UCLA Computer Science

STAR AI RESEARCH LAB UCLA

# Knowledge and Data in Neuro-Symbolic Learning

Guy Van den Broeck

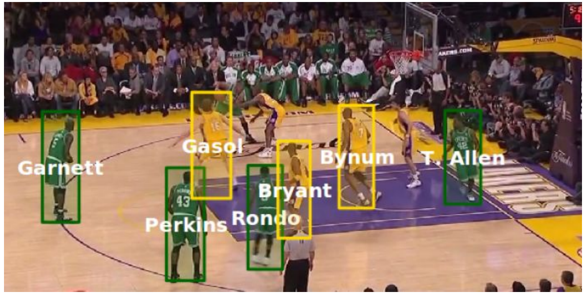
2022 Summit of the Mellichamp Mind & Machine Initiative - Feb 16, 2022

# The AI Dilemma

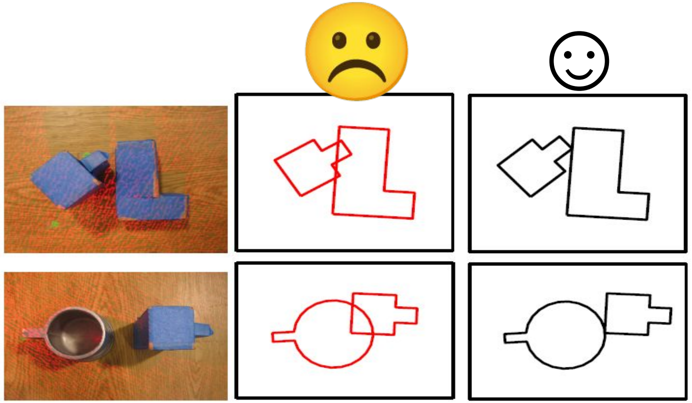

- Learn statistical models subject to symbolic knowledge
- Integrate reasoning into modern learning algorithms

*Today: Deep learning with constraints*
*Learning monotonic neural networks*

# Knowledge in Vision, Robotics, NLP




People appear at most once in a frame

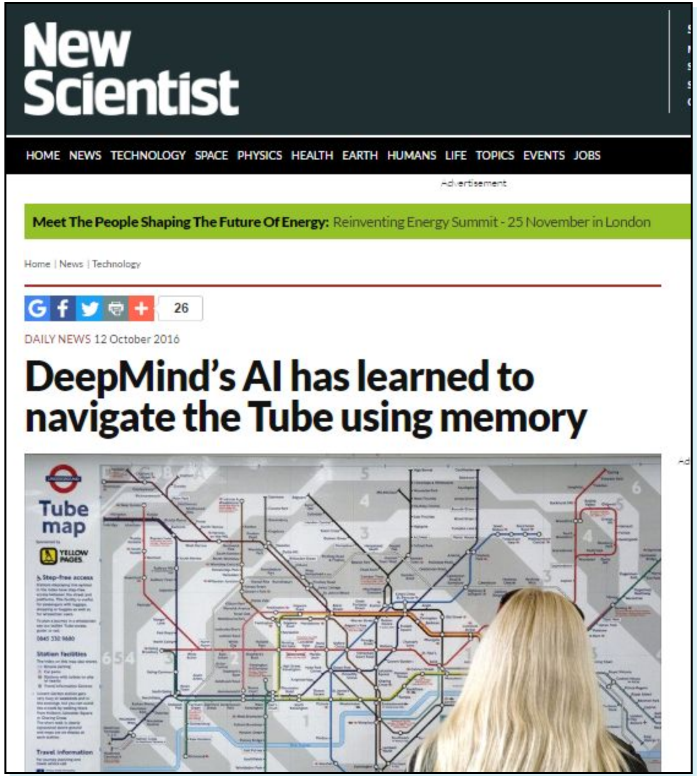


Rigid objects don’t overlap

At least one verb in each sentence.
If X and Y are married, then they are people.

[Lu, W. L., Ting, J. A., Little, J. J., & Murphy, K. P. (2013). Learning to track and identify players from broadcast sports videos.], [Wong, L. L., Kaelbling, L. P., & Lozano-Perez, T., Collision-free state estimation. ICRA 2012], [Chang, M., Ratinov, L., & Roth, D. (2008). Constraints as prior knowledge], [Ganchev, K., Gillenwater, J., & Taskar, B. (2010). Posterior regularization for structured latent variable models]… and many many more!

# Motivation: Deep Learning



New Scientist

HOME NEWS TECHNOLOGY SPACE PHYSICS HEALTH EARTH HUMANS LIFE TOPICS EVENTS JOBS



Home | News | Technology

26

DAILY NEWS 12 October 2016

DeepMind's AI has learned to navigate the Tube using memory



nature International weekly journal of science

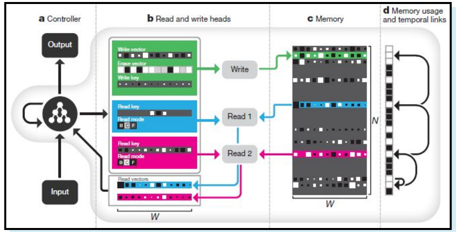
Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For

News & Comment > News > 2016 > November > Article

NATURE | NEWS

Google's AI reasons its way around the London Underground

DeepMind's latest technique uses external memory to solve tasks that require logic and reasoning — a step toward more human-like AI.



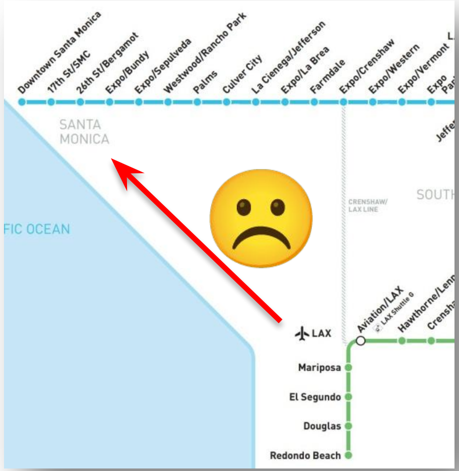


[Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., et al.. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471-476.]

# Motivation: Deep Learning

DeepMind's latest technique uses external memory to **solve tasks that require logic and reasoning — a step toward more human-like AI.**

## … but …




optimal planner recalculating a shortest path to the end node. To ensure that the network always moved to a valid node, the output distribution was renormalized over the set of possible triples outgoing from the current node. The performance

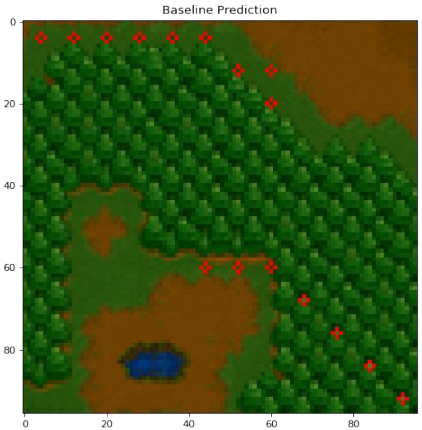
it also received input triples during the answer phase, indicating the actions chosen on the previous time-step. This makes the problem a 'structured prediction'

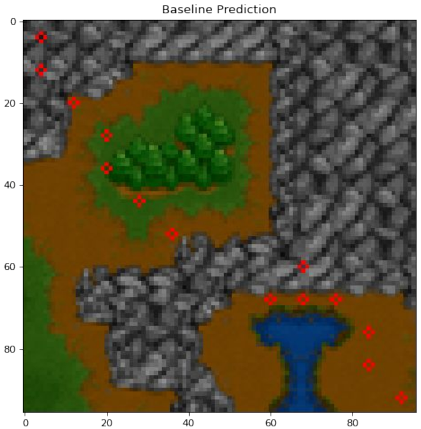

[Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., et al.. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471-476.]

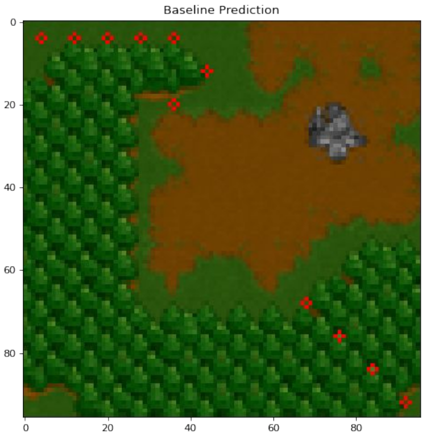
# Warcraft Shortest Path

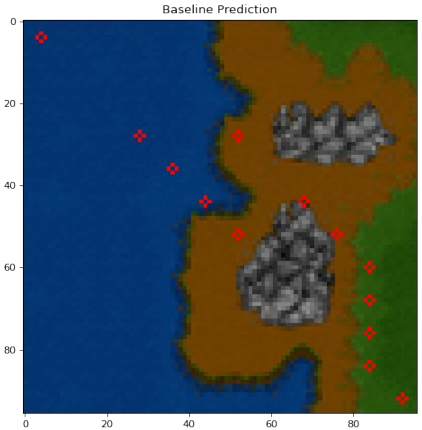
Predicting the minimum-cost path

[Vlastelica et al., 2019]

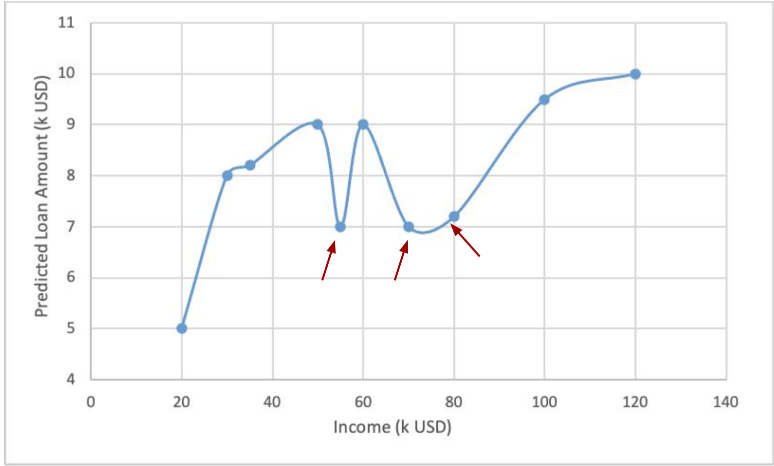

Baseline Prediction
0
20
40
60
80
0
20
40
60
80


Baseline Prediction
0
20
40
60
80
0
20
40
60
80


Baseline Prediction
0
20
40
60
80
0
20
40
60
80


Baseline Prediction
0
20
40
60
80
0
20
40
60
80

# Predict Loan Amount

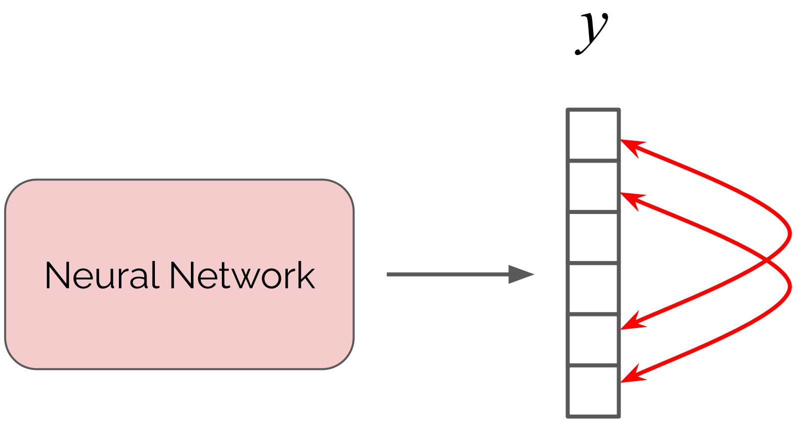




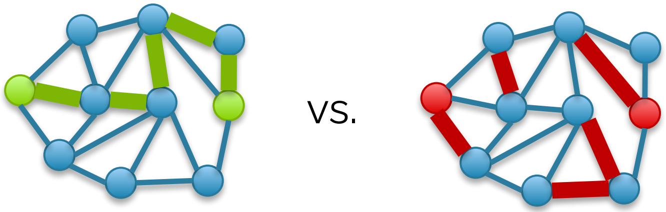
Neural Network Model: Increasing income can decrease the approved loan amount

Monotonicity (Prior Knowledge):
Increasing income should increase the approved loan amount

# Declarative Knowledge of the Output




How is the output structured?
Are all possible outputs valid?




How are the outputs related to each other?

Learning this from data is inefficient
Much easier to express this declaratively

How can do we inject declarative knowledge into PyTorch training code?

# pylon

Library that extends PyTorch to allow injection of declarative knowledge

- **Easy to Express Knowledge:** users write arbitrary constraints on the output
- **Integrates with PyTorch:** minimal change to existing code
- **Efficient Training:** compiles into loss that can be efficiently optimized
    - Exact semantic loss
    - Monte-carlo estimate of loss
    - T-norm approximation
    - *your solver?*

# pylon

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)
```

# pylon

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

2 Add as loss to training

```
loss += constraint_loss(check)
```

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)

    loss += constraint_loss(check)(py)
```

# pylon

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)

    loss += constraint_loss(check)(py)
```

1. Specify knowledge as a predicate

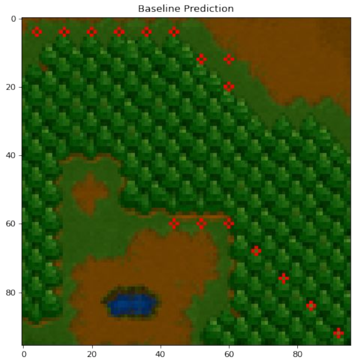
```
def check(y):
    ...
    return isValid
```

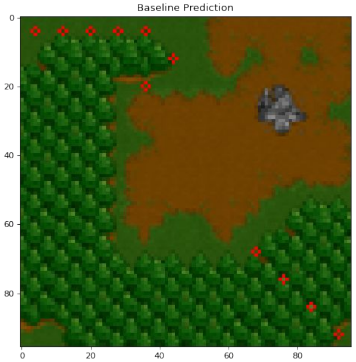
2. Add as loss to training

```
loss += constraint_loss(check)
```

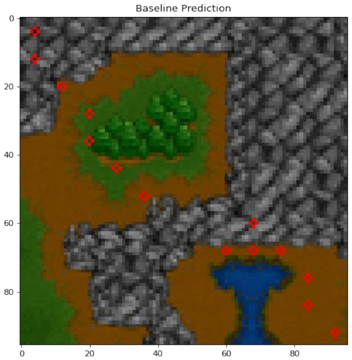
3. pylon derives the gradients
(solves a combinatorial problem)

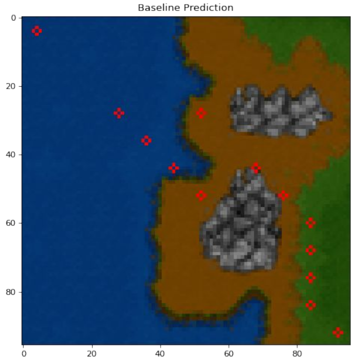
*without constraint*



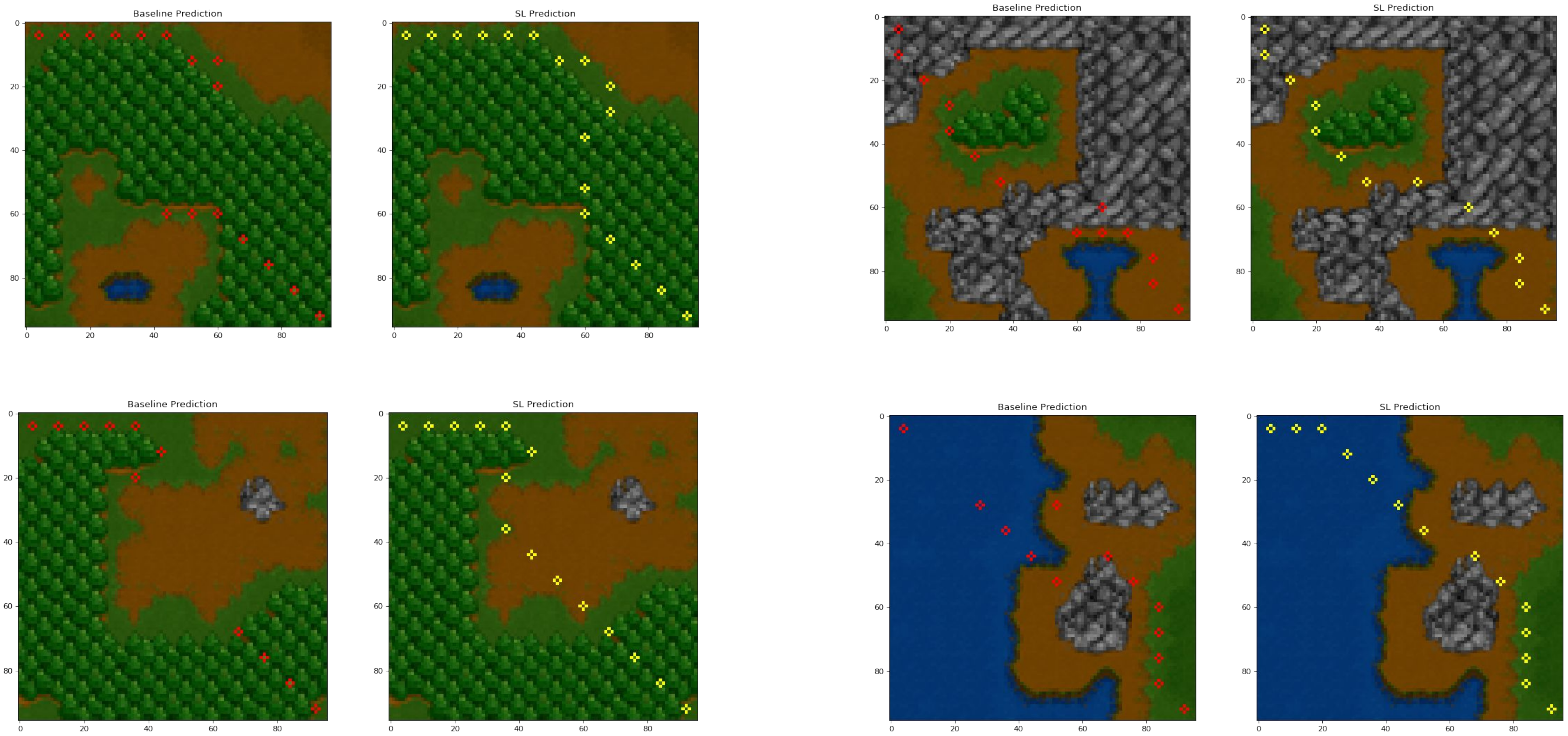



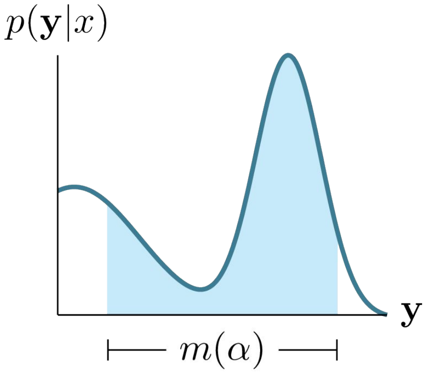

*without constraint*



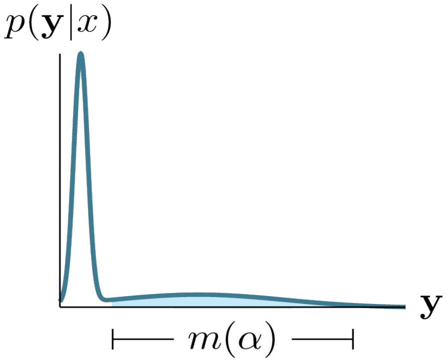



*without constraint* *with constraint* *without constraint* *with constraint*



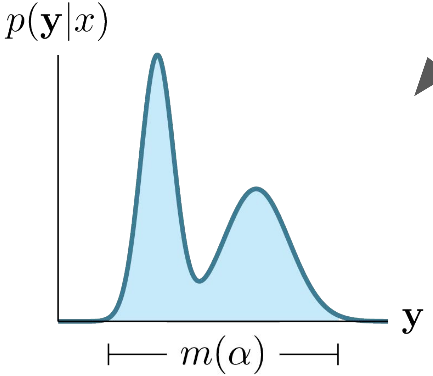



a) A network uncertain over both valid & invalid predictions




b) A network allocating most of its mass to an invalid prediction.

**Semantic Loss**




c) A network allocating most of its mass to models of the formula

$$P(\alpha|x) \uparrow : -\log P(\alpha|x) \downarrow$$

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, *ICML*, 2018.

# Warcraft min-cost simple-path prediction results

| Test accuracy % | Coherent | Incoherent | Constraint |
|---|---|---|---|
| ResNet-18 | 44.8 | **97.7** | 56.9 |

*Is prediction the shortest path?*
**This is the real task!**

*Are individual edge predictions correct?*

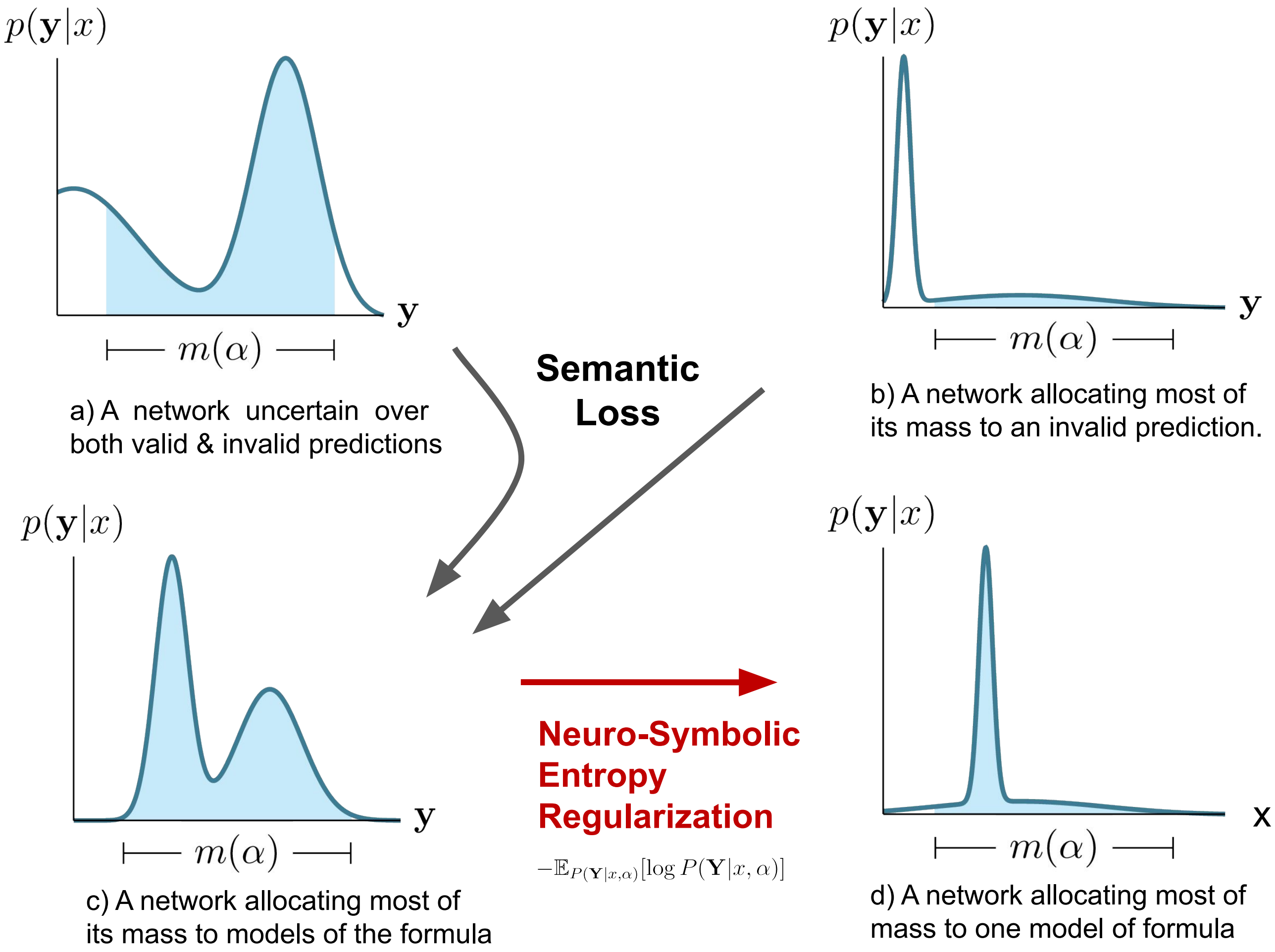
*Is output a path?*

Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2021.

## Warcraft min-cost simple-path prediction results

| Test accuracy % | Coherent | Incoherent | Constraint |
|---|---|---|---|
| ResNet-18 | 44.8 | **97.7** | 56.9 |
| + Semantic loss | **50.9** | **97.7** | **67.4** |

Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2021.




Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2021.

# Warcraft min-cost simple-path prediction results

| Test accuracy % | Coherent | Incoherent | Constraint |
|---|---|---|---|
| ResNet-18 | 44.8 | **97.7** | 56.9 |
| Semantic loss | **50.9** | **97.7** | **67.4** |
| + Entropy All | 51.5 | 97.6 | 67.7 |
| + Entropy Circuit | **55.0** | **97.9** | **69.8** |

Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2021.

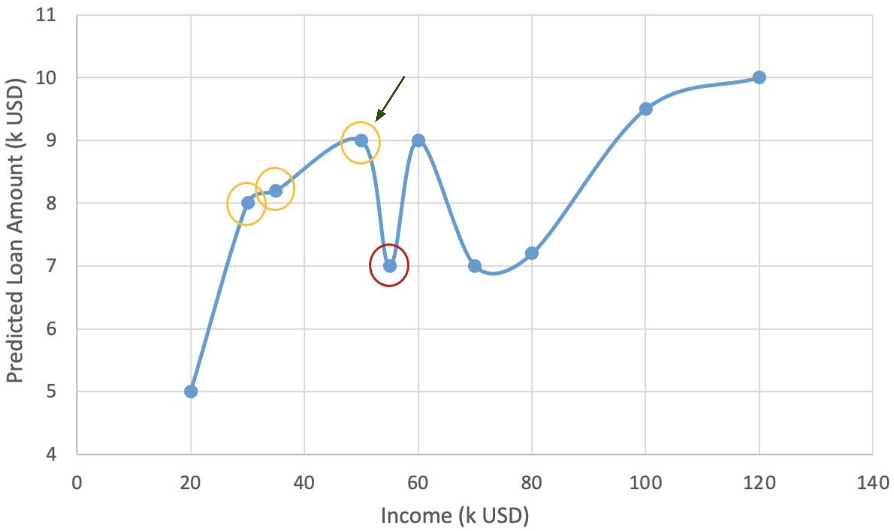
# Joint entity-relation extraction in natural language processing

| # Labels | | 3 | 5 | 10 | 15 | 25 | 50 | 75 |
|---|---|---|---|---|---|---|---|---|
| ACE05 | Baseline | $4.92 \pm 1.12$ | $7.24 \pm 1.75$ | $13.66 \pm 0.18$ | $15.07 \pm 1.79$ | $21.65 \pm 3.41$ | $28.96 \pm 0.98$ | $33.02 \pm 1.17$ |
| | Self-training | $7.72 \pm 1.21$ | $12.83 \pm 2.97$ | $16.22 \pm 3.08$ | $17.55 \pm 1.41$ | $27.00 \pm 3.66$ | $32.90 \pm 1.71$ | $37.15 \pm 1.42$ |
| | Product t-norm | $8.89 \pm 5.09$ | $14.52 \pm 2.13$ | $19.22 \pm 5.81$ | $21.80 \pm 7.67$ | $30.15 \pm 1.01$ | $34.12 \pm 2.75$ | $37.35 \pm 2.53$ |
| | Semantic Loss | $12.00 \pm 3.81$ | $14.92 \pm 3.14$ | $22.23 \pm 3.64$ | $27.35 \pm 3.10$ | $30.78 \pm 0.68$ | $36.76 \pm 1.40$ | $38.49 \pm 1.74$ |
| | + Entropy All | $\mathbf{14.80 \pm 3.70}$ | $15.78 \pm 1.90$ | $23.34 \pm 4.07$ | $28.09 \pm 1.46$ | $31.13 \pm 2.26$ | $36.05 \pm 1.00$ | $39.39 \pm 1.21$ |
| | + Entropy Circuit | $14.72 \pm 1.57$ | $\mathbf{18.38 \pm 2.50}$ | $\mathbf{26.41 \pm 0.49}$ | $\mathbf{31.17 \pm 1.68}$ | $\mathbf{35.85 \pm 0.75}$ | $\mathbf{37.62 \pm 2.17}$ | $\mathbf{41.28 \pm 0.46}$ |
| SciERC | Baseline | $2.71 \pm 1.1$ | $2.94 \pm 1.0$ | $3.49 \pm 1.8$ | $3.56 \pm 1.1$ | $8.83 \pm 1.0$ | $12.32 \pm 3.0$ | $12.49 \pm 2.6$ |
| | Self-training | $3.56 \pm 1.4$ | $3.04 \pm 0.9$ | $4.14 \pm 2.6$ | $3.73 \pm 1.1$ | $9.44 \pm 3.8$ | $14.82 \pm 1.2$ | $13.79 \pm 3.9$ |
| | Product t-norm | $\mathbf{6.50 \pm 2.0}$ | $8.86 \pm 1.2$ | $10.92 \pm 1.6$ | $13.38 \pm 0.7$ | $13.83 \pm 2.9$ | $19.20 \pm 1.7$ | $19.54 \pm 1.7$ |
| | Semantic Loss | $6.47 \pm 1.02$ | $\mathbf{9.31 \pm 0.76}$ | $11.50 \pm 1.53$ | $12.97 \pm 2.86$ | $14.07 \pm 2.33$ | $20.47 \pm 2.50$ | $23.72 \pm 0.38$ |
| | + Entropy All | $6.26 \pm 1.21$ | $8.49 \pm 0.85$ | $11.12 \pm 1.22$ | $14.10 \pm 2.79$ | $17.25 \pm 2.75$ | $\mathbf{22.42 \pm 0.43}$ | $24.37 \pm 1.62$ |
| | + Entropy Circuit | $6.19 \pm 2.40$ | $8.11 \pm 3.66$ | $\mathbf{13.17 \pm 1.08}$ | $\mathbf{15.47 \pm 2.19}$ | $\mathbf{17.45 \pm 1.52}$ | $22.14 \pm 1.46$ | $\mathbf{25.11 \pm 1.03}$ |

Table 5: Experimental results for joint entity-relation extraction on ACE05 and SciERC. #Labels indicates the number of labeled data points made available to the network per relation. The remaining training set is stripped of labels and is utilized in an unsupervised manner: enforce the constraint or minimize the entropy. We report averages and errors across 3 different runs.

Kareem Ahmed, Eric Wang, Kai-Wei Chang and Guy Van den Broeck. Neuro-Symbolic Entropy Regularization, 2021.

# Counterexamples



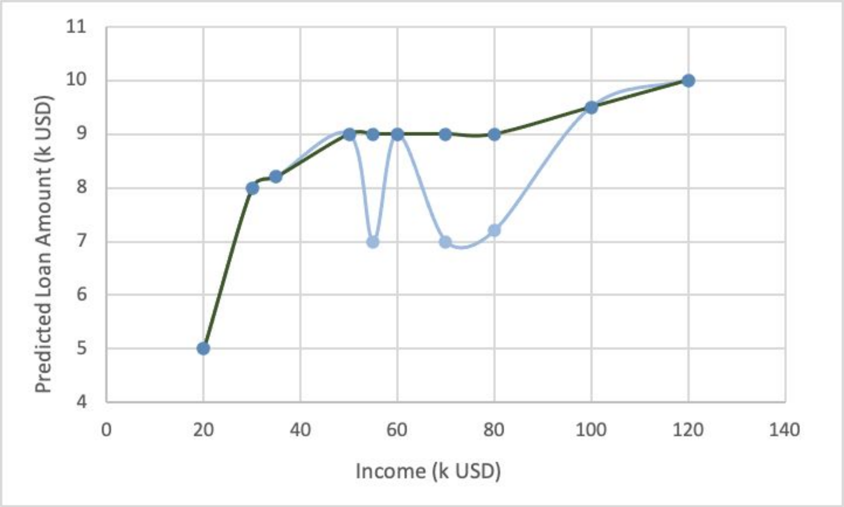

$$\exists x, y\ x \leq y \implies f(x) > f(y)$$

Computed using SMT(LRA) logical reasoning solver

Maximal counterexamples (largest violation) using OMT

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

# Counterexample-Guided Predictions




**Monotonic Envelope:**

- Replace each prediction by its maximal counterexample
- Envelope construction is online (during prediction)
- Guarantees monotonic predictions for any ReLU neural net

- Works for high-dimensional input
- Works for multiple monotonic features

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

# Monotonic Envelope: Performance

| Dataset | Feature | $NN_b$ | Envelope |
|---|---|---|---|
| Auto-MPG | Weight | 9.33±3.22 | **9.19±3.41** |
| | Displ. | 9.33±3.22 | 9.63±2.61 |
| | W,D | 9.33±3.22 | 9.63±2.61 |
| | W,D,HP | 9.33±3.22 | 9.63±2.61 |
| Boston | Rooms | 14.37±2.4 | **14.19±2.28** |
| | Crime | 14.37±2.4 | **14.02±2.17** |

| Dataset | Feature | $NN_b$ | Envelope |
|---|---|---|---|
| Heart | Trestbps | 0.85±0.04 | 0.85±0.04 |
| | Chol. | 0.85±0.04 | 0.85±0.05 |
| | T,C | 0.85±0.04 | 0.85±0.05 |
| Adult | Cap. Gain | 0.84 | 0.84 |
| | Hours | 0.84 | 0.84 |

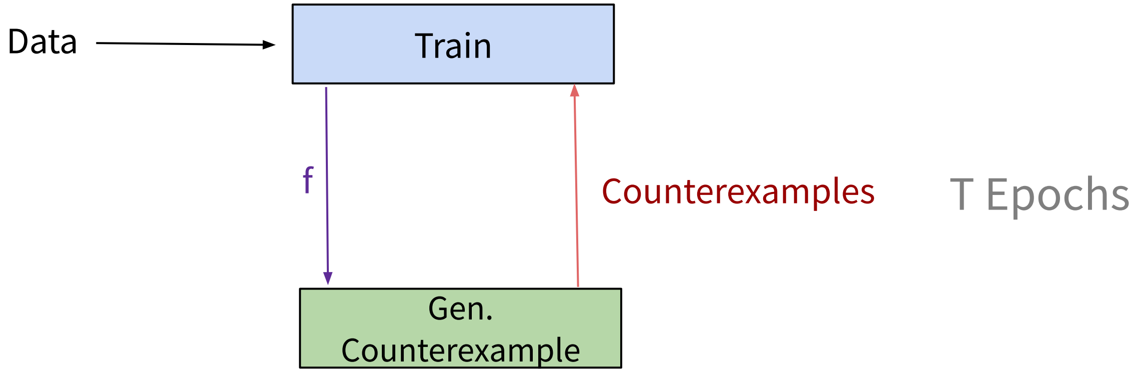
Guaranteed monotonicity at little to no cost

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

# Counterexample-Guided Learning

How to use monotonicity to improve model quality?

“Monotonicity as inductive bias”




Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

# Counterexample-Guided Learning: Performance

| Dataset | Feature | $NN_b$ | CGL |
|---|---|---|---|
| Auto-MPG | Weight | 9.33±3.22 | **9.04±2.76** |
| | Displ. | 9.33±3.22 | **9.08±2.87** |
| | W,D | 9.33±3.22 | **8.86±2.67** |
| | W,D,HP | 9.33±3.22 | **8.63±2.21** |
| Boston | Rooms | 14.37±2.4 | **12.24±2.87** |
| | Crime | 14.37±2.4 | **11.66±2.89** |

| Dataset | Feature | $NN_b$ | CGL |
|---|---|---|---|
| Heart | Trestbps | 0.85±0.04 | **0.86±0.02** |
| | Chol. | **0.85±0.04** | **0.85±0.05** |
| | T,C | 0.85±0.04 | **0.86±0.06** |
| Adult | Cap. Gain | **0.84** | **0.84** |
| | Hours | **0.84** | **0.84** |

Monotonicity is a *great* inductive bias for learning

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

# Counterexample-Guided Monotonicity Enforced Training (COMET)

Table 4: Monotonicity is an effective inductive bias. COMET outperforms Min-Max networks on all datasets. COMET outperforms DLN in regression datasets and achieves similar results in classification datasets.

| Dataset | Features | Min-Max | DLN | COMET |
|---|---|---|---|---|
| Auto-MPG | Weight | 9.91±1.20 | 16.77±2.57 | **8.92±2.93** |
| | Displ. | 11.78±2.20 | 16.67±2.25 | **9.11±2.25** |
| | W,D | 11.60±0.54 | 16.56±2.27 | **8.89±2.29** |
| | W,D,HP | 10.14±1.54 | 13.34±2.42 | **8.81±1.81** |
| Boston | Rooms | 30.88±13.78 | 15.93±1.40 | **11.54±2.55** |
| | Crime | 25.89±2.47 | 12.06±1.44 | **11.07±2.99** |

| Dataset | Features | Min-Max | DLN | COMET |
|---|---|---|---|---|
| Heart | Trestbps | 0.75±0.04 | 0.85±0.02 | **0.86±0.03** |
| | Chol. | 0.75±0.04 | 0.85±0.04 | **0.87±0.03** |
| | T,C | 0.75±0.04 | **0.86±0.02** | **0.86±0.03** |
| Adult | Cap. Gain | 0.77 | **0.84** | **0.84** |
| | Hours | 0.73 | **0.85** | 0.84 |

COMET = Provable Guarantees + SotA Results

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein and Guy Van den Broeck. Counterexample-Guided Learning of Monotonic Neural Networks, *NeurIPS*, 2020.

# The AI Dilemma


- Knowledge is (hidden) everywhere in ML
- A little bit of reasoning goes a long way!

*Today: Deep learning with constraints*
*Learning monotonic neural networks*

# Thanks

*This was the work of many wonderful students/postdoc/collaborators!*

References: http://starai.cs.ucla.edu/publications/