Empirical Study of Model Compression and Speed up for Vision Transformer

Feiyang Chen
fychen@cs.ucla.edu

Tianyi Xie
tianyixie77@g.ucla.edu

Yongqian Li
yongqianli@g.ucla.edu

Zhicheng Ren
franklinnwren@g.ucla.edu

Abstract

Vision transformers (ViT) have recently attracted considerable attention and achieved SOTA performance in many computer vision tasks. However, ViT models suffer from excessive computational and memory costs due to stacking multi-head self-attention modules and else. To allow its further application on resource-restricted and low-powered devices, model compression techniques are required to reduce the model size as well as speed up inference with acceptable precision loss. In this work, we study four main types of model compression methods for ViT, including quantization, low-rank approximation, knowledge distillation, and pruning, and make a comprehensive comparative analysis. Furthermore, we also explore combinations of different compression methods to verify whether better performance can be obtained. Extensive experimental results show our methods achieve a decent trade-off between accuracy and computational efficiency.

1 Introduction

Transformer [1] with self-attention has demonstrated high model capacity, easy scalability, and superior ability in capturing long-range dependency. They are first widely adopted in natural language processing (NLP) tasks, and recently received growing attention from the computer vision community as well. Prior works [2] in natural language processing (NLP) focus on addressing the quadratic complexity of the softmax-attention in transformer because inputs usually have a very long sequence in NLP tasks. Nevertheless, this is not the case in vision transformers, where the length of input sequence is fixed and usually not so large so that softmax-attention constitutes small fraction of the total FLOPs.

Vision Transformer (ViT) [3] is an architecture directly inherited from NLP, showing that embedding image patches into tokens and passing them through a sequence of transformer blocks can lead to comparable or even higher accuracy compared to previous CNN-based models. However, this great performance also comes with hundreds of millions of parameters, which makes ViT both memory and computation expensive during the inference and impedes its application on resource-restricted and low-powered devices. Therefore, the model compression technology of vision transformer models is urgently needed for deployment in industrial environments.

In contrast to previous popular deep learning models, model compression for vision transformers has not yet been thoroughly studied. Hence, in this paper, we aim to provide a systematic study of model compression for ViT, including quantization, low-rank approximation, knowledge distillation, and pruning. We make a comprehensive comparative analysis for different model compression techniques and also explore combinations of methods to verify whether better performance can be obtained. Extensive experimental results show our methods achieve a decent trade-off between accuracy and computational efficiency.

Preprint. Work in progress.

2 Related work

2.1 Quantization

Quantization is a widely adopted technique to enable efficient inference. By quantizing the network into low-bit representation, the inference of network requires less computation and less memory, while still achieves little performance degradation. A crucial point in this line of work is to determine the clipping range for the weights. [4] determines the clipping range by considering all of the weights in convolutional filters of a layer, while [5] quantizes Transformer using groupwise quantization. To remedy the accuracy loss induced by quantization, researchers also propose Quantization-Aware Training, in which the usual forward and forward pass are performed on the quantized model in floating point, but the model parameters are quantized after each gradient update.

2.2 Low-rank Approximation

The ViT utilizes the self-attention mechanism, and leads to quadratic complexity in the essence, and the attention matrices have low-rank properties[6]. Therefore, in order to reduce the complexity, an option is to use low-rank matrix approximation on the self-attention matrix. There are different method designs for this approximation: such as Nyström-based methods [7][8], Performer[9], and Linformer[10]. Some of them can be applied to pre-trained ViT models for fine-tuning and validation. It is also possible to unify low-rank approximation with sparse attention to get better approximations [6].

2.3 Knowledge Distillation

Knowledge distillation is a model compression method in which a small model (student model) leverages soft labels from a large model (teacher model) for transferring the knowledge [11]. The soft labels from the teacher are well known to be more informative than hard labels and lead to better student training ([12, 13]). [14] introduced a new distillation procedure based on a distillation token for vision transformer [3], which plays the same role as the class token, except that it aims at reproducing the label estimated by the teacher. Both tokens interact in the transformer through attention. This transformer-specific strategy outperforms vanilla distillation by a significant margin.

2.4 Pruning

Pruning is a popular way to reduce the dimensions of vision transformers [15]. The key idea behind pruning is to allocate an importance score for each dimension, and discard a great number of dimensions with small importance scores in order to achieve a high pruning ratio without significantly loss in accuracy. Meanwhile, dimensional redistribution can be applied together with the pruning process to achieve better performance [16]. It is also notable that the model after effective pruning might achieve better performance than the original model [17].

3 Methodology

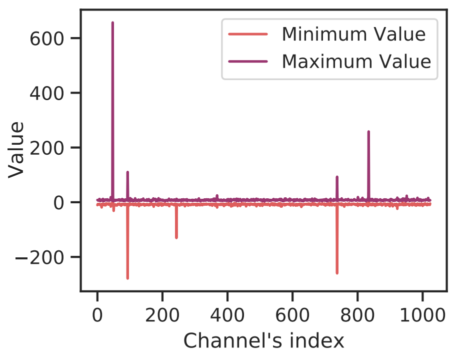
3.1 Quantization

3.1.1 Basic Concept

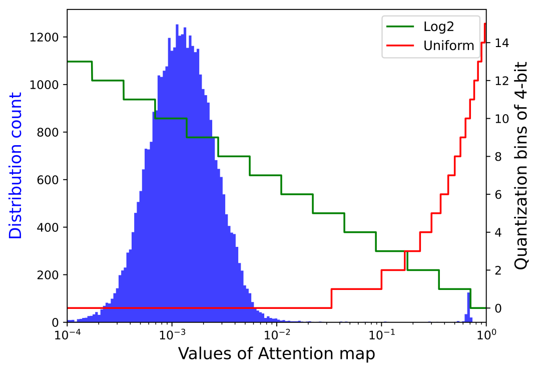
Generally, the goal of quantization is to reduce the precision of both the parameters (θ), as well as the intermediate activation maps to low-precision (e.g. 8 bit integer), with minimal impact on the generalizaiton performance of the model. To do this, we need to first define a function that can quantize weights and activations to a finite set of values. A popular choice for a quantization function is as follows:

$$Q(r) = \text{Int}(r/S) - Z, \tag{1}$$

where Q is the quantization mapping functions, r is a real valued input (i.e weights, activation), S is a real valued scaling factor, Z is an integer zero point. In essence, this function maps a real value r to some integer value. This method is known as *uniform quantization*, since the resulting values are uniformly spaced. Certainly there are other *non-uniform quantization* methods. Note it is also possible to the recover the real value r from its quantized value $Q(r)$ through *dequantization*:



(a) Data Distribution in LayerNorm



(b) Data Distribution in Attention Maps

Figure 1: **Left [19]**: Channel-wise minimum and maximum values of the last LayerNorm inputs in full precision Swin-B. **Right [19]**: Distribution of the attention map values from the first layer of ViT-L, and visualizing the quantized bins using uniform or Log2 quantization with 4-bit.

$$\tilde{r} = S(Q(r) + Z) \tag{2}$$

The the value of recovered $\tilde{r}$ may not be exactly the same as r due to the rounding error in quantization.

The most important thing in quantization is to determine the value of scaling factor S. This scaling factor essentially divides real values r into a number partitions:

$$S = \frac{\beta - \alpha}{2^b - 1} \tag{3}$$

where $[\alpha, \beta]$ denotes the clipping range and b is the quantization bit width. To determine the value of S, the clipping range $[\alpha, \beta]$ needs to be first defined and the process of choosing the clipping range is often referred to as *calibration*. One straightforward approach is to use the min/max of the input for this clipping range, i.e, $\alpha = r_{\text{min}}$ and $\beta = r_{\text{max}}$, which is an *asymmetric quantization* scheme since $-\alpha \neq \beta$. It is also possible to use a *symmetric quantization* scheme, e.g. $-\alpha = \beta = \max(|r_{\text{max}}|, |r_{\text{min}}|)$. In this case, the quantization function in Eq. 1 can be simplified by setting $Z = 0$.

3.1.2 Post Training Quantization

Post Training Quantization (PTQ) performs the quantization and the adjustments of weights without any fine-tuning. As such, the overhead is very low and often negligible. However this advantage sometimes come at the cost of much lower accuracy due to the precision loss in quantization. [18] found that there would be significant accuracy degradation when quantizing LayerNorm and Softmax of Transformer-based architectures. [19] pointed that this is due to the polarized data distribution of activations in the LayerNorm layers as well as values of attention maps. Specifically, as shown in the left side of Figure 1, serious inter-channel variation can be observed in the inputs of LayerNorm layers, as a result of which layer-wise quantization leads to unaccepetable quantization errors. Furthermore, we can observe a distribution centering at a fairly small value at the right side of Figure 1, while only few outliers have larger values close to 1. In this case, uniform quantization would also results in severe performance drop. To tackle these issues, [19] proposed Powers-of-Two Scale for LayerNorm quantization and Log-Int-Softmax for Softmax quantization.

3.1.3 Quantization Aware Training

Directly applying quantization to a trained model may introduce perturbation to the model parameters, resulting significant performance degradation. This can be addressed by re-training the model with quantized parameters so that model can converge to a point with better loss. Quantization Aware Training (QAT) is one popular approach to do this. Specifically, we perform the usually forward and backward procedure on the quantized model in floating float, but quantize the parameter again after each gradient update. LSQ [20] builds upon existing methods for learning weights in quantized networks by improving how the quantizer itself is configured and achieved state-of-the-art quantization performance. DIFFQ [21] is a differentiable method for model compression for quantizing model parameters without gradient approximations (e.g., Straight Through Estimator (STE)). It uses only pseudo quantization noise to approximate quantization at train time, as a differentiable alternative to STE, both with respect to the unquantized weights and number of bits used.

3.2 Knowledge Distillation

The classical distillation procedure includes soft distillation and hard distillation. Soft distillation [13] minimizes the Kullback-Leibler divergence between the softmax of the teacher and the softmax of the student model. The distillation objective is:

$$\mathcal{L}_{\text{global}} = (1-\lambda)\mathcal{L}_{\mathrm{CE}}\left(\psi\left(Z_{\mathrm{s}}\right), y\right) + \lambda\tau^{2}\mathrm{KL}\left(\psi\left(Z_{\mathrm{s}}/\tau\right), \psi\left(Z_{\mathrm{t}}/\tau\right)\right) \tag{4}$$

where Z_t is the logits of the teacher model, Z_s is the logits of the student model, τ is the temperature for the distillation, λ is the coefficient balancing the Kullback–Leibler divergence loss (KL) and the cross-entropy (L_{CE}) on ground truth labels y, and ψ is the softmax function.

Hard distillation takes the hard decision of the teacher as a true label. Let $y_{\mathrm{t}} = \operatorname{argmax}_c Z_{\mathrm{t}}(c)$ be the hard decision of the teacher, the objective associated with this hard-label distillation is:

$$\mathcal{L}_{\text{global}}^{\text{hardDistill}} = \frac{1}{2}\mathcal{L}_{\mathrm{CE}}\left(\psi\left(Z_s\right), y\right) + \frac{1}{2}\mathcal{L}_{\mathrm{CE}}\left(\psi\left(Z_s\right), y_{\mathrm{t}}\right) \tag{5}$$

For a given image, the hard label associated with the teacher may change depending on the specific data augmentation, which makes hard distillation become a better choice than soft distillation while being parameter-free and conceptually simpler: The teacher prediction y_t plays the same role as the true label y. DeiT [14] is a new distillation procedure based on hard distillation specific to transformers through adding a new token (distillation token) to the initial embeddings (patches and class token). It interacts with other embeddings through self-attention and is output by the network after the last layer. This distillation token is employed in a similar fashion as the class token, except that on the output of the network its objective is to reproduce the (hard) label predicted by the teacher, instead of the true label, which show its superiority. In our experiments, we directly fine-tune DeiT on the cifar dataset as the result of ViT compression due to the limitation of computational resources.

3.3 Pruning

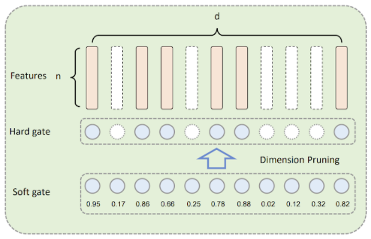
Most existing work on pruning focuses on reducing the number of parameters by reducing the dimensions of weight kernel between each of the hidden layers. In other words:

$$\begin{gathered} \min \alpha, \beta \\ \text{s.t. } \textstyle\sum_k loss(l_\beta^{(k)} W_{\alpha,\beta}^{(k)} l_\alpha^{(k+1)}) - loss(l_b^{(k)} W_{a,b}^{(k)} l_a^{(k+1)}) < \delta \end{gathered} \tag{6}$$

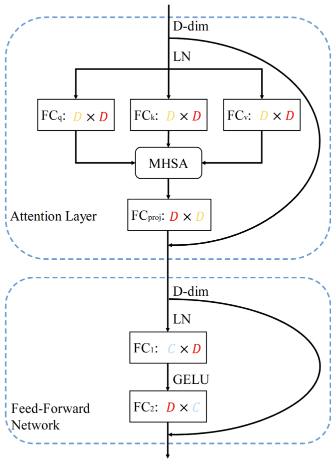
where a, b are the original number of dimensions of $W^{(k)}$ and α, β are the new number of dimensions of $W^{(k)}$. We want the increase of loss brought by parameter reduction to be below a certain predefined threshold δ so that the ViT model will retain its integrity in downstream tasks.

To decide whether a certain dimension will be pruned in the accelerated model, the concept of importance score are introduced. Importance scores are learned during the pre-training or fine-tuning stage. It takes several forms. Zhu el. al. [15] and Yang et. al [16] both adopt an importance score that are directly proportional to the gradient of the weight in a given dimension. Hence, the importance score could be learned by directly adding one more "soft gate" layer after the layer to be pruned. At the inference stage, the "soft gate" become "hard gate" and assign value zero the the pruned dimensions (Figure 2 (a)). More precisely, for a given block B in a weight layer W, the importance score takes the simple form:

$$s_B(W) = (\sum_{b \in B} \frac{\partial s}{\partial w_b} w_b)^2 \tag{7}$$



(a) Soft-gate based model proposed by Zhu et. al. [15]



(b) Skip-dimension model proposed by Yu et. al. [17]

Figure 2: Different pruning models

Yu et. al. [17] propose another importance score which takes consideration of KL divergence. The key concept behind this importance score is to evaluate the KL divergence in terms of accuracy with or without certain modules on a given data set Ω. Compare to the trivial importance score, this importance score allows both dimension-pruning within layers and layer-pruning within modules. To learn this importance score, a skip-dimension model is proposed, and the KL divergence is designed to be a loss directly added to the original loss function of the ViT model (Figure 2 (b)). More precisely:

$$s_B(W) = \sum_{i \in \Omega} D_{KL}(p_i || q_i) \tag{8}$$

where $q_i \in loss(W)$ and $p_i \in loss(W \setminus \{w_b\})$.

Nowadays, more sophisticated importance scores are proposed. Tang et. al. proposed an importance score that could quantify the impact any patch has to the final error, with theoretical guarantee [22]. This importance score allows more efficient patch slimming. Rao et. al. proposed a combination of both local and global features to obtain a local-global embedding and feed them to another MLP, in order to predict the probabilities to drop/keep the tokens [23]. Yi et. al. choose a variety of importance scores and unified them in a single loss function [24].

3.4 Low-rank Approximation

Generally, in each self-attention block in ViT, given the input sequence X and the weights W_Q, W_K, W_V, originally the first step is to project X using the weights to acquire the feature representations:

$$Q = W_Q X,\ K = W_K X,\ V = W_V X$$

Then the self-attention matrix will be $softmax(QK^T/\sqrt{d_q})V$. This process results in $O(n)$ complexities in both time and space, where n is the length of input sequences [8].

It is formally proved that Self-Attention is low rank[10]. Therefore, in order to reduce the computation time, one choice is to use the low-rank property of the attention matrix to approximate the result [6].

With proper method design and some constraints, both time and space complexities can be approximately $O(n)$ without major loss of accuracy [8] [7] [6].

The low-rank approximation can be applied not only when the model is trained from scratch, but also when there is a pre-trained model. When it is trained from scratch such as in [7], the low-rank approximation of attention matrices will reduce the training time significantly; for pre-trained models, since the weights W_Q, W_K, W_V of each self-attention block are given, the improvement can also be shown by the reduced calculation time during fine-tuning or validation. Normally compression methods based on low-rank approximation can improve computation time and reduce memory usage but cannot reduce the model size, since the weights W_Q, W_K, W_V still need to be saved and the approximation can only be performed after the input is given.

There exists different ways for low-rank approximation. Both Nyströmformer[8] and SOFT[7] are Nyström-based methods: they use the Nyström approximation method for the softmax matrix in self-attention. This approximation enables linearizing self-attention. There are also other low-rank approximation ways to linearize it, such as Linformer [10] and Performer [9]. Additionally, unifying the low-rank and sparse attention, such as in [6], will yield better approximation accuracy.

In our experiments, we will focus on Nyströmformer-based ViT, which is constructed by replacing the default calculation of softmax attention matrix by Nyström approximation method in all self-attention blocks. This construction enables loading the pre-trained weights of Vanilla ViT, making them comparable with other models in the experiments. Nyströmformer algorithm utilizes landmark (or Nyström) points to reconstruct an efficient Nyström approximation without calculating the full QK^T [8], and we want the number m of landmarks much smaller than n. In the experiments, we test m values of 24, 32, and 64.

4 Experiments

In this section, we comprehensively compare various model compression methods for Vision Transformers, including quantization, knowledge distillation, pruning, and low-rank approximation. Furthermore, we also explore combinations of different compression methods to verify whether better performance can be obtained.

4.1 Experiment Setting

Our experiments are based on Google Colab Pro+ powered by Tesla V100-SXM2 16GB GPU and code implementation is based on PyTorch [1]. Datasets only include CIFAR-10 and CIFAR-100 due to the limitation of computing resources. We focus on model size and inference speed for different model compression methods. In general, there is a trade-off between accuracy and model size/inference speed. An ideal algorithm is expected to have a small accuracy drop and significant model size reduction/speed up. The comparison results on CIFAR-10 and CIFAR-100 is presented on Table 1 and Table 2.

4.2 Comparison of Different Model Compression Methods

For **Model Size**, methods based on quantization and pruning can significantly reduce model size without losing too much accuracy, especially quantization-related methods achieve better results. Specifically, Dynamic Quantization [2] reduces the model size to 25% of the original model with minor accuracy loss by converting 32-bit floating point to 8-bit. Noted that FQ-ViT [19] is a method for solving the challenges in quantizing the LayerNorm and SoftMax, and successfully quantizes the whole model with acceptable performance drop. However, their current code implementation is a 'fake quantization' (doesn't really convert to 8-bit representation) and therefore it's meaningless to evaluate the speed as well as the model size. That's why we didn't record the relevant model size and inference speed results in our tables. DIFFQ achieves a more significant mode size reduction but also loses more accuracy. DIFFQ with LSQ has achieved an accuracy of 93.37%, which is almost 12% of the size of the original model. A more extreme example: our DIFFQ-based method retraining ViT (without pretrained model) achieves 60.29% accuracy with only 2MB model.

Compared with quantization, weight pruning with simple form importance score does not achieve a desirable model size vs accuracy trade off. When pruning rate is equal to 0.1 (10% of the model-parameters are pruned), the accuracy on CIFAR-10 dropped by ten percent comparing with vanilla ViT, and the accuracy on CIFAR-100 dropped by almost twenty percent. We did some further study to investigate the reason behind the result drop. From figure 3, we could observe that after training with soft gates, majority of the model parameters are still above the 0.99, means that the model

[1]Note currently Pytorch only supports quantization on CPU thus we also test the inference speed on CPU for some methods.

[2]We implement Dynamic Quantization directly through the PyTorch Quantization API: `https://pytorch.org/tutorials/advanced/dynamic_quantization_tutorial.html`

Model	Method	Accuracy	GPU Speed	CPU Speed	Size(MB)
Vanilla ViT [3]	-	98.94	4.48	0.050	327
Dynamic Quantization	Quantization (PTQ)	98.73	-	0.062	84
FQ-ViT [19]	Quantization (PTQ)	97.31	-	-	-
DIFFQ with LSQ [20]	Quantization (QAT)	93.37	2.10	-	41
DIFFQ with diffq [21]	Quantization (QAT)	60.29	12.20	-	2
DeiT base [14]	Knowledge Distillation	98.47	7.04	0.096	327
DeiT tiny [14]	Knowledge Distillation	95.43	16.78	-	21
ViT-Pruning(r=0.1) [15]	Pruning	88.36	4.86	-	301
ViT-Pruning(r=0.2) [15]	Pruning	80.56	5.54	-	254
ViT-Nyströmformer(m=24) [8]	Low-rank Approximation	65.91	4.67	-	327
ViT-Nyströmformer(m=32) [8]	Low-rank Approximation	75.94	4.57	-	327
ViT-Nyströmformer(m=64) [8]	Low-rank Approximation	91.70	4.38	-	327
DeiT base + Dynamic Quantization	Knowledge Distillation + PTQ	96.75	-	0.117	84

Table 1: Evaluation results on CIFAR-10. The Speed values are iterations per second.

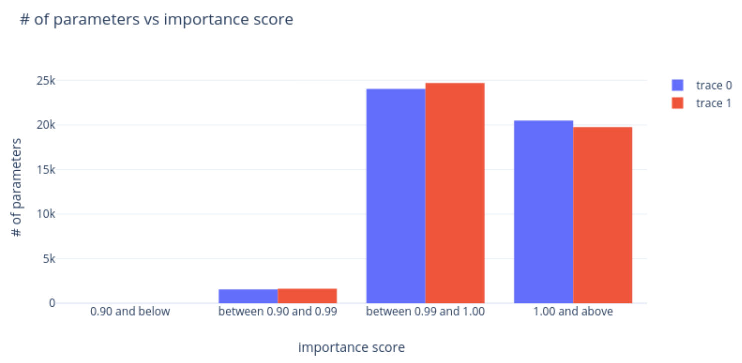


Figure 3: Number of parameters vs importance score. **Blue**: CIFAR-10. **Red**: CIFAR-100.

believes that they are of great importance. Hence, pruning those parameters will inevitable reduce much model integrity. This suggests that weight pruning with simple form importance score has some limitations. Potential means of improvements includes introduce more complicated important scores [17], or to implement input patch reduction or slimming instead of model weight pruning [23] [22].

For **Inference Speed**, most model compression methods achieve a range of speedups, especially knowledge distillation-based methods achieve better results. Specifically, although the model size of the Deit base is not significantly reduced, the inference speed is almost 2x that of the Vanilla ViT, while maintaining almost no loss in accuracy. Interestingly, on the CIFAR-10 dataset, Deit tiny achieves 95.43% accuracy with a 4x speedup compared to the Vanilla ViT but only at 6% the size of the original model. Besides, for Nyströmformer-based ViT, larger choices of number m of landmarks leads to more accurate approximations with a trade-off of speed. Dynamic Quantization also brings 10-20% inference speed improvement on CPU.

4.3 Exploration of Combinatorial Methods

Through the analysis of the above results, in the case of a minor loss of accuracy, combining the advantages of quantization in model size and knowledge distillation in inference speed could be a promising exploration. As is shown in Table 1 and Table 2, the mixed approach (DeiT base +

Model	Method	Accuracy	GPU Speed	CPU Speed	Size(MB)
Vanilla ViT [3]	-	92.87	4.34	0.093	327
Dynamic Quantization	Quantization (PTQ)	90.87	-	0.122	84
FQ-ViT [19]	Quantization (PTQ)	84.87	-	-	-
DIFFQ with LSQ [20]	Quantization (QAT)	76.08	2.10	-	41
DIFFQ with diffq [21]	Quantization (QAT)	41.02	12.00	-	2
DeiT base [14]	Knowledge Distillation	87.35	6.97	0.149	327
DeiT tiny [14]	Knowledge Distillation	75.90	16.16	-	21
ViT-Pruning(r=0.1) [15]	Pruning	74.46	4.69	-	302
ViT-Pruning(r=0.2) [15]	Pruning	64.27	5.19	-	272
ViT-Nyströmformer(m=24) [8]	Low-rank Approximation	38.51	4.77	-	327
ViT-Nyströmformer(m=32) [8]	Low-rank Approximation	50.31	4.65	-	327
ViT-Nyströmformer(m=64) [8]	Low-rank Approximation	74.01	4.46	-	327
DeiT base + Dynamic Quantization	Knowledge Distillation + PTQ	82.61	-	0.196	84

Table 2: Evaluation results on CIFAR-100. The Speed values are iterations per second.

Dynamic Quantization) enables over one times faster inference speed and reduces the model size to a quarter at a acceptable cost of accuracy loss.

5 Conclusion

In this work, we focus on an empirical study of model compression and speed up for Vision Transformer. We first investigate the four mainstream model compression methods, including quantization, low-rank approximation, knowledge distillation, and pruning, and do a literature review of recent pioneering work on compressing ViTs. To verify the efficiency of these different algorithms, we conduct a comparison study on both CIFAR-10 and CIFAR-100 datasets. According to the results of the experiment, we find post-training quantization and knowledge distillation are two promising methods to compress the vision transformer, which enables great model size reduction or much faster inference with an acceptable performance drop. On top of this, we also step further to see whether a combination of these two methods would allow us to achieve a better trade-off. We find this mixed approach works properly, especially on CIFAT-10, and yields over one times faster inference speed with only a quarter of the original model size. Such a multi-dimensional model compression approach, we believe, would be one promising research direction for Vision Transformer in the future.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[2] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[4] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[5] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In

Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8815–8821, 2020.

[6] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention approximation, 2021.

[7] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing Xu, Tao Xiang, and Li Zhang. Soft: Softmax-free transformer with linear complexity, 2021.

[8] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nystöm-based algorithm for approximating self-attention. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 35, page 14138. NIH Public Access, 2021.

[9] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[10] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[12] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[13] Longhui Wei, An Xiao, Lingxi Xie, Xiaopeng Zhang, Xin Chen, and Qi Tian. Circumventing outliers of autoaugment with knowledge distillation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 608–625. Springer, 2020.

[14] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[15] Mingjian Zhu, Kai Han, Yehui Tang, and Yunhe Wang. Visual transformer pruning. *arXiv preprint arXiv:2104.08500*, 2021.

[16] Huanrui Yang, Hongxu Yin, Pavlo Molchanov, Hai Li, and Jan Kautz. Nvit: Vision transformer compression and parameter redistribution. *arXiv preprint arXiv:2110.04869*, 2021.

[17] Hao Yu and Jianxin Wu. A unified pruning framework for vision transformers. *arXiv preprint arXiv:2111.15127*, 2021.

[18] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34, 2021.

[19] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Fully quantized vision transformer without retraining. *arXiv preprint arXiv:2111.13824*, 2021.

[20] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

[21] Alexandre Défossez, Yossi Adi, and Gabriel Synnaeve. Differentiable model compression via pseudo quantization noise. *arXiv preprint arXiv:2104.09987*, 2021.

[22] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. *ArXiv*, abs/2106.02852, 2021.

[23] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34, 2021.

[24] Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. In *International Conference on Learning Representations*, 2022.