{\tau})) + \mathbb{E}_{p^y_{\mathcal{D}}(\boldsymbol{\tau})}[\underbrace{D_{\mathrm{KL}}(p^y_{\bar{\boldsymbol{\theta}}}(\mathbf{z}|\boldsymbol{\tau}) \| p^y_{\boldsymbol{\theta}}(\mathbf{z}|\boldsymbol{\tau}))}_{\text{plan prediction}}], \end{aligned} \tag{7.9}$$

where $p^y_{\bar{\boldsymbol{\theta}}}(\mathbf{z}|\boldsymbol{\tau}) = p^y_{\bar{\boldsymbol{\theta}}}(\boldsymbol{\tau}, \mathbf{z}) / p^y_{\bar{\boldsymbol{\theta}}}(\boldsymbol{\tau})$ and $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}$ highlights these distributions have the same parameterization as $p^y_{\boldsymbol{\theta}}$ but are wrapped with `stop_grad()` operator when calculating gradients for $\boldsymbol{\theta}$ (Han et al., 2017). Comparing (7.8) and (7.9), it is now clear that $\mathbf{z}$ plays a similar role as RTG in promoting temporal consistency in autoregressive models. Uniquely, $p^y_{\bar{\boldsymbol{\theta}}}(\mathbf{z}|\boldsymbol{\tau})$ is the temporal abstraction intrinsic to the model, in contrast to stepwise rewards. From a sequential decision-making perspective, $\mathbf{z}$ is effectively a *plan* that the agent is persistent to. From a generative modeling perspective, $\mathbf{z}$ from different trajectory modes would decompose the density $p^y(\mathbf{a}_t|\mathbf{s}_{0:t}, \mathbf{a}_{0:t-1})$, relieving the burden of learning the autoregressive policy $p_{\boldsymbol{\beta}}(\mathbf{a}_t|\mathbf{s}_{0:t}, \mathbf{a}_{0:t-1}, \mathbf{z})$.

One caveat is that the *transition model estimation* should not be conditioned on y. Mixing up more trajectory regimes could provide additional regularization for its

estimation and generalization. Actually, environment stochasticity is a more concerning issue for autoregressive *behavior cloning*, as highlighted by several authors (Yang et al., 2022; Paster et al., 2022; Štrupl et al., 2022; Brandfonbrener et al., 2022; Villaflor et al., 2022; Eysenbach et al., 2022). Among them, Yang et al. (2022) pinpoint the issue by viewing *RTG*s as deterministic latent trajectory variables, closely related to what we present here. Uniquely, the latent variable $\mathbf{z}$ in our model is inherently multi-modal (hence very non-deterministic) and ignorant of step-wise rewards. We postulate that the overfitting issue can be mitigated. This is validated by our empirical study, which was inspired by the work of Paster et al. (2022).

Although *RTG prediction* and *plan prediction* both promote temporal consistency, they function very differently when mixing trajectories from multiple return-conditioned regimes. *RTG prediction* is a supervised learning over the joint distribution $p_{\mathcal{D}}(\boldsymbol{\tau}, \boldsymbol{\rho})$. Simply mixing trajectories from multiple regimes can't encourage generalization to trajectories that are *stitched* with those in the dataset. Yamagata et al. (2023) propose to resolve this by replacing *RTG* with RTG^Q. Intuitively, this augments the distribution $p_{\mathcal{D}}(\boldsymbol{\tau}, \boldsymbol{\rho})$ with $p_{\mathcal{D}}(\boldsymbol{\tau}', \boldsymbol{\rho}^Q)$, where $\boldsymbol{\tau}'$ denotes trajectories covered by the offline dynamic programming, such as Q learning, and $\boldsymbol{\rho}^Q = (RTG_0^Q, RTG_1^Q, \ldots, Q_T)$. It significantly improves tasks requiring trajectory *stitching*. Conversely, *plan prediction* is an unsupervised learning as it samples from $p_{\mathcal{D}}(\boldsymbol{\tau}, y)p_{\bar{\boldsymbol{\theta}}}(\mathbf{z}|\boldsymbol{\tau}, y)$. As $\mathbf{z}$ contains more trajectory-related information than step-wise *RTG*s, trajectories lying outside of $p_{\mathcal{D}}(\boldsymbol{\tau}, \boldsymbol{\rho})$ may be in-distribution for $p_{\mathcal{D}}(\boldsymbol{\tau}, y)p_{\bar{\boldsymbol{\theta}}}(\mathbf{z}|\boldsymbol{\tau}, y)$. The return prediction training further shapes the representation of $\mathbf{z}$, which can be benefited from denser coverage of y. With more return values covered, we may count on neural networks' strong interpolation capability to shift the trajectory distribution with y-conditioning.

7.5 Related Work

7.5.1 Decision-Making via Sequence Modeling

Chen et al. (2021) propose the Decision Transformer (DT), pioneering this paradigm shift. Concurrently, Janner et al. (2021) explore beam search upon the learned Transformer for model-based planning and inspired later work that searches over the latent state space (Zhang et al., 2022). Lee et al. (2022) report DT's capability in multi-task setting. Zheng et al. (2022) explore the online extension of DT. Yamagata et al. (2023) augment the Monte Carlo RTG in DT with a Q function and show improvement in tasks requiring trajectory *stitching*. Janner et al. (2022) explore diffusion models (Ho et al., 2020) as an alternative generative model family for decision-making. Our model differentiates from all above in data specification and model formulation.

7.5.2 Latent Trajectory Variables in Behavior Cloning

Yang et al. (2022) and Paster et al. (2022) investigate the DT's overfitting to environment contingencies and propose latent variable solutions. Our model is closely related to theirs but unique in an EM-style algorithm for MLE. Ajay et al. (2021) and Lynch et al. (2020) propose latent variable models to make Markovian policies temporally extended. Their models are more related to VAE (Kingma and Welling, 2014a).

7.5.3 Offline Reinforcement Learning

Since the offline static datasets only partially cover the state transition spaces, efforts from a conventional RL perspective focus on imposing pessimistic biases to value iteration (Kumar et al., 2020; Kostrikov et al., 2021; Uehara and Sun, 2021; Xie et al., 2021; Cheng et al., 2022). Fujimoto and Gu (2021) show that simply augmenting value-based methods with behavior cloning achieves impressive performance. Emmons et al. (2021) report that supervised learning on return-conditioned policies is competitive to value-based methods in offline RL. Our MLE objective is more related to the supervised learning methods. The

latent variable inference further imposes temporal consistency, acting as a replacement of value iteration.

7.5.4 Hierarchical RL

Methods like OPAL (Ajay et al., 2021), OPOSM (Freed et al., 2023) address TD-learning's limitations in long-range credit assignment using a two-stage approach: discovering skills from shorter subsequences to reduce the planning horizon, then applying skill-level CQL or online model-based planning on the reduced horizons. This chapter focuses on comparing various methods for long-range credit assignment on the original horizon. Future work includes first discovering skills and then modeling them with a skill-level LPT to further extend the effective horizon.

7.6 Experiments

The data specification of trajectory-return pairs distinguishes our empirical study from most existing works in offline RL. Omitting step-wise rewards naturally increases the challenges in decision-making.

7.6.1 Overview

Our empirical study adopts the convention from offline RL. We first train our model with the offline data and then test it as an agent in the corresponding task. More training details and ablation studies of LPT can be found in Section F.2 and Section F.3.

OpenAI Gym-Mujoco The D4RL offline RL dataset (Fu et al., 2020) features densely-rewarded locomotion tasks including *Halfcheetah, Hopper*, and *Walker2D*. We test for *medium* and *medium-replay*. It also includes *Antmaze*, a locomotion and goal-reaching task with extremely sparse reward. The agent will only receive a reward of 1 if hitting the target location and 0 otherwise. We use its *umaze* and *umaze-diverse* variants.

Franka Kitchen Franka Kitchen is a multitask environment where a Franka robot

with nine degrees of freedom operates within a kitchen setting, interacting with household objects to achieve specific configurations. Our experiments focus on two datasets of the environment: *mixed*, and *partial*, which consists of non-task-directed demonstrations and partially task-directed demonstrations respectively.

Maze2D Maze2D is a navigation task in which the agent reaches a fixed goal location from random starting positions. The agent is rewarded 1 point when it is around the goal. Experiments are conducted on three layouts: *umaze, medium*, and *large*, with increasing complexity. The training data of the Maze2D task contains only suboptimal trajectories from and to randomly selected locations.

Connect Four This is a tile-based game, where the agent plays against a stochastic opponent (Paster et al., 2022), receiving at the end of an episode 1 reward for winning, 0 for a draw, and -1 for losing.

Baselines We compare the performance of LPT with several representative baselines including CQL (Kumar et al., 2020), DT (Chen et al., 2021) and QDT (Yamagata et al., 2023). CQL baseline results are obtained from (Kumar et al., 2020). QDT baseline results are from (Yamagata et al., 2023). The DT results for Gym-Mujoco and Maze2D tasks are from (Yamagata et al., 2023), Antmaze from (Zheng et al., 2022), and Kitchen implemented based on the published source code. CQL and DT results in the Connect Four experiments are from (Paster et al., 2022). The mean and standard deviation of our model, shown as LPT and LPT-EI, are reported over 5 seeds.

7.6.2 Credit Assignment

When resolving the temporal consistency issue, our model doesn't have an explicit credit assignment mechanism that accounts for the actual contribution of each step. It is not aware of the step-wise rewards either. We are therefore curious about whether the inferred latent variable $\mathbf{z}$ can effectively assign fair credits to resolve compounding errors.

Distributing Sparse Rewards to High-Dimensional Actions The Gym-Mujoco environment was a standard testbed for high-dimensional continuous control during the

Dataset	Step-wise Reward			Final Return				
	CQL	DT	QDT	CQL	DT	QDT	LPT (Ours)	LPT-EI (Ours)
halfcheetah-medium	**44.4**	42.1	42.3	1.0	42.4	42.4	43.13 ± 0.38	**43.53** ± 0.08
halfcheetah-medium-replay	**46.2**	34.1	35.6	7.8	33.0	32.8	39.64 ± 0.83	**40.66** ± 0.12
hopper-medium	58.0	60.3	66.5	23.3	57.3	50.7	58.52 ± 1.92	**63.83** ± 1.47
hopper-medium-replay	48.6	63.7	52.1	7.7	50.8	38.7	82.29 ± 1.26	**89.93** ± 0.61
walker2d-medium	79.2	73.3	67.1	0.0	69.9	63.7	77.85 ± 3.18	**81.15** ± 0.33
walker2d-medium-replay	26.7	60.2	58.2	3.2	51.6	29.6	72.31 ± 1.92	**75.68** ± 0.34
kitchen-mixed	51.0	22.3	-	-	17.2	-	61.9 ± 1.22	**64.7** ± 0.51
kitchen-partial	49.8	20.4	-	-	10.5	-	61.2 ± 1.75	**65.3** ± 0.62

Table 7.1: Evaluation results of offline OpenAI Gym MuJoCo tasks. We provide results for data specification with step-wise reward (left) and final return (right). **Bold** highlighting indicates top scores. LPT outperforms all final-return baselines and most step-wise-reward baselines.

development of modern RL algorithms (Lillicrap et al., 2015). In this environment, step-wise rewards were believed to be critical for TD learning methods. In the setup of offline RL, Chen et al. (2021) reported the failure of the competitive CQL baseline when delaying step-wise rewards until the end of the trajectories. DT and QDT are reported to be robust to this alternation. As shown in Table 7.1, the proposed model, LPT, outperforms these baselines in the same data specification. Notably, LPT even excels in most of the control tasks when compared with the baselines with step-wise rewards.

Distributing Delayed Rewards to Long-Range Sequences Maze navigation tasks with fully delayed rewards align with our intuition of a planning problem, for it involves decision-making at certain critical states absent of instantaneous feedback. An ideal planner would take in the expected total return and calculate the sequential decisions, automatically distributing credits from the extremely sparse and fully delayed rewards. According to Yamagata et al. (2023), DT fails in these tasks. Our proposed model LPT outperforms QDT by a large margin in all three variants of the maze task, as shown in Table 7.2. Table 7.3 reports the performance in a locomotion maze task, Ant-Maze, where LPT also achieves the highest. These results validate our hypothesis that the additional plan prediction KL imposes temporal consistency on autoregressive policies.

Dataset	CQL	DT	QDT	LPT	LPT-EI
Maze2D-umaze	5.7	31.0 ± 21.3	57.3 ± 8.2	65.43 ± 2.91	$\mathbf{70.57} \pm 1.39$
Maze2D-medium	5.0	8.2 ± 4.4	13.3 ± 5.6	20.62 ± 1.81	$\mathbf{26.66} \pm 0.74$
Maze2D-large	12.5	2.3 ± 0.9	31.0 ± 19.8	37.21 ± 2.05	$\mathbf{45.89} \pm 2.98$

Table 7.2: Evaluation results of Maze2D tasks. **Bold** highlighting indicates top scores.

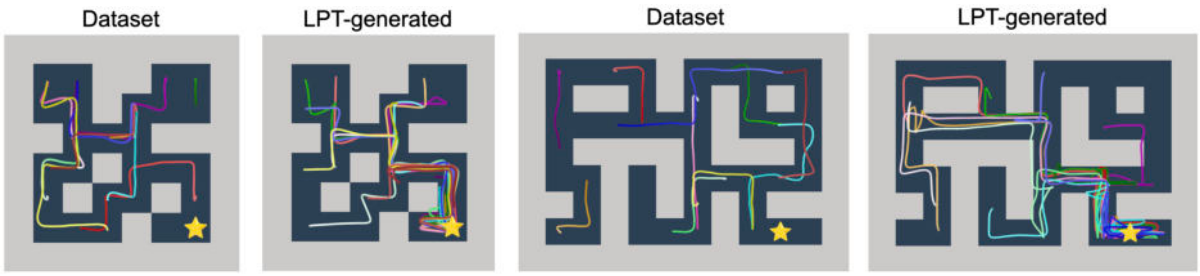
Dataset	CQL	DT	LPT	LPT-EI
Antmaze-umaze	74.0	53.3 ± 5.52	80.8 ± 4.83	$\mathbf{92.4} \pm 0.80$
Antmaze-umaze-diverse	84.0	52.5 ± 5.89	78.5 ± 1.66	$\mathbf{84.4} \pm 1.96$

Table 7.3: Evaluation results of Antmaze tasks. **Bold** highlighting indicates top scores.

7.6.3 Trajectory Stitching

In addition to credit assignment, the setup of offline RL further presents a challenge, trajectory *stitching* (Fu et al., 2020), which articulates the problem of shifting the trajectory distribution towards sparsely covered regimes with higher returns. In the Franka Kitchen environment, both the *mixed*, and *partial* datasets contain undirected data where the robot executes subtasks that do not necessarily achieve the goal configuration. The "mixed" dataset contains no complete solution trajectories, necessitating that the agent learn to piece together relevant sub-trajectories. A similar setting happens in Maze2D domain. Taking Maze2D-medium as an example, in the training set, the average return of all trajectories is 3.98 with a standard deviation of 10.44, where the max return is 47. DT's score is only marginally above the average return. Yamagata et al. (2023) attribute DT's failure in Maze2D to its difficulty with trajectory *stitching*.

Figure 7.2 visualizes samples from the training data and successful trajectories in testing. The left panels show that trajectories in training are suboptimal in terms of (1) being short in length and (2) containing very few goal-reaching instances. Trajectories on the right are generated by 10 random runs with LPT, where the agent successfully navigates to the end goal from random starting positions in an effective manner. This indicates that the agent can discover the correlation between different ys to facilitate such stitching.



(a) Maze2d-Medium (b) Maze2d-Large

Figure 7.2: (a) Maze2D-medium environment (b) Maze2D-large environment. Left panels show example trajectories from the training set and right panels show LPT generations. Yellow stars represent the goal states.

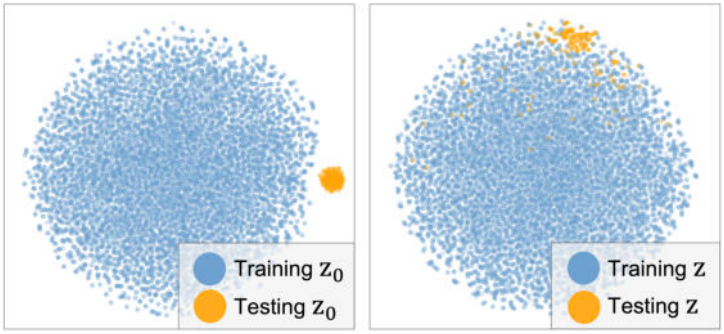


Figure 7.3: t-SNE visualization of latent variable in Maze2D-medium. Left: the distribution of $\mathbf{z}_0$, following an isotropic Gaussian. $\mathbf{z}_0$s from testing are disjoint from the training population. Right: the distribution of $z = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)$. $\mathbf{z}$s from the generated trajectories become "in-distribution."

To probe into the agent's understanding of trajectories' returns, we visualize the representation space of the latent variables. The left of Figure 7.3 is the distribution of $\mathbf{z}_0$. It follows an isotropic Gaussian, in which distance can roughly imply density. We can see that $\mathbf{z}_0$ from the generated trajectories are distant away from the training population. The agent understands they are not very likely in the training set. The right of Figure 7.3 is the distribution of $\mathbf{z}$, which is transformed from $\mathbf{z}_0$ with the UNet. We observe that $\mathbf{z}$s from the generated trajectories become "in-distribution" in the sense that some of them are mingled into the training population and the remaining lie inside a region coverable through linear interpolation of training samples. The agent understands what trajectories to generate even if they are unlikely among what it has seen.

Dataset	CQL	DT	ESPER	LPT
Connect Four	0.61 ± 0.05	0.8 ± 0.07	$\mathbf{0.99} \pm 0.03$	$\mathbf{0.99} \pm 0.01$

Table 7.4: Evaluation results on Connect Four. **Bold** highlighting indicates top scores.

7.6.4 Environment Contingencies

To live in a stochastic world, contingent planning that is adaptable to unforeseen noises is desirable. Paster et al. (2022); Yang et al. (2022) discover that DT's performance would degrade in stochastic environments due to inevitable overfitting towards contingencies. We examine LPT and other baselines in Connect Four from (Paster et al., 2022). Connect Four is a two-player game, where the opponent will make adversarial moves to deliberately disturb an agent's plan. According to the empirical study by Paster et al. (2022), the degradation of DT is more significant than in stochastic Gym tasks from (Yang et al., 2022). As shown in Table 7.4, LPT achieves the highest score with minimal variance. The ESPER baseline is from (Paster et al., 2022), which is very relevant to LPT as it is also a latent variable model. ESPER learns the latent variable model with an adversarial loss. It further adds a clustering loss in the latent space. LPT's on-par performance may justify that MLE upon a more flexible prior can play an equal role.

7.7 Limitation

An interesting future direction is to study LPT's continual learning potential. During planning, LPT explores with provably efficient posterior sampling (Osband et al., 2013; Osband and Van Roy, 2017). We include online learning experiments in Section F.4. Despite significance in some tasks, most improvements are within 1 standard deviation of the mean, similar to ODT (Zheng et al., 2022).

7.8 Summary

In this chapter, we presented a latent-variable auto-regressive model, LPT, which generates trajectories and returns from the latent attractions of plans. In learning, posterior sampling of the latent variables naturally gathers sub-trajectories to form an episode-wise abstraction despite finite context windows. In inference, the posterior sampling given the target final return explores the optimal regime of the latent space, producing a latent variable that guides the autoregressive policy to execute consistently. Across diverse evaluations, LPT demonstrates competitive capacities of nuanced credit assignments, trajectory stitching, and adaptation to environmental contingencies. Our analysis further showcased the connection between explicit inference of the latent temporal abstractions and step-wise credit assignment in terms of promoting temporal consistency.

Given the architectural similarity between our trajectory generator and state-of-the-art Large Language Models (LLMs) (Brown et al., 2020; Team et al., 2023), we anticipate that the advantages of latent abstraction learning and inference can be directly generalized to massive-scale Generative AI systems. This generalization has the potential to challenge the current paradigms of LLM development, which typically rely on pretraining with Maximum Likelihood Estimation followed by fine-tuning with Reinforcement Learning. Our findings suggest a more integrated approach that may potentially revolutionize the way we develop and optimize these powerful AI systems.

Part IV

Knowledge Distillation

CHAPTER 8

Distilling Data-Space Diffusion Models to Latent-Variable Models

8.1 Introduction

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020b) are another family of generative models, which can directly model the data-space distribution to enable the generation of high-quality images (Ramesh et al., 2022; Saharia et al., 2022; Rombach et al., 2022), videos (Ho et al., 2022; Brooks et al., 2024), and other modalities (Poole et al., 2022; Xu et al., 2022; Yang et al., 2023). Diffusion models use a forward process to create a sequence of distributions that transform the complex data distribution into a Gaussian distribution, and learn the score function for each of these intermediate distributions. Sampling from a diffusion model reverses this forward process to create data from random noise by solving an SDE, or an equivalent probability flow ODE (Song et al., 2020a). Typically, solving this differential equation requires a significant number of evaluations of the score function, resulting in a high computational cost. Reducing this cost to single-function evaluation as the generators in latent-variable models would enable applications in real-time generation.

To enable efficient sampling from diffusion models, two distinct approaches have emerged: (1) trajectory distillation methods (Salimans and Ho, 2022; Berthelot et al., 2023; Song et al., 2023; Gu et al., 2023; Zheng et al., 2023; Heek et al., 2024) that accelerate solving the differential equation, and (2) distribution matching approaches (Xiao et al., 2021; Xu et al., 2023; Sauer et al., 2023; Luo et al., 2023c; Yin et al., 2024) that learn implicit generators to match the marginals learned by the diffusion model. Trajec-

tory distillation-based approaches have greatly reduced the number of steps required to produce samples, but continue to face challenges in the 1-step generation regime. Distribution matching approaches can enable the use of arbitrary generators and produce more compelling results in the 1-step regime, but often fail to capture the full distribution due to the *mode-seeking* nature of the divergences they minimize.

In this chapter, we introduce EM Distillation (EMD), a diffusion distillation method that minimizes an approximation of the *mode-covering* divergence between a pre-trained diffusion teacher model and a latent-variable student model. The student enables efficient generation by mapping from noise to data in just one step. To achieve MLE of the marginal teacher distribution for the student, we propose a method similar to the EM framework (Dempster et al., 1977), which alternates between an Expectation-step (E-step) that estimates the learning gradients with Monte Carlo samples, and a Maximization-step (M-step) that updates the student through gradient ascent. As the target distribution is represented by the pre-trained score function, the E-step in the original EM that first samples a datapoint and then infers its implied latent variable would be expensive. We introduce an alternative MCMC sampling scheme that jointly updates the data and latent pairs initialized from student samples, and develop a reparameterized approach that simplifies hyperparameter tuning and improves performance for short-run MCMC (Nijkamp et al., 2019). For the optimization in the M-step given these joint samples, we discover a tractable linear noise term in the learning gradient, whose removal significantly reduces variances. Additionally, we identify a connection to Variational Score Distillation (Poole et al., 2022; Wang et al., 2024) and Diff-Instruct (Luo et al., 2023c), and show how the strength of the MCMC sampling scheme can interpolate between mode-seeking and mode-covering divergences. Empirically, we first demonstrate that a special case of EMD, which is equivalent to the Diff-Instruct (Luo et al., 2023c) baseline, can be readily scaled and improved to achieve strong performance. We further show that the general formulation of EMD that leverages multi-step MCMC can achieve even more competitive results. For ImageNet-64 and ImageNet-128 conditional generation, EMD outperforms existing one-step generation approaches with FID scores of 2.20

and 6.0. EMD also performs favorably on one-step text-to-image generation by distilling from Stable Diffusion models.

8.2 Preliminary

8.2.1 Diffusion Models and Score Matching

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020), also known as score-based generative models (Song and Ermon, 2019; Song et al., 2020b), consist of a forward process that gradually injects noise to the data distribution and a reverse process that progressively denoises the observations to recover the original data distribution $p_{\text{data}}(\mathbf{x}_0)$. This results in a sequence of noise levels $t \in (0, 1]$ with conditional distributions $q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\alpha_t \mathbf{x}_0, \sigma_t^2 \boldsymbol{I})$, whose marginals are $q_t(\mathbf{x}_t)$. We use a variance-preserving forward process (Song et al., 2020b; Kingma et al., 2021; Kingma and Gao, 2023) such that $\sigma_t^2 = 1 - \alpha_t^2$. Song et al. (2020b) showed that the reverse process can be simulated with a reverse-time Stochastic Differential Equation (SDE) that depends only on the time-dependent score function $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ of the marginal distribution of the noisy observations. This score function can be estimated by a neural network $s_{\boldsymbol{\phi}}(\mathbf{x}_t, t)$ through (weighted) denoising score matching (Hyvärinen and Dayan, 2005; Vincent, 2011):

$$\mathcal{J}(\boldsymbol{\phi}) = \mathbb{E}_{p_{data}(\mathbf{x}_0), p(t), q_t(\mathbf{x}_t|\mathbf{x}_0)} \left[w(t) \| s_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0) \|_2^2 \right], \tag{8.1}$$

where $w(t)$ is the weighting function and $p(t)$ is the noise schedule.

8.2.2 MCMC With Langevin Dynamics

While solving the reverse-time SDE results in a sampling process that traverses noise levels, simulating Langevin dynamics (Neal, 2011) results in a sampler that converges to and remains at the data manifold of a target distribution. As a particularly useful MCMC sampling method for continuous random variables, Langevin dynamics generate

samples from a target distribution $\rho(\mathbf{x})$ by iterating through

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \gamma \nabla_{\mathbf{x}} \log \rho(\mathbf{x}^i) + \sqrt{2\gamma}\mathbf{n}, \tag{8.2}$$

where γ is the stepsize, $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and i indexes the sampling timestep. Langevin dynamics has been widely adopted for sampling from diffusion models (Song and Ermon, 2019; Song et al., 2020b) and energy-based models (Xie et al., 2016, 2018; Gao et al., 2018; Nijkamp et al., 2021). Convergence of Langevin dynamics requires a large number of sampling steps, especially for high-dimensional data. In practice, short-run variants with early termination have been succesfully used for learning of EBMs (Nijkamp et al., 2019, 2020a; Gao et al., 2020).

8.2.3 Maximum Likelihood and Expectation-Maximization

Expectation-Maximization (EM) (Dempster et al., 1977) is a maximum likelihood estimation framework to learn latent variable models: $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, such that the marginal distribution $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})d\mathbf{z}$ approximates the target distribution $q(\mathbf{x})$. It originates from the generic training objective of *maximizing* the log-likelihood function over parameters: $\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x})]$, which is equivalent to *minimizing* the *forward* KL divergence $D_{\mathrm{KL}}(q(\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{x}))$ (Bishop, 2006). Since the marginal distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$ is usually analytically intractable, EM involves an E-step that expresses the gradients over the model parameters $\boldsymbol{\theta}$ with an expectation formula

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q(\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x})] = \mathbb{E}_{q(\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})], \tag{8.3}$$

where $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$ is the posterior distribution of $\mathbf{z}$ given $\mathbf{x}$. See Section G.1 for a detailed derivation. The expectation can be approximated by Monte Carlo samples drawn from the posterior using, *e.g.*, MCMC sampling techniques. The estimated gradients are then used in an M-step to optimize the parameters. Han et al. (2017) learned generator networks with an instantiation of this EM framework where E-steps leverage Langevin

dynamics for drawing samples.

8.2.4 Variational Score Distillation and Diff-Instruct

Our method is also closely related to Score Distillation Sampling (SDS) (Poole et al., 2022), Variational Score Distillation (VSD) (Wang et al., 2024) and Diff-Instruct (Luo et al., 2023c), which have been used for distilling diffusion models into a single-step generator (Yin et al., 2024; Nguyen and Tran, 2023). The generator produces clean images $\mathbf{x}_0 = g_{\boldsymbol{\theta}}(\mathbf{z})$ with $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and can be diffused to noise level t to form a latent variable model $p_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{z}) = p_{\boldsymbol{\theta},t}(\mathbf{x}_t|\mathbf{z})p(\mathbf{z})$, $p_{\boldsymbol{\theta},t}(\mathbf{x}_t|\mathbf{z}) = \mathcal{N}(\alpha_t g_{\boldsymbol{\theta}}(\mathbf{z}), \sigma_t^2 \boldsymbol{I})$. This model is trained to match the marginal distributions $p_{\boldsymbol{\theta},t}(\mathbf{x}_t)$ and $q_t(\mathbf{x}_t)$ by minimizing their *reverse* KL divergence. Integrating over all noise levels, the objective is to *minimize* $\mathcal{J}(\boldsymbol{\theta})$, where

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{p(t)}[\tilde{w}(t) D_{\mathrm{KL}}(p_{\boldsymbol{\theta},t}(\mathbf{x}_t)||q_t(\mathbf{x}_t))] = \mathbb{E}_{p(t)}\left[\tilde{w}(t) \int p_{\boldsymbol{\theta},t}(\mathbf{x}_t) \log \frac{p_{\boldsymbol{\theta},t}(\mathbf{x}_t)}{q_t(\mathbf{x}_t)} d\mathbf{x}_t\right]. \quad (8.4)$$

The gradient for this objective in (8.4) can be written as

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{p(t),p(\epsilon),p(\mathbf{z})}[-\tilde{w}(t)(\underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)}_{\text{teacher score}} - \underbrace{\nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta},t}(\mathbf{x}_t)}_{\text{learned } s_{\boldsymbol{\phi}}(\mathbf{x}_t,t)}) \nabla_{\boldsymbol{\theta}} \alpha_t g_{\boldsymbol{\theta}}(\mathbf{z})], \quad (8.5)$$

where the teacher score is provided by the pre-trained diffusion model. In SDS, $\nabla_{\mathbf{x}_t} \log p_{\boldsymbol{\theta},t}(\mathbf{x}_t)$ is the known analytic score function of the Gaussian generator. In VSD and Diff-Instruct, an auxiliary score network $s_{\boldsymbol{\phi}}(\mathbf{x}_t, t)$ is learned to estimate it. The training alternates between learning the generator network $g_{\boldsymbol{\theta}}$ with the gradient update in (8.5) and learning the score network $s_{\boldsymbol{\phi}}$ with the denoising score matching loss in (8.1).

(a) ImageNet (b) Text-to-image (SD embedding) (c) Text-to-image (SD image space)

Figure 8.1: Image samples before and after MCMC correction. In (a)(b), the left columns are $\mathbf{x} = g_{\boldsymbol{\theta}}(\mathbf{z})$, the right columns are updated $\mathbf{x}$ after 300 steps of MCMC sampling jointly on $\mathbf{x}$ and $\mathbf{z}$. (a) illustrates the effect of correction in ImageNet. Note that the off-manifold images are corrected. (b) illustrates the correction in the embedding space of Stable Diffusion v1.5, which are decoded to image space in (c). Note the disentanglement of the cats and sharpness of the sofa. Zoom in for better viewing.

8.3 Method

8.3.1 EM Distillation

We consider formulating the problem of distilling a pre-trained diffusion model to a deep latent-variable model $p_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{z})$ defined in Section 8.2.4 using the EM framework introduced in Section 8.2.3. For simplicity, we begin with discussing the framework at a single noise level and drop the subscript t. We will revisit the integration over all noise levels in Section 8.3.3. Assume the target distribution $q(\mathbf{x})$ is represented by the diffusion model where we can access the score function $\nabla_{\mathbf{x}} \log q(\mathbf{x})$. Theoretically speaking, the generator network $g_{\boldsymbol{\theta}}(\mathbf{z})$ can employ any architecture including ones where the dimensionality of the latents differs from the data dimensionality. In this chapter, we reuse the diffusion denoiser parameterization as in other work on one-step distillation: $g_{\boldsymbol{\theta}}(\mathbf{z}) = \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}, t^*)$, where $\hat{\mathbf{x}}_{\boldsymbol{\theta}}$ is the $\mathbf{x}$-prediction function inherited from the teacher diffusion model, and t^* remains a hyper-parameter.

A naive implementation of the E-step involves two steps: (1) draw samples from the target diffusion model $q(\mathbf{x})$ and (2) sample the latent variable $\mathbf{z}$ from $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with, *e.g.*, MCMC techniques. Both steps can be highly non-trivial and computationally expensive, so here we present an alternative approach to sampling the same target distribution that avoids directly sampling from the pretrained diffusion model, by instead running

MCMC from the joint distribution of $(\mathbf{x}, \mathbf{z})$. We initialize this sampling process using a joint sample from the student: drawing $\mathbf{z} \sim p(\mathbf{z})$ and $\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. This sampled $\mathbf{x}$ is no longer drawn from $q(\mathbf{x})$, but $\mathbf{z}$ is guaranteed to be a valid sample from the posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. We then run MCMC to correct the sampled pair towards the desired distribution: $\rho_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) := q(\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\frac{q(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$ (see Figure 8.1 for a visualization of this process). If $q(\mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{x})$ are close to each other, $\rho_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ is close to $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$. In that case, initializing the *joint* sampling of $\rho_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ with pairs of $(\mathbf{x}, \mathbf{z})$ from $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ could significantly accelerate both sampling of $\mathbf{x}$ and inference of $\mathbf{z}$. Assuming MCMC converges, we can use the resulting samples to estimate the learning gradients for EM:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\rho_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})} \left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] = \mathbb{E}_{\rho_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})} \left[-\frac{\nabla_{\boldsymbol{\theta}} \|\mathbf{x} - \alpha g_{\boldsymbol{\theta}}(\mathbf{z})\|_2^2}{2\sigma^2}\right]. \tag{8.6}$$

We abbreviate our method as EMD hereafter. To successfully learn the student network with EMD, we need to identify efficient approaches to sample from $\rho_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$.

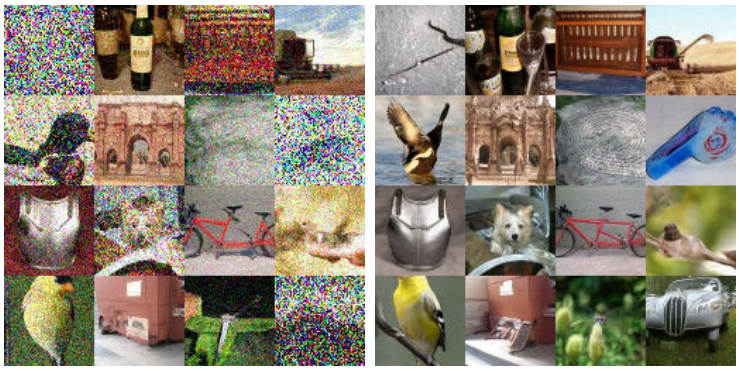
8.3.2 Reparametrized Sampling and Noise Cancellation

As an initial strategy, we consider Langevin dynamics which only requires the score functions

$$\begin{aligned} \nabla_{\mathbf{x}} \log \rho_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) &= \underbrace{\nabla_{\mathbf{x}} \log q(\mathbf{x})}_{\text{teacher score}} - \underbrace{\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})}_{\text{learned } s_{\boldsymbol{\phi}}(\mathbf{x})} + \underbrace{\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{-\frac{\mathbf{x}-\alpha g_{\boldsymbol{\theta}}(\mathbf{z})}{\sigma^2}}, \\ \nabla_{\mathbf{z}} \log \rho_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) &= \nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) + \nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{z}) = -\frac{\mathbf{x} - \alpha g_{\boldsymbol{\theta}}(\mathbf{z})}{\sigma^2} \nabla_{\mathbf{z}} g_{\boldsymbol{\theta}}(\mathbf{z}) - \mathbf{z}. \end{aligned} \tag{8.7}$$

While we do not have access to the score of the student, $\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$, we can approximate it with a learned score network $s_{\boldsymbol{\phi}}$ estimated with denoising score matching as in VSD (Wang et al., 2024) and Diff-Instruct (Luo et al., 2023c). As will be covered in Section 8.3.3, this score network is estimated at all noise levels. The Langevin dynamics defined in (8.7) can therefore be simulated at any noise level.

Running Langevin MCMC is expensive and requires careful tuning, and we found this



(a) $\mathbf{x}$ w/ accumulated noise (b) $\mathbf{x}$ w/o accumulated noise

Figure 8.2: Images after 8-step Langevin updates with and without accumulated noise.

challenging in the context of diffusion model distillation where different noise levels have different optimal step sizes. We leverage a reparametrization of $\mathbf{x}$ and $\mathbf{z}$ to accelerate the joint MCMC sampling and simplify step size tuning, similar to (Nijkamp et al., 2021; Xiao et al., 2020). Specifically, $\mathbf{x} = \alpha g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma\boldsymbol{\epsilon}$ defines a deterministic transformation from the pair of $(\boldsymbol{\epsilon}, \mathbf{z})$ to the pair of $(\mathbf{x}, \mathbf{z})$, which enables us to push back the joint distribution $\rho_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ to the $(\boldsymbol{\epsilon}, \mathbf{z})$-space. The reparameterized distribution is given by

$$\rho_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathbf{z}) = \frac{q(\alpha g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma\boldsymbol{\epsilon})}{p_{\boldsymbol{\theta}}(\alpha g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma\boldsymbol{\epsilon})} p(\boldsymbol{\epsilon}) p(\mathbf{z}). \tag{8.8}$$

See Section G.2 for a detailed derivation. We found that this parameterization admits the same step sizes across noise levels and results in better performance empirically (Table 8.1).

Still, learning the student with these samples continued to present challenges. When visualizing samples $\mathbf{x}$ produced by MCMC (see Figure 8.2a), we found that samples contained substantial noise. While this makes sense given the level of noise in the marginal distributions, we found that this inhibited learning of the student. We identify that, due to the structure of Langevin dynamics, there is noise added to $\mathbf{x}$ at each step that can be linearly accumulated across iterations. By removing this accumulated noise along with the temporally decayed initial ϵ, we recover cleaner $\mathbf{x}$ samples (Figure 8.2b). Since $\mathbf{x}$ is effectively a regression target in (8.6), and the expected value of the noises is $\mathbf{0}$,

canceling these noises reduces variance of the gradient without introducing bias. This noise cancellation was critical to the success of EMD, and is detailed in Section G.2 and ablated in experiments (Figure 8.3ab).

8.3.3 Maximum Likelihood Across All Noise Levels

The derivation above assumes smoothing the data distribution with a single noise level. In practice, the diffusion teachers always employ multiple noise levels t, coordinated by a noise schedule $p(t)$. Therefore, we optimize a weighted loss over all noise levels of the diffusion model, to encourage that the marginals of the student network match the marginals of the diffusion process at all noise levels:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{p(t), q_t(\mathbf{x}_t)} \left[\tilde{w}(t) \log p_{\boldsymbol{\theta}, t}(\mathbf{x}_t) \right] = \mathbb{E}_{p(t), \rho_t(\mathbf{x}_t, \mathbf{z})} \left[\tilde{w}(t) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}, t}(\mathbf{x}_t, \mathbf{z}) \right], \tag{8.9}$$

where $p_{\boldsymbol{\theta}, t}(\mathbf{x}_t, \mathbf{z}) = p_{\boldsymbol{\theta}, t}(\mathbf{x}_t | \mathbf{z}) p(\mathbf{z})$, $p_{\boldsymbol{\theta}, t}(\mathbf{x}_t | \mathbf{z}) = \mathcal{N}(\alpha_t g_{\boldsymbol{\theta}}(\mathbf{z}), \sigma_t^2 \boldsymbol{I})$, with a shared generator $g_{\boldsymbol{\theta}}(\mathbf{z})$ across all noise levels. Empirically, we find $\tilde{w}(t) = \sigma_t^2 / \alpha_t$ or $\tilde{w}(t) = \sigma_t^2 / \alpha_t^2$ perform well.

Denote the resulted distribution after K steps of MCMC sampling with noise cancellation as $\rho_t^K(\mathbf{x}_t^K, \mathbf{z}^K)$, the final gradient for the generator network $g_{\boldsymbol{\theta}}$ is

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{p(t), \rho_t^K(\mathbf{x}_t^K, \mathbf{z}^K)} \left[-\tilde{w}(t) \frac{\nabla_{\boldsymbol{\theta}} \| \mathbf{x}_t^K - \alpha_t g_{\boldsymbol{\theta}}(\mathbf{z}^K) \|_2^2}{2\sigma_t^2} \right]. \tag{8.10}$$

The final gradient for the score network $s_{\boldsymbol{\phi}}(\mathbf{x}_t, t)$ is

$$\nabla_{\boldsymbol{\phi}} \mathcal{J}(\boldsymbol{\phi}) = \mathbb{E}_{p(t), p_{\boldsymbol{\theta}, t}(\mathbf{x}_t, \mathbf{z})} \left[w(t) \nabla_{\boldsymbol{\phi}} \| s_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | g_{\boldsymbol{\theta}}(\mathbf{z})) \|_2^2 \right]. \tag{8.11}$$

Similar to VSD (Wang et al., 2024; Luo et al., 2023c), we employ alternating update for the generator network $g_{\boldsymbol{\theta}}$ and the score network $s_{\boldsymbol{\phi}}(\mathbf{x}_t, t)$. See summarization in Algorithm 8.

Algorithm 8 EM Distillation

Input: Teacher score functions $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$, generator network $g_{\boldsymbol{\theta}}$, prior $p(\mathbf{z})$, score network $s_{\boldsymbol{\phi}}$, noise scheduler $p(t)$, weighting functions $w(t)$ and $\tilde{w}(t)$, # of MCMC steps K, MCMC step size γ.
Output: Generator network $g_{\boldsymbol{\theta}}$, score network $s_{\boldsymbol{\phi}}$.
while not converged **do**
  Sampling a batch of t, $\mathbf{z}$, $\boldsymbol{\epsilon}$ from $p(t)$, $p(\mathbf{z})$, $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to obtain $\mathbf{x}_t$
  Updating $s_{\boldsymbol{\phi}}$ via Stochastic Gradient Descent with the batch estimate of (8.11)
  Sampling $\mathbf{x}_t^K$ and $\mathbf{z}^K$ with $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector$(\mathbf{x}_0, \boldsymbol{\epsilon}, \mathbf{z}, t, \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t), g_{\boldsymbol{\theta}}, s_{\boldsymbol{\phi}}, K, \gamma)$ as in Algorithm 9
  Updating $g_{\boldsymbol{\theta}}$ via Stochastic Gradient Ascent with the batch estimate of (8.10)
end while

Algorithm 9 $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector

Input: $\mathbf{x}_0$, $\boldsymbol{\epsilon}$, $\mathbf{z}$, t, teacher score function $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$, generator network $g_{\boldsymbol{\theta}}$, prior $p_0(\mathbf{z})$, score network $s_{\boldsymbol{\phi}}$, # of MCMC steps K, MCMC step size γ.
Output: $\mathbf{x}_t^K$, $\mathbf{z}^K$.
Sampling Langevin noise $\mathbf{n}^1, \mathbf{n}^2, \ldots, \mathbf{n}^K$ from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, letting $\boldsymbol{\epsilon}^0 = \boldsymbol{\epsilon}, \mathbf{z}^0 = \mathbf{z}$
for i in $[1, K]$ **do**
  Updating $(\boldsymbol{\epsilon}^i, \mathbf{z}^i)$ with 1-step Langevin update over scores (G.3), with $\boldsymbol{\epsilon}^i$ updated using $\mathbf{n}^i$
end for
Pushing $(\boldsymbol{\epsilon}^K, \mathbf{z}^K)$ forward to $(\mathbf{x}_t^K, \mathbf{z}^K)$ and then canceling the noises in $\mathbf{x}_t^K$

8.3.4 Connection With VSD and Diff-Instruct

In this subsection, we reveal an interesting connection between EMD and Variational Score Distillation (VSD) (Wang et al., 2024; Luo et al., 2023c), *i.e.*, although motivated by optimizing different types of divergences, VSD (Wang et al., 2024; Luo et al., 2023c) is equivalent to EMD with a special sampling scheme.

To see this, consider the 1-step EMD with noise cancellation, stepsize $\gamma = 1$ in $\mathbf{x}$, and no update on $\mathbf{z}$

$$\mathbf{x}_t^0 = \alpha_t g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma_t \epsilon, \quad \mathbf{x}_t^1 = \alpha_t g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma_t^2 \nabla_{\mathbf{x}} \log \frac{q(\mathbf{x}_t^0)}{p_{\boldsymbol{\theta},t}(\mathbf{x}_t^0)} + \cancel{\sqrt{2}\sigma \mathbf{n}^1}. \tag{8.12}$$

Substitute it into (8.10), we have

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) &= \mathbb{E}_{p(t),p(\epsilon),p(\mathbf{z})}\left[-\tilde{w}(t)\frac{\nabla_{\boldsymbol{\theta}}\|\mathbf{x}_t^1-\alpha_t g_{\boldsymbol{\theta}}(\mathbf{z})\|_2^2}{2\sigma_t^2}\right] \\ &= \mathbb{E}_{p(t),p(\epsilon),p(\mathbf{z})}\left[\tilde{w}(t)(\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)-\nabla_{\mathbf{x}_t}\log p_{\boldsymbol{\theta},t}(\mathbf{x}_t))\nabla_{\boldsymbol{\theta}}\alpha_t g_{\boldsymbol{\theta}}(\mathbf{z})\right],\end{aligned} \tag{8.13}$$

which is exactly the gradient for VSD (8.5), up to a sign difference. This insight demonstrates that, EMD framework can flexibly interpolate between mode-seeking and mode-covering divergences, by leveraging different sampling schemes from 1-step sampling in only $\mathbf{x}$ (a likely biased sampler) to many-step joint sampling in $(\mathbf{x}, \mathbf{z})$ (closer to a mixing sampler).

If we further assume the marginal $p_{\boldsymbol{\theta}}(\mathbf{x})$ is a Gaussian, then the EMD update in (8.13) would resemble Score Distillation Sampling (SDS) (Poole et al., 2022).

8.4 Related work

Diffusion Acceleration Diffusion models have the notable issue of slowness in inference, that motivates many research efforts to accelerate the sampling process. One line of work focuses on developing numerical solvers (Luhman and Luhman, 2021; Song et al., 2020a; Lu et al., 2022a,b; Bao et al., 2022; Karras et al., 2022) for the PF-ODE. Another line of work leverages the concept of knowledge distillation (Hinton et al., 2015) to condense the sampling trajectory of PF-ODE into fewer steps (Salimans and Ho, 2022; Meng et al., 2023; Song et al., 2023; Berthelot et al., 2023; Li et al., 2023; Luo et al., 2023a; Heek et al., 2024; Kim et al., 2023; Zheng et al., 2024; Yan et al., 2024; Liu et al., 2023). However, both approaches have significant limitations and have difficulty in substantially reducing the sampling steps to the single-step regime without significant loss in perceptual quality.

Single-Step Diffusion Models Recently, several methods for one-step diffusion sampling have been proposed, sharing the same goal as our approach. Some methods fine-tune the pre-trained diffusion model into a single-step generator via adversarial train-

ing (Xu et al., 2023; Sauer et al., 2023, 2024), where the adversarial loss enhances the sharpness of the diffusion model's single-step output. Adversarial training can also be combined with trajectory distillation techniques to improve performance in few or single-step regimes (Kim et al., 2023; Lin et al., 2024; Ren et al., 2024). Score distillation techniques (Poole et al., 2022; Wang et al., 2024) have been adopted to match the distribution of the one-step generator's output with that of the teacher diffusion model, enabling single-step generation (Luo et al., 2023c; Nguyen and Tran, 2023). Additionally, Yin et al. (2024) introduce a regression loss to further enhance performance. These methods achieve more impressive 1-step generation, some of which enjoy additional merits of being data-free or flexible in the selection of generator architecture. However, they often minimizes over mode-seeking divergences that can fail to capture the full distribution and therefore causes mode collapse issues. We discuss the connection between our method and this line of work in Section 8.3.4.

8.5 Experiments

We employ EMD to learn one-step image generators on ImageNet 64×64, ImageNet 128×128 (Deng et al., 2009) and text-to-image generation. The student generators are initialized with the teacher model weights. Results are compared according to Frechet Inception Distance (FID) (Heusel et al., 2017), Inception Score (IS) (Salimans et al., 2016), Recall (Rec.) (Kynkäänniemi et al., 2019) and CLIP Score (Radford et al., 2021). Throughout this section, we will refer to the proposed EMD with K steps of Langevin updates on $(\mathbf{x}, \mathbf{z})$ as EMD-K, and we use EMD-1 to describe the DiffInstruct/VSD-equivalent formulation with only one update in $\mathbf{x}$ as presented in Section 8.3.4.

8.5.1 ImageNet

We start from showcasing the effect of the key components of EMD, namely noise cancellation, multi-step joint sampling, and reparametrized sampling. We then summarize results on ImageNet 64×64 with Karras et al. (2022) as teacher, and ImageNet 128×128

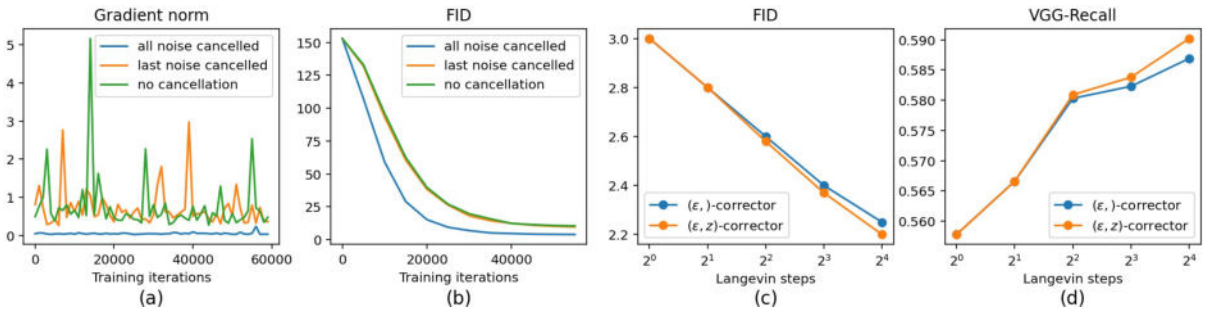


Figure 8.3: (a)(b) Gradient norms and FIDs for complete noise cancellation, last-step noise cancellation and no noise cancellation. (c)(d) FIDs and Recalls of EMD with different numbers of Langevin steps.

with Kingma and Gao (2023) as teacher.

Noise Cancellation During our development, we observed the vital importance of canceling the noise after the Langevin update. Even though theoretically speaking our noise cancellation technique does not guarantee reducing the variance of the gradients for learning, we find removing the accumulated noise term from the samples (including the initial diffusion noise $\boldsymbol{\epsilon}$) does give us seemingly clean images empirically. See Figure 8.2 for a comparison. These updated $\mathbf{x}^K$ can be seen as regression targets in (8.10). Intuitively speaking, regressing a generator towards clean images should result in more stable training than towards noisy images. Reflected in the training process, canceling the noise significantly decreases the variance in the gradient (Figure 8.3a) and boosts the speed of convergence (Figure 8.3b). We also compare with another setting where only the noise in the last step gets canceled, which is only marginally helpful.

Multi-Step Joint Sampling We scrutinize the effect of multi-step joint update on $(\boldsymbol{\epsilon}, \mathbf{z})$. Empirically, we find a constant step size of Langevin dynamics across all noise levels in the $(\boldsymbol{\epsilon}, \mathbf{z})$-space works well: $\boldsymbol{\gamma} = (\gamma_{\boldsymbol{\epsilon}}, \gamma_{\mathbf{z}}) = (0.4^2, 0.004^2)$, which simplifies the process of step size tuning. Figure 8.1 shows results of running this $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector for 300 steps. We can see that the $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector removes visual artifacts and improves structure. Figure 8.3cd illustrates the relation between the distilled generator's performance and the number of Langevin steps per distillation iteration, measured by FID and Recall respectively. Both metrics show clear improvement monotonically as the number of

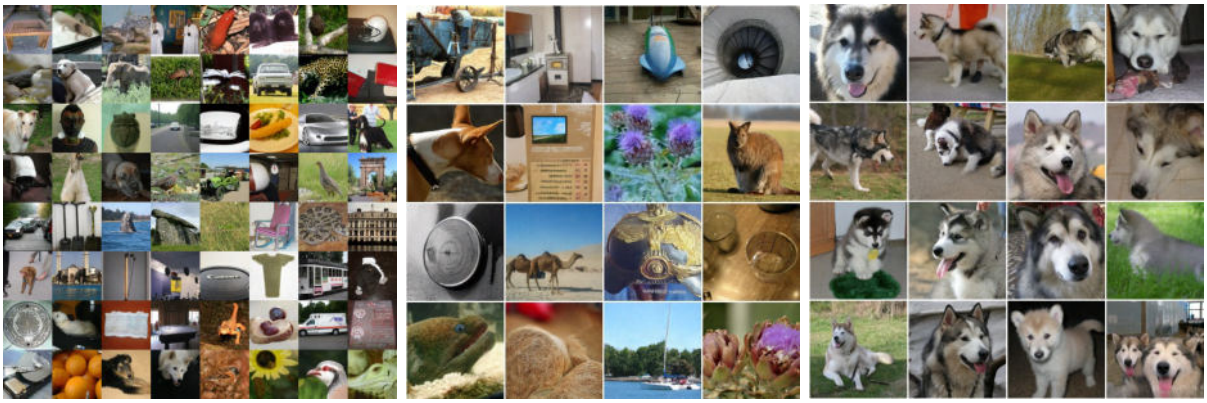
	FID (↓)	IS (↑)
$(\mathbf{x},)/(\boldsymbol{\epsilon},)$	<u>2.829</u>	<u>62.31</u>
$(\mathbf{x}, \mathbf{z})$	3.11	61.08
$(\boldsymbol{\epsilon}, \mathbf{z})$	**2.77**	**62.98**

Table 8.1: EMD-8 on ImageNet 64×64, 100k steps of training.

Langevin steps increases. Recall is designed for measuring mode coverage (Kynkäänniemi et al., 2019), and has been widely adopted in the GAN literature. A larger number of Langevin steps encourages better mode coverage, likely because it approximates the mode-covering *forward* KL better. Sampling $\mathbf{z}$ is more expensive than sampling $\boldsymbol{\epsilon}$, requiring back-propagation through the generator $g_{\boldsymbol{\theta}}$. An alternative is to only sample $\boldsymbol{\epsilon}$ while keeping $\mathbf{z}$ fixed, with the hope that if $\mathbf{x}$ does not change dramatically with a finite number of MCMC updates, the initial $\mathbf{z}$ remains a good approximation of samples from $\rho_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. As shown in Figure 8.3cd, sampling $\boldsymbol{\epsilon}$ performs similarly to the joint sampling of $(\boldsymbol{\epsilon}, \mathbf{z})$ when the number of sampling steps is small, but starts to fall behind with more sampling steps.

Reparametrized Sampling As shown in Section G.2, the noise cancellation technique does not depend on the reparametrization. One can start from either the score functions of $(\mathbf{x}, \mathbf{z})$ in (8.7) or the score functions of $(\boldsymbol{\epsilon}, \mathbf{z})$ in (G.3) to derive something similar. We conduct a comparison between the two parameterizations for joint sampling, $(\mathbf{x}, \mathbf{z})$-corrector and $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector.

For the $(\mathbf{x}, \mathbf{z})$-corrector, we set the step size of $\mathbf{x}$ as $\sigma_t^2 \gamma_{\boldsymbol{\epsilon}}$ to align the magnitude of update with the one of the $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector, and keep the step size of $\mathbf{z}$ the same (see Section G.2 for details). This also promotes numerical stability in $(\mathbf{x}, \mathbf{z})$-corrector by canceling the σ_t^2 in the denominator of the term $\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = -\frac{\mathbf{x} - \alpha g_{\boldsymbol{\theta}}(\mathbf{z})}{\sigma^2}$ in the score function (8.7). A similar design choice was proposed in Song and Ermon (2019). Also note that adjusting the step sizes in this way results in an equivalence between $(\boldsymbol{\epsilon},)$-corrector and $(\mathbf{x},)$-corrector without sampling in $\mathbf{z}$, which serves as the baseline for the joint sampling.



(a) ImageNet 64 multi-class (b) ImageNet 128 multi-class (c) ImageNet 128 single-class

Figure 8.4: ImageNet samples from the distilled 1-step generator. Models are trained class-conditionally with all classes. We provide single-class samples in (c) to demonstrate good mode coverage.

Table 8.1 reports the quantitative comparisons with EMD-8 on ImageNet 64×64 after 100k steps of training. While joint sampling with $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector improves over $(\boldsymbol{\epsilon},)$-corrector, $(\mathbf{x}, \mathbf{z})$-corrector struggles to even match the baseline. Possible explanations include that the space of $(\boldsymbol{\epsilon}, \mathbf{z})$ is more MCMC friendly, or it requires more effort on searching for the optimal step size of $\mathbf{z}$ for the $(\mathbf{x}, \mathbf{z})$-corrector. We leave further investigation to future work.

Comparsion With Existing Methods We report the results from our full-fledged method, EMD-16, which utilizes a $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector with 16 steps of Langevin updates, and compare with existing approaches. We train for 300k steps on ImageNet 64×64, and 200k steps on ImageNet 128×128. Other hyperparameters can be found in Section G.3. Samples from the distilled generator can be found in Figure 8.4. We also include additional samples in Section G.4.1. We summarize the comparison with existing methods for few-step diffusion generation in Table 8.2 and Table 8.3 for ImageNet 64×64 and ImageNet 128×128 respectively. Note that we also tune the baseline EMD-1, which in formulation is equivalent to Diff-Instruct (Luo et al., 2023c), to perform better than their reported numbers. The improvement mainly comes from a fine-grained tuning of learning rates and enabling dropout for both the teacher and student score functions. Our final models outperform existing approaches for one-step distillation of diffusion models

Method	NFE (↓)	FID (↓)	Rec. (↑)
Multiple Steps			
DDIM (Song et al., 2020a)	50	13.7	
EDM-Heun (Karras et al., 2022)	10	17.25	
DPM Solver (Lu et al., 2022a)	10	7.93	
PD (Salimans and Ho, 2022)	2	8.95	0.65
CD (Song et al., 2023)	2	4.70	0.64
Multistep CD (Heek et al., 2024)	2	2.0	-
Single Step			
BigGAN-deep (Brock et al., 2018)	1	4.06	0.48
EDM (Karras et al., 2022)	1	154.78	-
PD (Salimans and Ho, 2022)	1	15.39	0.62
BOOT (Gu et al., 2023)	1	16.30	0.36
DFNO (Zheng et al., 2023)	1	7.83	-
TRACT (Berthelot et al., 2023)	1	7.43	-
CD-LPIPS (Song et al., 2023)	1	6.20	0.63
Diff-Instruct (Luo et al., 2023c)	1	5.57	-
DMD (Yin et al., 2024)	1	2.62	-
EMD-1 (baseline)	1	3.1	0.55
EMD-16 (ours)	1	**2.20**	0.59
Teacher	256	1.43	-

Table 8.2: Class-conditional generation on ImageNet 64×64.

in terms of FID scores on both tasks. On ImageNet 64×64, EMD achieves a competitive recall among distribution matching approaches but falls behind trajectory distillation approaches which maintain individual trajectory mappings from the teacher.

8.5.2 Text-to-Image Generation

We further test the potential of EMD on text-to-image models at scale by distilling the Stable Diffusion v1.5 (Rombach et al., 2022) model. Note that the training is image-free and we only use text prompts from the LAION-Aesthetics-6.25+ dataset (Schuhmann et al., 2022). On this task, DMD (Yin et al., 2024) is a strong baseline, which introduced an additional regression loss to VSD or Diff-Instruct to avoid mode collapse. However, we find the baseline without regression loss, or equivalently EMD-1, can be improved by simply tuning the hyperparameter t^*. Empirically, we find it is better to set t^* to

Method	NFE (↓)	FID (↓)	IS (↑)
Multiple Steps			
Multistep CD (Heek et al., 2024)	8	2.1	164
Multistep CD (Heek et al., 2024)	4	2.3	157
Multistep CD (Heek et al., 2024)	2	3.1	147
Single Step			
CD (Song et al., 2023)	1	7.0	-
EMD-1 (baseline)	1	6.3	134 ± 2.75
EMD-16 (ours)	1	**6.0**	**140** ± 2.83
Teacher	512	1.75	171.1 ± 2.7

Table 8.3: Class-conditional generation on ImageNet 128×128.

Family	Method	Latency (↓)	FID (↓)
Unaccelerated	DALL·E (Ramesh et al., 2021)	-	27.5
	DALL·E 2 (Ramesh et al., 2022)	-	10.39
	Parti-3B (Yu et al., 2022a)	6.4s	8.10
	Make-A-Scene (Gafni et al., 2022)	25.0s	11.84
	GLIDE (Nichol et al., 2021)	15.0s	12.24
	Imagen (Saharia et al., 2022)	9.1s	7.27
	eDiff-I (Balaji et al., 2022)	32.0s	6.95
GANs	StyleGAN-T (Karras et al., 2018)	0.10s	13.90
	GigaGAN (Kang et al., 2023)	0.13s	9.09
Accelerated	DPM++ (4 step)† (Lu et al., 2022b)	0.26s	22.36
	UniPC (4 step)† (Zhao et al., 2024)	0.26s	19.57
	LCM-LoRA (1 step)† (Luo et al., 2023b)	0.09s	77.90
	LCM-LoRA (4 step)† (Luo et al., 2023b)	0.19s	23.62
	InstaFlow-0.9B† (Liu et al., 2023)	0.09s	13.10
	UFOGen (Xu et al., 2023)	0.09s	12.78
	DMD (tCFG=3)† (Yin et al., 2024)	0.09s	11.49
	EMD-1 (baseline, tCFG=3)	0.09s	10.96
	EMD-1 (baseline, tCFG=2)	0.09s	9.78
	EMD-8 (ours, tCFG=2)	0.09s	**9.66**
Teacher	SDv1.5† (Rombach et al., 2022)	2.59s	8.78

Table 8.4: † Results are evaluated by Yin et al. (2024).

Family	Method	Latency (↓)	CLIP (↑)
Accelerated	DPM++ (4 step) (Lu et al., 2022b)†	0.26s	0.309
	UniPC (4 step)† (Zhao et al., 2024)	0.26s	0.308
	LCM-LoRA (1 step)† (Luo et al., 2023b)	0.09s	0.238
	LCM-LoRA (4 step)† (Luo et al., 2023b)	0.19s	0.297
	DMD† (Yin et al., 2024)	0.09s	0.320
	EMD-8 (ours)	0.09s	0.316
Teacher	SDv1.5 † (Rombach et al., 2022)	2.59s	0.322

Table 8.5: CLIP Scores in high CFG regime

Figure 8.5: Text-to-image samples from the model distilled from Stable Diffusion v1.5.

intermediate noise levels, consistent with the observation from Luo et al. (2023c). Other hyperparameters can be found in Section G.3.

We evaluate the distilled one-step generator for text-to-image generation with zero-shot generalization on MSCOCO (Lin et al., 2014) and report the FID-30k in Table 8.4 and CLIP Score in Table 8.5. Yin et al. (2024) uses the guidance scale of 3.0 to compose the classifer-free guided teacher score (we refer to this guidance scale of teacher as tCFG) in the learning gradient of DMD, for it achieves the best FID for DDIM sampler. However, we find EMD achieves a lower FID at the tCFG of 2.0. Our method, EMD-8, trained on 256 TPU-v5e for 5 hours (5000 steps), achieves the FID=9.66 for one-step text-to-image generation. Using a higher tCFG, similar to DMD, produces a model with competitive CLIP Score. In Figure 8.5, we include some samples for qualitative evalu-

ation. Additional qualitative results (Table G.8 and Table G.9), as well as side-by-side comparisons (Table G.4, Table G.5, Table G.6, and Table G.7) with trajectory-based distillation baselines (Liu et al., 2023; Luo et al., 2023b) and adversarial distillation baselines (Sauer et al., 2023) can be found in Section G.4.2.

8.6 Summary

We presented EMD, a maximum likelihood-based method that leverages the EM framework with novel sampling and optimization techniques to learn a one-step latent-variable student model whose marginal distributions match the marginals of a pretrained diffusion model. EMD demonstrates strong performance in class-conditional generation on ImageNet and text-to-image generation. Despite exhibiting compelling results, EMD has a few limitations that call for future work. Empirically, we find that EMD still requires the student model to be initialized from the teacher model to perform competitively, and is sensitive to the choice of t^* (fixed timestep conditioning that repurposes the diffusion denoiser to become a one-step genertor) at initialization. While training a student model entirely from scratch is supported theoretically by our framework, empirically we were unable to achieve competitive results. Improving methods to enable generation from randomly initialized generator networks with distinct architectures and lower-dimensional latent variables is an exciting direction for future work. Although being efficient in inference, EMD introduces additional computational cost in training by running multiple sampling steps per iteration, and the step size of MCMC sampling can require careful tuning. There remains a fundamental trade-off between training cost and model performance. Analysis and further improving on the Pareto frontier of this trade-off would be of interest for future work.

CHAPTER 9

Conclusions

9.1 Summary of Contributions

The motivating theme for this dissertation was to develop Artificial Intelligence (AI) agents that can effectively organize the patterns relevant to their own experiences for better prediction, reasoning, and decision-making. Among the broad spectrum of AI methodologies, our focus was particularly on Machine Learning (ML)-based AI systems that combine data and computation following pattern theoretic principles (Grenander, 1970; Grenander and Miller, 2007). These systems are currently known as Generative Models or Generative AI. We identified *the learning and inference of latent abstractions* as a shortcoming of these powerful systems and a blocker to aligning their understanding and intrinsic values with those of humans. To extend the validated principles of generative modeling from data space to the latent space, we discussed the relevant theory and practice across the four parts of this dissertation. We now summarize the contributions and interconnections of these parts.

Part I and Part II share similar structures. We started from two latent-variable models, LDEBM and DRC, as holistic prototypes for applying the pattern theoretic principles to abstraction learning. Leveraging the established workflow in well-developed research topics, we focused on technical contributions in terms of designing models, learning objectives, and inference algorithms. Chapter 2 and Chapter 4 affirm the effectiveness of learning latent abstractions in generative modeling. Experiments presented in these two chapters, though limited by their controllable scopes, demonstrate the niche role of test-time inference over latent abstractions in promoting better interpretability and broader

generalization.

Armed with these formalizations, we ventured into the uncharted territories of iconic symbols (Chapter 3) and object compositionality (Chapter 5), and we were rewarded with refreshed perspectives. In Chapter 3, we replicated a seminal experiment about the formation of abstraction in human visual communication. When facing the unique representations in the modality of sketches, we designed evaluation metrics following mathematical principles to measure their alignment with human intuitions, making experimental contributions to the underexplored research topic of emergent communication. The emergent phenomenon of coadapted agents switching between abstract and iconic drawings to communicate seen and unseen concepts offers an intuitive mental model for thinking about the convoluted relationship between abstraction and generalization. While the metrics employed in Chapter 3 trace back to some established metrics in representation learning, the COAT measure proposed in Chapter 5 is more original. We went beyond the established problems of categorization and disentanglement, scrutinizing the algebraic and geometrical structures of multi-object scenes, and redefined the problem of object-centric abstraction learning. Our metric revealed existing models' lack of understanding about an object's absence and identity, pinpointing a fundamental misalignment with object abstractions in human cognition.

The first two parts of the dissertation prepared us for Part III, where we combined the contributions of initiating and shaping novel research topics with principled methodological development. In Chapter 6, we challenged the widely adopted probabilistic model for decision-making and the associated learning algorithm. We extended the scientific prototypes from Chapter 2 and Chapter 4 to propose a novel solution. By viewing decisions as latent abstractions, this new model formalizes the key concepts of "understanding", "intrinsic values", and "consistency", offering profound insights into the connection between decision-making and generative modeling. In Chapter 7, we further developed and extended this quest by introducing a latent temporal abstraction — the plan. Utilizing the most common language in generative modeling, distribution matching, we provided an affirmative answer to the advantage of latent abstraction inference over the more pop-

ular data-space auto-regressive models: it inherently enforces temporal consistency. Due to the generality of this theoretical analysis, it may challenge the status quo pretraining-postraining pipeline of Large Language Models (LLMs).

While the first three parts of the dissertation are sufficiently comprehensive with regard to its theme, we included Part IV as a dialectic discussion. We demonstrated how to re-purpose and extend the general techniques developed in the previous chapters to tackle a research problem that is of contemporary value but lies beyond the presented learning paradigm: distilling from foundational data-space diffusion models. Despite the paradigm shift, we showed that our technical contributions can help improve state-of-the-art massive-scale foundational models, independent of the high-level hypotheses of whether the latent abstractions should be learned directly from raw data or distilled from data-space models.

9.2 Future Work

The contributions of this dissertation suggest new opportunities and challenging questions, giving impetus to future research, some of which we highlight below:

Web-Scale Latent Abstraction Learning How do we scale up the abstraction learning methods to leverage the abundant web-scale data? Just one day before the submission of this dissertation, OpenAI released their latent-variable LLM, OpenAI o1 (El-Kishky et al., 2024), which achieves top expert-level performance in competitive programming questions and the Math Olympiad. It is effectively a scaled-up version of the model presented in Chapter 7, but learns the distribution over latent abstractions with reinforcement learning, similar to the method presented in Chapter 3. Looking forward, we anticipate that the fundamental connection between latent-variable generative modeling and decision-making we derived herein will open up new avenues for future research.

Test-Time Compute How do we design more efficient latent abstraction inference algorithms to achieve adaptive test-time compute? The inference algorithms that we presented in this dissertation are mostly MCMC-based sampling. While this captures

the intuition of iterative refinement in principle, its intractability has been a long-standing challenge. However, lifting MCMC to the latent space should potentially alleviate this issue. Our recent publication Yu et al. (2023) proposes to learn a diffusion model during the training of latent EBM to amortize the posterior inference. The learned latent energy space is more traversible than in prior art (Nijkamp et al., 2020a). If we can achieve a Bayesian-principled test-time computation in the future, we can expect it to be more efficiently scalable than scaling the model parameters (Snell et al., 2024).

Unsupervised Machine Formalization What are the missing pieces to achieve unsupervised formalization, the ultimate form of abstraction that facilitates our logical and mathematical reasoning? To date, machine formalization is still a supervised translation task (Wu et al., 2022). Given the symbol-vector duality that we presented in Part I, as well as some current progress in scaling up mechanistic interpretability (Cunningham et al., 2023; Gao et al., 2024; Templeton, 2024), we are optimistic that an interpretable hierarchy of categories will emerge automatically in the latent space of generative models. Inference over these latent abstractions will potentially produce chains-of-thought that align with human understanding.

Memory Systems as Generative Models Does the brain have the analog of a computer hard disk drive? While we cannot be certain, the impressive capabilities of generative models suggest that they could be viable candidates for modeling our memory systems. This analogy opens up intriguing avenues for complementary memory systems in neocortex and hippocampus that cognitive scientists have long postulated for language (Ullman, 2004) and intelligence in general (Kumaran et al., 2016). Are latent abstractions and generators two different but reciprocal memory systems? What would be the role of data-space models? We hope that the models and methods presented in this dissertation can provide the language and tools for studying these fundamental problems.

In closing, we are confident that future advancements of abstraction learning and inference in generative modeling will have a profound impact not only in next generation artificial intelligence, but also in the next epoch of human cultural evolution.

APPENDIX A

Derivations and Experimental Details for Chapter 2

A.1 Extended Derivations and Further Discussion

A.1.1 Derivation of Conditional EBMs

We first define the marginal EBMs at each diffusion step:

$$\begin{cases} p_\alpha(\mathbf{z}_t) = \dfrac{1}{Z_{\alpha,t}} \exp(F_\alpha(\mathbf{z}_t, t)) p_0(\mathbf{z}_t), \ t = T-1 \\ p_\alpha(\mathbf{z}_t) = \dfrac{1}{Z_{\alpha,t}} \exp(F_\alpha(\mathbf{z}_t, t)), \ t = 0, 1, \ldots, T-2, \end{cases} \tag{A.1}$$

where the marginal energy term is in a log-sum-exponential form $F_\alpha(\mathbf{z}_t, t) = \log \sum_{\mathbf{y}} \exp(\langle \mathbf{y}, f_\alpha(\mathbf{z}_t, t) \rangle)$; it serves to aggregate the energy score from each category. Of note, the marginal EBM corresponding with the last diffusion step has a slightly different definition. We set this term as exponential tilting of a non-informative Gaussian prior $p_0(\mathbf{z}_t)$ which helps to stabilize training in practice.

Recall that $\mathbf{z}_{t+1} = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t + \sigma_{t+1}\boldsymbol{\epsilon}_{t+1}$. Let $\tilde{\mathbf{z}}_t = \sqrt{1-\sigma_{t+1}^2}\mathbf{z}_t$. For $t = 0, 1, \ldots, T-2$, we have

$$\begin{aligned} p_\alpha(\tilde{\mathbf{z}}_t | \mathbf{z}_{t+1}) &= \frac{p_\alpha(\tilde{\mathbf{z}}_t) p(\mathbf{z}_{t+1} | \tilde{\mathbf{z}}_t)}{p_\alpha(\mathbf{z}_{t+1})} \\ &= \frac{1}{\tilde{Z}_{\alpha,t}} \frac{\exp(F_\alpha(\tilde{\mathbf{z}}_t, t))}{p_\alpha(\mathbf{z}_{t+1})} \exp\left(-\frac{1}{2\sigma_{t+1}^2} ||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2 \right) \\ &= \frac{1}{\tilde{Z}_{\alpha,t}(\mathbf{z}_{t+1})} \exp\left(F_\alpha(\tilde{\mathbf{z}}_t, t) - \frac{1}{2\sigma_{t+1}^2} ||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2 \right), \end{aligned} \tag{A.2}$$

where $\tilde{Z}_{\alpha,t} = (2\pi\sigma_{t+1}^2)^{\frac{n}{2}} Z_{\alpha,t}$; we slightly abuse the notation and use $p(\mathbf{z}_{t+1}|\tilde{\mathbf{z}}_t)$ to represent the forward transition $q(\mathbf{z}_{t+1}|\mathbf{z}_t)$ defined in (2.4) for notation consistency.

The diffused samples at time step T are close to Gaussian white noise; $p_\alpha(\tilde{\mathbf{z}}_{T-1}|\mathbf{z}_T)$ therefore falls to its marginal distribution $p(\tilde{\mathbf{z}}_{T-1})$ defined in (A.1).

A.1.2 Derivation of the ELBO

Recall that the ELBO in SVEBM is

$$\begin{aligned}
\text{ELBO}_{\boldsymbol{\theta},\boldsymbol{\phi}} &= \log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_\alpha(\mathbf{z})) \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_\alpha(\mathbf{z})\right],
\end{aligned} \tag{A.3}$$

where D_{KL} denotes the Kullback-Leibler divergence. Let us consider the full trajectory of the perturbed samples $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T$. The above equation can be written as

$$\begin{aligned}
\text{ELBO}_{\boldsymbol{\theta},\boldsymbol{\phi}} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})\right] \\
+ \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\log \int_{\mathbf{z}_{1:T}} p_\alpha(\mathbf{z}_{0:T}) d\mathbf{z}_{1:T}\right],
\end{aligned} \tag{A.4}$$

where the last term is further lower-bounded by introducing the forward trajectory distribution; the inequality holds by applying Jensen's Inequality:

$$\begin{aligned}
& \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\log \int_{\mathbf{z}_{1:T}} p_\alpha(\mathbf{z}_{0:T}) d\mathbf{z}_{1:T}\right] \\
= \; & \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\log \int_{\mathbf{z}_{1:T}} q(\mathbf{z}_{1:T}|\mathbf{z}_0) \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)} d\mathbf{z}_{1:T}\right] \\
\geq \; & \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\int_{\mathbf{z}_{1:T}} q(\mathbf{z}_{1:T}|\mathbf{z}_0) \log \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)} d\mathbf{z}_{1:T}\right] \\
= \; & \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x}), q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log \frac{p_\alpha(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\right].
\end{aligned} \tag{A.5}$$

Further, we can decompose the joint distribution of forward and backward trajectories as

$$\begin{aligned}
&\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x}),q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log \frac{p_{\boldsymbol{\alpha}}(\mathbf{z}_{0:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\right] = \\
&\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|x),q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log p(\mathbf{z}_T) + \sum_{t=0}^{T-1} \log \frac{p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1})}{q(\mathbf{z}_{t+1}|\mathbf{z}_t)}\right] = \\
&\mathbb{E}\left[\log p(\mathbf{z}_T) + \sum_{t=0}^{T-1} \log p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1})\right] + \sum_{t=0}^{T-1} \mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t),
\end{aligned} \tag{A.6}$$

where $p(\mathbf{z}_T)$ is standard Gaussian distribution; $\mathbb{E}$ is the abbreviation of $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|x),q(\mathbf{z}_{1:T}|\mathbf{z}_0)}$. $\mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t), t = 0, \ldots, 1$ is the conditional entropy under the forward trajectory distribution. We obtain $\mathbf{z}_t$ by sampling $\tilde{\mathbf{z}}_t$ from $p_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})$ and then applying $\mathbf{z}_t = \tilde{\mathbf{z}}_t/\sqrt{1-\sigma_{t+1}^2}$; the reverse trajectory in our model is primarily defined by $p_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})$ for $t > 0$. We use $[\mathbf{z}_t|\mathbf{z}_{t+1}]$ to represent this process in the following sections; we may interchangeably use the notation of $\tilde{\mathbf{z}}_t$ and $\mathbf{z}_t$ for simplicity.

Note that the entropies can be analytically computed and do not involve learnable parameters. The joint training of inference, prior and generation models can be largely reduced to finding the agreement of the forward and reverse Markov transitions defined by $q_{\boldsymbol{\phi}}$ and $p_{\boldsymbol{\theta}}$ respectively.

A.1.3 Detailed Discussion of Symbol Coupling

In Section 2.2, we briefly describe how to introduce the symbolic one-hot vector $\mathbf{y}$. Since only $\mathbf{z}_0$ is connected with $\mathbf{y}$, we can first define the joint prior $p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}_0)$ as in (A.1) by substituting $F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_0, 0)$ with $\langle \mathbf{y}, f_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_0, 0)\rangle$. Then the conditional symbol-vector coupling joint distribution follows as

$$p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1) = \frac{1}{\tilde{Z}_{\boldsymbol{\alpha},t=0}} \exp\left(\langle \mathbf{y}, f_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_0, 0)\rangle\right) \exp\left(-\frac{1}{2\sigma_1^2}||\tilde{\mathbf{z}}_0 - \mathbf{z}_1||^2\right). \tag{A.7}$$

Note that $p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1) = p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0)p_{\boldsymbol{\alpha}}(\mathbf{z}_0|\mathbf{z}_1)$, *i.e.*, $\mathbf{z}_0$ is sufficient for inferring $\mathbf{y}$ in this formulation:

$$\begin{aligned} p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0, \mathbf{z}_1) &= \frac{p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}_0|\mathbf{z}_1)}{p_{\boldsymbol{\alpha}}(\mathbf{z}_0|\mathbf{z}_1)} \\ &= \frac{\exp\left(\langle \mathbf{y}, f_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_0, 0)\rangle\right)}{\exp\left(F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_0, 0)\right)}, \end{aligned} \tag{A.8}$$

so that given $\mathbf{z}_0$,

$$p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0) \propto \exp(\langle \mathbf{y}, f_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_0, 0)\rangle). \tag{A.9}$$

It similarly becomes a softmax classifier where $f_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_0, 0)$ provides the logit scores for the K categories.

A.1.4 Derivation of the Information Bottleneck

We first define the mutual information term between $\mathbf{z}_0$ and $\mathbf{y}$. Consider the joint distribution of $\mathbf{x}, \mathbf{z}_0$ and $\mathbf{y}$, $\pi(\mathbf{y}, \mathbf{z}_0, \mathbf{x}) = p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0)q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})q_{\text{data}}(\mathbf{x})$; the mutual information $\mathcal{I}(\mathbf{z}_0, \mathbf{y})$ defined under π then follows as

$$\begin{aligned} \mathcal{I}(\mathbf{z}_0, \mathbf{y}) &= \mathcal{H}(\mathbf{y}) - \mathcal{H}(\mathbf{y}|\mathbf{z}_0) \\ &= -\sum_{\mathbf{y}} q(\mathbf{y}) \log q(\mathbf{y}) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0)} \sum_{\mathbf{y}} p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0) \log p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0), \end{aligned} \tag{A.10}$$

where $q(\mathbf{y}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0)}[p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0)]$; $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0)$ is the softmax probability over K categories in (A.9).

We then show how to obtain the quantities defined in Section 2.3.2. For the marginal distribution of $\mathbf{z}_0$,

$$\begin{aligned} q_{\boldsymbol{\phi}}(\mathbf{z}_0) &= \int_{\mathbf{x}, \mathbf{z}_{1:T}} Q_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z}_{0:T}) d\mathbf{x} d\mathbf{z}_{1:T} \\ &= \mathbb{E}_{q_{\text{data}}(\mathbf{x})}[q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})]. \end{aligned} \tag{A.11}$$

The entropy and conditional entropy of $\mathbf{z}_0$ are thus

$$\begin{aligned} \mathcal{H}(\mathbf{z}_0) &= -\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0)}[\log q_{\boldsymbol{\phi}}(\mathbf{z}_0)]; \\ \mathcal{H}(\mathbf{z}_0|\mathbf{x}) &= -\mathbb{E}_{Q_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z}_0)}[\log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})]. \end{aligned} \tag{A.12}$$

Taking together, we can then decompose the KL-Divergence, $D_{\mathrm{KL}}(Q_{\boldsymbol{\phi}} \| P_{\boldsymbol{\theta}})$ in (2.8) as

$$\begin{aligned} D_{\mathrm{KL}}(Q_{\boldsymbol{\phi}} \| P_{\boldsymbol{\theta}}) = \mathbb{E}_{Q_{\boldsymbol{\phi}}}\left[q_{\mathrm{data}}(\mathbf{x})\right] + \mathbb{E}_{Q_{\boldsymbol{\phi}}}\left[q_{\boldsymbol{\phi}}(\mathbf{z}_{0:T}|x)\right] \\ - \mathbb{E}_{Q_{\boldsymbol{\phi}}}\left[p_{\boldsymbol{\alpha}}(\mathbf{z}_{0:T})\right] - \mathbb{E}_{Q_{\boldsymbol{\phi}}}\left[p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0)\right], \end{aligned} \tag{A.13}$$

and further as

$$\begin{aligned} -\mathcal{H}(\mathbf{x}) + \sum_{t=0}^{T-1} \mathcal{H}(\mathbf{z}_{t+1}|\mathbf{z}_t) - \mathcal{H}(\mathbf{z}_0|\mathbf{x}) + \mathcal{H}(\mathbf{z}_0) - \mathcal{H}(\mathbf{z}_0) \\ - \mathbb{E}_{Q_{\boldsymbol{\phi}}}\left[p_{\boldsymbol{\alpha}}(\mathbf{z}_{0:T})\right] - \mathbb{E}_{Q_{\boldsymbol{\phi}}}\left[p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0)\right], \end{aligned} \tag{A.14}$$

by plugging in $\mathcal{H}(\mathbf{z}_0) - \mathcal{H}(\mathbf{z}_0) = 0$. Rearranging (A.14), we can obtain

$$\begin{aligned} D_{\mathrm{KL}}(Q_{\boldsymbol{\phi}} \| P_{\boldsymbol{\theta}}) = \mathcal{C} - \mathbb{E}_{Q_{\boldsymbol{\phi}}}\left[p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0)\right] \\ + D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}_0) \| p_{\boldsymbol{\alpha}}(\mathbf{z}_{0:T})) + \mathcal{I}(\mathbf{x}, \mathbf{z}_0), \end{aligned} \tag{A.15}$$

which leads to our result in (2.9).

A.1.5 Derivation of the Learning Gradient

Recall that we derive the extended version of (2.6) in Section A.1.2. To calculate the gradient of $\boldsymbol{\alpha}$, we have

$$\begin{aligned} \nabla_{\boldsymbol{\alpha}} \mathrm{ELBO}_{\mathrm{Diff},\boldsymbol{\theta},\boldsymbol{\phi}} &= \nabla_{\boldsymbol{\alpha}}\, \mathbb{E}\left[\sum_{t=0}^{T-1} \log p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1})\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1})\right], \end{aligned} \tag{A.16}$$

where $\mathbb{E}$ is the abbreviation of $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|x), q(\mathbf{z}_{1:T}|\mathbf{z}_0)}$; in practice, we use Monte-Carlo average to approximate the expectation. We next examine the learning gradient for each diffusion step t:

$$\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1}) = \nabla_{\boldsymbol{\alpha}} F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t) - \nabla_{\boldsymbol{\alpha}} \tilde{Z}_{\boldsymbol{\alpha},t}(\mathbf{z}_{t+1}), \tag{A.17}$$

where the quadratic term $\frac{1}{2\sigma_{t+1}^2}||\tilde{\mathbf{z}}_t - \mathbf{z}_{t+1}||^2$ is not related to $\boldsymbol{\alpha}$ and gets cancelled. According to the definition of the partition function in Section 2.2, we can similarly derive

$$\nabla_{\boldsymbol{\alpha}} \tilde{Z}_{\boldsymbol{\alpha},t}(\mathbf{z}_{t+1}) = \mathbb{E}_{p_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})} \left[\nabla_{\boldsymbol{\alpha}} F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t)\right], \tag{A.18}$$

as in (Pang et al., 2020a). For the prior model, we thus have

$$\nabla_{\boldsymbol{\alpha}} \text{ELBO}_t = \mathbb{E}_{q_{\boldsymbol{\phi}}(\tilde{\mathbf{z}}_t, \mathbf{z}_0|\mathbf{x})}[\nabla_{\boldsymbol{\alpha}} F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t)] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_{t+1}, \mathbf{z}_0|\mathbf{x}), p_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t|\mathbf{z}_{t+1})}[\nabla_{\boldsymbol{\alpha}} F_{\boldsymbol{\alpha}}(\tilde{\mathbf{z}}_t, t)], \tag{A.19}$$

where $q_{\boldsymbol{\phi}}(\tilde{\mathbf{z}}_t, \mathbf{z}_0|x) = q(\tilde{\mathbf{z}}_t|\mathbf{z}_0) q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})$. Note that we can sample $\mathbf{z}_t,\ t > 0$ directly from

$$q(\mathbf{z}_t|\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_t; \sqrt{\bar{\gamma}_t}\mathbf{z}_{t-1}, (1 - \bar{\gamma}_t)\mathbf{I}) \tag{A.20}$$

by merging the Gaussian noises during forward diffusion process; we denote $\gamma_t = 1 - \sigma_t^2$ and $\bar{\gamma}_t = \prod_{i=1}^{t} \gamma_t$.

For the encoder and decoder, based on (2.6) and (A.6), we have

$$\begin{aligned} \nabla_{\psi} \text{ELBO} = \nabla_{\psi}\, \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}[\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}_0) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})] \\ - \nabla_{\boldsymbol{\phi}}\, \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_{0:T}|\mathbf{x})} \left[\log p(\mathbf{z}_T) + \sum_{t=0}^{T-1} \log p_{\boldsymbol{\alpha}}(\mathbf{z}_t|\mathbf{z}_{t+1})\right], \end{aligned} \tag{A.21}$$

where the summation of energy terms provides extra guidance for the optimization of encoder.

A.2 Extra Experimental Details and Discussion

A.2.1 Network Architecture and Hyperparameters

We provide detailed network architecture for the latent space model of this chapter in Table A.1; we adopt the same architecture throughout the experiments. Spectral normalization (Miyato et al., 2018) is used to regularize parameters in linear layers. The

Layers	Output size	Note
Time Embedding		
Input: t	1	Index of diffusion step
Sin. embedding	200	
Linear, LReLU	200	negative_slope 0.2
Linear	200	
Input Embedding		
Input: $\mathbf{z}$	d_{lat}	
Linear, LReLU	200	negative_slope 0.2
Linear	200	
Context Embedding (for response generation only)		
Input: $\mathbf{z}_{\text{ctx}}$	512	ctx. embedding
Linear, LReLU	200	negative_slope 0.2
Linear	200	
LDEBM Prior		
Input: $\mathbf{z}, t$	1, d_{lat}	optional $\mathbf{z}_{\text{ctx}}$
Input: $^*\mathbf{z}_{\text{ctx}}$	512	
Embedding	200	Embedding of each input
Concatenate	400	w/o ctx.
	600	w/ ctx.
LReLU, Linear	200	negative_slope 0.2
N ResBlocks	200	LReLU, Linear + Input
LReLU, Linear	K	K class logits
Log-Sum-Exp	1	energy score

Table A.1: Network architecture for the LDEBM prior. N is set to 12 for all the experiments.

Dataset	d_{lat}	K	λ_1	λ_2	λ_3
2D Gaussian	2	16	1	0.05	0.05
2D Pinwheel	2	10	1	0.05	0.05
PTB	40	20	0.1	0.05	0.05
Jericho	40	20	0.1	0.05	0.05
DD-CLS	32	125	0.01	0.05	0.5
DD-GEN	32	125	1	0.05	0.05
SMD	32	125	10	10	5
Yelp	40	2	50	50	200
AGNews	20	4	1e-3	5	200

Table A.2: Hyperparameters of LDEBM. DD-CLS presents the set of hyperparameters used in unsupervised clustering on DD dataset. DD-GEN presents the set of hyperparameters used in conditional response generation on DD dataset.

encoder and decoder in all models are the same as in (Pang and Wu, 2021), implemented with a single-layer GRU with a hidden size of 512. The key hyperparameters of LDEBM for each dataset are listed in Table A.2. Of note, we use the same dimension of the latent space as in (Pang and Wu, 2021) for a fair comparison.

λ_1 is the hyperparameter that reweights the term in (A.6); it generally controls how fast $q_{\boldsymbol{\phi}}$ and $p_{\boldsymbol{\theta}}$ run towards each other. λ_2 refers to the hyperparameter in (2.9); it controls the trade-off between the compressivity of $\mathbf{z}_0$ about $\mathbf{x}$ and its expressivity to $\mathbf{y}$. λ_3 controls the weight of classification loss mentioned in Section 2.3.3; recall that we use pseudo-label $\hat{\mathbf{y}}$ inferred by the geometric clustering algorithm or the ground-truth label $\mathbf{y}$ to supervise $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0)$ in our modeling. For controllable generation and semi-supervised classification, we find it important to have a larger weight on the classification loss so that the model is forced to capture the major modes of the data.

For optimization, we use Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for all the experiments. On all the datasets but 2D synthetic datasets and AGNews dataset, we use a batch size of 128 and a constant learning rate of $1e-3$ for encoder and decoder without weight decay. For LDEBM, we use a constant learning rate of $1e-4$. We use a larger batch size of 1000 on 2D synthetic datasets. On the AGNews dataset, we use the same set of hyperparameters as in (Pang and Wu, 2021) for optimization. The batch size is set to 200; the initial learning rate is $1e-4$ for encoder

and decoder, and $1e-5$ for LDEBM. Learning rates are exponentially decayed with a decay rate of 0.998 for each model. Encoder and LDEBM have a weight decay rate of $2e-3$ and $1e-3$, respectively.

A.2.2 Experiment Settings and Baselines

Experiment Settings For generative modeling, following previous methods (Shi et al., 2020; Pang and Wu, 2021), the NLL term is computed with importance sampling (Burda et al., 2016) using 500 importance samples. To compute rPPL, we set the generated sample size as $40,000$, which is the same size as PTB training set. We recruit ASGD Weight-Dropped LSTM (Merity et al., 2018) to compute rPPL as in previous works.

In terms of conditional response generation, for word-embedding-based evaluation on SMD and DD, we use the publicly available GloVe (Pennington et al., 2014) word embeddings of 300 dimension trained on 840B tokens, and report the score from 1 response per context. We use a context window size of 5 during training and evaluation.

The maximum length of each sentence is set to 40 words for most datasets and 70 words for the JerichoWorld dataset. On JerichoWorld dataset, we extract the description of each state as the text data.

Baselines On **PTB**, DD and **SMD**, our model is compared with the following baselines: (1) RNNLM (Mikolov et al., 2010), the language model implemented with GRU (Cho et al., 2014); (2) AE (Vincent et al., 2010), the deterministic auto-encoder which has no regularization to the latent space; (3) DAE, the AE with a discrete latent space; (4) VAE (Kingma and Welling, 2014a), the vanilla VAE with a continuous latent space and a non-informative Gaussian prior; (5) DVAE, the VAE with a discrete latent space; (6) DI-VAE (Zhao et al., 2018b), a DVAE variant with a mutual information term between the observed piece of text $\mathbf{x}$ and its inferred latent variable $\mathbf{z}$; (7) semi-VAE (Kingma et al., 2014), the semi-supervised VAE model with independent discrete and continuous latent variables; (8) GM-VAE, the VAE with a Gaussian mixture prior; (9) DGM-VAE (Shi et al., 2020), the GM-VAE with a dispersion term that avoids the mode-

collapse of Gaussian mixture prior; (10) semi-VAE $+\mathcal{I}(\mathbf{x}, \mathbf{y})$, GM-VAE $+\mathcal{I}(\mathbf{x}, \mathbf{y})$, DGM-VAE $+\mathcal{I}(\mathbf{x}, \mathbf{y})$, are the same models as (7), (8), and (9) respectively, but with a mutual information term between $\mathbf{x}$ and $\mathbf{y}$ computed using separate inference networks for $\mathbf{y}$ and $\mathbf{z}$. We compare with the close competitors (11) SVEBM, the symbol-vector coupling prior model and (12) SVEBM-IB, SVEBM with regularization based on information-bottleneck.

On **Yelp** dataset, we additionally include text conditional GAN (Subramanian et al., 2018) as a baseline for controllable generation. On **AGNews** dataset, we further compare our model to VAMPIRE (Gururangan et al., 2019), a VAE-based semi-supervised text learning model. Other baselines include its supervised learning variants: (1) the model trained with Glove embedding pre-trained on 840 billion words (Glove-OD); (2) the model trained with Glove embedding on in-domain unlabeled data (Glove-ID). We also include more recent baselines such as Hard EM and CatVAE (Jin et al., 2020) that improve over VAMPIRE.

A.2.3 Extra Details for Experiments

More Ablation Study We conduct additional experiments on both PTB and DD datasets to inspect the contribution of the proposed techniques. In Section 2.4.1, we have reported results on PTB and datasets of Ours w/o GC which represents the model with Information Bottleneck but without Geometric Clustering (GC); Ours denotes the full model.

We further conduct experiments on the proposed model without using IB or GC. We observe that the proposed model using only diffusion-based sampling scheme has a rPPL of 166.26, BLEU of 11.30, wKL of 0.07 and NLL of 80.76 on PTB; it has a MI of 0.01, BLEU of 19.28, Act. of 0.12 and Emo. of 0.06 on DD, which is better than SVEBMs (see Table 2.1 and Table 2.3 in Section 2.4.1).

We also add GC to SVEBM (denoted as SVE-IB w/ GC). We find that SVE-IB w/ GC does perform better compared with SVE-IB, showing a rPPL of 179.95, BLEU

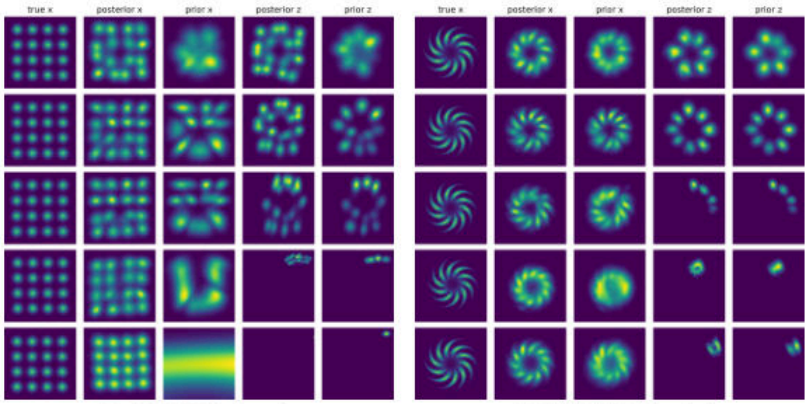
of 10.08, wKL of 0.15 and NLL of 93.28 on PTB; it has a MI of 2.88, BLEU of 11.75, Act. of 0.61 and Emo. of 0.60 on DD. Notably, SVE-IB w/ GC is still inferior to LDEBMs.

In summary, we think these additional experiments (1) emphasize the importance of our diffusion-based modeling approach, and (2) demonstrate the effectiveness of GC as additional regularization.

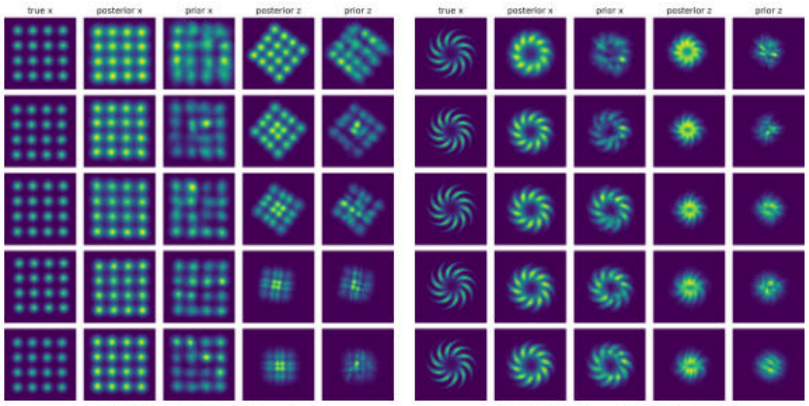
2D Synthetic Data We provide the full evolution of SVEBM-IB and our models as visualized in Figure A.1. Though SVEBM-IB can capture some regularities of the data in the early stages of training, the model is prone to collapse due to the degenerated sampling quality. This features an exploding KL-term and leads to poor performance on generation. Our preliminary experiments indicate that common deep learning heuristics for improving the model capacity barely help. These include but are not limited to increasing the number of parameters in SVEBM, *i.e.*, using larger models, and adopting deliberately designed activation functions or normalization modules. LDEBM w/o geometric clustering has a better sampling quality and effectively mitigates the instability in training. However, the mode coverage is not satisfying in data space; the structure is unclear in latent space. In contrast, LDEBM w/ geometric clustering shows superior generation quality with better mode coverage. It demonstrates a better-structured latent space.

Sentence Completion To perform sentence completion, we adopt a two-stage training scheme. We first train the LDEBM with inference, prior and generation models on the JerichoWorld dataset. After the first-stage training, the parameters of prior, inference and generation models are fixed. We then train a shallow MLP in the latent space to project the inferred posterior $\mathbf{z}_0$ to a disentangled space; the variables in the projected $\mathbf{z}_0$ can be grouped as: (a) the representation of observable words $\hat{\mathbf{z}}_{\text{obs}}$ in the input sentence and (b) the representation of unknown words $\hat{\mathbf{z}}_{\text{unk}}$. Conditional sampling in the latent space then refers to updating $\hat{\mathbf{z}}_{\text{unk}}$ based on the fixed $\hat{\mathbf{z}}_{\text{obs}}$ by running Langevin dynamics guided by the latent space model.

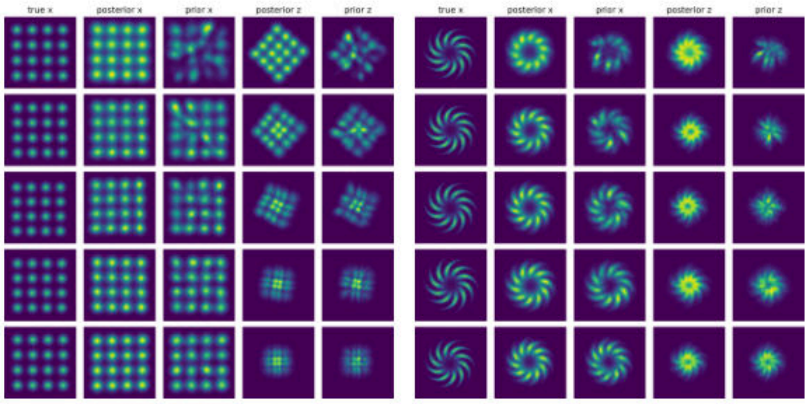
We mask half of the words in the sentences with `<unk>` token to prepare the inputs.



(a) SVEBM-IB Gaussian (b) SVEBM-IB Pinwheel



(c) LDEBM w/o PL Gaussian (d) LDEBM w/o PL Pinwheel



(e) LDEBM Gaussian (f) LDEBM Pinwheel

Figure A.1: Full evolution of SVEBM-IB and our models. In each sub-figure, we provide the typical states of the model trained on the corresponding dataset, sequentially from the top row to the bottom row.

In the second stage of training, we supervise the MLP by minimizing the reconstruction error between only the observable words of the input the sentence and the corresponding outputs of the model.

Sentence Sentiment Control Recall that in our formulation only $\mathbf{z}_0$ is connected to $\mathbf{y}$. We therefore condition only the final reverse diffusion step $[\mathbf{z}_0|\mathbf{z}_1]$ on $\mathbf{y}$ when performing controllable generation, *i.e.*, using $\mathbf{y}$ to guide the generation only when $t = 0$ in Algorithm 2. This can be a bit counter-intuitive since no label information is injected in previous reverse steps. Theoretically, $\mathbf{y}$ and $\mathbf{z}_{1:T}$ are independent given $\mathbf{z}_0$ in our formulation; however, we empirically observe that $\mathbf{y}$ and $\mathbf{z}_t$ for $t > 0$ are nearly independent even marginally after we integrating out $\mathbf{z}_{0:t-1}$ in our model. In other words, $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t),\ t > 0$ are in general non-informative since adding noise in the latent space could be much more corrupting than adding noise in the data space. The model learns to enjoy the less multi-modal energy landscape in previous reverse steps; it then seeks the given mode only in the most informative final reverse step. We examine $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t),\ t > 0$ for the model trained on Yelp dataset by marginalizing out $\mathbf{z}_{t-1}$ of $p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}_{t-1}|\mathbf{z}_t),\ t > 0$. For example, for $t = 1$, we may calculate

$$\begin{aligned} p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_1) &= \int_{\mathbf{z}_0} p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0) p_{\boldsymbol{\alpha}}(\mathbf{z}_0|\mathbf{z}_1) d\mathbf{z}_0 \\ &= \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{z}_0|\mathbf{z}_1)} \left[p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0) \right] \\ &\approx \frac{1}{M} \sum_{i=1}^{M} p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_0^{(i)}). \end{aligned} \tag{A.22}$$

See Figure A.2 for the visualization of $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t)$ over t.

A more intuitive method is to use the data label $\mathbf{y}$ to supervise each $[\mathbf{y}, \mathbf{z}_t|\mathbf{z}_{t+1}]$, so that we can propagate the label information through the whole trajectory. Given $\mathbf{z}_0$, $\mathbf{y}$ and $\mathbf{z}_{1:T}$ are independent. But if we marginalize out $\mathbf{z}_0$, $\mathbf{y}$ will depend on $\mathbf{z}_1$. Similarly, if we continue to marginalize out $\mathbf{z}_1$, $\mathbf{y}$ will depend on $\mathbf{z}_2$. Repeating this process results in $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t)$ for each t after integrating out $\mathbf{z}_{0:t-1}$. Supervising $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t),\ t > 0$ using $\mathbf{y}$ therefore effectively encodes the label information into the whole trajectory.

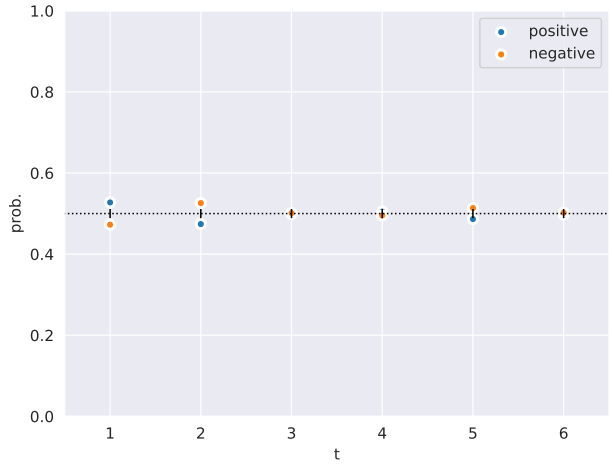


Figure A.2: Visualization of $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t)$ over t. $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t)$ is constantly around the probability of 0.5 over t.

While the marginalization can be difficult, we may approximate it by learning the amortized version of $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{z}_t),\ t > 0$ as $p_{\boldsymbol{\alpha}}(\mathbf{y}, \mathbf{z}_{t-1} = \mu_{\boldsymbol{\phi},t-1}|\mathbf{z}_t),\ t > 0$, where $\mu_{\boldsymbol{\phi},t}$ is the posterior mean of $\mathbf{z}_t$. We may therefore circumvent the intractable integration in practice and guide the whole trajectory for controllable generation.

APPENDIX B

Derivations and Experimental Details for Chapter 3

B.1 Category List

We include 30 categories for training and 10 held-out categories for testing in our game; see Table B.1.

B.2 Category Embedding for Other Game Settings

Figure B.1 shows the t-SNE visualization for other game settings. Agents under *max-step*, *sender-fixed*, and *one-step* settings fail to form clear boundaries between different categories, which makes it hard to observe semantic relations.

training categories							
apple	axe	bell	blimp	camel	cannon	car_(sedan)	chicken
cow	cup	deer	dolphin	duck	frog	giraffe	guitar
hamburger	horse	knife	mushroom	pig	pistol	pizza	rabbit
sailboat	seal	shark	sheep	snail	turtle		
unseen categories							
pear	hammer	pickup truck	songbird	violin	sword	elephant	fish
penguin	swan						

Table B.1: Categories used in the visual communication game

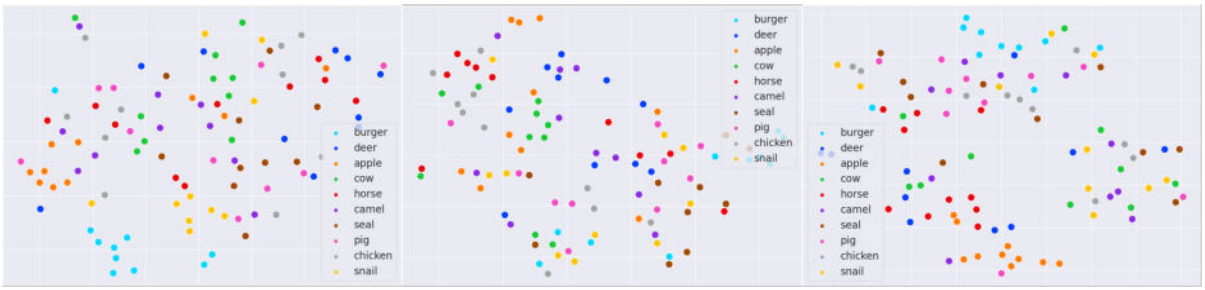


Figure B.1: t-SNE of visual embedding. These embeddings are extracted from the fine-tuned VGGNet used for evolved sketch classification under the *max-step* (left), *sender-fixed* (middle), and *one-step* (right) settings, respectively. Neither of them forms a clear boundary between different categories.

B.3 Learning Objectives and Training Algorithm

Agents are trained jointly to maximize the objective

$$\pi_S^*, \pi_R^* = \arg\max_{\pi_S, \pi_R} \mathbb{E}_{\tau \sim (\pi_S, \pi_R)}[\sum_{t=0} \gamma^t r_t], \tag{B.1}$$

where $\tau = \{\mathbf{C}_0, \mathbf{a}_{S0}, \mathbf{C}_1, \mathbf{a}_{R1}, \mathbf{a}_{S1}, \dots\}$ is the simulated episodic trajectory. To further expand the objective,

$$\begin{aligned}
\mathbb{E}_{(\pi_S, \pi_R)}[\sum_{t=0} \gamma^t r_t] = & \int p(\mathbf{I}_S) p(\mathbf{I}_R^1) \dots p(\mathbf{I}_R^M) p(\mathbf{C}_0) \\
& \int \pi_S(\mathbf{a}_{S0} | \mathbf{I}_S, \mathbf{C}_0) \pi_R(\mathbf{a}_{R1} | \mathbf{C}_0, G(\mathbf{C}_0, \mathbf{a}_{S0}), \mathbf{I}_R^1, \dots, \mathbf{I}_R^M) \\
& \cdot \left[r_0 + \mathbb{E}_{(\pi_S, \pi_R)} \left[\sum_{t=1} \gamma^t r_t \right] \right] d\mathbf{a}_{S0} d\mathbf{a}_{R1} d\mathbf{I}_S d\mathbf{I}_R^1 \dots d\mathbf{I}_R^M d\mathbf{C}_0 \\
= & \mathbb{E}_{\mathbf{I}_S, \mathbf{I}_R^1, \dots, \mathbf{I}_R^M, \mathbf{C}_0} \left[\mathbb{E}_{(\pi_S, \pi_R)} \left[r_0 + \mathbb{E}_{(\pi_S, \pi_R)} \left[\sum_{t=1} \gamma^t r_t \right] \right] \right].
\end{aligned} \tag{B.2}$$

We calculate $\mathbb{E}_{\mathbf{I}_S, \mathbf{I}_R^1, \dots, \mathbf{I}_R^M, \mathbf{C}_0}[\cdot]$ by sampling $\mathbf{I}_S$, $\mathbf{I}_R$, and initializing $\mathbf{C}_0$ to blank at each round. We represent the $\mathbb{E}_{(\pi_S, \pi_R)}[\cdot]$ as $\mathcal{V}(\mathbf{X}_0)$ and use $V_\lambda(\mathbf{X}_1)$ to estimate the reward expectation $\mathbb{E}_{(\pi_S, \pi_R)}[\sum_{t=1} \gamma^t r_t]$:

$$\mathcal{V}(\mathbf{X}_0) = \mathbb{E}_{(\pi_S(\mathbf{a}_{S0} | \mathbf{I}_S, \mathbf{C}_0), \pi_R(\mathbf{a}_{R1} | \mathbf{C}_0, G(\mathbf{C}_0, \mathbf{a}_{S0}), \mathbf{I}_R^1, \dots, \mathbf{I}_R^M))}[(r_0 + \gamma \delta(\mathbf{a}_{R1}) V_\lambda(\mathbf{X}_1)], \tag{B.3}$$

where $\mathbf{X}_t = [\mathbf{I}_S, \mathbf{I}_R^1, \ldots, \mathbf{I}_R^M, \mathbf{C}_t, \mathbf{C}_{t+1}], t = 0, 1 \ldots,$ $\delta(\cdot)$ is the Dirac delta function that returns 1 when the action is *wait* and 0 otherwise.

The sender policy is parametrized as a Gaussian distribution,

$$\pi_S = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}^2), \quad \boldsymbol{\mu}_t = h_S(\mathbf{I}_S, \mathbf{C}_t), \quad \boldsymbol{\sigma}^2 = c \cdot \mathbf{I}, \tag{B.4}$$

such that $\mathbf{a}_{S0}$ can be written as

$$\mathbf{a}_{S0} = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \tag{B.5}$$

Therefore, we can expand $\mathcal{V}(\mathbf{X}_0)$ as

$$\begin{aligned}
\mathcal{V}(\mathbf{X}_0) &= \int \pi_S(\mathbf{a}_{S0}|\mathbf{C}_0, \mathbf{I}_S)\mathbb{E}_{\pi_R(\mathbf{a}_{R1}|\mathbf{C}_0, G(\mathbf{C}_0, \mathbf{a}_{S0}), \mathbf{I}_R^1, \ldots, \mathbf{I}_R^M)} \cdot [r_0 + \gamma\delta(\mathbf{a}_{R1})V_\lambda(\mathbf{X}_1)]d\mathbf{a}_{S0} \\
&= \int p(\epsilon)\mathbb{E}_{\pi_R(\mathbf{a}_{R1}|\mathbf{C}_0, G(\mathbf{C}_0, \mu_0 + \sigma\epsilon), \mathbf{I}_R^1, \ldots, \mathbf{I}_R^M)} \cdot [r_0 + \gamma\delta(\mathbf{a}_{R1})V_\lambda(\mathbf{X}_1)]d\epsilon \\
&= \mathbb{E}_\epsilon[\mathbb{E}_{\pi_R}[r_0 + \gamma\delta(\mathbf{a}_{R1})V_\lambda(\mathbf{X}_1)]].
\end{aligned} \tag{B.6}$$

$\mathbb{E}_\epsilon[\cdot]$ is approximated with a point estimate. Since π_R is a categorical distribution, we expand $\mathbb{E}_{\pi_R}$ as

$$\mathbb{E}_{\pi_R}[r_0 + \gamma\delta(\mathbf{a}_{R1})V_\lambda(\mathbf{X}_1) = \sum_{j=1}^{M+1} p(\mathbf{a}_{R1}^j)[r_0^j + \gamma\delta(\mathbf{a}_{R1})V_\lambda(\mathbf{X}_1)]. \tag{B.7}$$

$V_\lambda(\mathbf{X}_t)$ in (B.3) is an eligibility trace approximation of the ground-truth value function (Sutton and Barto, 2018). Considering the early termination in our setting, we set the time step when the receiver makes the prediction as T_{choice}. When t is the time step less or equal than T_{choice}, V_λ mixes Monte Carlo estimate at different roll-out lengths.

Otherwise, we only have an estimated value $v_{\boldsymbol{\phi}}(\mathbf{X}_t)$.

$$V_\lambda(\mathbf{X}_t) = \begin{cases} (1-\lambda)\sum_{n=1}^{H-1}\lambda^{n-1}V_N^n(\mathbf{X}_t) + \lambda^{H-1}V_N^H(\mathbf{X}_t) \\ \qquad\qquad\qquad\qquad \text{if } t \leq T_{\text{choice}} \\ v_{\boldsymbol{\phi}}(\mathbf{X}_t) \qquad\qquad\qquad \text{otherwise,} \end{cases} \tag{B.8}$$

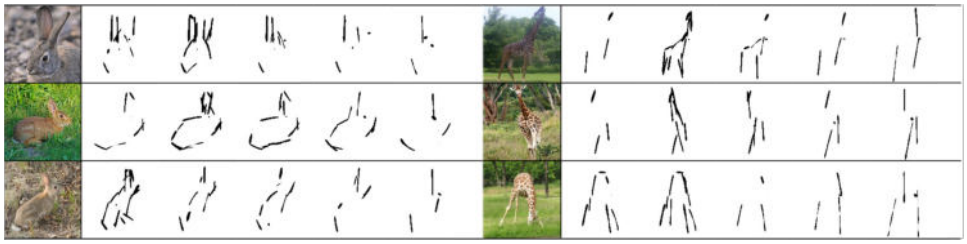
where $H = T_{\text{choice}} - t + 1$, and $V_N^k(\mathbf{X}_t)$ is the Monte Carlo estimate at k roll-out lengths. $V_N^k(\mathbf{X}_t) = \mathbb{E}_{\pi_S,\pi_R}[\sum_{n=t}^{h-1}\gamma^{n-t}r_n + \gamma^{h-t}\delta(\mathbf{a}_{Rh})v_{\boldsymbol{\phi}}(\mathbf{X}_h)]$, with $h = \min(t+k, T_{\text{choice}})$ being the maximal timestep. Due to the error reduction property (Sutton and Barto, 2018), the eligibility trace estimation $V_\lambda(\cdot)$ is less biased than $v_{\boldsymbol{\phi}}(\cdot)$. When regressing $v_{\boldsymbol{\phi}}(\mathbf{X}_t)$ towards the bootstrapped $V_\lambda(\mathbf{X}_t)$,

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}} \mathbb{E}_{\pi_S,\pi_R}[\sum\nolimits_t \frac{1}{2}||v_{\boldsymbol{\phi}}(\mathbf{X}_t) - V_\lambda(\mathbf{X}_t)||^2]. \tag{B.9}$$

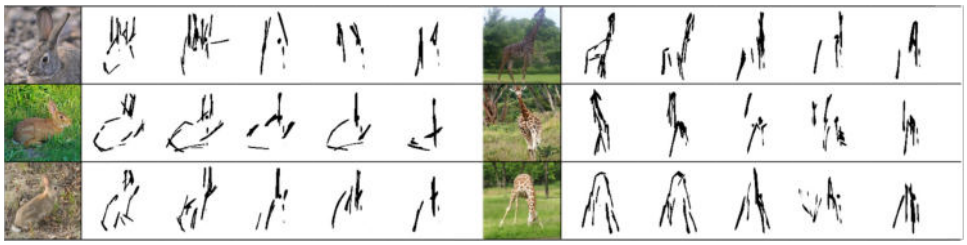
$v_{\boldsymbol{\phi}}(\mathbf{X}_t)$ will be improved towards the fixed point.

B.4 Visualizing Sketch Evolution

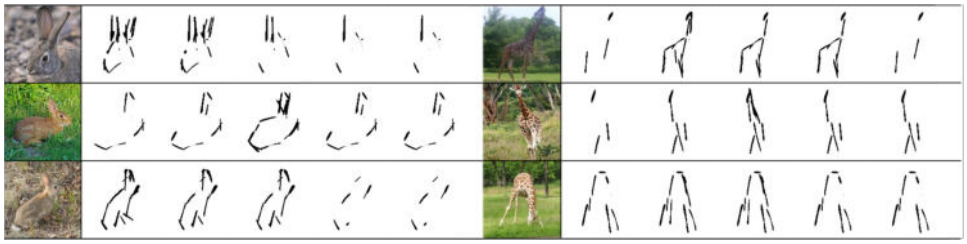
Visualizing the evolution process helps us understand what the agents have learned through communication regarding different categories. By comparing the evolved sketches with the intermediate results, we can know (i) how the agents abstract the sketch, (ii) which parts of the visual concept they highlight, and (iii) which parts are de-emphasized. Figure B.2, Figure B.3, and Figure B.4 show some evolution examples under different settings. Agents under *max-step* seem to abstract their drawings by repeatedly placing new strokes near old strokes, resulting in bold drawings. The number of strokes under *sender-fixed* gradually decreases, but the way of the drawing will not change. Senders under *one-step* change more wildly but cannot form a consistent drawing behavior. Overall, compared with the *complete* setting, agents under the control settings do not form patterns to draw sketches, which echoes their relatively low classification results.



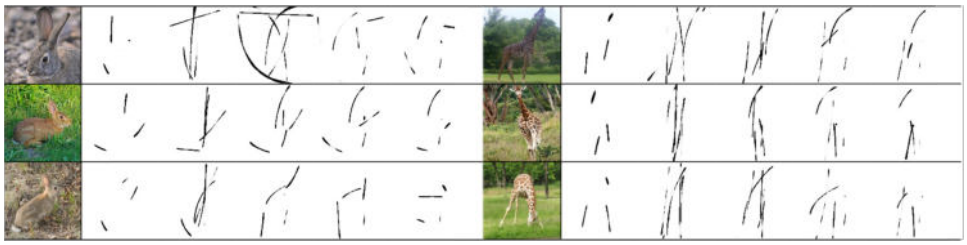
(a) *complete* example 1



(b) *max-step* example 1

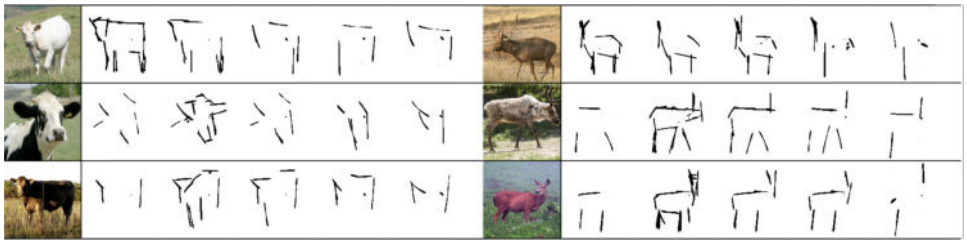


(c) *sender-fixed* example 1

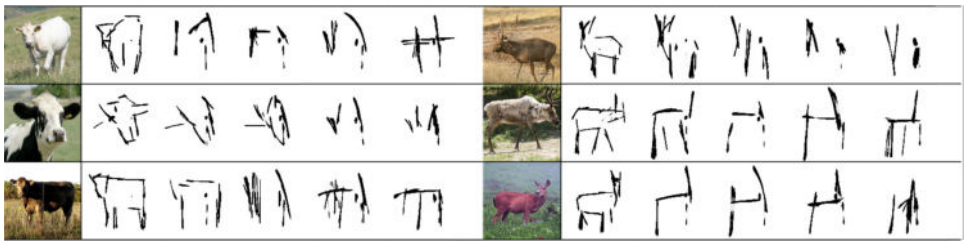


(d) *one-step* example 1

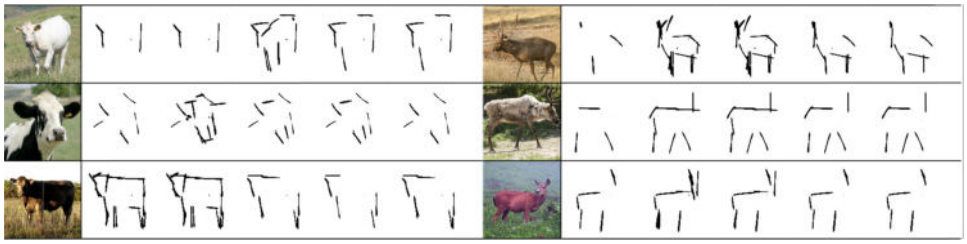
Figure B.2: Evolution of *rabbit* and *giraffe* under different settings.



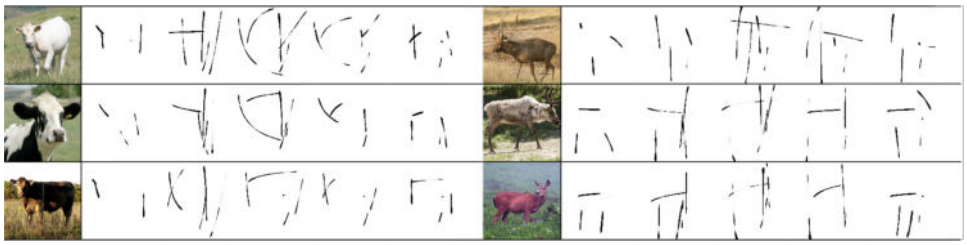
(a) *complete* example 2



(b) *max-step* example 2

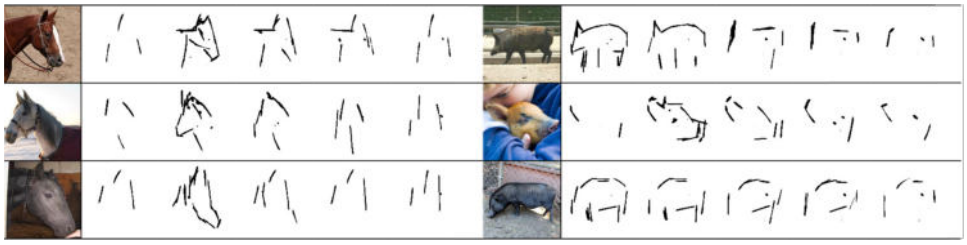


(c) *sender-fixed* example 2

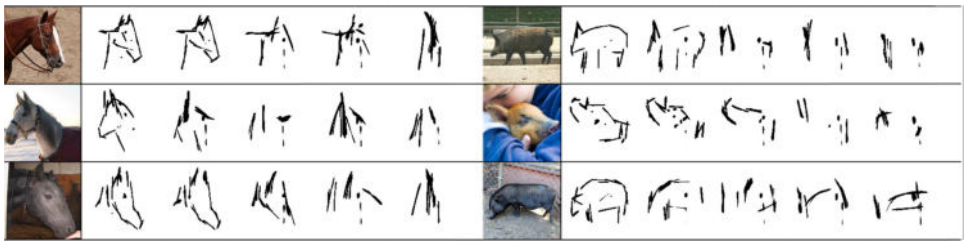


(d) *one-step* example 2

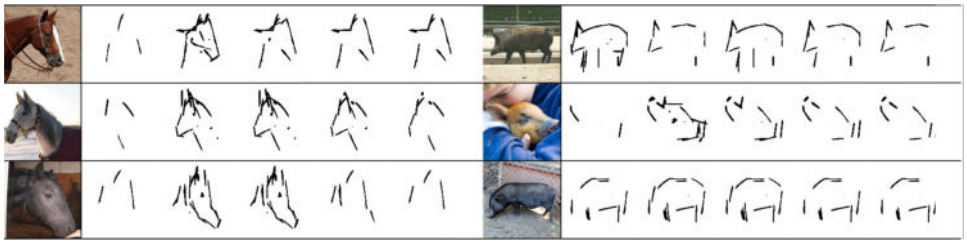
Figure B.3: Evolution of *cow* and *deer* under different settings.



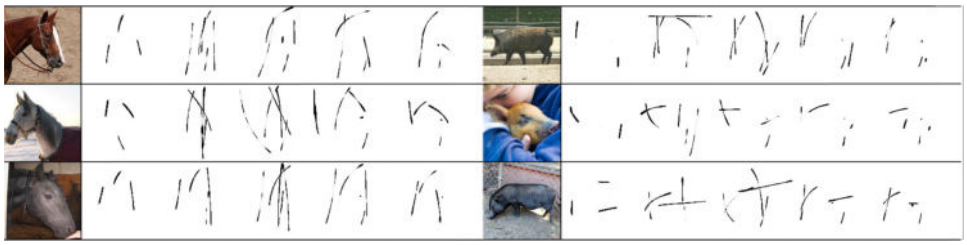
(a) *complete* example 3



(b) *max-step* example 3



(c) *sender-fixed* example 3



(d) *one-step* example 3

Figure B.4: Evolution of *horse* and *pig* under different settings.

APPENDIX C

Derivations and Experimental Details for Chapter 4

C.1 Dataset Details

C.1.1 Caltech-UCSD Birds-200-2011

Birds dataset consists of 11,788 images of 200 classes of birds annotated with high-quality segmentation masks. Each image is further annotated with 15 part locations, 312 binary attributes, and 1 bounding box. We use the provided bounding box to extract a center square from the image, and scale it to 128×128 pixels. Each scene contains exactly one foreground object.

C.1.2 Stanford Dogs

Dogs dataset consists of 20,580 images of 120 classes annotated with bounding boxes. We first use the provided bounding box to extract the center square, and then scale it to 128×128 pixels. We approximate ground-truth masks for the pre-processed images with Mask R-CNN (He et al., 2017), pre-trained on the MS COCO (Lin et al., 2014) dataset with a ResNet-101 (He et al., 2016) backend. The pretrained model is acquired from the detectron2 toolkit (Wu et al., 2019). We exclude the images where no dog is detected. We then manually exclude those images where the foreground object has occupied more than $\sim 90\%$ of the image, those with poor masks, and those with significant foreground distractors such as humans (see Figure C.1). The filtering strategy results in 5,024 images with a clear foreground-background setup and high-quality mask.

Figure C.1: Examples of excluded images. From left to right: (i) image with a foreground object that occupied too much space, (ii) image with a low-quality mask, and (iii) image with significant foreground distractors.

C.1.3 Stanford Cars

Cars dataset consists of 16,185 images of 196 classes annotated with bounding boxes. Though also being primarily designed for fine-grained categorization, it has a much clearer foreground-background setup compared with the Dogs dataset. We employ a similar process as used for Dogs dataset to approximate the ground-truth masks, and only exclude those images where cars are not properly detected. It finally produces 12,322 images for our experiments.

C.1.4 CLEVR6

CLEVR6 dataset is a subset of the original CLEVR dataset (Johnson et al., 2017) with masks, generated by Greff et al. (2019). We follow the evaluation protocol adopted by IODINE (Greff et al., 2019) and Slot-attention (Locatello et al., 2020), which takes the first 70K samples from CLEVR. These samples are then filtered to only include scenes with at most 6 objects. Additionally, we perform a center square crop of 192×192 from the original 240×320 image, and scale it to 128×128 pixels. The resulting CLEVR6 dataset contains 3-6 foreground objects that could be with partial occlusion and truncation in each visual scene.

C.1.5 Textured Multi-dSprites

TM-dSprites dataset, which is based on the dSprites dataset (Matthey et al., 2017) and Textured MNIST (Greff et al., 2016), consists of 20,000 images with a resolution of

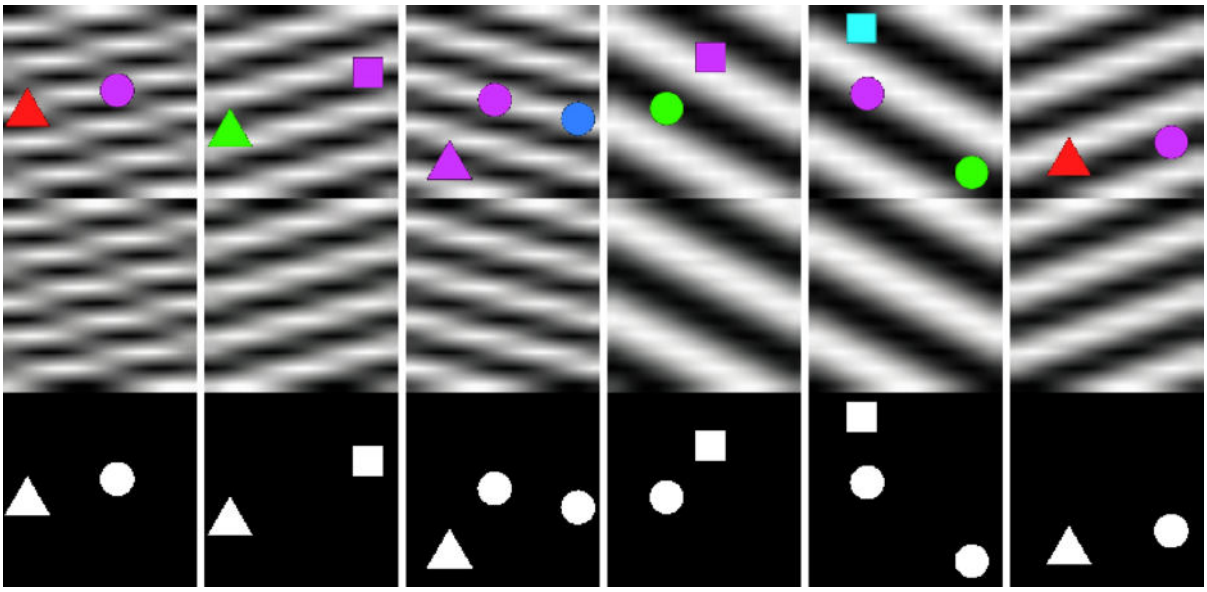


Figure C.2: Samples from TM-dSprites. From top to bottom: (1) observed images, (ii) background textures, and (iii) ground-truth masks.

Dataset	Foreground	Background	Pixel Re-assignment
Birds	256	256	512
Dogs	256	256	512
Cars	256	192	512
CLEVR6	256	2	256
TM-dSprites	256	4	1024

Table C.1: Dimension of latent variables on each dataset.

128×128. Each image contains 2-3 random sprites, which vary in terms of shape (square, circle, or triangle), color (uniform saturated colors), and position (continuous). The background regions are borrowed from Textured MNIST dataset (Greff et al., 2016). The textures for the background are randomly shifted samples from a bank of 20 sinusoidal textures with different frequencies and orientations. We adopt a simpler foreground setting compared with the vanilla Multi-dSprites dataset used by Greff et al. (2019), *i.e.*, the foreground objects are not occluded as the dataset is designed to emphasize the background part. Some samples are presented in Figure C.2.

C.2 Details on Models and Hyperparameters

Architecture We use the same overall architecture for different datasets (while the size of latent variables may vary). The details for the generators and LEBMs are summarized in Table C.1 and Table C.2.

Hyperparameters and Training Details For the Langevin dynamics sampling (Welling and Teh, 2011), we use K_0 and K_1 to denote the number of prior and posterior sampling steps with step sizes s_0 and s_1 respectively. Our hyperparameter choices are: $K_0 = 60, K_1 = 40, s_0 = 0.4$ and $s_1 = 0.1$. These are identical across different datasets. During testing, we set the posterior sampling steps to 300 for Dogs and Cars, and 2.5K, 5K and 5K for Birds, CLEVR6 and TM-dSprites respectively. The parameters of the generators and LEBMs are initialized with orthogonal initialization (Saxe et al., 2014). The gain is set to 1.0 for all the models. We use the ADAM optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Generators are trained with a constant learning rate of 0.0001, and LEBMs with 0.00002. We run experiments on a single V100 GPU with 16GB of RAM and with a batch size of 48. We set the maximum training iterations to 10K and run for at most 48hrs for each dataset.

C.3 Details on Learning Objective and Regularization

C.3.1 Learning Objective

Derivation of Surrogate Learning Objective $\mathcal{J}(\boldsymbol{\theta}) = \mathbf{E}_{\mathbf{w}\sim p_{\boldsymbol{\beta}}(\mathbf{w}|\mathbf{x},\mathbf{z})}\left[\mathcal{L}(\boldsymbol{\theta})\right]$ is the conditional expectation of $\mathbf{w}$,

$$\begin{aligned}\mathcal{J}(\boldsymbol{\theta}) &= \mathbf{E}_{\mathbf{w}\sim p_{\boldsymbol{\beta}}(\mathbf{w}|\mathbf{x},\mathbf{z})}\left[\mathcal{L}(\boldsymbol{\theta})\right] \\ &= \log p_{\boldsymbol{\alpha}}(\mathbf{z}) + \mathbf{E}\left[\sum_{i=1}^{D}\sum_{k=1}^{2} w_{ik}\left(\log \pi_{ik} + \log p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)\right)\right] \\ &= \log p_{\boldsymbol{\alpha}}(\mathbf{z}) + \sum_{i=1}^{D}\sum_{k=1}^{2} \mathbf{E}\left[w_{ik}\right]\left(\log \pi_{ik} + \log p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)\right),\end{aligned} \tag{C.1}$$

Layers	In-Out size	Comment
LEBM for Foreground/Background Models		
Input: $\mathbf{z}$	D^*	
Linear, LReLU	200	
Linear, LReLU	200	
Linear	$K^\dagger$	
LEBM for Pixel Re-assignment Model		
Input: $\mathbf{z}$	D^*	
Linear, LReLU	200	
Linear, LReLU	200	
Linear, LReLU	200	
Linear	1	
Generator for Foreground/Background Model and Re-assignment Model		
Input: $\mathbf{z}$	D^*	
Linear, LReLU	$4 \times 4 \times 128$	reshaped output
UpConv3x3Norm, LReLU	$8 \times 8 \times 1024$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$16 \times 16 \times 512$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$32 \times 32 \times 256$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$64 \times 64 \times 128$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$128 \times 128 \times 64$	stride 1 & padding 1
Conv3x3	$128 \times 128 \times (3+1)$	RGB & Mask
	$128 \times 128 \times 2$	Re-assignment grid
Auxiliary classifier for Foreground/Background Model		
Input: $\mathbf{x}$	$128 \times 128 \times 3$	generated image
Conv4x4Norm, LReLU	$64 \times 64 \times 64$	stride 2 & padding 1
Conv4x4Norm, LReLU	$32 \times 32 \times 128$	stride 2 & padding 1
Conv4x4Norm, LReLU	$16 \times 16 \times 256$	stride 2 & padding 1
Conv4x4Norm, LReLU	$8 \times 8 \times 512$	stride 2 & padding 1
Conv4x4Norm, LReLU	$4 \times 4 \times 1024$	stride 2 & padding 1
Conv4x4	$1 \times 1 \times K^\dagger$	

Table C.2: Architecture of the generators, LEBMs and auxiliary classifiers (see Section C.3.2). UpConv3x3Norm denotes a Upsampling-Convolutional-InstanceNorm layer with a convolution kernel size of 3. Similarly, Conv4x4Norm denotes a Convolutional-InstanceNorm layer with a kernel size of 4. LReLU denotes the Leaky-ReLU activation function. The leak factor for LReLU is 0.2 in LEBMs and auxiliary classifiers, and 0.01 in generators. $^*D$ represents the dimensions of the latent variables for different datasets; see Table C.1. $\dagger K$ represents the pre-specified category number for latent variables. We use 200 for both the foreground and background LEBMs on real-world datasets, and 30 and 10 in the foreground and background LEBMs on multi-object datasets respectively.

where $\mathbf{E}$ is the conditional expectation of $\mathbf{w}$. Recall that $w_{ik} \in \{0, 1\}$. The expectation becomes

$$\begin{aligned} \mathbf{E}\left[w_{ik}\right] &= 0 \times p(w_{ik} = 0|\mathbf{x}_i, \mathbf{z}) + 1 \times p(w_{ik} = 1|\mathbf{x}_i, \mathbf{z}) \\ &= \gamma_{ik}, \end{aligned} \tag{C.2}$$

which is the posterior responsibility of w_{ik}. We can further decompose $\mathcal{J}(\boldsymbol{\theta})$ into

$$\mathcal{J}(\boldsymbol{\theta}) = \underbrace{\log p_{\boldsymbol{\alpha}}(\mathbf{z})}_{\text{objective for LEBM}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2}\gamma_{ik}\log\pi_{ik}}_{\text{foreground-background partitioning}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2}\gamma_{ik}\log p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_k)}_{\text{objective for image generation}}. \tag{C.3}$$

Understanding the Optimization Process Note that the surrogate learning objective is an expectation w.r.t $\mathbf{z}$,

$$\max_{\boldsymbol{\theta}} \ \mathbf{E}_{\mathbf{z}\sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\left[\mathcal{J}(\boldsymbol{\theta})\right], \ \text{s.t.} \ \forall i, \sum_{k=1}^{2}\pi_{ik} = 1, \tag{C.4}$$

which is generally intractable to calculate. We therefore need to approximate the expectation by sampling from the distributions, and calculating the Monte Carlo average. In practice, this can be done by gradient-based MCMC sampling method, such as Langevin Dynamics (Welling and Teh, 2011).

Given $\mathbf{x}$, we have $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \propto p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\alpha}}(\mathbf{z})$. Note that

$$\begin{aligned} \nabla_{\mathbf{z}}\log p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}) &= \frac{1}{p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})}\nabla_{\mathbf{z}}p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}) \\ &= \int_{\mathbf{w}} p_{\boldsymbol{\beta}}(\mathbf{w}|\mathbf{x}, \mathbf{z})\nabla_{\mathbf{z}}\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{w}|\mathbf{z})d\mathbf{w} \\ &= \mathbf{E}_{\mathbf{w}\sim p_{\boldsymbol{\beta}}(\mathbf{w}|\mathbf{x},\mathbf{z})}\left[\nabla_{\mathbf{z}}\log p_{\boldsymbol{\beta}}(\mathbf{x}, \mathbf{w}|\mathbf{z})\right]. \end{aligned} \tag{C.5}$$

Therefore, the log-likelihood of surrogate target distribution for the Langevin dynam-

ics at the t-th step is

$$\begin{aligned}\log \tilde{Q}(\mathbf{z}_t) &= \log p_{\boldsymbol{\alpha}}(\mathbf{z}_t) + \mathbf{E}_{\mathbf{w}\sim p_{\boldsymbol{\beta}}(\mathbf{w}|\mathbf{x},\mathbf{z}_t)}\left[\sum_{i=1}^{D}\sum_{k=1}^{2} w_{ik}\left(\log \pi_{ik} + \log p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_{k,t})\right)\right] \\ &= \log p_{\boldsymbol{\alpha}}(\mathbf{z}_t) + \sum_{i=1}^{D}\sum_{k=1}^{2} \gamma_{ik,t}\left(\log \pi_{ik} + \log p_{\boldsymbol{\beta}_k}(\mathbf{x}_i|\mathbf{z}_{k,t})\right),\end{aligned} \tag{C.6}$$

which has the same form as $\mathcal{J}(\boldsymbol{\theta})$. However, instead of updating parameters $\boldsymbol{\theta}$, Langevin dynamics updates the latent variables $\mathbf{z}$ with the calculated gradients.

The two-step learning process of the DRC models can be understood as follows: (1) in the first step, the algorithm optimizes $\mathcal{J}$ by updating latent variables $\mathbf{z}$, where the posterior responsibility γ_{ik} inferred at each step serves to gradually disentangle the foreground and background components, and (2) in the second step, the updated $\mathbf{z}$ is fed again into the models to generate the observation $\mathbf{x}$, where the algorithm optimizes $\mathcal{J}$ by updating the model parameters $\boldsymbol{\theta}$.

It is worth mentioning that learning LEBMs requires an extra sampling step (Pang et al., 2020a), as the gradients are given as follows:

$$\delta_{\boldsymbol{\alpha}}(\mathbf{x}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{z})\right] - \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{z})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{z})\right], \tag{C.7}$$

where the second terms should be computed by sampling with $p_{\boldsymbol{\alpha}}(\mathbf{z})$.

Further Details About the Loss Functions For the generative models $p_{\boldsymbol{\beta}_k}(\mathbf{x}|\mathbf{z}_k),\ k = 1, 2$, we assume that $\mathbf{x} = g_{\boldsymbol{\beta}_k}(\mathbf{z}_k) + \epsilon$, where $g_{\boldsymbol{\beta}_k}(\mathbf{z}_k),\ k = 1, 2$ are the generator networks for foreground and background regions, and ϵ is random noise sampled from a zero-mean Gaussian or Laplace distribution. Assuming a global fixed variance σ^2 for Gaussian, we have $\log p_{\boldsymbol{\beta}_k}(\mathbf{x}|\mathbf{z}_k) = -\frac{1}{2\sigma^2}\|g_{\boldsymbol{\beta}_k}(\mathbf{z}_k) - \mathbf{x}\|^2 + C,\ k = 1, 2$, where C is a constant unrelated to $\boldsymbol{\beta}_k$ and $\mathbf{z}_k$. Similarly for Laplace distribution, we have $\log p_{\boldsymbol{\beta}_k}(\mathbf{x}|\mathbf{z}_k) = -\frac{1}{\lambda}|g_{\boldsymbol{\beta}_k}(\mathbf{z}_k) - \mathbf{x}| + C,\ k = 1, 2$. These two log-likelihoods correspond to the MSE loss and L1 loss commonly used for image reconstruction, respectively.

C.3.2 Regularization

Pseudo Label Learning We exploit the symbolic vector $\mathbf{y}$ emitted by the LEBM for additional regularization. Let the target distribution of $\mathbf{y}_k$ be P_k given by $p_{\boldsymbol{\alpha}_k}(\mathbf{y}|\mathbf{z}_k),\ k = 1, 2$, which represents the distribution of symbolic vector for foreground and background regions respectively. We can optimize the following objective as a regularization to our original learning objective:

$$\max_{\boldsymbol{\beta},\tau}\ \mathcal{L}_{\text{pseudo-label}} = \sum_{k=1}^{2} H(P_k, Q_k), \tag{C.8}$$

$$H(P_k, Q_k) = -\langle p_{\boldsymbol{\alpha}_k}(\mathbf{y}|\mathbf{z}_k), \log q_{\tau_k}(\mathbf{y}|g_{\boldsymbol{\beta}_k}(\mathbf{z}_k))\rangle,\ k = 1, 2, \tag{C.9}$$

where $q_{\tau_k},\ k = 1, 2$ represents the jointly trained auxiliary classifier network (see Section C.2 for architecture details) for foreground and background. $g_{\boldsymbol{\beta}_k}(\mathbf{z}_k),\ k = 1, 2$ represents the output of generator network. We set the weight of this regularization term to 0.1 for all the models.

Total Variation Norm The Total Variation norm (Rudin et al., 1992) is commonly used for image denoising, and has been extended as an effective technique for in-painting. We use Total Variation norm (TV-norm) as a regularization for learning the background generator:

$$\min_{\boldsymbol{\beta}_2}\ \mathcal{L}_{\text{TV-norm}} = \sum_{h,w} \left(|\frac{\partial g_{\boldsymbol{\beta}_2}(\mathbf{z}_2)}{\partial x}(h, w)| + |\frac{\partial g_{\boldsymbol{\beta}_2}(\mathbf{z}_2)}{\partial y}(h, w)| \right), \tag{C.10}$$

where $\partial_x g_{\boldsymbol{\beta}_2}(\mathbf{z}_2)(h, w)$ and $\partial_y g_{\boldsymbol{\beta}_2}(\mathbf{z}_2)(h, w)$ represent the horizontal and vertical image gradients at the pixel coordinate (h, w) respectively. We set the weight of this regularization term to 0.01 for all the models.

Orthogonal Regularization We use orthogonal regularization (Brock et al., 2016) for the convolutional layers only. Let $\mathbf{W}$ be the flattened kernel weights of the convolutional layers, *i.e.*, the size of $\mathbf{W}$ is $C \times K$ where C is the output channel number. The

orthogonal regularization is calculated according to

$$\min_{\boldsymbol{\beta}} \mathcal{L}_{\text{orthogonal-reg}} = \|\mathbf{W}\mathbf{W}^T \odot (\mathbf{1} - \mathbf{I})\|_F, \tag{C.11}$$

where $\odot$ is the Hadamard product. $\mathbf{I}$ denotes the identity matrix, and $\mathbf{1}$ denotes the matrix filled with ones. We set the weight of this regularization term to 0.1 for Birds models, and 1.0 for the rest of the models.

C.4 Evaluation Protocols

Intersecion of Union (IoU) The IoU score measures the overlap of two regions A and B by calculating the ratio of intersection over union, according to

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{C.12}$$

where we use the inferred mask and ground-truth mask as A and B respectively for evaluation.

Dice (F1) score Similarly, the Dice (F1) score is

$$\text{Dice}(A, B) = \frac{2|A \cap B|}{|A| + |B|}. \tag{C.13}$$

Higher is better for both scores.

Evaluation IODINE (Greff et al., 2019) and Slot-attention (Locatello et al., 2020) are designed for segmenting complex multi-object scenes using slot-based object representations. Ideally, the output of these models consists of masks for each individual object, while the background is viewed as a virtual "object" as well. In practice, however, it is possible that the model distributes the background over all the slots as mentioned by Locatello et al. (2020). Taking both cases into consideration (see Figure C.3 and Figure C.4), we propose two approaches to convert the multiple output masks into a foreground-background partition, and report the best results of these two options: (1)

Figure C.3: An example situation when using each individual mask as the background mask gives higher scores. Note that if we threshold the output of each individual slot and compose them, the result would be the mask shown in the last column.

Figure C.4: An example situation when thresholding and combining the output of each individual slot gives higher scores. We can see from the last column that the combined mask fits the foreground objects well.

we compute the scores by making each mask as the background mask at a time, and then choose the best one; this works better when the background is treated as a virtual "object"; (2) we threshold and combine all the masks into a foreground mask; this is for when background is distributed to all slots.

C.5 Additional Illustrations and Baseline Results

C.5.1 More Examples

We provide more foreground extraction results of our model for each dataset; see Figure C.5, Figure C.6, Figure C.7 and Figure C.8. From top to bottom, we display: (i) observed images, (ii) generated images, (iii) masked generated foregrounds, (iv) generated backgrounds, (v) ground-truth foreground masks, and (vi) inferred foreground masks in each figure.

C.5.2 Failure Modes

We provide examples for illustrating typical failure modes of the proposed model; see Figure C.9. On Birds dataset, we observe that the method can perform worse on samples where the foreground object has colors and textures quite similar to the background

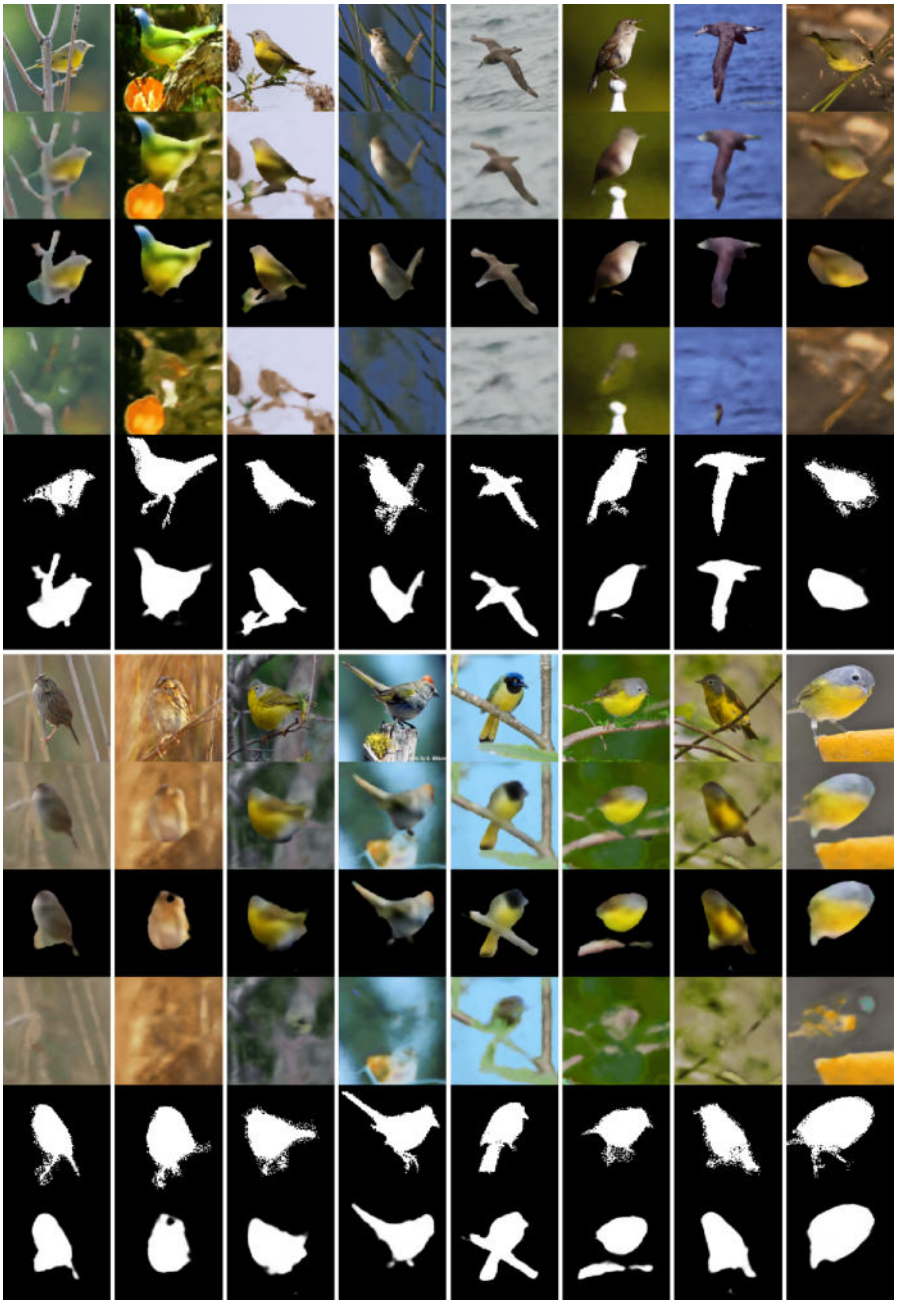


Figure C.5: Additional foreground extraction results on Birds dataset.

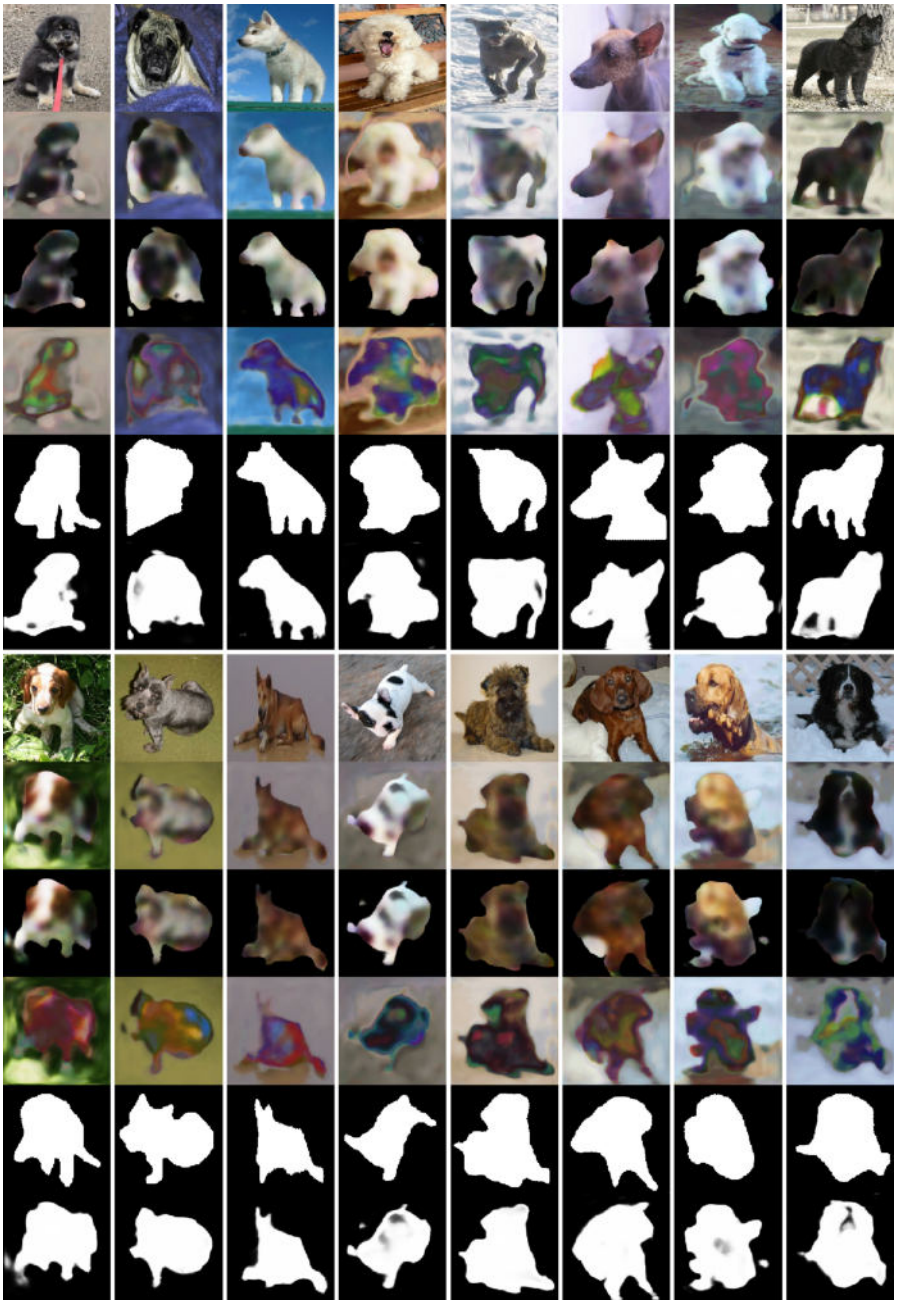


Figure C.6: Additional foreground extraction results on Dogs dataset.

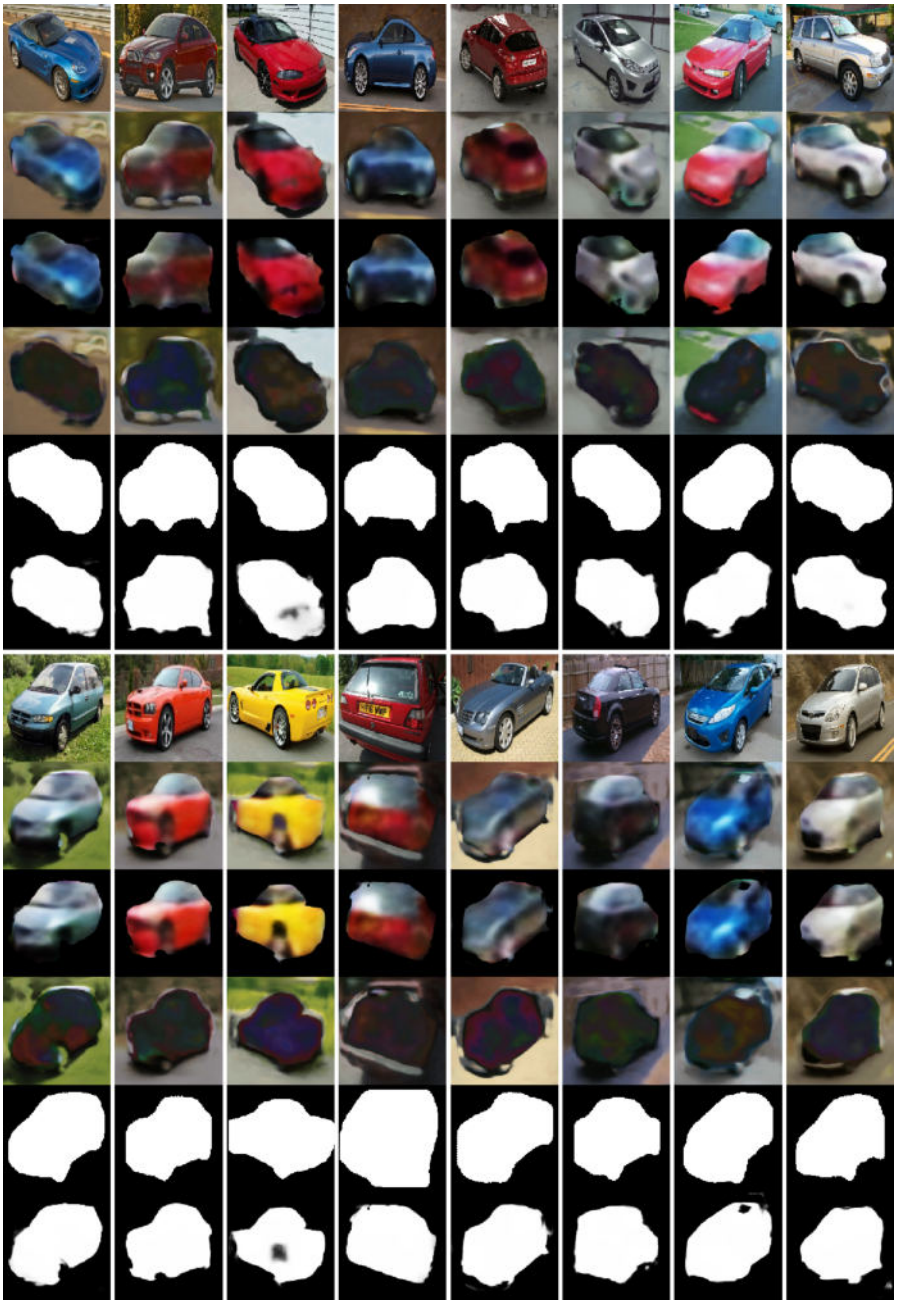


Figure C.7: Additional foreground extraction results on Cars dataset.

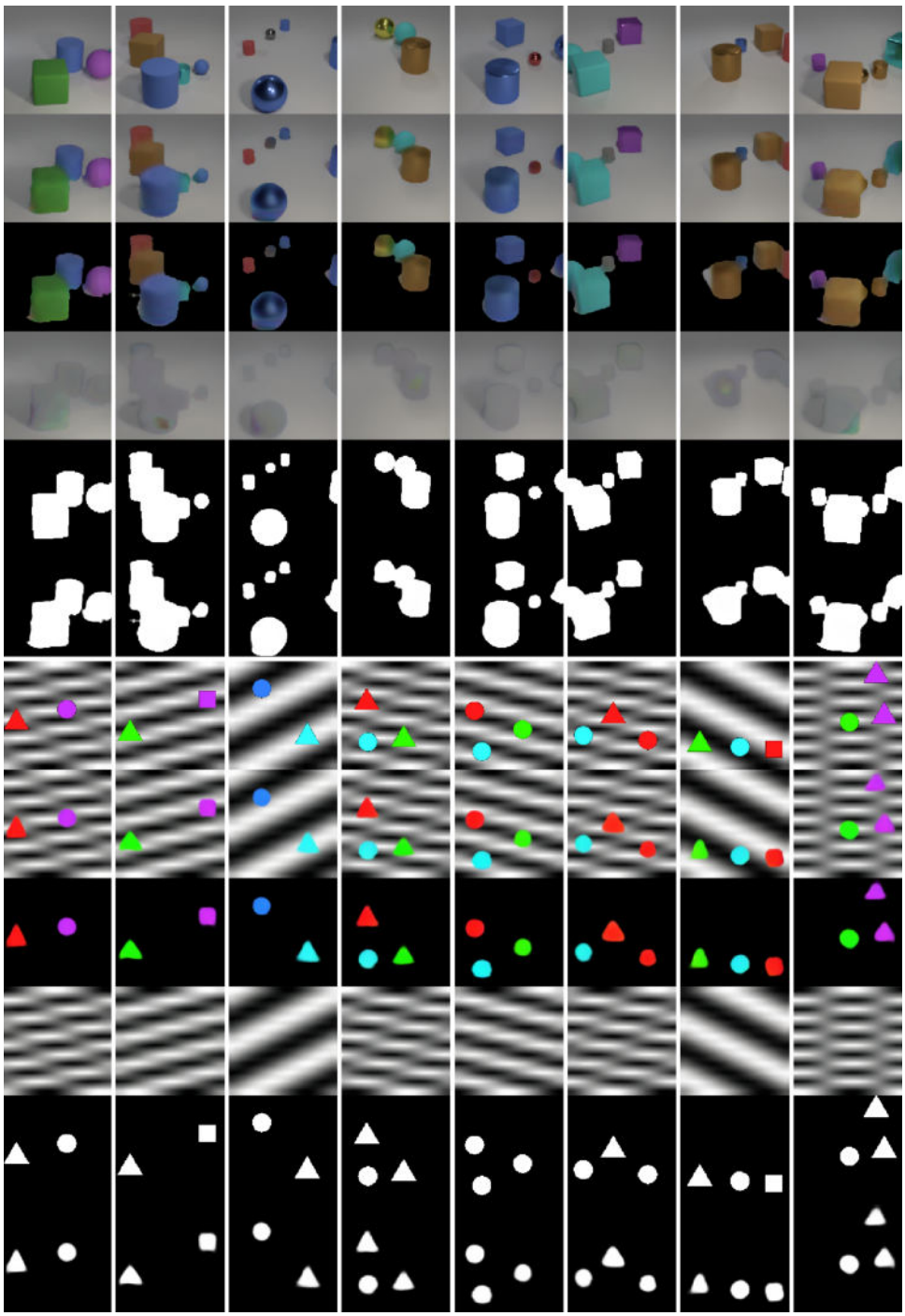


Figure C.8: Additional foreground extraction results on CLEVR6 and TM-dSprites datasets.

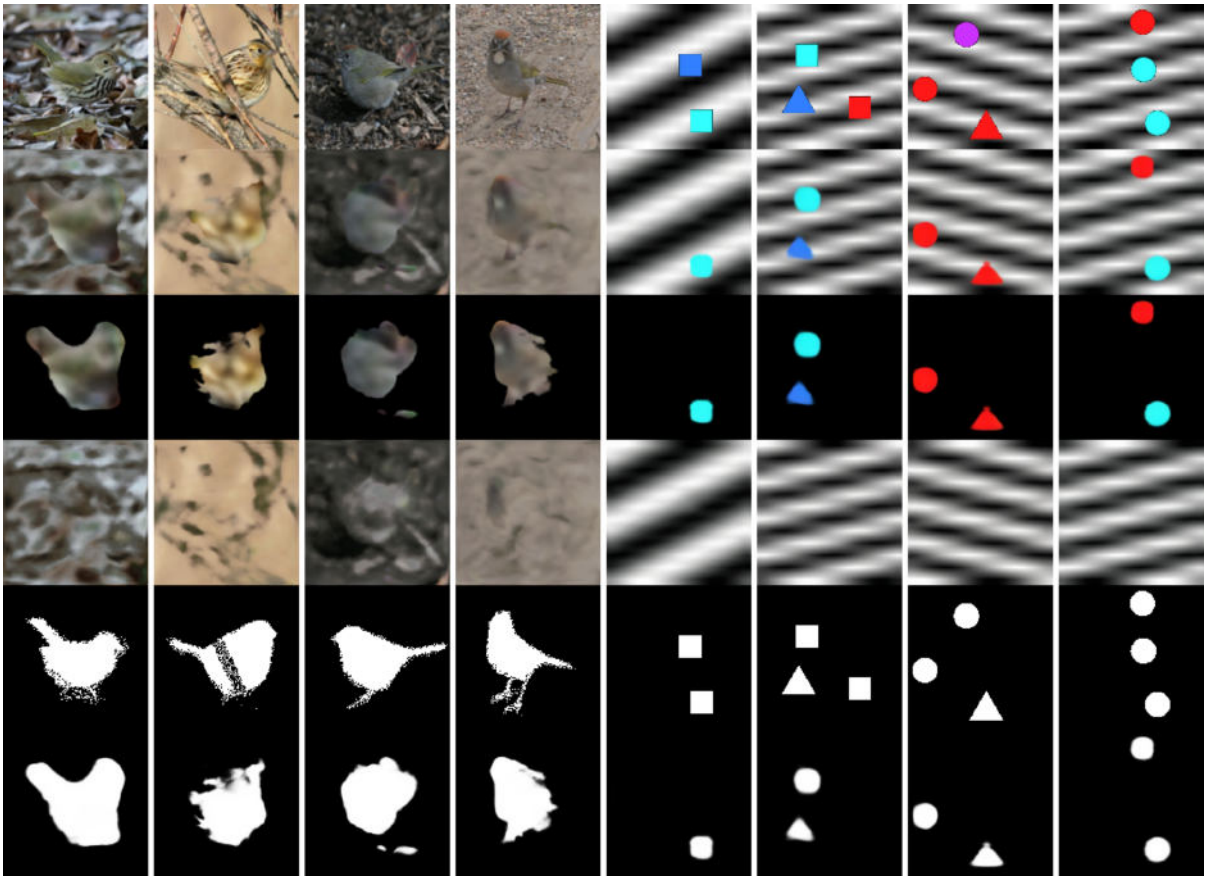


Figure C.9: Typical failure modes on Birds and TM-dSprites.

regions. Although the method can still capture the rough shape of the foreground object, some details can be missing. On TM-dSprites dataset, we observe that the method may occasionally miss one of the foreground objects. We conjecture that the problem can be mitigated with more powerful generator and further fine-tuning on this dataset.

C.5.3 Baseline Results

GrabCut We provide results of GrabCut (Rother et al., 2004) on Birds dataset and TM-dSprites dataset, shown in Figure C.10. We can see that GrabCut algorithm may fail when the foreground object and background region have moderately similar colors and textures. On TM-dSprites dataset, GrabCut algorithm outperforms other baselines, but is still inferior to the proposed method and exhibits a similar failure pattern.

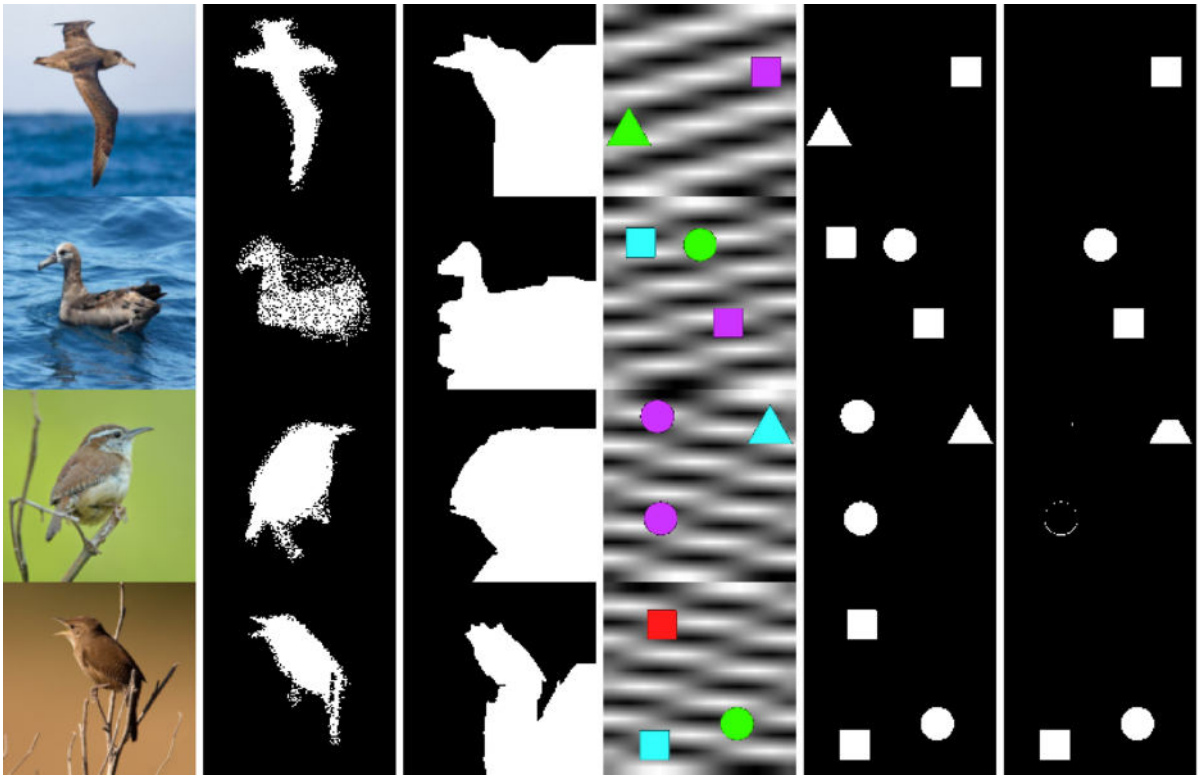


Figure C.10: Results of GrabCut on Birds and TM-dSprites datasets. The first three columns are results from Birds dataset, and the last three are from TM-dSprites. From left to right, we display the observed image, ground-truth mask, and the foreground extraction results respectively.

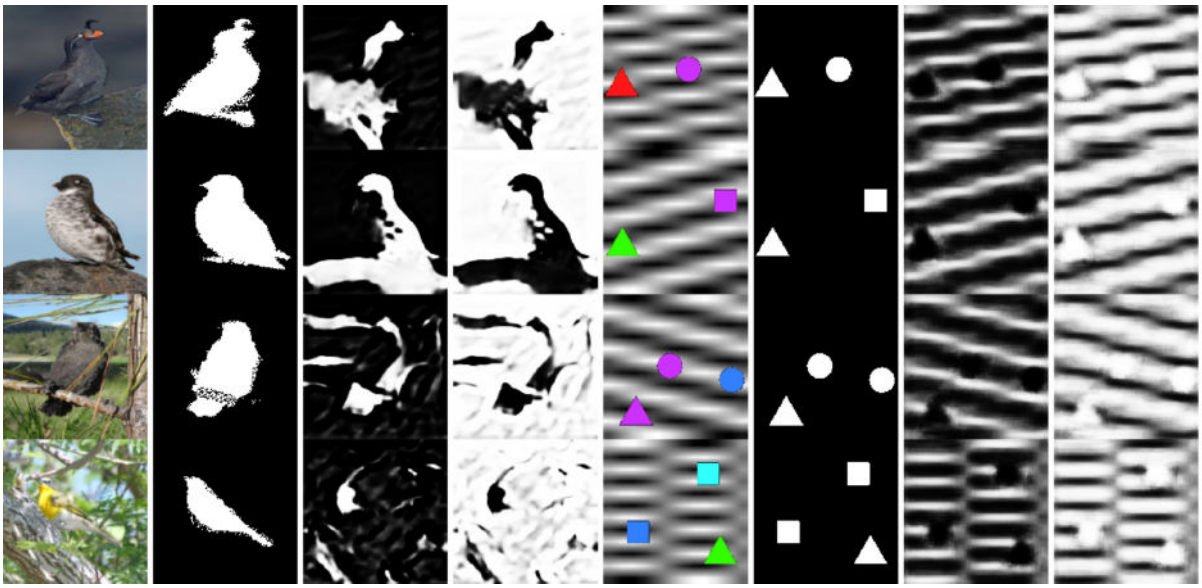


Figure C.11: Results of ReDO on Birds and TM-dSprites datasets. The first four columns are results from Birds dataset, and the last four are from TM-dSprites. From left to right, we display the observed image, ground-truth mask, mask from the first output channel and from the second channel respectively.

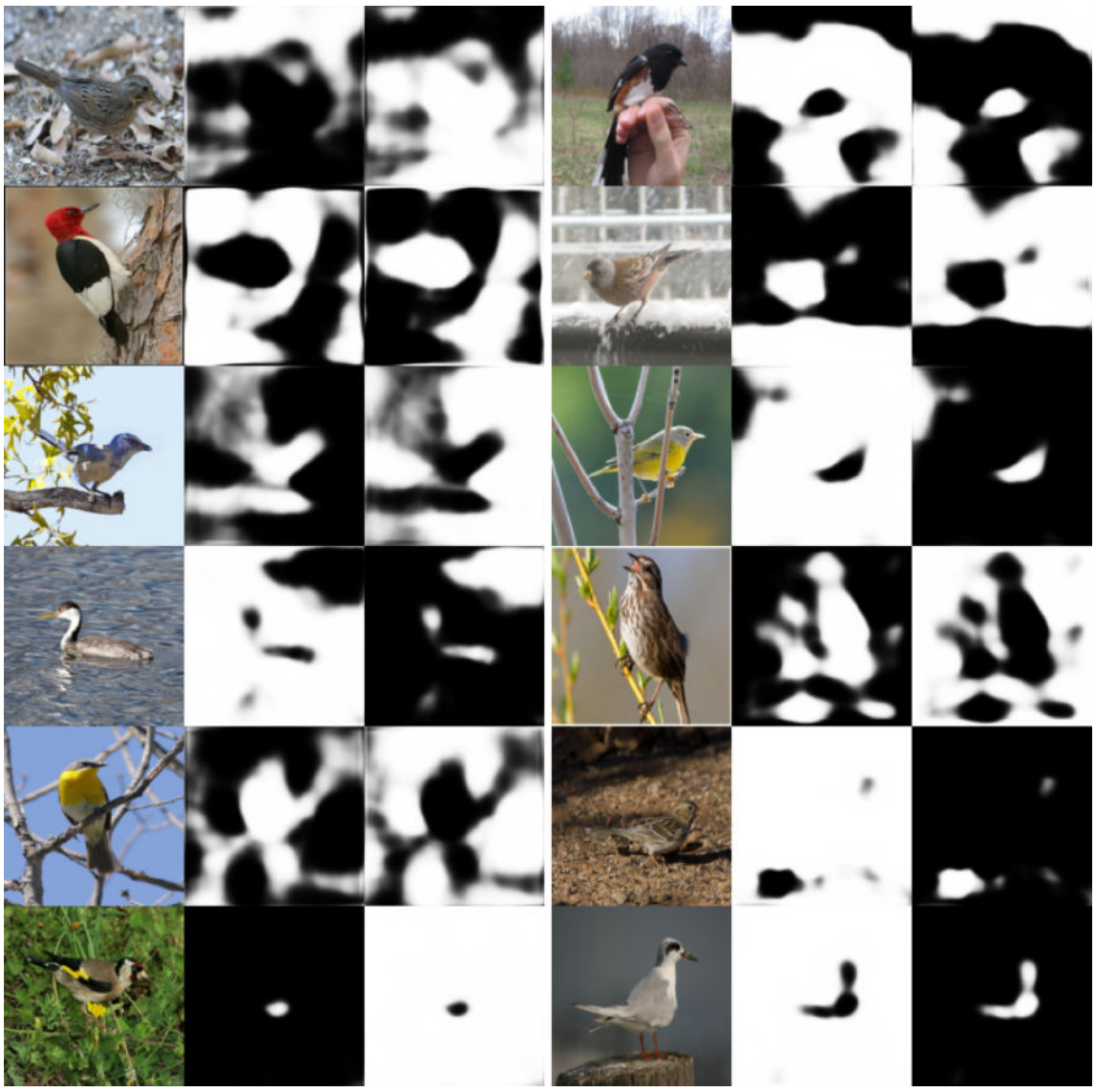


Figure C.12: Results of IODINE on Birds datasets. We provide the observed image, mask from the first slot and from the second slot respectively.

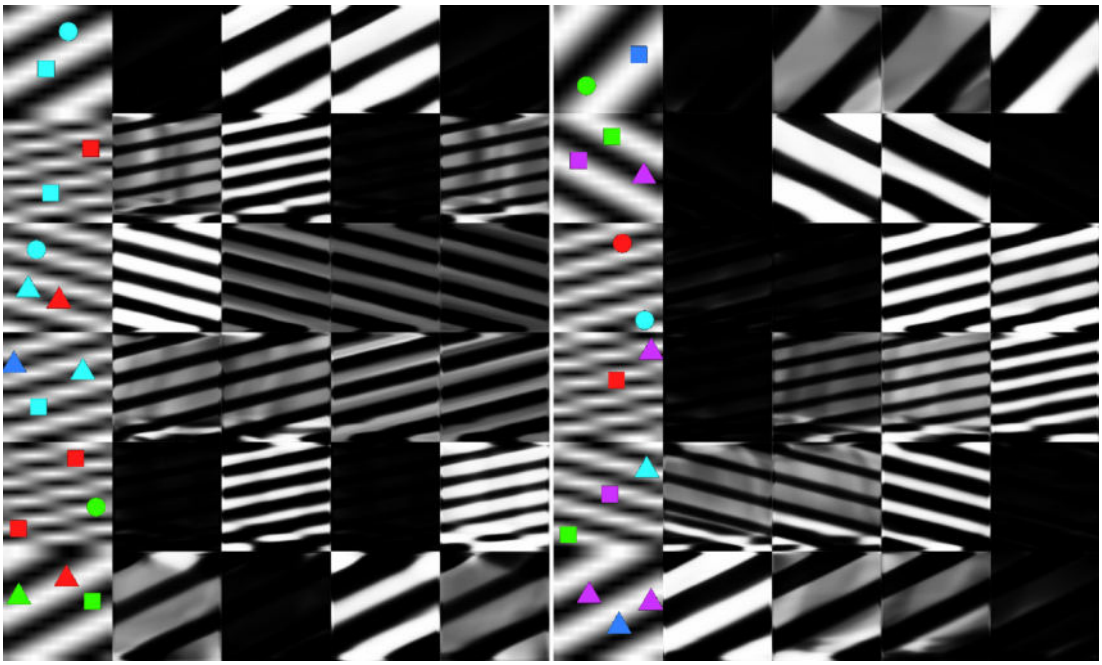


Figure C.13: Results of IODINE on TM-dSprites datasets. We provide the observed image and masks from four object slots respectively.

ReDO We provide results of ReDO (Chen et al., 2019a) on Birds dataset and TM-dSprites dataset, shown in Figure C.11. ReDO overall performs better than GrabCut on Birds dataset, while it may fail when the background regions become more complex. We can also observe that ReDO relies heavily on the pixel intensities for foreground-background grouping on TM-dSprites dataset.

IODINE On the Birds dataset, we observe that IODINE (Greff et al., 2019) tends to use color as a strong cue for segmentation, see Figure C.12. On TM-dSprites dataset, IODINE is distracted by the background; see Figure C.13. These two findings are consistent with those reported by Greff et al. (2019).

Slot-Attention On the Birds dataset, Slot-attention learns to roughly locate the position of foreground objects, but mostly fails to provide foreground masks when the background region becomes complex; see Figure C.14. Similarly, we can observe that Slot-Attention tends to use color as a strong cue for segmentation. On TM-dSprites dataset, Slot-attention is distracted by the background; see Figure C.15.

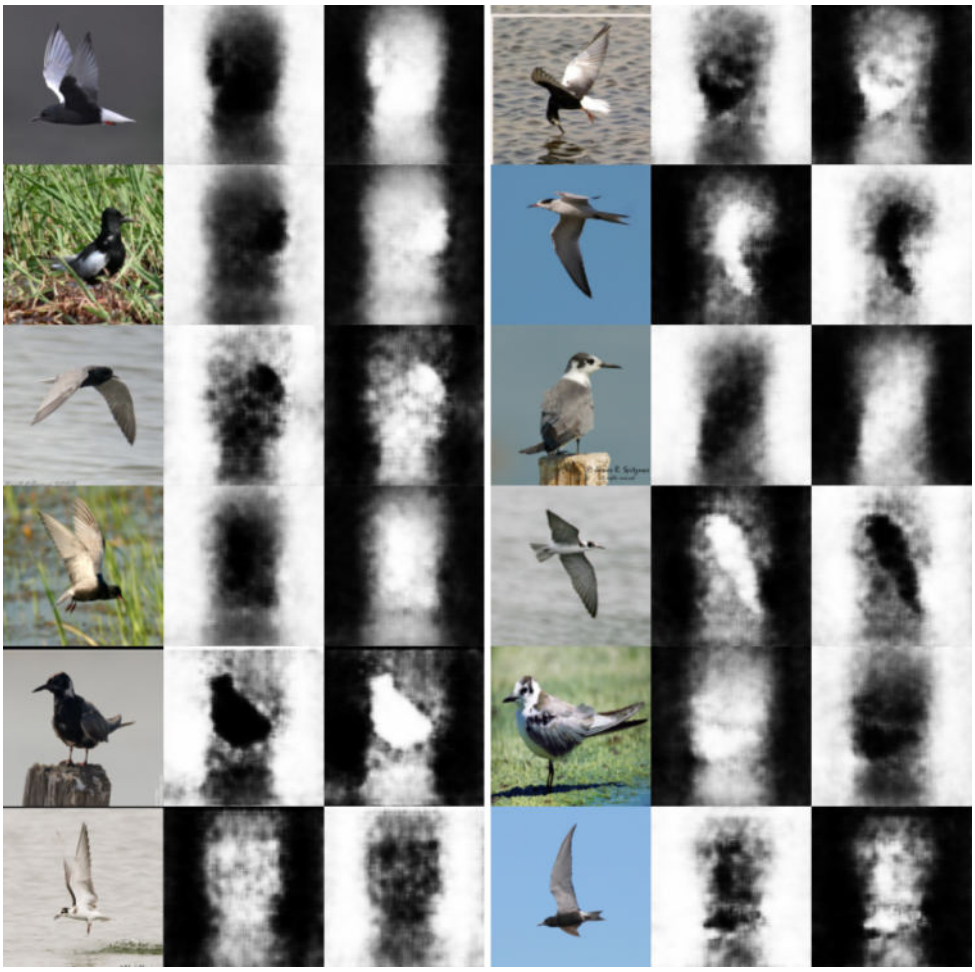


Figure C.14: Results of Slot-Attention on Birds datasets. We provide the observed image, mask from the first slot and from the second slot respectively.

C.6 Further Discussion

C.6.1 Preliminary Analysis of Real-World Datasets

We provide preliminary analysis of the statistics of the three real-world datasets. To measure the similarity of colors and textures for these datasets, we calculate the image histogram for the foreground objects and background regions of each dataset; see Figure C.16. To probe the similarity of shape distributions, we also provide the heatmap of foreground masks, as shown in Figure C.17. The heatmaps are calculated by overlapping the ground-truth masks and normalizing the summarized intensities with the maximum

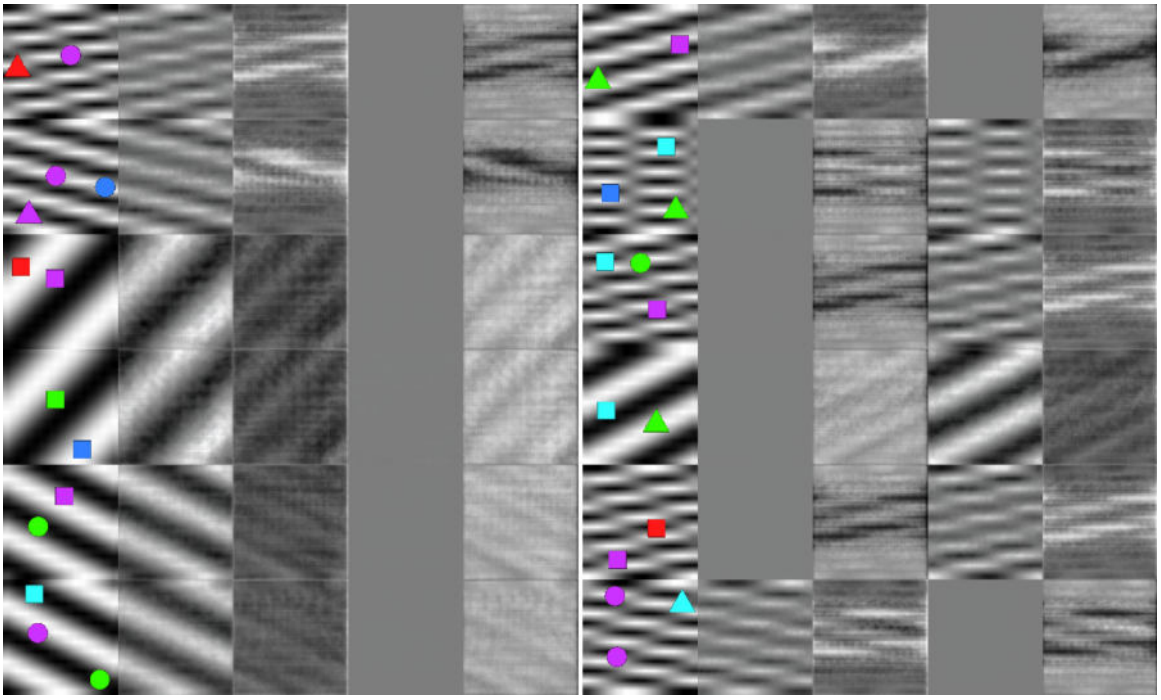


Figure C.15: Results of Slot-Attention on the TM-dSprites datasets. We provide the observed image and masks from four object slots respectively.

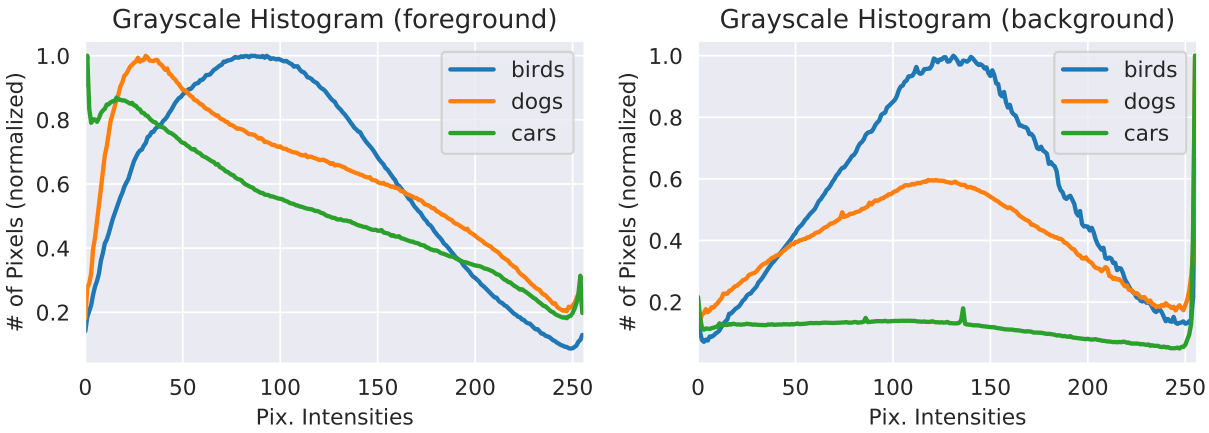


Figure C.16: Image histograms for foreground objects and background regions from each dataset.

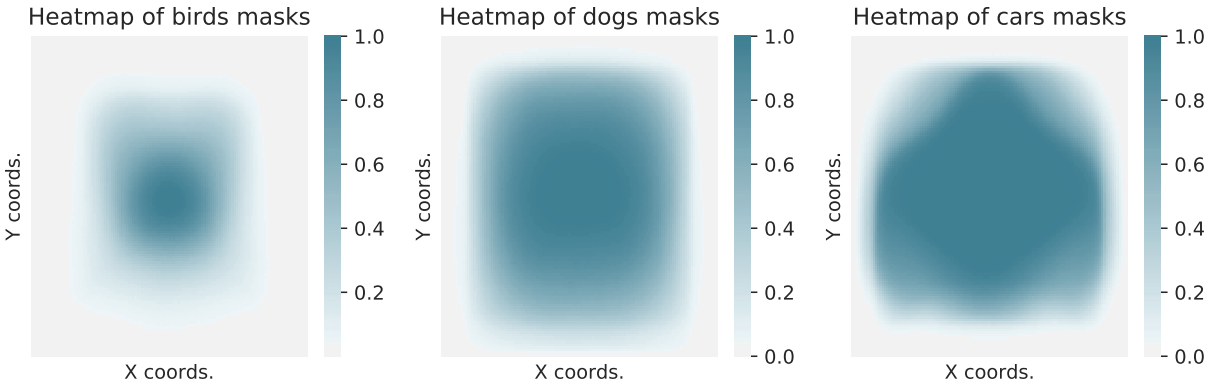


Figure C.17: Heatmaps of ground-truth masks for each dataset.

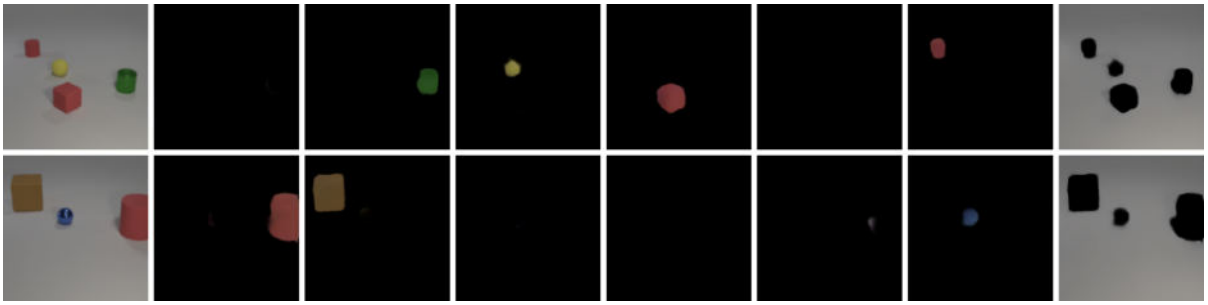


Figure C.18: Preliminary results on learning slot-based object representation.

values. Despite the apparent difference in Birds vs Dogs and Cars, we can see that the data distribution of Birds dataset is more similar to that of Dogs dataset than to that of Cars dataset. We can also observe the similarity between the distributions of Dogs and Cars datasets. This could partly explain why the proposed method shows relatively strong performance on objects from unseen categories, *i.e.*, it effectively combines the colors, textures and shapes for foreground extraction.

C.6.2 Possible Extension to Multi-Object Segmentation

We explore the possibility of using our model for segmenting and disentangling multiple objects. As shown in Figure C.18, the proposed method can disentangle the foreground objects, while providing explicit identification of the background region. However, we find that the model occasionally distributes a single object into several slots based on the difference in texture and shading; see Figure C.19. We conjecture that this is due to the lack of objectness modeling. We would like to investigate further in this direction in

Figure C.19: Failure modes of energy-based slot representation model.

future work.

APPENDIX D

Derivations and Experimental Details for Chapter 5

D.1 Relation Between Compositionality and Disentanglement

D.1.1 Slot-Based Disentanglement is Compositional

Consider a slot-based representation space $\mathbf{z} = [\mathbf{z}_0, \cdots, \mathbf{z}_K]$ where each $\mathbf{z}_i \in \mathbb{R}^{\frac{D}{K}}$ with a functional constraint that uses individual slots to decode them back into the pixel space, namely $f_i(\mathbf{z}_i) = \mathbf{u}_i$, where $\mathbf{u}_i \in \mathbb{R}^N$ as explained in Section 5.3. Further, assume that $f_i = f_j$ for all $i, j \in 1, \cdots, K$ (this constraint is typically followed by models as well). Also, assume that the model represents all blank slots consistently, specifically, assume that $\mathbf{z}_k = \mathbf{0}$ where $\mathbf{0}$ represents a vector in $\mathbf{R}^{\frac{D}{K}}$ where each entry is 0, without loss of generality.

Now, assume image A is the original image we add an object to, in order to obtain image B, for the analogy test in Figure 5.1. Further assume that A has k objects. Thus, by assumption that blank slots are consistently represented, $\forall k' > k$, $\mathbf{z}'_k = \mathbf{0}$. Let $\delta \in \mathbb{R}^{\frac{D}{K}}$ be the offset to be added to a blank slot $\mathbf{z}_{k+1}$ in A in order to obtain B, which essentially corresponds to the addition of a blue object. Then, in order to add the same object to image C with $\hat{k}$ objects, one would add the same offset vector δ to the blank slot $\mathbf{z}_{\hat{k}+1}$. This is true, since we assume that the function $f_i(\mathbf{z}_i)$ is independent for each slot, and also $f_i = f_j$ for all $i, j \in 1, \cdots, K$.

Thus, by construction one achieves a parallelogram in the full original vector space $\mathbf{z}$ in which the representation exists, meaning that an object-centric disentangled slot-based representation is also compositional with respect to addition and subtration of objects

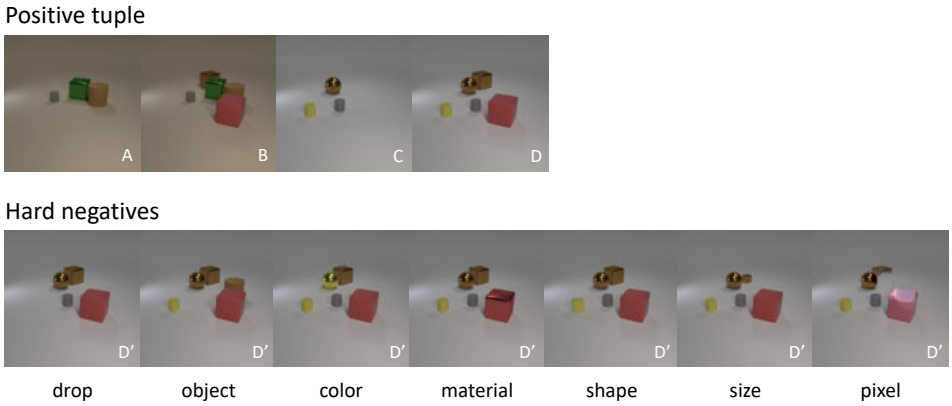


Figure D.1: Example of the test corpus of COAT. A, B, C, D form the positive tuple, where the same transformation leads A to B and C to D. In the second row there are hard negatives D', where "drop" is dropping one object from D, "object" is changing one object from D, "color" is changing the color of one object from D, "material" is changing the material of one object from D, "shape" is changing the shape of one object from D, "size" is changing the size of one object from D, "pixel" is the result of $B - A + C$.

from the scene.

D.2 Example Test Cases

In Figure D.1 and Figure D.2, we illustrate two examples of the COAT test. COAT features a positive tuple of images A, B, C, D, and a set of negative D's.

D.2.1 Full-Rank Transformed Disentanglement is Compositional

Given a disentangled representation $q(\mathbf{x})$, and let $\mathbf{z}_A = q(A)$, $\mathbf{z}_B = q(B)$, $\mathbf{z}_C = q(C)$ and $\mathbf{z}_D = q(D)$, where $\mathbf{z} \in \mathbb{R}^D$ as in the rest of the chapter. be the representations for four scenes in the analogy test Figure 5.1. Now, since disentangling implies compositionality, we have

$$\mathbf{z}_B - \mathbf{z}_A + \mathbf{z}_C = \mathbf{z}_D. \tag{D.1}$$

Next, consider a full rank matrix $W \in \mathbb{R}^{D \times D}$. Then, multiplying by W on both sides above, we get

$$W\mathbf{z}_B - W\mathbf{z}_A + W\mathbf{z}_C = W\mathbf{z}_D. \tag{D.2}$$

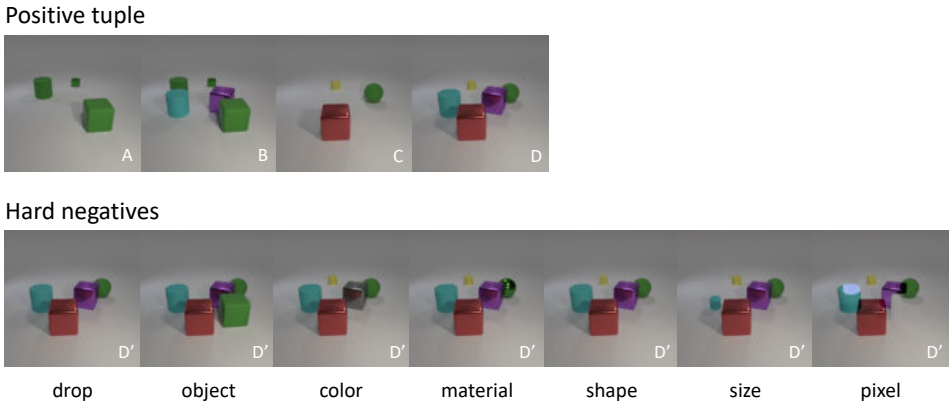


Figure D.2: Example of the test corpus of COAT. A, B, C, D form the positive tuple, where the same transformation leads A to B and C to D. In the second row there are hard negatives D', where "drop" is dropping one object from D, "object" is changing one object from D, "color" is changing the color of one object from D, "material" is changing the material of one object from D, "shape" is changing the shape of one object from D, "size" is changing the size of one object from D, "pixel" is the result of $B - A + C$.

Now, notice that in general, $W\mathbf{z}_B$ is no longer disentangled with respect to the original factors of variation (*e.g.*, W could be a rotation matrix or a permutation matrix), but the resultant representation still satisfies the parallelogram property. Moreover, given invertibility of W because of it being full rank, it has all the information present in the original disentangled space meaning that it avoids collapse of the representation which might enable it to lose information of attributes or find some other "shortcut" to pass the analogy test trivially.

D.3 CLEVR With Colorful Background

D.3.1 Independently Identically Distributed Dataset

The training set is built upon the renderer of CLEVR. We start from a setup commonly used for multi-object scene understanding, where all objects are independently identically distributed. Figure D.3 shows some example samples from this training set.

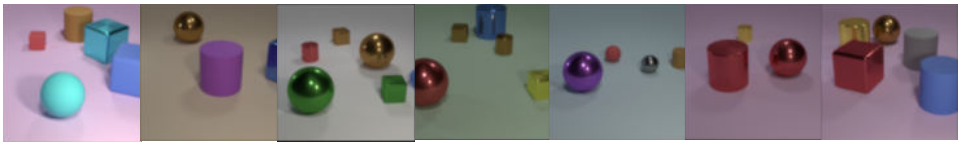


Figure D.3: Example of the IID training data with colorful background.

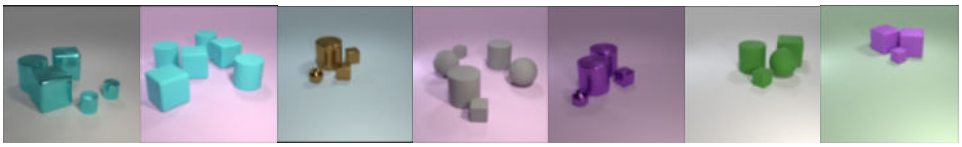


Figure D.4: Example of the correlated training data with colorful background.

D.3.2 Correlated Dataset

We also explored a correlated dataset where all objects have identical color and materials and are densely cluttered together, as shown in Figure D.4.

D.3.3 Impact on COAT of Using Colorful Background

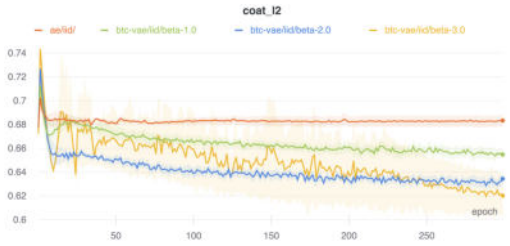
To investigate the impact of colorful background on COAT, we apply the COAT measure to a test corpus with only white background. The COAT score of slot-based models (*e.g.*, IODINE) are not affected much by this change, but slot-free models improve by ≈3% (although they still fail the hard-negative tests). IODINE segments white background inconsistently in correlated images (see Figure 5.7), which might explain this lack of improvement.

D.4 Greedy Matching Algorithm

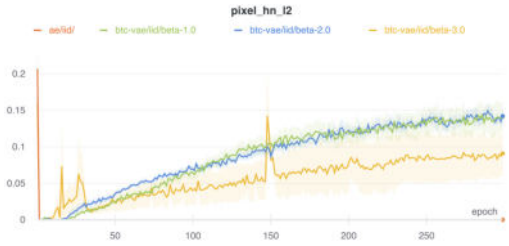
When applying COAT metric to slot-based representations, a 4-way slot matching needs to be conducted to find the lowest matching cost. In this chapter, this is realized by iteratively picking one slot greedily from each representation $\mathbf{z}_A, \mathbf{z}_B, \mathbf{z}_C, \mathbf{z}_D$ without replacement. And the criterion for this matching is the L2 residual $||\mathbf{z}_B - \mathbf{z}_A + \mathbf{z}_C - \mathbf{z}_D||_2$ from these slots. Empirically, we find this greedy matching algorithm perform very well even if it does not guarantee global optimum.

D.5 Additional Empirical Results

In Figure D.5 and Figure D.6 we provide the training curves of adjusted COAT metric and the sampled success rate $\hat{p}$ on hard negative tests for the Autoencoder and VAEs.

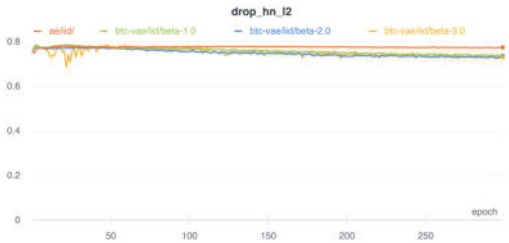


(a) COAT l2 score

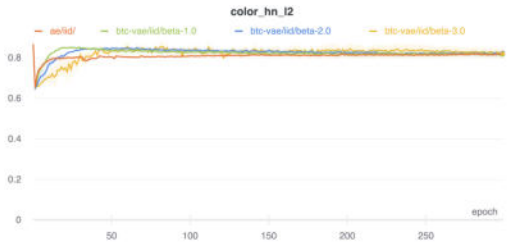


(b) $\hat{p}$ on pixel negative

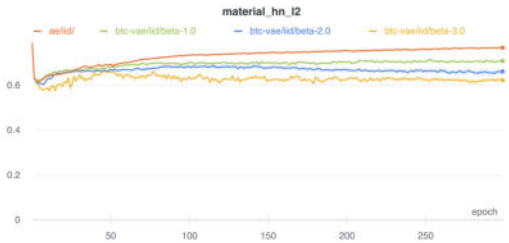
(c) $\hat{p}$ on object negative



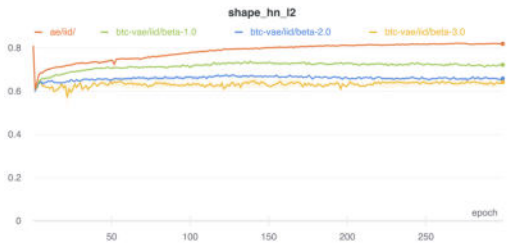
(d) $\hat{p}$ on drop negative



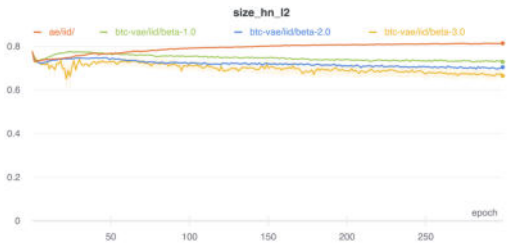
(e) $\hat{p}$ on color negative



(f) $\hat{p}$ on material negative

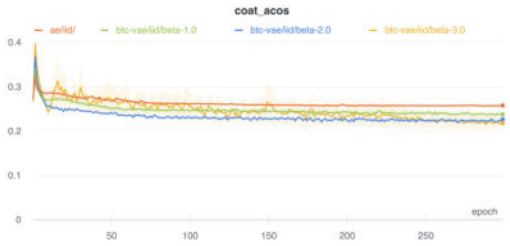


(g) $\hat{p}$ on shape negative

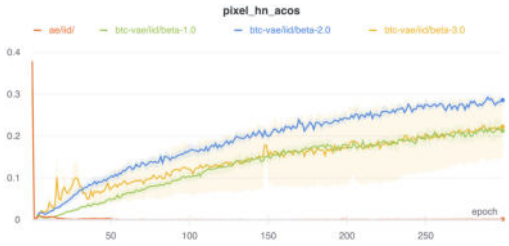


(h) $\hat{p}$ on size negative

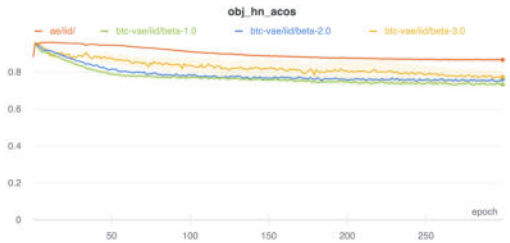
Figure D.5: COAT l2 for Autoencoder and VAEs



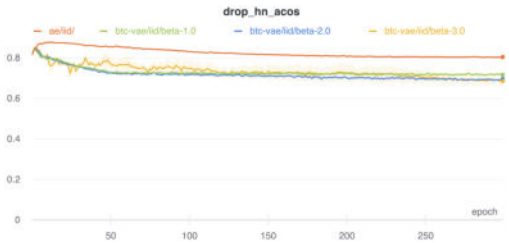
(a) COAT acos score



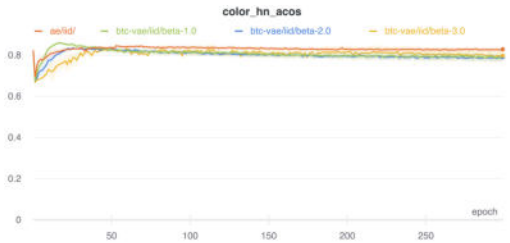
(b) $\hat{p}$ on pixel negative



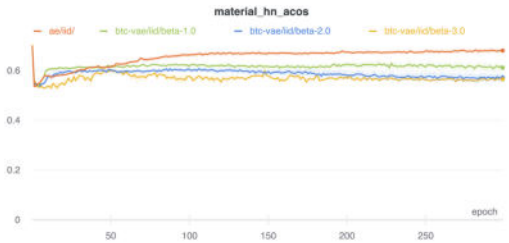
(c) $\hat{p}$ on object negative



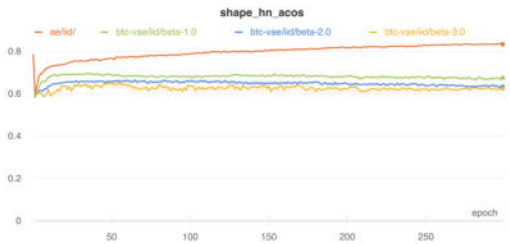
(d) $\hat{p}$ on drop negative



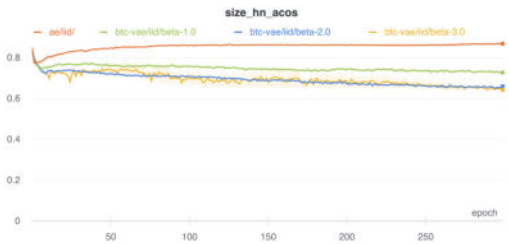
(e) $\hat{p}$ on color negative



(f) $\hat{p}$ on material negative



(g) $\hat{p}$ on shape negative



(h) $\hat{p}$ on size negative

Figure D.6: COAT acos for Autoencoder and VAEs

APPENDIX E

Derivations and Experimental Details for Chapter 6

E.1 Learning and Sampling in LanMDP

E.1.1 Maximum Likelihood Learning

We need to estimate $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$. Suppose we observe training examples: $\{\xi^i\}, i = 1, 2, \cdots, n, \quad \xi^i = [\mathbf{s}_0^i, \mathbf{s}_1^i, \ldots, \mathbf{s}_T^i]$. The log-likelihood function is

$$L(\boldsymbol{\theta}) = \sum\nolimits_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\xi^i). \tag{E.1}$$

Denote posterior distribution of action sequence $p_{\boldsymbol{\theta}}(\mathbf{a}_{0:T-1}|\mathbf{s}_{0:T})$ as $p_{\boldsymbol{\theta}}(A|S)$ for convenience where A and S means the complete action and state sequences in a trajectory. The gradient of log-likelihood is

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\xi) &= \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_T) \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_T)] \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_T)] + \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(A|S)] \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{a}_{T-1}, \mathbf{s}_T)] \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}[\nabla_{\boldsymbol{\theta}} \log p(\mathbf{s}_0) p_{\boldsymbol{\alpha}}(\mathbf{a}_0|\mathbf{s}_0) \cdots p_{\boldsymbol{\alpha}}(\mathbf{a}_{T-1}|\mathbf{s}_{0:T-1}) p_{\boldsymbol{\beta}}(\mathbf{s}_T|\mathbf{s}_{T-1}, \mathbf{a}_{T-1})] \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}[\nabla_{\boldsymbol{\theta}} \sum\nolimits_{t=0}^{T-1} (\log p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) + \log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t))] \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}[\sum\nolimits_{t=0}^{T-1} (\underbrace{\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}_{\text{policy/prior}} + \underbrace{\nabla_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}_{\text{transition}})],
\end{aligned} \tag{E.2}$$

where the third equation is because of a simple identity $\mathrm{E}_{\pi_{\boldsymbol{\theta}}(a)}\left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a)\right] = 0$ for any probability distribution $\pi_{\boldsymbol{\theta}}(a)$. Applying this simple identiy, we also have

$$\begin{aligned} 0 &= \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})\right] \\ &= \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t}) - \nabla_{\boldsymbol{\alpha}} \log Z(\boldsymbol{\alpha}, \mathbf{s}_{0:t})\right] \\ &= \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right] - \nabla_{\boldsymbol{\alpha}} \log Z(\boldsymbol{\alpha}, \mathbf{s}_{0:t}). \end{aligned} \tag{E.3}$$

Due to the normalizing constant $Z(\boldsymbol{\alpha}, \mathbf{s}_{0:t})$ in the energy-based prior $p_{\boldsymbol{\alpha}}$, the gradient for the policy term involves both posterior and prior samples is

$$\begin{aligned} \delta_{\boldsymbol{\alpha},t}(S) &= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}\left[\nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})\right] \\ &= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t}) - \nabla_{\boldsymbol{\alpha}} \log Z(\boldsymbol{\alpha}, \mathbf{s}_{0:t})\right] \\ &= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t}) - \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[\nabla f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right]\right] \\ &= \mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right] - \mathbb{E}_{p_{\boldsymbol{\alpha}}(a_t|\mathbf{s}_{0:t})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(a_t; \mathbf{s}_{0:t})\right], \end{aligned} \tag{E.4}$$

where $\delta_{\boldsymbol{\alpha},t}(S)$ denotes the surrogate loss of policy term for time step t. Intuition can be gained from the perspective of adversarial training (Finn et al., 2016; Ho and Ermon, 2016): On one hand, the model utilizes action samples from the posterior $p_{\boldsymbol{\theta}}(A|S)$ as pseudo-labels to supervise the unnormalized prior at each step $p_{\boldsymbol{\alpha}}(a_t|\mathbf{s}_{0:t})$. On the other hand, it discourages action samples directly sampled from the prior. The model converges when prior samples and posterior samples are indistinguishable.

To ensure the transition model's validity, it needs to be grounded in real-world dynamics when jointly learned with the policy. Otherwise, the agent would be purely hallucinating based on the demonstrations. Throughout the training process, we allow the agent to periodically collect self-interaction data with $p_{\boldsymbol{\alpha}}(a_t|s_{0:t})$ and mix transition data from two sources with weight $w_{\boldsymbol{\beta}}$:

$$\delta_{\boldsymbol{\beta},t}(S) = w_{\boldsymbol{\beta}}\mathbb{E}_{p_{\boldsymbol{\theta}}(A|S)}\left[\nabla_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right] + (1 - w_{\boldsymbol{\beta}})\mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}),Tr}\left[\nabla_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right]. \tag{E.5}$$

E.1.2 General Transition Model

We need to compute the gradient of $\boldsymbol{\beta}$ for the logarithm of transition probability in (E.5), and as we will see in Section 6.3.3, we also need to compute the gradient of the action during sampling actions. The reparameterization (Kingma and Welling, 2014a) is useful since it can be used to rewrite an expectation w.r.t $p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ such that the Monte Carlo estimate of the expectation is differentiable, so we use delta function $\delta(.)$ to rewrite probability as an expectation:

$$\begin{aligned} p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) &= \int \delta(\mathbf{s}_{t+1} - \mathbf{s}'_{t+1}) p_{\boldsymbol{\beta}}(\mathbf{s}'_{t+1}|\mathbf{s}_t, \mathbf{a}_t) d\mathbf{s}'_{t+1} \\ &= \int \delta(\mathbf{s}_{t+1} - g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t, \boldsymbol{\epsilon})) p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}. \end{aligned} \tag{E.6}$$

Taking advantage of the properties of $\delta(.)$:

$$\int f(x)\delta(x)dx = f(0), \quad \delta(f(x)) = \Sigma_n \frac{1}{|f'(x_n)|}\delta(x - x_n), \tag{E.7}$$

where f is differentiable and have isolated zeros, which is x_n, we can rewrite the transition probability as

$$\begin{aligned} p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) &= \int \sum_n \frac{1}{|\frac{\partial}{\partial \boldsymbol{\epsilon}} g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t, \boldsymbol{\epsilon})|_{\boldsymbol{\epsilon}=\boldsymbol{\epsilon}_n}} \delta(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_n) p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} \\ &= \sum_n \frac{p(\boldsymbol{\epsilon}_n)}{|\frac{\partial}{\partial \boldsymbol{\epsilon}} g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t, \boldsymbol{\epsilon})|_{\boldsymbol{\epsilon}=\boldsymbol{\epsilon}_n}}, \end{aligned} \tag{E.8}$$

where $\boldsymbol{\epsilon}_n$ is the zero of $\mathbf{s}_{t+1} = g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t, \boldsymbol{\epsilon})$. Therefore, if we have a differentiable simulator $\nabla_{\mathbf{a}_t} \log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and the analytical form of $p(\boldsymbol{\epsilon})$, then gradient of both $\mathbf{a}_t$ and $\boldsymbol{\beta}$ for $\log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ can be computed.

The simplest situation is

$$\mathbf{s}_{t+1} = g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t) + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) = \mathcal{N}(0, \sigma^2). \tag{E.9}$$

In this case, there is only one zero $\boldsymbol{\epsilon}^*$ for the transition function, $\mathbf{s}_{t+1} = g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t) + \boldsymbol{\epsilon}^*$,

and the gradient of log probability is

$$\begin{aligned}\nabla \log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) &= \nabla \log \frac{p(\boldsymbol{\epsilon}^*)}{|\frac{\partial}{\partial \boldsymbol{\epsilon}}(g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t) + \boldsymbol{\epsilon})|_{\boldsymbol{\epsilon}=\boldsymbol{\epsilon}^*}} \\ &= \nabla \log p(\boldsymbol{\epsilon}^*) \\ &= \nabla \log p(\mathbf{s}_{t+1} - g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t)) \\ &= \frac{1}{\sigma^2}(\mathbf{s}_{t+1} - g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t))\nabla g_{\boldsymbol{\beta}}(\mathbf{s}_t, \mathbf{a}_t).\end{aligned} \tag{E.10}$$

E.1.3 Prior and Posterior Sampling

The maximum likelihood estimation requires samples from the prior and the posterior distributions of actions. It would not be a problem if the action space is quantized. However, since we target general latent action learning, we proceed to introduce sampling techniques for continuous actions.

When sampling from a continuous energy space, short-run Langevin dynamics (Nijkamp et al., 2019) can be an efficient choice. For a target distribution $\pi(a)$, Langevin dynamics iterates $\mathbf{a}_{k+1} = \mathbf{a}_k + s\nabla_{\mathbf{a}_k} \log \pi(\mathbf{a}_k) + \sqrt{2s}\boldsymbol{\epsilon}_k$, where k indexes the number of iteration, s is a small step size, and $\boldsymbol{\epsilon}_k$ is the Gaussian white noise. $\pi(a)$ can be either the prior $p_{\boldsymbol{\alpha}}(a_t|\mathbf{s}_{0:t})$ or the posterior $p_{\boldsymbol{\theta}}(A|S)$. One property of Langevin dynamics that is particularly amenable for EBM is that we can get rid of the normalizing constant. So for each t the iterative update for prior samples is

$$a_{t,k+1} = \mathbf{a}_{t,k} + s\nabla_{\mathbf{a}_{t,k}} f_{\boldsymbol{\alpha}}(\mathbf{a}_{t,k}; \mathbf{s}_{0:t}) + \sqrt{2s}\boldsymbol{\epsilon}_k. \tag{E.11}$$

Given a state sequence $\mathbf{s}_{0:T}$ from the demonstrations, the posterior samples at each time step $\mathbf{a}_t$ come from the conditional distribution $p(\mathbf{a}_t|\mathbf{s}_{0:T})$. Notice that with Markov transition, we can derive

$$p_{\boldsymbol{\theta}}(\mathbf{a}_{0:T-1}|\mathbf{s}_{0:T}) = \prod_{t=0}^{T-1} p_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_{0:T}) = \prod_{t=0}^{T-1} p_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_{0:t+1}). \tag{E.12}$$

The point is, given the previous and the next subsequent state, the posterior can be

sampled at each step independently. So the posterior iterative update is

$$\begin{aligned} a_{t,k+1} &= \mathbf{a}_{t,k} + s\nabla_{\mathbf{a}_{t,k}} \log p_{\boldsymbol{\theta}}(\mathbf{a}_{t,k}|\mathbf{s}_{0:t+1}) + \sqrt{2s}\boldsymbol{\epsilon}_k \\ &= \mathbf{a}_{t,k} + s\nabla_{\mathbf{a}_{t,k}} \log p_{\boldsymbol{\theta}}(\mathbf{s}_{0:t}, \mathbf{a}_{t,k}, \mathbf{s}_{t+1}) + \sqrt{2s}\boldsymbol{\epsilon}_k \\ &= \mathbf{a}_{t,k} + s\nabla_{\mathbf{a}_{t,k}} (\underbrace{\log p_{\boldsymbol{\alpha}}(\mathbf{a}_{t,k}|\mathbf{s}_{0:t})}_{\text{policy/prior}} + \underbrace{\log p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t))}_{\text{transition}} + \sqrt{2s}\boldsymbol{\epsilon}_k. \end{aligned} \tag{E.13}$$

Intuitively, action samples at each step are updated with the energy of all subsequent actions and a single-step forward by back-propagation. However, while gradients from the transition term are analogous to the inverse dynamics in BCO (Torabi et al., 2018a), it may lead to poor training performance due to non-injectiveness in forward dynamics (Zhu et al., 2020).

We develop an alternative posterior sampling method with importance sampling to overcome this challenge. Leveraging the learned transition, we have

$$p_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_{0:t+1}) = \frac{p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}{\mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right]} p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}). \tag{E.14}$$

Let $c(\mathbf{a}_t; \mathbf{s}_{0:t+1}) = \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right]$, posterior sampling from $p_{\boldsymbol{\theta}}(\mathbf{a}_{0:T-1}|\mathbf{s}_{0:T})$ can be realized by adjusting importance weights of independent samples from the prior $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$, in which the estimation of weights involves another prior sampling. In this way, we avoid back-propagating through non-injective dynamics and save some computation overhead in (E.13).

To train the policy, (E.4) can now be rewritten as

$$\delta_{\boldsymbol{\alpha},t}(S) = \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[\frac{p_{\boldsymbol{\beta}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}{c(\mathbf{a}_t; \mathbf{s}_{0:t+1})} \nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right] - \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_{0:t})}\left[\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})\right]. \tag{E.15}$$

E.1.4 Algorithm

The learning and sampling algorithms with MCMC and with importance sampling for posterior sampling are described in Algorithm 10 and Algorithm 5.

Algorithm 10 LanMDP without importance sampling

Input: Learning iterations N, learning rate for energy-based policy $\eta_{\boldsymbol{\alpha}}$, learning rate for transition model $\eta_{\boldsymbol{\beta}}$, initial parameters $\boldsymbol{\theta}_0 = (\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0)$, expert demonstrations $\{\mathbf{s}_{0:H}\}$, context length L, batch size m, number of prior and posterior sampling steps $\{K_0, K_1\}$, prior and posterior sampling step sizes $\{\mathbf{s}_0, \mathbf{s}_1\}$.
Output: $\boldsymbol{\theta}_N = (\boldsymbol{\alpha}_N, \boldsymbol{\beta}_N)$.
Reorganize $\{\mathbf{s}_{0:H}\}$ to to state sequenec segments $(\mathbf{s}_{t-L+1}, \cdots, \mathbf{s}_{t+1})$ with length $L+1$.
Use energy-based policy with $\boldsymbol{\alpha}_0$ collect transitions to fill in the replay buffer.
Use transitions in replay buffer to pre-train transition model $\boldsymbol{\beta}_0$.
for $t = 0$ **to** $N-1$ **do**
  Demo sampling Sample observed examples $(\mathbf{s}_{t-L+1}, \cdots, \mathbf{s}_{t+1})_{i=1}^m$.
  Posterior sampling: Sample $\{\mathbf{a}_t\}_{i=1}^m$ using (E.13) with K_1 iterations and stepsize $\mathbf{s}_1$.
  Prior sampling: Sample $\{\hat{a}_t\}_{i=1}^m$ using (E.11) with K_0 iterations and stepsize $\mathbf{s}_0$.
  Policy learning: Update $\boldsymbol{\alpha}_t$ to $\boldsymbol{\alpha}_{t+1}$ by (E.4) with learning rate $\eta_{\boldsymbol{\alpha}}$.
  Transition learning: Update replay buffer with trajectories from current policy model $\boldsymbol{\alpha}_{t+1}$, then update $\boldsymbol{\beta}_t$ to $\boldsymbol{\beta}_{t+1}$ by (E.5) with learning rate $\eta_{\boldsymbol{\beta}}$.
end for

E.2 A Decision-making Problem in MLE

Let the ground-truth distribution of demonstrations be $p^*(\mathbf{s}_{0:T})$, and the learned marginal distributions of state sequences be $p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T})$. Equation (E.1) is an empirical estimate of

$$\mathbb{E}_{p^*(\mathbf{s}_{0:T})}[\log p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T})] = \mathbb{E}_{p^*(\mathbf{s}_0)}\left[\log p^*(\mathbf{s}_0) + \mathbb{E}_{p^*(\mathbf{s}_{1:T}|\mathbf{s}_0)}[\log p_{\boldsymbol{\theta}}(\mathbf{s}_{1:T}|\mathbf{s}_0)]\right]. \tag{E.16}$$

We can show that a sequential decision-making problem can be constructed to maximize the same objective. To start off, suppose the MLE yields the maximum, we will have $p_{\boldsymbol{\theta}^*} = p^*$.

Define $V^*(\mathbf{s}_0) := \mathbb{E}_{p^*(\mathbf{s}_{1:T}|\mathbf{s}_0)}[\log p^*(\mathbf{s}_{1:T}|\mathbf{s}_0)]$, we can generalize it to have a V *function*

$$V^*(\mathbf{s}_{0:t}) := \mathbb{E}_{p^*(\mathbf{s}_{t+1:T}|\mathbf{s}_{0:t})}[\log p^*(\mathbf{s}_{t+1:T}|\mathbf{s}_{0:t})], \tag{E.17}$$

which comes with a Bellman optimality equation

$$V^*(\mathbf{s}_{0:t}) = \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})}\left[r(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) + V^*(\mathbf{s}_{0:t+1})\right], \tag{E.18}$$

with $r(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) := \log p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t}) = \log \int p_{\boldsymbol{\alpha}^*}(\mathbf{a}_t|\mathbf{s}_{0:t}) p_{\boldsymbol{\beta}^*}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) d\mathbf{a}_t$, $V^*(\mathbf{s}_{0:T}) := 0$. It is worth noting that the r defined above involves the optimal policy, which may not be known a priori. We can resolve this by replacing it with $r_{\boldsymbol{\alpha}}$ for an arbitrary policy $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$. All Bellman identities and updates should still hold. Anyways, involving the current policy in the reward function should not appear to be too odd given the popularity of maximum entropy RL (Ziebart, 2010; Levine, 2018).

The entailed Bellman update, *value iteration*, for arbitrary V and $\boldsymbol{\alpha}$ is

$$V(\mathbf{s}_{0:t}) = \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})} \left[r_{\boldsymbol{\alpha}}(\mathbf{s}_{0:t}, \mathbf{s}_{t+1}) + V(\mathbf{s}_{0:t+1}) \right]. \tag{E.19}$$

We then define $r(\mathbf{s}_{t+1}, \mathbf{a}_t, \mathbf{s}_{0:t}) := r(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) + \log p_{\boldsymbol{\alpha}^*}(\mathbf{a}_t|\mathbf{s}_{0:t})$ to construct a Q function,

$$Q^*(\mathbf{a}_t; \mathbf{s}_{0:t}) := \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})} \left[r(\mathbf{s}_{t+1}, \mathbf{a}_t, \mathbf{s}_{0:t}) + V^*(\mathbf{s}_{0:t+1}) \right], \tag{E.20}$$

which entails a Bellman update, *Q backup*, for arbitrary $\boldsymbol{\alpha}$, Q and V:

$$Q(\mathbf{a}_t; \mathbf{s}_{0:t}) = \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})} \left[r_{\boldsymbol{\alpha}}(\mathbf{s}_{0:t}, \mathbf{a}_t, \mathbf{s}_{t+1}) + V(\mathbf{s}_{0:t+1}) \right]. \tag{E.21}$$

Also note that the V and Q in identities (E.19) and (E.21) respectively are not necessarily associated with the policy $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$. Slightly overloading the notations, we use $Q^{\boldsymbol{\alpha}}$, $V^{\boldsymbol{\alpha}}$ to denote the expected returns from policy $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$.

By now, we finish the construction of atomic algebraic components and move on to check if the relations between them align with the algebraic structure of a sequential decision-making problem (Sutton and Barto, 2018).

We first prove the construction above is valid at optimality.

Lemma 1. *When $f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t}) = Q^*(\mathbf{a}_t; \mathbf{s}_{0:t}) - V^*(\mathbf{s}_{0:t})$, $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$ is the optimal policy.*

Proof. Note that the construction gives us

$$\begin{aligned} Q^*(\mathbf{a}_t; \mathbf{s}_{0:t}) &= \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})} \left[r(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) + \log p_{\boldsymbol{\alpha}^*}(\mathbf{a}_t|\mathbf{s}_{0:t}) + V^*(\mathbf{s}_{0:t+1}) \right] \\ &= \log p_{\boldsymbol{\alpha}^*}(\mathbf{a}_t|\mathbf{s}_{0:t}) + \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})} \left[r(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) + V^*(\mathbf{s}_{0:t+1}) \right] \\ &= \log p_{\boldsymbol{\alpha}^*}(\mathbf{a}_t|\mathbf{s}_{0:t}) + V^*(\mathbf{s}_{0:t}). \end{aligned} \tag{E.22}$$

Obviously, $Q^*(\mathbf{a}_t; \mathbf{s}_{0:t})$ lies in the hypothesis space of $f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})$. □

Lemma 1 indicates that we need to either parametrize $f_{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})$ or $Q(\mathbf{a}_t; \mathbf{s}_{0:t})$.

While Q^* and V^* are constructed from the optimality, the derived $Q^{\boldsymbol{\alpha}}$ and $V^{\boldsymbol{\alpha}}$ measure the performance of an interactive agent when it executes with the policy $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$. They should be consistent with each other.

Lemma 2. *$V^{\boldsymbol{\alpha}}(\mathbf{s}_{0:t})$ and $\mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}[Q^{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})]$ yield the same optimal policy $p_{\boldsymbol{\alpha}^*}(\mathbf{a}_t|\mathbf{s}_{0:t})$.*

Proof.

$$\begin{aligned} &\mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}[Q^{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})] := \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})} \left[\mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})} \left[r(\mathbf{s}_{t+1}, \mathbf{a}_t, \mathbf{s}_{0:t}) + V^{\boldsymbol{\alpha}}(\mathbf{s}_{0:t+1}) \right] \right] \\ =& \mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})} \left[\mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})} \left[\log p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) + r(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) + V^{\boldsymbol{\alpha}}(\mathbf{s}_{0:t+1}) \right] \right] \\ =& \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})} \left[r(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) - \mathcal{H}_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) + V^{\boldsymbol{\alpha}}(\mathbf{s}_{0:t+1}) \right] \\ =& V^{\boldsymbol{\alpha}}(\mathbf{s}_{0:t}) - \mathcal{H}_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) - \sum\nolimits_{k=t+1}^{T-1} \mathbb{E}_{p^*(\mathbf{s}_{t+1:k}|\mathbf{s}_{0:t})}[\mathcal{H}_{\boldsymbol{\alpha}}(\mathbf{a}_k|\mathbf{s}_{0:k})], \end{aligned} \tag{E.23}$$

where the last line is derived by recursively applying the Bellman equation in the line above until $\mathbf{s}_{0:T}$ and then applying backup with (E.19). As an energy-based policy, $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$'s entropy is inherently maximized (Jaynes, 1957). Therefore, within the hypothesis space, $p_{\boldsymbol{\alpha}^*}(\mathbf{a}_t|\mathbf{s}_{0:t})$ that optimizes $V^{\boldsymbol{\alpha}}(\mathbf{s}_{0:t})$ also leads to optimal expected return $\mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}[Q^{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t})]$. □

If we parametrize the policy as $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) \propto \exp(Q^{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t}))$, the logarithmic normalizing constant $\log Z^{\boldsymbol{\alpha}_k}(\mathbf{s}_{0:t})$ will be the *soft V function* in maximum entropy RL (Fox

et al., 2016; Haarnoja et al., 2017, 2018)

$$V^{\boldsymbol{\alpha}}_{soft}(\mathbf{s}_{0:t}) := \log \int \exp(Q^{\boldsymbol{\alpha}}(\mathbf{a}_t; \mathbf{s}_{0:t}))d\mathbf{a}_t, \tag{E.24}$$

even if the reward function is defined differently. We can further show that Bellman identities and backup updates above can entail RL algorithms that achieve optimality of the decision-making objective $V^{\boldsymbol{\alpha}}$, including *soft policy iteration* (Ziebart, 2010)

$$p_{\boldsymbol{\alpha}_{k+1}}(\mathbf{a}_t|\mathbf{s}_{0:t}) \leftarrow \frac{\exp(Q^{\boldsymbol{\alpha}_k}(\mathbf{a}_t; \mathbf{s}_{0:t}))}{Z^{\boldsymbol{\alpha}_k}(\mathbf{s}_{0:t})}, \forall \mathbf{s}_{0:t}, k \in [0, 1, \ldots, M] \tag{E.25}$$

and *soft Q iteration* (Fox et al., 2016)

$$\begin{aligned} Q^{\boldsymbol{\alpha}_{k+1}}(\mathbf{a}_t; \mathbf{s}_{0:t}) &\leftarrow \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})}\left[r_{\boldsymbol{\alpha}}(\mathbf{s}_{0:t}, \mathbf{a}_t, \mathbf{s}_{t+1}) + V^{\boldsymbol{\alpha}_k}_{soft}(\mathbf{s}_{0:t+1})\right], \forall \mathbf{s}_{0:t}, \mathbf{a}_t, \\ V^{\boldsymbol{\alpha}_{k+1}}_{soft}(\mathbf{s}_{0:t}) &\leftarrow \log \int \exp(Q^{\boldsymbol{\alpha}_k}(a; \mathbf{s}_{0:t}))da, \forall \mathbf{s}_{0:t}, k \in [0, 1, \ldots, M]. \end{aligned} \tag{E.26}$$

Lemma 3. *If $p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})$ is accessible and $p_{\boldsymbol{\beta}^*}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is known, soft policy iteration and soft Q learning both converge to $p_{\boldsymbol{\alpha}^*}(\mathbf{a}_t|\mathbf{s}_{0:t}) = p_{Q^*}(\mathbf{a}_t|\mathbf{s}_{0:t}) \propto \exp(Q^*(\mathbf{a}_t; \mathbf{s}_{0:t}))$ under certain conditions.*

Proof. See the convergence proof by Ziebart (2010) for *soft policy iteration* and the proof by Fox et al. (2016) for *soft Q learning*. The latter requires Markovian assumption. But under some conditions, it can be extended to non-Markovian domains in the same way as proposed by Majeed and Hutter (2018). □

Lemma 3 means given $p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})$ and $p_{\boldsymbol{\beta}^*}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, we can recover $p_{\boldsymbol{\alpha}^*}$ through reinforcement learning methods, instead of the proposed MLE. So $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$ is a viable policy space for the constructed sequential decision-making problem.

Together, Lemma 1, Lemma 2, and Lemma 3 provide constructive proof for a valid sequential decision-making problem that maximizes the same objective of MLE, described by Theorem 1.

Theorem 1. *Assuming the Markovian transition $p_{\boldsymbol{\beta}^*}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is known, the ground-truth conditional state distribution $p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})$ for demonstration sequences is accessible, we can construct a sequential decision-making problem, based on a reward function $r_{\boldsymbol{\alpha}}(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) := \log \int p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) p_{\boldsymbol{\beta}^*}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) d\mathbf{a}_t$ for an arbitrary energy-based policy $p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$. Its objective is*

$$\sum\nolimits_{t=0}^{T} \mathbb{E}_{p^*(\mathbf{s}_{0:t})}[V^{p_{\boldsymbol{\alpha}}}(\mathbf{s}_{0:t})] = \mathbb{E}_{p^*(\mathbf{s}_{0:T})}\left[\sum\nolimits_{t=0}^{T} \sum\nolimits_{k=t}^{T} r_{\boldsymbol{\alpha}}(\mathbf{s}_{k+1}; \mathbf{s}_{0:k})\right],$$

where $V^{p_{\boldsymbol{\alpha}}}(\mathbf{s}_{0:t}) := E_{p^*(\mathbf{s}_{t+1:T}|\mathbf{s}_{0:t})}[\sum_{k=t}^{T} r_{\boldsymbol{\alpha}}(\mathbf{s}_{k+1}; \mathbf{s}_{0:k})]$ is the value function for $p_{\boldsymbol{\alpha}}$. This objective yields the same optimal policy as the Maximum Likelihood Estimation $\mathbb{E}_{p^*(\mathbf{s}_{0:T})}[\log p_{\boldsymbol{\theta}}(\mathbf{s}_{0:T})]$.

If we further define a reward function $r_{\boldsymbol{\alpha}}(\mathbf{s}_{t+1}, \mathbf{a}_t, \mathbf{s}_{0:t}) := r_{\boldsymbol{\alpha}}(\mathbf{s}_{t+1}, \mathbf{s}_{0:t}) + \log p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})$ to construct a Q function for $p_{\boldsymbol{\alpha}}$

$$Q^{p_{\boldsymbol{\alpha}}}(\mathbf{a}_t; \mathbf{s}_{0:t}) := \mathbb{E}_{p^*(\mathbf{s}_{t+1}|\mathbf{s}_{0:t})}\left[r_{\boldsymbol{\alpha}}(\mathbf{s}_{t+1}, \mathbf{a}_t, \mathbf{s}_{0:t}) + V^{p_{\boldsymbol{\alpha}}}(\mathbf{s}_{0:t+1})\right].$$

The expected return of $Q^{p_{\boldsymbol{\alpha}}}(\mathbf{a}_t; \mathbf{s}_{0:t})$ forms an alternative objective

$$\mathbb{E}_{p_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t})}[Q^{p_{\boldsymbol{\alpha}}}(\mathbf{a}_t; \mathbf{s}_{0:t})] = V^{p_{\boldsymbol{\alpha}}}(\mathbf{s}_{0:t}) - \mathcal{H}_{\boldsymbol{\alpha}}(\mathbf{a}_t|\mathbf{s}_{0:t}) - \sum\nolimits_{k=t+1}^{T-1} \mathbb{E}_{p^*(\mathbf{s}_{t+1:k}|\mathbf{s}_{0:t})}[\mathcal{H}_{\boldsymbol{\alpha}}(\mathbf{a}_k|\mathbf{s}_{0:k})]$$

that yields the same optimal policy, for which the optimal $Q^*(\mathbf{a}_t; \mathbf{s}_{0:t})$ can be the energy function.

Only under certain conditions, this sequential decision-making problem is solvable through non-Markovian extensions of the maximum entropy reinforcement learning algorithms.

E.3 More results of Curve Planning

The energy function is parameterized by a small MLP with one hidden layer and $4 * L$ hidden neurons, where L is the context length. In short-run Langevin dynamics, the number of samples, the number of sampling steps, and the stepsize are 4, 20 and 1

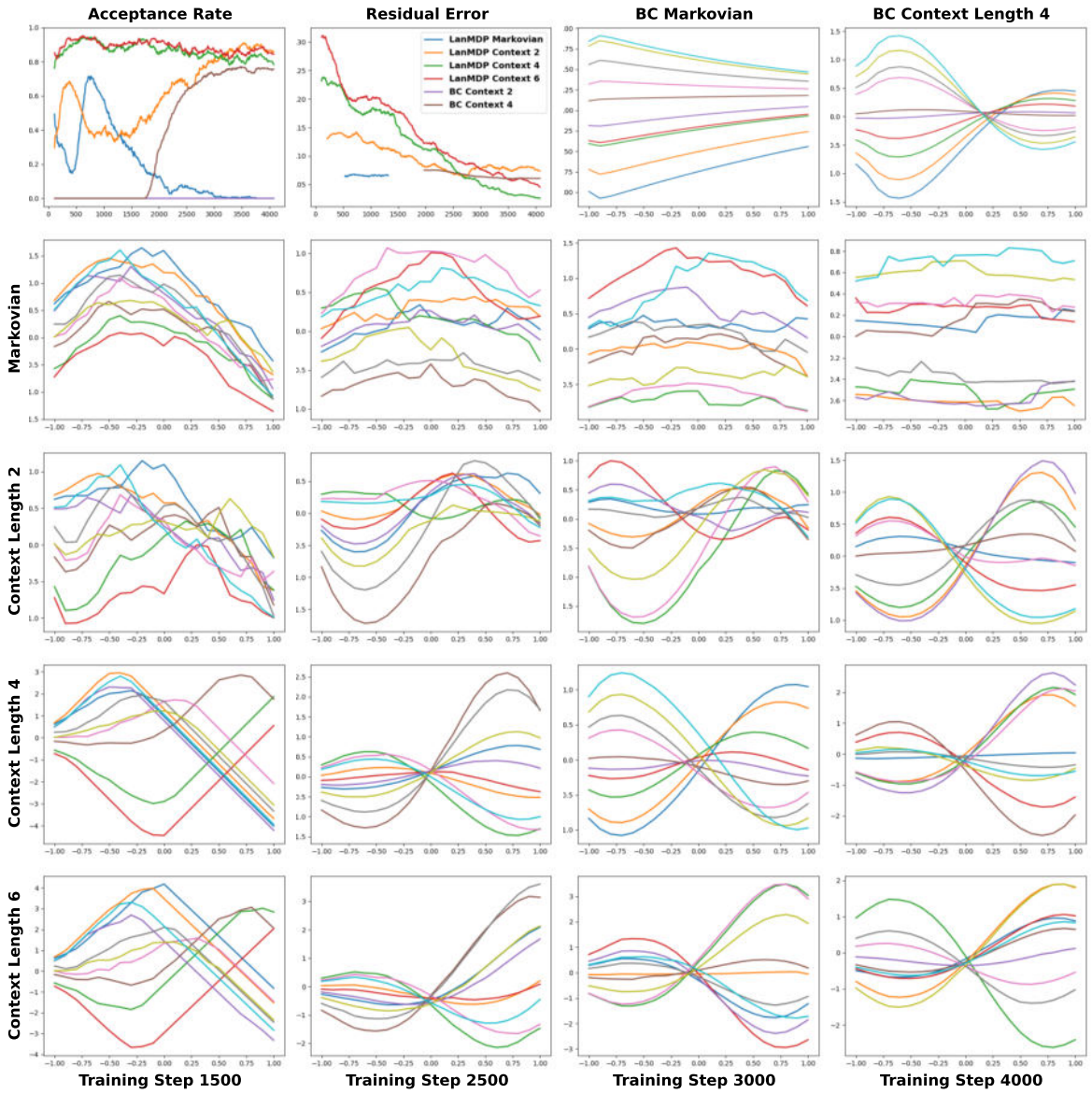


Figure E.1: More results for cubic curve generation

respectively. We use Adam optimizer with a learning rate 1e-4 and batch size 64. Here we present the complete result in Figure E.1 with different training steps under context length 1 2 4 6, the acceptance rate and residual error of the testing trajectories, as well as the behavior cloning results. We can see that even with sufficient context, BC performs worse than LanMDP. Also, from the result of context length 6 we can see that excessive expressivity does not impair performance, it just requires more training.

E.4 Implementation Details of MuJoCo Environment

This section delineates the configurations for the MuJoCo environments utilized in our research. In particular, we employ standard environment horizons of 500 and 50 for Cartpole-v1 and Reacher-v2, respectively. Meanwhile, for Swimmer-v2, Hopper-v2, and Walker2d-v2, we operate within an environment horizon set at 400 as referenced in previous literature (Kidambi et al., 2021, 2020; Kurutach et al., 2018; Luo et al., 2018; Nagabandi et al., 2018; Rajeswaran et al., 2020). Additional specifications are made for Hopper-v2 and Walker2d-v2, where the velocity of the center of mass was integrated into the state parameterization (Kidambi et al., 2021, 2020; Luo et al., 2018; Rajeswaran et al., 2020). We leverage PPO (Schulman et al., 2017) approach to train the expert policy until it reaches (approximately) 450, -10, 40, 3000, 2000 for Cartpole-v1, Reacher-v2, Swimmer-v2, Hopper-v2, Walker2d-v2 respectively. It should be noted that all results disclosed in the experimental section represent averages over five random seeds. Comparative benchmarks include BC (Ross and Bagnell, 2010), BCO (Torabi et al., 2018a), GAIL (Ho and Ermon, 2016), and GAIFO (Torabi et al., 2018b). MobILE (Kidambi et al., 2021) is a recent method for Markovian model-based imitation from observation. However, we failed to reproduce the expected performance utilizing various sets of demonstrations, so it is prudently omitted from the present displayed result. We specifically point out that BC/GAIL algorithms are privy to expert actions, however, our algorithm is not. We report the mean of the best performance achieved by BC/BCO with five random seeds, even though these peak performances may transpire at varying epochs. For

BC, we executed the supervised learning algorithm for 200 iterations. The BCO/GAIL algorithms are run with an equivalent number of online samples as LanMDP for a fair comparison. All benchmarking is performed using a single 3090Ti GPU and implemented using the PyTorch framework. Notably, in our codebase, the modified environments of Hopper-v2 and Walker2d-v2 utilize MobILE's implementation (Kidambi et al., 2021). Referring to the results in the main text, our presentation of normalized results in bar graph form is derived by normalizing each algorithm's performance (mean/standard deviation) against the expert mean. For Reacher-v2, due to the inherently negative rewards, we first add a constant offset of 20 to each algorithm' s performance, thus converting all values to positive before normalizing them against the mean of expert policy.

We parameterize both the policy model and the transition model as MLPs, and the non-linear activation function is Swish and LeakyReLU respectively. We use AdamW to optimize both policy and transition. To stabilize training, we prefer using actions around which the transition model is more certain for computing the expectation over importance-weighted prior distribution in (E.15). Therefore, we use a model ensemble with two transition models and use the disagreement between these two models to measure the uncertainty of the sampled actions. We implement Algorithm 5 for all experiments to avoid expensive computation of the gradient for the transition model in posterior sampling. As for better and more effective short-run Langevin sampling, we use a polynomially decaying schedule for the step size as recommended in (Grathwohl et al., 2019). We also use weakly L2 regularized energy magnitudes and clip gradient steps like (Du and Mordatch, 2019), choosing to clip the total amount of change value, i.e. after the gradient and noise have been combined. To realize more delicate decision-making, another trick in Implicit Behavior Clone (Florence et al., 2022) is also adopted for the inference/testing stage that we continue running the MCMC chain after the step size reaches the smallest in the polynomial schedule until we get twice as many inference Langevin steps as were used during training.

Hyper-parameters are listed in Table E.1. Other hyperparameters that are not mentioned here are left as default in PyTorch. Also, note that the Cartpole-v1 task has no

Parameter	Cartpole-v1	Reacher-v2	Swimmer-v2	Hopper-v2	Walker2d-v2
Environment Specification					
Horizon	500	50	400	400	400
Expert Performance ($\approx$)	450	-10	40	3000	2000
Transition Model					
Architecture	MLP(64;4)	MLP(64;4)	MLP(128;4)	MLP(512;4)	MLP(512;4)
Optimizer(LR)	3e-3	3e-3	3e-3	3e-3	3e-3
Batch Size	2500	20000	20000	32768	32768
Replay Buffer Size	2500	20000	20000	200000	200000
Policy Model (with context length L)					
Architecture	MLP(150*L;4)	MLP(150*L;4)	MLP(150*L;4)	MLP(512*L;4)	MLP(512*L;4)
Learning rate	1e-3	1e-2	1e-2	1e-2	5e-3
Batch Size	2500	20000	20000	32768	32768
Number of test trajectories	5	20	20	50	50
Sampling Parameters					
Number of prior samples	\	8	8	8	8
Number of Langevin steps	\	100	100	100	100
Langevin initial stepsize	\	10	10	10	10
Langevin ending stepsize	\	1	1	1	1

Table E.1: Hyper-parameters in MuJoCo experiments

parameters for sampling because expectation can be calculated analytically.

APPENDIX F

Derivations and Experimental Details for Chapter 7

F.1 Details About the Model and Learning

Given a trajectory $\boldsymbol{\tau}$, $\mathbf{z} \in \mathbb{R}^d$ is the latent vector to represent the variable-length trajectory. $y \in \mathbb{R}$ is the return of the trajectory. With offline training trajectory-return pairs $\{(\boldsymbol{\tau}_i, y_i), i = 1, \ldots, n\}$. The log-likelihood function is $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, y_i)$, with learning gradient $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, y_i)$. We derive the form of $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, y_i)$, proving (7.4) below, dropping index subscript $_i$ for simplicity.

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y) &= \frac{1}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y)} \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y) \\
&= \frac{1}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y)} \int \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y, \mathbf{z}) d\mathbf{z} \\
&= \frac{1}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y)} \int \nabla_{\boldsymbol{\theta}} \left(p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y | \mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) p_0(\mathbf{z}_0) \right) d\mathbf{z}_0 \\
&= \int \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y | \mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0))}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y)} \nabla_{\boldsymbol{\theta}} \log \left(p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y | \mathbf{z} = U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) p_0(\mathbf{z}_0) \right) d\mathbf{z}_0 \\
&= \int p_{\boldsymbol{\theta}}(\mathbf{z}_0 | \boldsymbol{\tau}, y) \nabla_{\boldsymbol{\theta}} \log \left(p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y | U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) p_0(\mathbf{z}_0) \right) d\mathbf{z}_0 \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_0 | \boldsymbol{\tau}, y)} \left[\nabla_{\boldsymbol{\theta}} \log \left(p_{\boldsymbol{\theta}}(\boldsymbol{\tau}, y | U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) p_0(\mathbf{z}_0) \right) \right] \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_0 | \boldsymbol{\tau}, y)} \left[\nabla_{\boldsymbol{\theta}} \log \left(p_{\boldsymbol{\beta}}(\boldsymbol{\tau} | U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) p_{\gamma}(y | U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) p_0(\mathbf{z}_0) \right) \right] \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_0 | \boldsymbol{\tau}, y)} \left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\beta}}(\boldsymbol{\tau} | U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) + \nabla_{\boldsymbol{\theta}} \log p_{\gamma}(y | U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) + \nabla_{\boldsymbol{\theta}} \log p_0(\mathbf{z}_0) \right] \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}_0 | \boldsymbol{\tau}, y)} \left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\beta}}(\boldsymbol{\tau} | U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) + \nabla_{\boldsymbol{\theta}} \log p_{\gamma}(y | U_{\boldsymbol{\alpha}}(\mathbf{z}_0)) \right].
\end{aligned}
$$

Parameter	HalfCheetah	Walker2D	Hopper	AntMaze
Number of layers	3	3	3	3
Number of attention heads	1	1	1	1
Embedding dimension	128	128	128	192
Context length	32	64	64	64
Learning rate	1e-4	1e-4	1e-4	1e-3
Langevin step size	0.3	0.3	0.3	0.3
Nonlinearity function	ReLU	ReLU	ReLU	ReLU

Table F.1: Gym-Mujoco Environments LPT Model Parameters

Parameter	Umaze	Medium	Large
Number of layers	1	3	4
Number of attention heads	8	1	4
Embedding dimension	128	192	192
Context length	32	64	64
Learning rate	1e-3	1e-3	2e-4
Langevin step size	0.3	0.3	0.3
Nonlinearity function	ReLU	ReLU	ReLU

Table F.2: Maze2D Environments LPT Model Parameters

F.2 Training Details

For Gym-Mujoco offline training, as shown in Table F.1, most of the hyperparameters were shared across all tasks except context length and hidden size. However, due to the significant variations in the scale of the maze maps and the lengths of the trajectories within the Maze2D environments — spanning umaze, medium, and large categories — model sizes were adjusted accordingly to accommodate these differences, where the detailed setting can be found in Table F.2. We also show the parameters for Franka Kitchen environment in Table F.3 and Connect Four in Table F.4.

Training time for the Gym-Mujoco tasks using a single Nvidia A6000 GPU is 18 hours on average. We train Maze2d tasks using a single Nvidia A100 GPU using 30 hours on average. Kitchen tasks using a single Nvidia A6000 GPU takes 60 hours on average. Connect-4 on a single Nvidia A6000 GPU takes 10 hours.

Parameter	Mixed	Partial
Number of layers	4	3
Number of attention heads	4	16
Embedding dimension	128	128
Context length	16	16
Learning rate	1e-3	1e-3
Langevin step size	0.3	0.3
Nonlinearity function	ReLU	ReLU

Table F.3: Franka Kitchen Environments LPT Model Parameters

Parameter	Value
Number of layers	3
Number of attention heads	4
Embedding dimension	128
Context length	4
Learning rate	1e-3
Langevin step size	0.3
Nonlinearity function	ReLU

Table F.4: Connect 4 LPT Model Parameters

F.3 Ablation Study

We investigate the role of the expressive prior $p_{\boldsymbol{\alpha}}(z)$ by removing the UNet, which transforms z_0 from a non-informative Gaussian. Table F.5 reports the result in three Mujoco control tasks and Connect Four. We observe that the performance of LPT drops in all of these environments. For example, in the stochastic environment Connect Four, LPT drops from 0.99 to 0.90, while DT, the baseline without latent variable, obtains 0.8. The result indicates that more flexible prior benefits the learning and inference of LPT.

Dataset	DT	LPT	LPT w/o UNet
halfcheetah-medium-replay	33.0 ± 4.8	39.64 ± 0.83	34.70 ± 1.58
hopper-medium-replay	50.8 ± 14.3	71.17 ± 3.01	53.41 ± 6.95
walker2d-medium-replay	51.6 ± 24.7	72.31 ± 1.92	56.88 ± 4.20
Connect Four	0.8 ± 0.07	0.99 ± 0.01	0.90 ± 0.06

Table F.5: Ablation study on Gym and Connect Four.

Dataset	Step-wise Reward		Final Return	
	ODT	Δ	LPT	Δ
halfcheetah-medium	42.16 ± 1.48	-0.56	43.26 ± 0.59	0.13
halfcheetah-medium-replay	40.2 ± 1.61	0.43	40.63 ± 0.28	0.99
hopper-medium	97.54 ± 2.1	30.59	64.84 ± 10.29	6.32
hopper-medium-replay	88.89 ± 6.33	2.25	72.44 ± 8.07	1.27
walker2d-medium	76.79 ± 2.30	4.60	79.54 ± 5.11	1.69
walker2d-medium-replay	76.86 ± 4.04	7.94	78.99 ± 3.84	6.68
Antmaze-umaze	88.5 ± 5.88	35.4	83.5 ± 3.28	22.3
Antmaze-diverse	56.00 ± 5.69	5.8	75.6 ± 2.8	8.0

Table F.6: Evaluation results of online Open AI Gym MuJoCo and Antmaze tasks. ODT baselines are sourced from (Zheng et al., 2022). Our results are reported over 5 seeds.

F.4 Continual Learning With Online Data

We are also interested in LPT's potential in finetuning or even continual learning. Inspired by ODT (Zheng et al., 2022), we employ a trajectory replay butter (Mnih et al., 2015) to store samples from online interaction in a first-in-first-out manner. After the completion of each episode, we update LPT with the same learning algorithm as with the offline data. Note that ODT introduces some techniques additional to DT. In contrast, LPT explores with the provably efficient posterior sampling (Osband et al., 2013; Osband and Van Roy, 2017). We report the results in Table F.6. Despite the significance in a few tasks, the improvement is within 1 standard deviation of the mean for the majority. We observe a similar pattern in ODT (Zheng et al., 2022).

APPENDIX G

Derivations and Experimental Details for Chapter 8

G.1 Expectation-Maximization

To learn a latent variable model $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})$, $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})d\mathbf{z}$ from a target distribution $q(\mathbf{x})$, the EM-like transformation on the gradient of the log-likelihood function is

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}}\, \mathbb{E}_{q(\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x})] \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}[\mathbb{E}_{q(\mathbf{x})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})]] \\
&= \mathbb{E}_{q(\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) + \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{q(\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})].
\end{aligned} \tag{G.1}$$

Line 3 is due to the simple equality that $\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})] = 0$.

G.2 Reparametrized Sampling and Noise Cancellation

Reparametrization. The EM-like algorithm we propose requires joint sampling of $(\mathbf{x}, \mathbf{z})$ from $\rho_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$. Like Nijkamp et al. (2021) andXiao et al. (2020), we utilize a reparameterization of $\mathbf{x}$ and $\mathbf{z}$ to overcome challenges in joint MCMC sampling, such as slow convergence and complex step size tuning. Notice that $\mathbf{x} = \alpha g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma\boldsymbol{\epsilon}$ defines a

deterministc mapping from $(\boldsymbol{\epsilon}, \mathbf{z})$ to $(\mathbf{x}, \mathbf{z})$. Then by *change of variable* we have

$$\begin{aligned}
\rho_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathbf{z}) d\boldsymbol{\epsilon} d\mathbf{z} &= \rho_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{x} d\mathbf{z} \\
&= \frac{q(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{x} d\mathbf{z} \\
&= \frac{q(\alpha g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma\boldsymbol{\epsilon})}{p_{\boldsymbol{\theta}}(\alpha g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma\boldsymbol{\epsilon})} p_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathbf{z}) d\boldsymbol{\epsilon} d\mathbf{z} \\
\Rightarrow \rho_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathbf{z}) &= \frac{q(\alpha g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma\boldsymbol{\epsilon})}{p_{\boldsymbol{\theta}}(\alpha g_{\boldsymbol{\theta}}(\mathbf{z}) + \sigma\boldsymbol{\epsilon})} p(\boldsymbol{\epsilon}) p(\mathbf{z}),
\end{aligned} \tag{G.2}$$

where $p(\boldsymbol{\epsilon})$ and $p(\mathbf{z})$ are standard Normal distributions.

The score functions become

$$\begin{aligned}
\nabla_{\boldsymbol{\epsilon}} \log \rho(\boldsymbol{\epsilon}, \mathbf{z}) &= \sigma(\nabla_{\mathbf{x}} \log q(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})) - \boldsymbol{\epsilon}, \\
\nabla_{\mathbf{z}} \log \rho(\boldsymbol{\epsilon}, \mathbf{z}) &= \alpha(\nabla_{\mathbf{x}} \log q(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})) \nabla_{\mathbf{z}} g_{\boldsymbol{\theta}}(\mathbf{z}) - \mathbf{z}.
\end{aligned} \tag{G.3}$$

Noise Cancellation The single-step Langevin update on $\boldsymbol{\epsilon}$ is then

$$\boldsymbol{\epsilon}^{i+1} = (1-\gamma)\boldsymbol{\epsilon}^i + \gamma\sigma\nabla_{\mathbf{x}} \log \frac{q(\mathbf{x}^i)}{p_{\boldsymbol{\theta}}(\mathbf{x}^i)} + \sqrt{2\gamma}\mathbf{n}^i. \tag{G.4}$$

Interestingly, we find the particular form of $\nabla_{\boldsymbol{\epsilon}} \log \rho(\boldsymbol{\epsilon}, \mathbf{z})$ results in a closed-form accumulation of multi-step updates

$$\boldsymbol{\epsilon}^{i+1} = (1-\gamma)^{i+1}\boldsymbol{\epsilon}^0 + \gamma \sum\nolimits_{k=0}^{i} (1-\gamma)^{i-k} \sigma \nabla_{\mathbf{x}} \log \frac{q(\mathbf{x}^i)}{p_{\boldsymbol{\theta}}(\mathbf{x}^i)} + \sum\nolimits_{k=0}^{i} (1-\gamma)^{i-k} \sqrt{2\gamma}\mathbf{n}^k, \tag{G.5}$$

which, after the push-forward, gives us

$$\begin{aligned}
\mathbf{x}^{i+1} =& \alpha g(\mathbf{z}^{i+1}) + \underbrace{\gamma \sum\nolimits_{k=0}^{i} (1-\gamma)^{i-k} \sigma^2 \nabla_{\mathbf{x}} \log \frac{q(\mathbf{x}^i)}{p_{\boldsymbol{\theta}}(\mathbf{x}^i)}}_{drift} \\
&+ \underbrace{(1-\gamma)^{i+1}\sigma\boldsymbol{\epsilon}^0 + \sum\nolimits_{k=0}^{i} (1-\gamma)^{i-k} \sqrt{2\gamma}\sigma\mathbf{n}^k}_{noise}.
\end{aligned} \tag{G.6}$$

lr_g	lr_s	batch size	Adam b_1	Adam b_2	$\gamma_{\boldsymbol{\epsilon}}$	$\gamma_{\mathbf{z}}$	K	λ^*	$\tilde{w}(t)$
2×10^{-6}	1×10^{-5}	128	0.0	0.99	0.4^2	0.004^2	16	3.2189	$\frac{\sigma_t^2}{\alpha_t^2}$

Table G.1

As $\mathbf{x}^{i+1}$ is effectively a regression target in (8.6), and the expected value of the *noise* is $\mathbf{0}$, we can remove it without biasing the gradient. Empirically, we find book-keeping the sampled noises in the MCMC chain and canceling these noises after the loop significantly stabilize the training of the generator network.

The same applies to the $(\mathbf{x}, \mathbf{z})$ sampling (with step size $\gamma\sigma^2$):

$$\begin{aligned}\mathbf{x}^{i+1} =& \mathbf{x}^i + \gamma\sigma^2(\nabla_{\mathbf{x}} \log q(\mathbf{x}^i) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}^i)) - \gamma(\mathbf{x}^i - \alpha g(\mathbf{z}^i)) + \sqrt{2\gamma} n^i \\ =& \gamma \sum_{k=1}^{i} (1-\gamma)^{i-k} \alpha g(\mathbf{z}^k) + \gamma \sum_{k=0}^{i} (1-\gamma)^{i-k} \sigma^2 (\nabla_{\mathbf{x}} \log q(\mathbf{x}^i) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}^i)) \\ &+ (1-\gamma)^i \alpha g(\mathbf{z}^0) + \underbrace{(1-\gamma)^{i+1} \sigma \boldsymbol{\epsilon}^0 + \sum_{k=0}^{i} (1-\gamma)^{i-k} \sqrt{2\gamma} \sigma n^k}_{noises}.\end{aligned} \tag{G.7}$$

G.3 Implementation Details

G.3.1 ImageNet 64x64

We train the teacher model using the best setting of EDM (Karras et al., 2022) with the ADM UNet architecture (Dhariwal and Nichol, 2021). We inherit the noise schedule and the score matching weighting function from the teacher. We run the distillation training for 300k steps (roughly 8 days) on 64 TPU-v4. We use $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector, in which both the teacher and the student score networks have a dropout probability of 0.1. We list other hyperparameters in Table G.1. Instead of listing t^*, we list the corresponding log signal-to-noise ratio λ^*.

lr_g	lr_s	batch size	Adam b_1	Adam b_2	$\gamma_{\boldsymbol{\epsilon}}$	$\gamma_{\mathbf{z}}$	K	λ^*	$\tilde{w}(t)$
2×10^{-6}	1×10^{-5}	1024	0.0	0.99	0.4^2	0.004^2	16	6	$\frac{\sigma_t^2}{\alpha_t}$

Table G.2

lr_g	lr_s	batch size	Adam b_1	Adam b_2	$\gamma_{\boldsymbol{\epsilon}}$	$\gamma_{\mathbf{z}}$	K	t^*	$\tilde{w}(t)$
2×10^{-6}	1×10^{-5}	1024	0.0	0.99	0.3^2	0.005^2	8	500	$\frac{\sigma_t^2}{\alpha_t}$

Table G.3

G.3.2 ImageNet 128x128

We train the teacher model following the best setting of VDM++ (Kingma and Gao, 2023) with the 'U-ViT, L' architecture (Hoogeboom et al., 2023). We use the 'cosine-adjusted' noise schedule (Hoogeboom et al., 2023) and 'EDM monotonic' weighting for student score matching. We run the distillation training for 200k steps (roughly 10 days) on 128 TPU-v5p. We use $(\boldsymbol{\epsilon}, \mathbf{z})$-corrector, in which both the teacher and the student score networks have a dropout probability of 0.1. We list other hyperparameters in Table G.2.

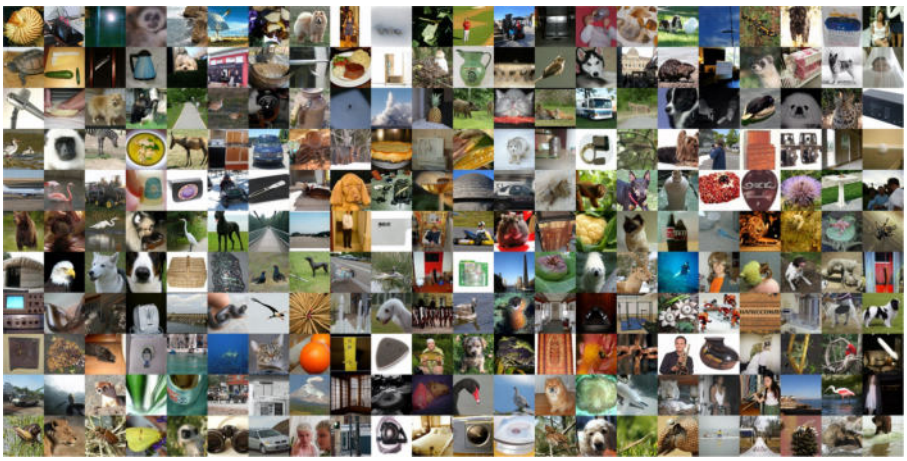
G.3.3 Text-to-Image Generation

We adopt the public checkpoint of Stable Diffusion v1.5 (Rombach et al., 2022) as the teacher. We inherit the noise schedule from the teacher model. The student score matching uses the same weighting function as the teacher. We list other hyperparameters in Table G.3.

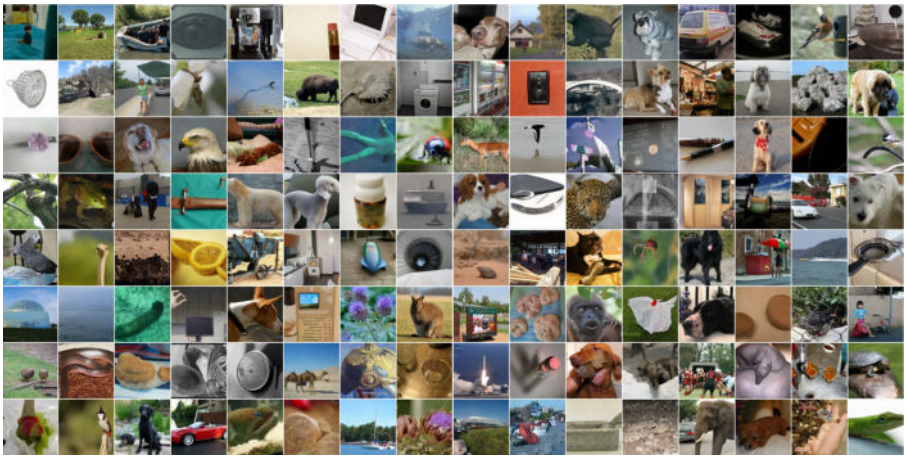
G.4 Additional Qualitative Results

G.4.1 Additional ImageNet Results

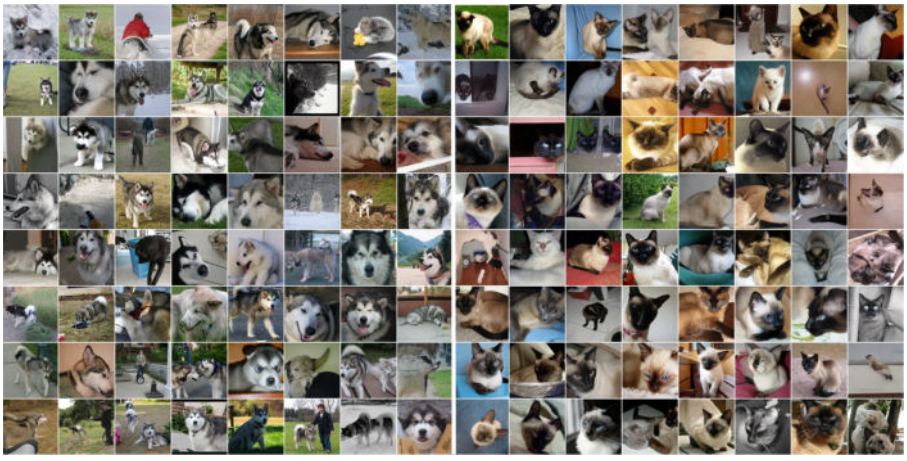
In this section, we present additional qualitative samples for our one-step generator on ImageNet 64×64 and ImageNet 128×128 in Figure G.1 to help further evaluate the generation quality and diversity.



(a) ImageNet 64×64 Multi-class



(b) ImageNet 128×128 Multi-class



(c) ImageNet 128×128 Single-class (Left: Husky, right: Siamese)

Figure G.1: Zoom in for better view.

G.4.2 Additional Text-to-Image Results

In this section, we present additional qualitative samples from our one-step generator distilled from Stable Diffusion 1.5. In Table G.4, Table G.5, Table G.6, and Table G.7, we visually compare the sample quality of our method with open-source competing methods for few- or single-step generation. We also include the teacher model in our comparison. We use the public checkpoints of LCM[1] and InstaFlow[2], where both checkpoints share the same Stable Diffusion 1.5 as teachers. Note that the SD-turbo results are obtained from the public checkpoint [3] fine-tuned from Stable Diffusion 2.1, which is different from our teacher model.

From the comparison, we observe that our model significantly outperforms distillation-based methods including LCM and InstaFlow, and it demonstrates better diversity and quality than GAN-based SD-turbo. The visual quality is on par with 50-step generation from the teacher model.

We show additional samples from our model on a more diverse set of prompts in Table Table G.8, Table G.9 and Table G.10.

[1] https://huggingface.co/latent-consistency/lcm-lora-sdv1-5

[2] https://huggingface.co/XCLiu/instaflow_0_9B_from_sd_1_5

[3] https://huggingface.co/stabilityai/sd-turbo

Teacher (50 step)

EMD (1 step)

LCM (1 step)

LCM (2 steps)

SD-Turbo (1 step)

InstaFlow (1 step)

Table G.4: More results on text-to-image. Prompt: *Dog graduation at university.*

Teacher (50 step)

EMD (1 step)

LCM (1 step)

LCM (2 steps)

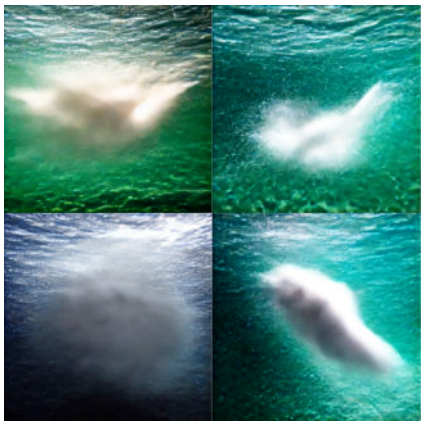
SD-Turbo (1 step)

InstaFlow (1 step)

Table G.5: More results on text-to-image. Prompt: *3D animation cinematic style young caveman kid, in its natural environment.*

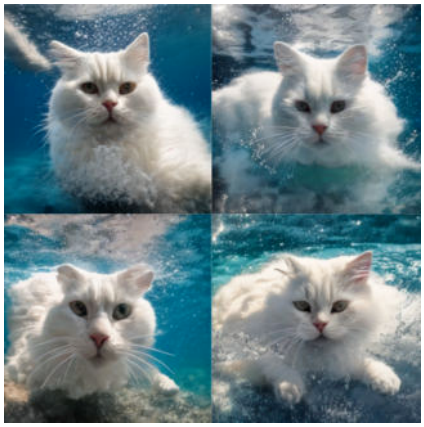
Teacher (50 step)

EMD (1 step)



LCM (1 step)

LCM (2 steps)



SD-Turbo (1 step)

InstaFlow (1 step)

Table G.6: More results on text-to-image. Prompt: *An underwater photo portrait of a beautiful fluffy white cat, hair floating. In a dynamic swimming pose. The sun rays filters through the water. High-angle shot. Shot on Fujifilm X.*

Teacher (50 step)

EMD (1 step)

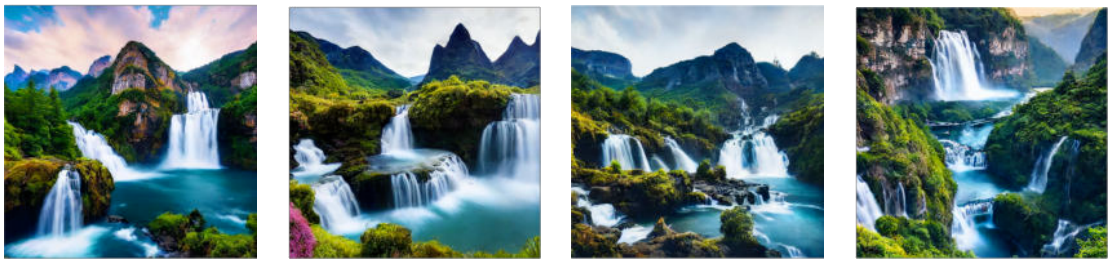
LCM (1 step)

LCM (2 steps)

SD-Turbo (1 step)

InstaFlow (1 step)

Table G.7: More results on text-to-image. Prompt: *A minimalist Teddy bear in front of a wall of red roses.*



A close-up photo of a intricate beautiful natural landscape of mountains and waterfalls.

A hyperrealistic photo of a fox astronaut; perfect face, artstation.

Large plate of delicious fried chicken, with a side of dipping sauce, realistic advertising photo, 4k.

A DSLR photo of a golden retriever in heavy snow.

Table G.8: Zoom-in for better viewing.

Masterpiece color pencil drawing of a horse, bright vivid color.

Oil painting of a wise old man with a white beard in the enchanted and magical forest.

3D render baby parrot, Chibi, adorable big eyes. In a garden with butterflies, greenery, lush whimsical and soft, magical, octane render, fairy dust.

Dreamy puppy surrounded by floating bubbles.

Table G.9: Zoom-in for better viewing.

A painting of an adorable rabbit sitting on a colorful splash.

Macro photo of a miniature toy sloth drinking a soda, shot on a light pastel cyclorama.

A traditional tea house in a tranquil garden with blooming cherry blossom trees.

Three cats having dinner at a table at new years eve, cinematic shot, 8k.

Table G.10: Zoom-in for better viewing.

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. 121

Abel, D., Dabney, W., Harutyunyan, A., Ho, M. K., Littman, M., Precup, D., and Singh, S. (2021). On the expressivity of markov reward. *Advances in Neural Information Processing Systems*, 34:7799–7812. 103, 105

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. (2018). Learning representations and generative models for 3d point clouds. In *International Conference on Machine Learning*, pages 40–49. PMLR. 81

Adams, R. and Bischof, L. (1994). Seeded region growing. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 16(6):641–647. 55

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J. B., Jaakkola, T. S., and Agrawal, P. (2023). Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*. 103, 122

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. (2021). OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*. 135, 136

Allen, C., Kirtland, A., Tao, R. Y., Lobel, S., Scott, D., Petrocelli, N., Gottesman, O., Parr, R., Littman, M. L., and Konidaris, G. (2024). Mitigating partial observability in sequential decision processes via the lambda discrepancy. *arXiv preprint arXiv:2407.07333*. 105

Amit, R. and Matari, M. (2002). Learning movement sequences from demonstration. In *Proceedings 2nd International Conference on Development and Learning. ICDL 2002*, pages 203–208. IEEE. 121

Ammanabrolu, P. and Riedl, M. O. (2021). Modeling worlds in text. In *Advances in Neural Information Processing Systems (NeurIPS)*. 23

Andreas, J. (2019). Measuring compositionality in representation learning. *arXiv preprint arXiv:1902.07181*. 81, 83, 84

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433. 74

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483. 121

Atkeson, C. G. and Schaal, S. (1997). Robot learning from demonstration. In *ICML*, volume 97, pages 12–20. Citeseer. 121

Attias, H. (2003). Planning by probabilistic inference. In *International Workshop on Artificial Intelligence and Statistics*, pages 9–16. PMLR. 104

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*. 30

Balaji, Y., Nah, S., Huang, X., Vahdat, A., Song, J., Zhang, Q., Kreis, K., Aittala, M., Aila, T., Laine, S., et al. (2022). eDIFF-I: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*. 160

Bao, F., Li, C., Zhu, J., and Zhang, B. (2022). Analytic-DPM: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. *arXiv preprint arXiv:2201.06503*. 154

Bapst, V., Sanchez-Gonzalez, A., Doersch, C., Stachenfeld, K., Kohli, P., Battaglia, P., and Hamrick, J. (2019). Structured agents for physical construction. In *International Conference on Machine Learning*, pages 464–474. PMLR. 74

Barratt, S. T. and Sharma, R. (2018). A note on the inception score. *CoRR*, abs/1801.01973. 83

Barrett, D., Hill, F., Santoro, A., Morcos, A., and Lillicrap, T. (2018). Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning*, pages 511–520. PMLR. 74, 79

Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013). Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*. 14

Benny, Y. and Wolf, L. (2020). OneGAN: Simultaneous unsupervised learning of conditional image generation, foreground segmentation, and fine-grained clustering. In *European Conference on Computer Vision (ECCV)*. 57, 68, 69

Berthelot, D., Autef, A., Lin, J., Yap, D. A., Zhai, S., Hu, S., Zheng, D., Talbott, W., and Gu, E. (2023). Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*. 144, 154, 159

Bhunia, A. K., Das, A., Muhammad, U. R., Yang, Y., Hospedales, T. M., Xiang, T., Gryaditskaya, Y., and Song, Y.-Z. (2020). Pixelor: A competitive sketching AI agent. so you think you can sketch? *ACM Transactions on Graphics (TOG)*, 39(6):1–15. 37

Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer Google Schola*, 2:1122–1128. 147

Bloom, P. (2002). *How children learn the meanings of words*. MIT press. 3

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*. 6

Botvinick, M. and Toussaint, M. (2012). Planning as inference. *Trends in Cognitive Sciences*, 16(10):485–488. 104

Bouchacourt, D. and Baroni, M. (2018). How agents see things: On visual representations in an emergent language game. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 35

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *Conference on Computational Natural Language Learning (CoNLL)*. 10, 29

Boykov, Y. Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *International Conference on Computer Vision (ICCV)*. 54

Brafman, R. I. and De Giacomo, G. (2019). Regular decision processes: A model for non-markovian domains. In *IJCAI*, pages 5516–5522. 105

Brandfonbrener, D., Bietti, A., Buckman, J., Laroche, R., and Bruna, J. (2022). When does return-conditioned supervised learning work for offline reinforcement learning? *Advances in Neural Information Processing Systems*, 35:1542–1553. 134

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*. 159

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2016). Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*. 66, 195

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov chain Monte Carlo*. CRC press. 60

Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., and Ramesh, A. (2024). Video generation models as world simulators. *OpenAI Technical Report*. 1, 144

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*. 9

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901. 1, 6, 103, 142

Burda, Y., Grosse, R. B., and Salakhutdinov, R. (2016). Importance weighted autoencoders. In *International Conference on Learning Representations (ICLR)*. 175

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*. 55, 57, 74, 79, 80, 82

Cao, K., Lazaridou, A., Lanctot, M., Leibo, J. Z., Tuyls, K., and Clark, S. (2018). Emergent communication through negotiation. In *International Conference on Learning Representations (ICLR)*. 33, 34

Card, D., Tan, C., and Smith, N. A. (2018). Neural models for documents with metadata. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 28

Carey, S. (2000). The origin of concepts. *Journal of Cognition and Development*, 1(1):37–41. 2

Chattopadhyay, P., Vedantam, R., Selvaraju, R. R., Batra, D., and Parikh, D. (2017). Counting everyday objects in everyday scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1135–1144. 74, 75, 83

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, 34:15084–15097. 103, 121, 122, 123, 124, 127, 135, 137, 138

Chen, M., Artières, T., and Denoyer, L. (2019a). Unsupervised object segmentation by redrawing. In *Advances in Neural Information Processing Systems (NeurIPS)*. 56, 68, 71, 206

Chen, R. T., Li, X., Grosse, R., and Duvenaud, D. (2019b). Isolating sources of disentanglement in VAEs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2615–2625. 80, 82, 89

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR. 122

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*. 66, 80

Cheng, C.-A., Xie, T., Jiang, N., and Agarwal, A. (2022). Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*, pages 3852–3878. PMLR. 135

Cheng, M.-M., Mitra, N. J., Huang, X., Torr, P. H., and Hu, S.-M. (2014). Global contrast based salient region detection. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(3):569–582. 54

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 175

Cohen, L. D. (1991). On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218. 55

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. (2023). Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*. 166

Dai, J., He, K., and Sun, J. (2015). BoxSup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *International Conference on Computer Vision (ICCV)*. 58

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38. 4, 145, 147

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee. 155

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794. 131, 238

Di Pellegrino, G., Fadiga, L., Fogassi, L., Gallese, V., and Rizzolatti, G. (1992). Understanding motor events: A neurophysiological study. *Experimental Brain Research*, 91:176–180. 102

Dosovitskiy, A. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 76

Du, Y., Li, S., Tenenbaum, J., and Mordatch, I. (2021). Improved contrastive divergence training of energy based models. In *International Conference on Machine Learning (ICML)*. 30

Du, Y. and Mordatch, I. (2019). Implicit generation and generalization in energy-based models. *CoRR*, abs/1903.08689. 230

Eastwood, C. and Williams, C. K. (2018). A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*. 80

Eccles, T., Bachrach, Y., Lever, G., Lazaridou, A., and Graepel, T. (2019). Biases for emergent communication in multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. 33

Eitz, M., Hays, J., and Alexa, M. (2012). How do humans sketch objects? *ACM Transactions on Graphics (TOG)*, 31(4):1–10. 36

El-Kishky, A., Selsam, D., Song, Parascandolo, F. G., Ren, H., Lightman, H., Chung, H. W., Akkaya, I., Sutskever, I., Wei, J., Gordon, J., Cobbe, K., Yu, K., Kondraciuk, L., Schwarzer, M., Rohaninejad, M., Brown, N., Zhao, S., Bansal, T., Kosaraju, V., and Zhou, W. (2024). Learning to reason with LLMs. *OpenAI Technical Report*. 165

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. (2021). RvS: What is essential for offline RL via supervised learning? *arXiv preprint arXiv:2112.10751*. 135

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. (2020). GENESIS: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations (ICLR)*. 55, 57, 79

Eric, M., Krishnan, L., Charette, F., and Manning, C. D. (2017). Key-value retrieval networks for task-oriented dialogue. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDial)*. 27

Eslami, S. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., and Hinton, G. E. (2016). Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*. 55, 57, 74

Eslami, S. A., Rezende, D. J., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., et al. (2018). Neural scene representation and rendering. *Science*, 360(6394):1204–1210. 81, 82

Evtimova, K., Drozdov, A., Kiela, D., and Cho, K. (2018). Emergent communication in a multi-modal, multi-step referential game. In *International Conference on Learning Representations (ICLR)*. 33, 34, 36

Eysenbach, B., Udatha, S., Salakhutdinov, R. R., and Levine, S. (2022). Imitating past successes can be very suboptimal. *Advances in Neural Information Processing Systems*, 35:6047–6059. 134

Fan, J. E., Hawkins, R. D., Wu, M., and Goodman, N. D. (2020). Pragmatic inference and visual abstraction enable contextual flexibility during visual communication. *Computational Brain & Behavior*, 3(1):86–101. 37

Fang, L., Li, C., Gao, J., Dong, W., and Chen, C. (2019). Implicit deep latent variable models for text generation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 10, 29

Fay, N., Ellison, M., and Garrod, S. (2014). Iconicity: From sign to system in human communication and language. *Pragmatics & Cognition*, 22(2):244–263. 32, 33, 34

Fay, N., Garrod, S., Roberts, L., and Swoboda, N. (2010). The interactive evolution of human communication systems. *Cognitive Science*, 34(3):351–386. 35, 36

Fay, N., Walker, B., Swoboda, N., and Garrod, S. (2018). How to create shared symbols. *Cognitive Science*, 42:241–269. 34

Ferrer-Mestres, J., Dietterich, T. G., Buffet, O., and Chades, I. (2020). Solving k-MDPs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 110–118. 116

Finn, C., Christiano, P. F., Abbeel, P., and Levine, S. (2016). A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *CoRR*, abs/1611.03852. 108, 121, 219

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. (2022). Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR. 230

Foerster, J. N., Assael, Y. M., De Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. 34

Forgues, G., Pineau, J., Larchevêque, J.-M., and Tremblay, R. (2014). Bootstrapping dialog systems with word embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*. 27

Fox, R., Pakman, A., and Tishby, N. (2016). Taming the noise in reinforcement learning via soft updates. In *32nd Conference on Uncertainty in Artificial Intelligence 2016, UAI 2016*, pages 202–211. Association For Uncertainty in Artificial Intelligence (AUAI). 104, 225, 226

Freed, B., Venkatraman, S., Sartoretti, G. A., Schneider, J., and Choset, H. (2023). Learning temporally abstractworld models without online experimentation. In *International Conference on Machine Learning*, pages 10338–10356. PMLR. 136

Fu, H., Li, C., Liu, X., Gao, J., Çelikyilmaz, A., and Carin, L. (2019). Cyclical annealing schedule: A simple approach to mitigating kl vanishing. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 29

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2020). D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*. 125, 136, 139

Fujimoto, S. and Gu, S. S. (2021). A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:20132–20145. 135

Gafni, O., Polyak, A., Ashual, O., Sheynin, S., Parikh, D., and Taigman, Y. (2022). Make-a-scene: Scene-based text-to-image generation with human priors. In *European Conference on Computer Vision*, pages 89–106. Springer. 160

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. (2024). Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*. 166

Gao, R., Lu, Y., Zhou, J., Zhu, S., and Wu, Y. N. (2018). Learning generative convnets via multi-grid modeling and sampling. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9155–9164. 147

Gao, R., Song, Y., Poole, B., Wu, Y. N., and Kingma, D. P. (2020). Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*. 10, 11, 14, 19, 30, 147

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680. 56

Gordon, J., Lopez-Paz, D., Baroni, M., and Bouchacourt, D. (2019). Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*. 83, 84

Goyal, A., Lamb, A., Gampa, P., Beaudoin, P., Levine, S., Blundell, C., Bengio, Y., and Mozer, M. (2020). Object files and schemata: Factorizing declarative and procedural knowledge in dynamical systems. *arXiv preprint arXiv:2006.16225*. 74

Graesser, L., Cho, K., and Kiela, D. (2019). Emergent linguistic phenomena in multi-agent communication games. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 33, 36

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. (2019). Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*. 10, 18, 30, 230

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning (ICML)*. 55, 57, 58, 64, 65, 70, 74, 75, 79, 80, 81, 82, 89, 92, 97, 189, 190, 196, 206

Greff, K., Rasmus, A., Berglund, M., Hao, T. H., Schmidhuber, J., and Valpola, H. (2016). Tagger: Deep unsupervised perceptual grouping. In *Advances in Neural Information Processing Systems (NeurIPS)*. 55, 57, 70, 189, 190

Greff, K., van Steenkiste, S., and Schmidhuber, J. (2017). Neural expectation maximization. In *Advances in Neural Information Processing Systems (NeurIPS)*. 55, 57, 79

Greff, K., Van Steenkiste, S., and Schmidhuber, J. (2020). On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*. 74

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1462–1471. 57

Grenander, U. (1970). A unified approach to pattern analysis. In *Advances in Computers*, volume 10, pages 175–216. Elsevier. 1, 163

Grenander, U. and Miller, M. I. (2007). *Pattern theory: from representation to inference*. Oxford University Press. 1, 163

Gu, J., Zhai, S., Zhang, Y., Liu, L., and Susskind, J. M. (2023). Boot: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*. 144, 159

Guo, C.-e., Zhu, S.-C., and Wu, Y. N. (2007). Primal sketch: Integrating structure and texture. *Computer Vision and Image Understanding (CVIU)*, 106(1):5–19. 55, 62

Gupta, A., Agarwal, A., Singh, P., and Rai, P. (2018). A deep generative framework for paraphrase generation. In *AAAI Conference on Artificial Intelligence (AAAI)*. 9, 29

Gururangan, S., Dang, T., Card, D., and Smith, N. A. (2019). Variational pretraining for semi-supervised text classification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 176

Ha, D. and Eck, D. (2018). A neural representation of sketch drawings. In *International Conference on Learning Representations (ICLR)*. 34, 35, 36, 81

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR. 104, 105, 112, 226

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR. 104, 112, 226

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2019). Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*. 40

Han, T., Lu, Y., Zhu, S., and Wu, Y. N. (2017). Alternating back-propagation for generator network. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 1976–1984. 130, 133, 147

Han, T., Nijkamp, E., Fang, X., Hill, M., Zhu, S.-C., and Wu, Y. N. (2019). Divergence triangle for joint training of generator model, energy-based model, and inferential model. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 17

Han, T., Nijkamp, E., Zhou, L., Pang, B., Zhu, S.-C., and Wu, Y. N. (2020). Joint training of variational auto-encoder and latent energy-based model. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 30

Harispe, S., Ranwez, S., Janaqi, S., and Montmain, J. (2015). Semantic similarity from natural language and ontology analysis. *Synthesis Lectures on Human Language Technologies*, 8(1):1–254. 35

Havrylov, S. and Titov, I. (2017). Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in Neural Information Processing Systems (NeurIPS)*. 33, 36, 40

Hawkins, R. X., Sano, M., Goodman, N. D., and Fan, J. E. (2019). Disentangling contributions of visual information and interaction history in the formation of graphical conventions. In *Annual Meeting of the Cognitive Science Society (CogSci)*. 32, 33, 34, 35, 36

Hayes, G. M. and Demiris, J. (1994). *A robot controller using learning by imitation*. University of Edinburgh, Department of Artificial Intelligence. 102, 121

He, J., Spokoyny, D., Neubig, G., and Berg-Kirkpatrick, T. (2018). Lagging inference networks and posterior collapse in variational autoencoders. In *International Conference on Learning Representations (ICLR)*. 29

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738. 89, 98, 122

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-CNN. In *International Conference on Computer Vision (ICCV)*. 68, 188

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 68, 76, 188

Heek, J., Hoogeboom, E., and Salimans, T. (2024). Multistep consistency models. *arXiv preprint arXiv:2403.06807*. 144, 154, 159, 160

Heider, F. and Simmel, M. (1944). An experimental study of apparent behavior. *The American Journal of Psychology*, 57(2):243–259. 5

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6626–6637. 155

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2016). beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*. 29, 74, 75, 80, 82, 83

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. 154

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., et al. (2022). Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*. 144

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, 29. 108, 118, 121, 219, 229

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851. 11, 15, 19, 21, 30, 135, 144, 146

Ho, J. and Salimans, T. (2022). Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*. 131

Hoffmann, D. L., Standish, C. D., García-Diez, M., Pettitt, P. B., Milton, J. A., Zilhão, J., Alcolea-González, J. J., Cantalejo-Duarte, P., Collado, H., De Balbín, R., et al. (2018). U-Th dating of carbonate crusts reveals neandertal origin of Iberian cave art. *Science*, 359(6378):912–915. 32, 82

Hoogeboom, E., Heek, J., and Salimans, T. (2023). Simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pages 13213–13232. PMLR. 239

Horner, V. and Whiten, A. (2005). Causal knowledge and imitation/emulation switching in chimpanzees (pan troglodytes) and children (homo sapiens). *Animal Cognition*, 8:164–181. 104, 117

Huang, J. and Murphy, K. (2015). Efficient inference in occlusion-aware generative models of images. *arXiv preprint arXiv:1511.06362*. 74, 82

Huang, Z., Heng, W., and Zhou, S. (2019). Learning to paint with model-based deep reinforcement learning. In *International Conference on Computer Vision (ICCV)*. 34, 36, 39, 40

Huang, Z., Wang, X., Wang, J., Liu, W., and Wang, J. (2018). Weakly-supervised semantic segmentation network with deep seeded region growing. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 58

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218. 79

Hutter, M. (2009). Feature reinforcement learning: Part i. unstructured MDPs. *Journal of Artificial General Intelligence*, 1(1):3–24. 105

Hyvärinen, A. and Dayan, P. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4). 146

Icarte, R. T., Klassen, T., Valenzano, R., and McIlraith, S. (2018). Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR. 105

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456. 66

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87. 55, 60

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. (2022). Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR. 103, 122, 135

Janner, M., Li, Q., and Levine, S. (2021). Offline reinforcement learning as one big sequence modeling problem. *Advances in Neural Information Processing Systems*, 34:1273–1286. 103, 120, 121, 122, 123, 135

Jayaraman, D. and Grauman, K. (2015). Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1413–1421. 81, 84

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical Review*, 106(4):620. 225

Ji, X., Henriques, J. F., and Vedaldi, A. (2019). Invariant information clustering for unsupervised image classification and segmentation. In *International Conference on Computer Vision (ICCV)*. 58

Jiang, H., Wang, J., Yuan, Z., Wu, Y., Zheng, N., and Li, S. (2013). Salient object detection: A discriminative regional feature integration approach. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 54

Jin, S., Wiseman, S., Stratos, K., and Livescu, K. (2020). Discrete latent variable representations for low-resource text classification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 176

Jing, M., Ma, X., Huang, W., Sun, F., and Liu, H. (2019). Task transfer by preference-based cost learning. In *AAAI Conference on Artificial Intelligence (AAAI)*. 30

Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*. 20

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. (2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 67, 69, 76, 79, 189

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214. 55, 60

Julesz, B. (1981). Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97. 62

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134. 105

Kahneman, D., Treisman, A., and Gibbs, B. J. (1992). The reviewing of object files: Object-specific integration of information. *Cognitive Psychology*, 24(2):175–219. 4

Kampelmuhler, M. and Pinz, A. (2020). Synthesizing human-like sketches from natural images using a conditional convolutional decoder. In *Proceedings of Winter Conference on Applications of Computer Vision (WACV)*. 36, 39

Kanezaki, A. (2018). Unsupervised image segmentation by backpropagation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 58

Kang, M., Zhu, J.-Y., Zhang, R., Park, J., Shechtman, E., Paris, S., and Park, T. (2023). Scaling up GANs for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10124–10134. 160

Kappen, H. J., Gómez, V., and Opper, M. (2012). Optimal control as a graphical model inference problem. *Machine Learning*, 87:159–182. 110

Karazija, L., Laina, I., and Rupprecht, C. (2021). ClevrTex: A texture-rich benchmark for unsupervised multi-object segmentation. *arXiv preprint arXiv:2111.10265*. 79, 89

Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577. 154, 155, 159, 238

Karras, T., Laine, S., and Aila, T. (2018). A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*. 160

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision (IJCV)*, 1(4):321–331. 55

Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., et al. (2019). Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*. 81

Khoreva, A., Benenson, R., Hosang, J., Hein, M., and Schiele, B. (2017). Simple does it: Weakly supervised instance and semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 58

Khosla, A., Jayadevaprakash, N., Yao, B., and Li, F.-F. (2011). Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*. 67, 71

Kidambi, R., Chang, J., and Sun, W. (2021). MobILE: Model-based imitation learning from observation alone. *arXiv preprint arXiv:2102.10769*. 121, 229, 230

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. (2020). Morel: Model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21810–21823. 229

Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. (2023). Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*. 154, 155

Kim, H. and Mnih, A. (2018). Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR. 79, 80, 83

Kim, I. Y. and De Weck, O. L. (2005). Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29(2):149–158. 46

Kingma, D. and Gao, R. (2023). Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36. 146, 156, 239

Kingma, D., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models. *Advances in Neural Information Processing Systems*, 34:21696–21707. 146

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 174, 191

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 44

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*. 175

Kingma, D. P. and Welling, M. (2014a). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 10, 13, 29, 71, 112, 135, 175, 220

Kingma, D. P. and Welling, M. (2014b). Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*. 41

Kong, D., Xu, D., Zhao, M., Pang, B., Xie, J., Lizarraga, A., Huang, Y., Xie, S., and Wu, Y. N. (2024). Latent plan transformer: Planning as latent variable inference. *arXiv preprint arXiv:2402.04647*. 6

Kosiorek, A., Kim, H., Teh, Y. W., and Posner, I. (2018). Sequential attend, infer, repeat: Generative modelling of moving objects. *Advances in Neural Information Processing Systems*, 31. 74

Kostrikov, I., Nair, A., and Levine, S. (2021). Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*. 135

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243. 89

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3D object representations for fine-grained categorization. In *ICCV workshops*. 67, 71

Kumar, A., Sattigeri, P., and Balakrishnan, A. (2018). Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*. 80

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191. 124, 135, 137

Kumaran, D., Hassabis, D., and McClelland, J. L. (2016). What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534. 166

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. (2018). Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*. 229

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., and Aila, T. (2019). Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32. 155, 157

Lake, B. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR. 75, 81, 83

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. (2020). Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33:19884–19895. 122

Lazaridou, A. and Baroni, M. (2020). Emergent multi-agent communication in the deep learning era. *arXiv preprint arXiv:2006.02419*. 33

Lazaridou, A., Hermann, K. M., Tuyls, K., and Clark, S. (2018). Emergence of linguistic communication from referential games with symbolic and pixel input. In *International Conference on Learning Representations (ICLR)*. 33, 34, 36, 40

Lazaridou, A., Peysakhovich, A., and Baroni, M. (2017). Multi-agent cooperation and the emergence of (natural) language. In *International Conference on Learning Representations (ICLR)*. 33, 34, 36, 37, 40

Leclerc, Y. G. (1989). Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision (IJCV)*, 3(1):73–102. 55

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, 1(0). 121

Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., et al. (2022). Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936. 135

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*. 104, 110, 224

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*. 103, 123

Lewis, D. K. (1969). *Convention: A Philosophical Study*. John Wiley & Sons. 44, 47

Li, B., He, J., Neubig, G., Berg-Kirkpatrick, T., and Yang, Y. (2019a). A surprisingly effective fix for deep latent variable modeling of text. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 29

Li, J., Jia, R., He, H., and Liang, P. (2018). Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 27

Li, M., Lin, Z., Mech, R., Yumer, E., and Ramanan, D. (2019b). Photo-sketching: Inferring contour drawings from images. In *Proceedings of Winter Conference on Applications of Computer Vision (WACV)*. 39

Li, P., Lam, W., Bing, L., and Wang, Z. (2017a). Deep recurrent generative decoder for abstractive text summarization. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 9, 29

Li, Y., Su, H., Shen, X., Li, W., Cao, Z., and Niu, S. (2017b). DailyDialog: A manually labelled multi-turn dialogue dataset. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 25

Li, Y., Wang, H., Jin, Q., Hu, J., Chemerys, P., Fu, Y., Wang, Y., Tulyakov, S., and Ren, J. (2023). SnapFusion: Text-to-image diffusion model on mobile devices within two seconds. *arXiv preprint arXiv:2306.00980*. 154

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*. 138

Lin, S., Wang, A., and Yang, X. (2024). SDXL-Lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*. 155

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*. 68, 161, 188

Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. (2020). SPACE: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations (ICLR)*. 55, 57

Liu, X., Zhang, X., Ma, J., Peng, J., et al. (2023). InstaFlow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*. 154, 160, 162

Liu, Y., Gupta, A., Abbeel, P., and Levine, S. (2018). Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125. IEEE. 121

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*. 55, 57, 58, 64, 65, 70, 75, 79, 80, 81, 82, 83, 88, 89, 92, 93, 99, 189, 196

Lorenz, D., Bereska, L., Milbich, T., and Ommer, B. (2019). Unsupervised part-based disentangling of object shape and appearance. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 58

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. (2022a). DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787. 154, 159

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. (2022b). DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*. 154, 160, 161

Luhman, E. and Luhman, T. (2021). Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*. 154

Luo, S., Tan, Y., Huang, L., Li, J., and Zhao, H. (2023a). Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*. 154

Luo, S., Tan, Y., Patil, S., Gu, D., von Platen, P., Passos, A., Huang, L., Li, J., and Zhao, H. (2023b). LCM-LoRA: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*. 160, 161, 162

Luo, W., Hu, T., Zhang, S., Sun, J., Li, Z., and Zhang, Z. (2023c). Diff-Instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36. 144, 145, 148, 150, 152, 153, 155, 158, 159, 161

Luo, Y., Xu, H., Li, Y., Tian, Y., Darrell, T., and Ma, T. (2018). Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*. 229

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. (2020). Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR. 135

Lyons, D. E., Young, A. G., and Keil, F. C. (2007). The hidden structure of overimitation. *Proceedings of the National Academy of Sciences*, 104(50):19751–19756. 104

Majeed, S. J. and Hutter, M. (2018). On Q-learning convergence for non-Markov decision processes. In *IJCAI*, volume 18, pages 2546–2552. 226

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330. 23

Marino, J., Yue, Y., and Mandt, S. (2018). Iterative amortized inference. In *International Conference on Machine Learning (ICML)*. 57

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. (2017). dSprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/. 67, 69, 189

Meng, C., Rombach, R., Gao, R., Kingma, D., Ermon, S., Ho, J., and Salimans, T. (2023). On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306. 154

Merity, S., Keskar, N. S., and Socher, R. (2018). Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations (ICLR)*. 175

Miao, Y., Yu, L., and Blunsom, P. (2016). Neural variational inference for text processing. In *International Conference on Machine Learning (ICML)*. 28

Mihai, D. and Hare, J. (2021). Learning to draw: Emergent communication through sketching. In *Advances in Neural Information Processing Systems (NeurIPS)*. 36, 37

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. 36, 49

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*. 23, 175

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26. 75, 77, 80

Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 27, 75

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 172

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533. 235

Mordatch, I. and Abbeel, P. (2018). Emergence of grounded compositional language in multi-agent populations. In *AAAI Conference on Artificial Intelligence (AAAI)*. 33

Muhammad, U. R., Yang, Y., Song, Y.-Z., Xiang, T., and Hospedales, T. M. (2018). Learning deep sketch abstraction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 37

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. (2018). Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE. 229

Neal, R. M. (2011). MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2. 129, 146

Ng, A. Y. and Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2. 103, 121

Nguyen, T. H. and Tran, A. (2023). SwiftBrush: One-step text-to-image diffusion model with variational score distillation. *arXiv preprint arXiv:2312.05239*. 148, 155

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. (2021). Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*. 160

Nie, W., Vahdat, A., and Anandkumar, A. (2021). Controllable and compositional generation with latent-space energy-based models. *Advances in Neural Information Processing Systems*, 34:13497–13510. 11, 31

Nijkamp, E., Gao, R., Sountsov, P., Vasudevan, S., Pang, B., Zhu, S.-C., and Wu, Y. N. (2021). Mcmc should mix: Learning energy-based model with neural transport latent space mcmc. In *International Conference on Learning Representations*. 147, 151, 236

Nijkamp, E., Hill, M., Han, T., Zhu, S.-C., and Wu, Y. N. (2020a). On the anatomy of MCMC-based maximum likelihood learning of energy-based models. In *AAAI Conference on Artificial Intelligence (AAAI)*. 10, 30, 147, 166

Nijkamp, E., Hill, M., Zhu, S.-C., and Wu, Y. N. (2019). Learning non-convergent non-persistent short-run MCMC toward energy-based model. *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, Canada*. 10, 30, 71, 104, 108, 145, 147, 221

Nijkamp, E., Pang, B., Han, T., Zhou, L., Zhu, S.-C., and Wu, Y. N. (2020b). Learning multi-layer latent variable model via variational optimization of short run MCMC for approximate inference. In *European Conference on Computer Vision (ECCV)*, pages 361–378. 130

Oh, S. J., Benenson, R., Khoreva, A., Akata, Z., Fritz, M., and Schiele, B. (2017). Exploiting saliency for object segmentation from image level labels. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 58

Osband, I., Russo, D., and Van Roy, B. (2013). (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26. 141, 235

Osband, I. and Van Roy, B. (2017). Why is posterior sampling better than optimism for reinforcement learning? In *International Conference on Machine Learning*, pages 2701–2710. PMLR. 131, 141, 235

Ostyakov, P., Suvorov, R., Logacheva, E., Khomenko, O., and Nikolenko, S. I. (2018). SeiGAN: Towards compositional image generation by simultaneously learning to segment, enhance, and inpaint. *arXiv preprint arXiv:1811.07630*. 56

Pang, B., Han, T., Nijkamp, E., Zhu, S.-C., and Wu, Y. N. (2020a). Learning latent space energy-based prior model. In *Advances in Neural Information Processing Systems (NeurIPS)*. 10, 30, 55, 59, 71, 103, 107, 121, 127, 129, 172, 194

Pang, B., Han, T., and Wu, Y. N. (2020b). Learning latent space energy-based prior model for molecule generation. *arXiv preprint arXiv:2010.09351*. 30

Pang, B. and Wu, Y. N. (2021). Latent space energy-based model of symbol-vector coupling for text generation and classification. In *International Conference on Machine Learning (ICML)*. 10, 12, 13, 21, 23, 27, 28, 29, 30, 60, 174, 175

Pang, B., Zhao, T., Xie, X., and Wu, Y. N. (2021). Trajectory prediction with latent belief energy-based model. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 30

Papandreou, G., Chen, L.-C., Murphy, K. P., and Yuille, A. L. (2015). Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *International Conference on Computer Vision (ICCV)*. 58

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 23

Paster, K., McIlraith, S., and Ba, J. (2022). You can' t count on luck: Why decision transformers and rvs fail in stochastic environments. *Advances in Neural Information Processing Systems*, 35:38966–38979. 134, 135, 137, 141

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. 65

Pathak, D., Krahenbuhl, P., and Darrell, T. (2015). Constrained convolutional neural networks for weakly supervised segmentation. In *International Conference on Computer Vision (ICCV)*. 58

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 175

Pham, T. T., Do, T.-T., Sünderhauf, N., and Reid, I. (2018). SceneCut: Joint geometric and object segmentation for indoor scenes. In *International Conference on Robotics and Automation (ICRA)*. 58

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. (2022). DreamFusion: Text-to-3D using 2D diffusion. *arXiv preprint arXiv:2209.14988*. 144, 145, 148, 154, 155

Qin, A., Gao, F., Li, Q., Zhu, S.-C., and Xie, S. (2023). Learning non-Markovian decision-making from state-only sequences. In *Thirty-Seventh Conference on Neural Information Processing Systems*. 5

Qiu, S., Xie, S., Fan, L., Gao, T., Joo, J., Zhu, S.-C., and Zhu, Y. (2022). Emergent graphical conventions in a visual communication game. *Advances in Neural Information Processing Systems*, 35:13119–13131. 3

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR. 155

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 66, 75, 80

Rajeswaran, A., Mordatch, I., and Kumar, V. (2020). A game theoretic framework for model based reinforcement learning. In *International Conference on Machine Learning*, pages 7953–7963. PMLR. 229

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3. 144, 160

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. Pmlr. 1, 160

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. (2006). Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 729–736. 121

Ren, Y., Guo, S., Labeau, M., Cohen, S. B., and Kirby, S. (2020). Compositional languages emerge in a neural iterated learning model. In *International Conference on Learning Representations (ICLR)*. 33, 34, 36

Ren, Y., Xia, X., Lu, Y., Zhang, J., Wu, J., Xie, P., Wang, X., and Xiao, X. (2024). Hyper-SD: Trajectory segmented consistency model for efficient image synthesis. *arXiv preprint arXiv:2404.13686*. 155

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1278–1286. 10, 29

Rizzolatti, G., Fogassi, L., and Gallese, V. (2001). Neurophysiological mechanisms underlying the understanding and imitation of action. *Nature Reviews Neuroscience*, 2(9):661–670. 102

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695. 144, 159, 160, 161, 239

Ronca, A., Licks, G. P., and De Giacomo, G. (2022). Markov abstractions for PAC reinforcement learning in non-markov decision processes. *arXiv preprint arXiv:2205.01053*. 105

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer. 127

Ross, S. and Bagnell, D. (2010). Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 661–668. JMLR Workshop and Conference Proceedings. 118, 132, 229

Rother, C., Kolmogorov, V., and Blake, A. (2004). GrabCut interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314. 54, 72, 202

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268. 67, 195

Rus, V. and Lintean, M. (2012). An optimal assessment of natural language student input using word-to-word similarity metrics. In *International Conference on Intelligent Tutoring Systems*. 27

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494. 1, 144, 160

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234. 155

Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*. 144, 154, 159

Sangkloy, P., Burnell, N., Ham, C., and Hays, J. (2016). The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12. 36, 40, 42

Sauer, A., Boesel, F., Dockhorn, T., Blattmann, A., Esser, P., and Rombach, R. (2024). Fast high-resolution image synthesis with latent adversarial diffusion distillation. *arXiv preprint arXiv:2403.12015*. 155

Sauer, A., Lorenz, D., Blattmann, A., and Rombach, R. (2023). Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*. 144, 155, 162

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*. 66, 191

Sayim, B. and Cavanagh, P. (2011). What line drawings reveal about the visual brain. *Frontiers in Human Neuroscience*, 5:118. 39

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634. 79, 81, 82

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. (2022). Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294. 159

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. 118, 229

Serban, I., Sordoni, A., Bengio, Y., Courville, A., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI Conference on Artificial Intelligence (AAAI)*. 9, 29

Serban, I., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio, Y. (2017). A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI Conference on Artificial Intelligence (AAAI)*. 29

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905. 54

Shi, W., Zhou, H., Miao, N., and Li, L. (2020). Dispersed exponential family mixture vaes for interpretable text generation. In *International Conference on Machine Learning (ICML)*. 10, 23, 27, 29, 30, 175

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision (ECCV)*. 58

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*. 41

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*. 40

Singh, G., Deng, F., and Ahn, S. (2021). Illiterate DALL-E learns to compose. *arXiv preprint arXiv:2110.11405*. 74, 75

Singh, K. K., Ojha, U., and Lee, Y. J. (2019). FineGAN: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 57

Singh, S. P., Jaakkola, T., and Jordan, M. I. (1994). Learning without state-estimation in partially observable markovian decision processes. In *Machine Learning Proceedings 1994*, pages 284–292. Elsevier. 103

Sinha, A., Song, J., Meng, C., and Ermon, S. (2021). D2C: Diffusion-denoising models for few-shot conditional generation. In *Advances in Neural Information Processing Systems (NeurIPS)*. 31

Sinha, S., Mandlekar, A., and Garg, A. (2022). S4RL: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*, pages 907–917. PMLR. 122

Snell, C., Lee, J., Xu, K., and Kumar, A. (2024). Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*. 166

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*. 11, 13, 30, 144, 146

Song, J., Meng, C., and Ermon, S. (2020a). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*. 144, 154, 159

Song, J., Pang, K., Song, Y.-Z., Xiang, T., and Hospedales, T. M. (2018). Learning to sketch with shortcut cycle consistency. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 35

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023). Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR. 144, 154, 159, 160

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*. 30, 146, 147, 157

Song, Y. and Ermon, S. (2020). Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*. 11, 30

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*. 11, 30, 31, 144, 146, 147

Štrupl, M., Faccio, F., Ashley, D. R., Schmidhuber, J., and Srivastava, R. K. (2022). Upside-down reinforcement learning can diverge in stochastic environments with episodic resets. *arXiv preprint arXiv:2205.06595*. 134

Subramanian, S., Rajeswar, S., Sordoni, A., Trischler, A., Courville, A., and Pal, C. (2018). Towards text generation with adversarially learned neural outlines. In *Advances in Neural Information Processing Systems (NeurIPS)*. 176

Sun, W., Vemula, A., Boots, B., and Bagnell, D. (2019). Provably efficient imitation learning from observation alone. In *International Conference on Machine Learning*, pages 6036–6045. PMLR. 103, 121

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press. 3, 5, 34, 41, 103, 105, 124, 183, 184, 224

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063. 41

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. (2023). Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*. 1, 6, 142

Templeton, A. (2024). *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic. 166

Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine learning*, pages 1064–1071. 130

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*. 16, 32

Todorov, E. (2006). Linearly-solvable Markov decision problems. *Advances in Neural Information Processing Systems*, 19. 105

Todorov, E. (2008). General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pages 4286–4292. IEEE. 110

Torabi, F., Warnell, G., and Stone, P. (2018a). Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*. 109, 118, 222, 229

Torabi, F., Warnell, G., and Stone, P. (2018b). Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*. 103, 118, 121, 229

Toro Icarte, R., Waldie, E., Klassen, T., Valenzano, R., Castro, M., and McIlraith, S. (2019). Learning reward machines for partially observable reinforcement learning. *Advances in Neural Information Processing Systems*, 32. 105

Toussaint, M. (2009). Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1049–1056. 110

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*. 1

Tu, Z. and Zhu, S.-C. (2002). Image segmentation by data-driven markov chain Monte Carlo. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):657–673. 54, 79

Uehara, M. and Sun, W. (2021). Pessimistic model-based offline reinforcement learning under partial coverage. In *International Conference on Learning Representations*. 135

Ullman, M. T. (2004). Contributions of memory circuits to language: The declarative/procedural model. *Cognition*, 92(1-2):231–270. 166

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*. 66

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2020). Deep image prior. *International Journal of Computer Vision (IJCV)*, 128(7). 58

Vahdat, A. and Kautz, J. (2020). NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems (NeurIPS)*. 31

Vahdat, A., Kreis, K., and Kautz, J. (2021). Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems (NeurIPS)*. 11, 31

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)*, 9(11). 50

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *International Conference on Learning Representations (ICLR)*. 55, 57, 74

Varin, C., Reid, N., and Firth, D. (2011). An overview of composite likelihood methods. *Statistica Sinica*, pages 5–42. 108

Vedantam, R., Szlam, A., Nickel, M., Morcos, A., and Lake, B. M. (2021). Curi: A benchmark for productive concept learning under uncertainty. In *International Conference on Machine Learning*, pages 10519–10529. PMLR. 74

Villaflor, A. R., Huang, Z., Pande, S., Dolan, J. M., and Schneider, J. (2022). Addressing optimism bias in sequence modeling for reinforcement learning. In *International Conference on Machine Learning*, pages 22270–22283. PMLR. 134

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Computation*. 30, 146

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., and Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research (JMLR)*. 175

Voynov, A., Morozov, S., and Babenko, A. (2021). Object segmentation without labels with large-scale generative models. In *International Conference on Machine Learning (ICML)*. 58

Wang, A., Ren, M., and Zemel, R. (2021). SketchEmbedNet: Learning novel concepts by imitating drawings. In *International Conference on Machine Learning (ICML)*. 35

Wang, W., Gan, Z., Xu, H., Zhang, R., Wang, G., Shen, D., Chen, C., and Carin, L. (2019). Topic-guided variational auto-encoder for text generation. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 10

Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., and Zhu, J. (2024). ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36. 145, 148, 150, 152, 153, 155

Wehenkel, A. and Louppe, G. (2021). Diffusion priors in variational autoencoders. *arXiv preprint arXiv:2106.15671*. 31

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010). Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology. 67

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688. 13, 19, 60, 65, 191, 193

Wen, T.-H., Miao, Y., Blunsom, P., and Young, S. (2017). Latent intention dialogue models. In *International Conference on Machine Learning (ICML)*. 10, 29

Wen, Y., Yang, Y., Luo, R., Wang, J., and Pan, W. (2019). Probabilistic recursive reasoning for multi-agent reinforcement learning. In *International Conference on Learning Representations (ICLR)*. 34

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256. 46

Wu, Y., Jiang, A. Q., Li, W., Rabe, M., Staats, C., Jamnik, M., and Szegedy, C. (2022). Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368. 166

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019). Detectron2. `https://github.com/facebookresearch/detectron2`. 68, 188

Xia, X. and Kulis, B. (2017). W-net: A deep model for fully unsupervised image segmentation. *arXiv preprint arXiv:1711.08506*. 58

Xiao, Z., Kreis, K., Kautz, J., and Vahdat, A. (2020). VAEBM: A symbiosis between variational autoencoders and energy-based models. In *International Conference on Learning Representations*. 151, 236

Xiao, Z., Kreis, K., and Vahdat, A. (2021). Tackling the generative learning trilemma with denoising diffusion GANs. In *International Conference on Learning Representations*. 144

Xie, J., Lu, Y., Gao, R., and Wu, Y. N. (2018). Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32. 147

Xie, J., Lu, Y., Zhu, S., and Wu, Y. N. (2016). A theory of generative convnet. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2635–2644. 30, 147

Xie, S., Morcos, A. S., Zhu, S.-C., and Vedantam, R. (2022). COAT: Measuring object compositionality in emergent representations. In *International Conference on Machine Learning*, pages 24388–24413. PMLR. 4

Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *International Conference on Computer Vision (ICCV)*. 39

Xie, S., Xiao, Z., Kingma, D. P., Hou, T., Wu, Y. N., Murphy, K. P., Salimans, T., Poole, B., and Gao, R. (2024). EM distillation for one-step diffusion models. *arXiv preprint arXiv:2405.16852*. 6

Xie, T., Cheng, C.-A., Jiang, N., Mineiro, P., and Agarwal, A. (2021). Bellman-consistent pessimism for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:6683–6694. 135

Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., and Tang, J. (2022). GeoDIFF: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*. 144

Xu, Y., Zhao, Y., Xiao, Z., and Hou, T. (2023). UFOGen: You forward once large scale text-to-image generation via diffusion GANs. *arXiv preprint arXiv:2311.09257*. 144, 155, 160

Yamagata, T., Khalil, A., and Santos-Rodriguez, R. (2023). Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline RL. In *International Conference on Machine Learning*, pages 38989–39007. PMLR. 126, 134, 135, 137, 138, 139

Yan, H., Liu, X., Pan, J., Liew, J. H., Liu, Q., and Feng, J. (2024). PeRFlow: Piecewise rectified flow as universal plug-and-play accelerator. *arXiv preprint arXiv:2405.07510*. 154

Yang, J., Kannan, A., Batra, D., and Parikh, D. (2017). LR-GAN: Layered recursive generative adversarial networks for image generation. In *International Conference on Learning Representations (ICLR)*. 56

Yang, M., Cho, K., Merchant, A., Abbeel, P., Schuurmans, D., Mordatch, I., and Cubuk, E. D. (2023). Scalable diffusion for materials generation. *arXiv preprint arXiv:2311.09235*. 144

Yang, S., Schuurmans, D., Abbeel, P., and Nachum, O. (2022). Dichotomy of control: Separating what you can control from what you cannot. In *The Eleventh International Conference on Learning Representations*. 134, 135, 141

Yang, Y., Lai, B., and Soatto, S. (2021). DyStaB: Unsupervised object segmentation via dynamic-static bootstrapping. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 58

Yang, Y., Loquercio, A., Scaramuzza, D., and Soatto, S. (2019). Unsupervised moving object detection via contextual information separation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 58

Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W. T., and Park, T. (2024). One-step diffusion with distribution matching distillation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 144, 148, 155, 159, 160, 161

Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013). POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*. 9

Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., et al. (2022a). Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5. 160

Yu, P., Xie, S., Ma, X., Jia, B., Pang, B., Gao, R., Zhu, Y., Zhu, S.-C., and Wu, Y. N. (2022b). Latent diffusion energy-based model for interpretable text modeling. *arXiv preprint arXiv:2206.05895*. 3

Yu, P., Xie, S., Ma, X., Zhu, Y., Wu, Y. N., and Zhu, S.-C. (2021). Unsupervised foreground extraction via deep region competition. In *Advances in Neural Information Processing Systems (NeurIPS)*. 4, 30, 97

Yu, P., Zhu, Y., Xie, S., Ma, X. S., Gao, R., Zhu, S.-C., and Wu, Y. N. (2023). Learning energy-based prior model with diffusion-amortized MCMC. *Advances in Neural Information Processing Systems*, 36:42717–42747. 166

Yu, Q., Liu, F., Song, Y.-Z., Xiang, T., Hospedales, T. M., and Loy, C.-C. (2016). Sketch me that shoe. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 36

Yuan, L., Gao, X., Zheng, Z., Edmonds, M., Wu, Y. N., Rossano, F., Lu, H., Zhu, Y., and Zhu, S.-C. (2022). In situ bidirectional human-robot value alignment. *Science Robotics*, 7(68). 36

Zaslavsky, N., Kemp, C., Regier, T., and Tishby, N. (2018). Efficient compression in color naming and its evolution. *Proceedings of the National Academy of Sciences (PNAS)*, 115(31):7937–7942. 46

Zeng, Y., Zhuge, Y., Lu, H., and Zhang, L. (2019). Joint learning of saliency detection and weakly supervised semantic segmentation. In *International Conference on Computer Vision (ICCV)*. 58

Zhang, B., Xiong, D., Su, J., Duan, H., and Zhang, M. (2016). Variational neural machine translation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 29

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 49

Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., Tian, Y., et al. (2022). Efficient planning in a compact latent action space. In *The Eleventh International Conference on Learning Representations*. 135

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*. 28

Zhao, J., Kim, Y., Zhang, K., Rush, A., and LeCun, Y. (2018a). Adversarially regularized autoencoders. In *International Conference on Machine Learning*, pages 5902–5911. 9, 23, 29

Zhao, T., Lee, K., and Eskenazi, M. (2018b). Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 10, 23, 25, 29, 175

Zhao, T., Zhao, R., and Eskenazi, M. (2017). Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 9, 29

Zhao, W., Bai, L., Rao, Y., Zhou, J., and Lu, J. (2024). UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36. 160, 161

Zheng, H., Nie, W., Vahdat, A., Azizzadenesheli, K., and Anandkumar, A. (2023). Fast sampling of diffusion models via operator learning. In *International Conference on Machine Learning*, pages 42390–42402. PMLR. 144, 159

Zheng, J., Hu, M., Fan, Z., Wang, C., Ding, C., Tao, D., and Cham, T.-J. (2024). Trajectory consistency distillation. *arXiv preprint arXiv:2402.19159*. 154

Zheng, Q., Zhang, A., and Grover, A. (2022). Online decision transformer. In *International Conference on Machine Learning*, pages 27042–27059. PMLR. 135, 137, 141, 235

Zhu, S. C., Wu, Y., and Mumford, D. (1998). Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126. 55, 121

Zhu, S.-C. and Yuille, A. (1996). Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(9):884–900. 4, 54, 55, 60, 64, 65, 79

Zhu, W., Liang, S., Wei, Y., and Sun, J. (2014). Saliency optimization from robust background detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 54

Zhu, Z., Lin, K., Dai, B., and Zhou, J. (2020). Off-policy imitation learning from observations. *Advances in Neural Information Processing Systems*, 33:12402–12413. 109, 118, 121, 222

Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University. 104, 110, 112, 224, 226

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1433–1438. 103, 104, 105, 106, 121

Zou, C., Yu, Q., Du, R., Mo, H., Song, Y.-Z., Xiang, T., Gao, C., Chen, B., and Zhang, H. (2018). SketchyScene: Richly-annotated scene sketches. In *European Conference on Computer Vision (ECCV)*. 35