ISBN-978-989-9121-05-8
PORTO 20th International Conference on "Innovations in Engineering & Sciences"
(PICIES-22)
Porto (Portugal) May 31-June 2, 2022

Searching Intrinsic Dimensions of Vision Transformers

Fanghui Xue[1], Biao Yang[1], Yingyong Qi[1], and Jack Xin[1]

[1]University of California, Irvine, CA 92697, USA

***Abstract*:** *It has been shown by many researchers that transformers perform as well as convolutional neural networks in many computer vision tasks. Meanwhile, the large computational costs of its attention module hinder further studies and applications on edge devices. Some pruning methods have been developed to construct efficient vision transformers, but most of them have considered image classification tasks only. Inspired by these results, we propose SiDT, a method for pruning vision transformer backbones on more complicated vision tasks like object detection, based on the search of transformer dimensions. Experiments on CIFAR-100 and COCO datasets show that the backbones with 20% or 40% dimensions/parameters pruned can have similar or even better performance than the unpruned models. Moreover, we have also provided the complexity analysis and comparisons with the previous pruning methods.*

Keywords: *deep neural networks, vision transformers, pruning*

1. Introduction

Unlike the convolutional or recurrent neural networks (CNN or RNN) [3], transformers are the models based completely or partially on the attention mechanisms. They are originally proposed to learn global dependency for sequence transduction tasks [10] and have obtained better performance and training efficiency. Besides its success in language models, transformers have also been widely studied in computer vision tasks. One of the directions is to replace the CNN backbones by transformers. In other words, transformers are used to extract features from images, and the features are processed by various heads to solve various tasks after that. Among these transformer-based models, ViT [2], DeiT [9] and Swin Transformer [8] have achieved high performance in multiple tasks like image classification, object detection, segmentation, etc.

The general architecture of the transformer for sequence modeling is composed of an encoder module and a subsequent decoder module. The encoder module is a stack of a few sequential encoder blocks, with each of them containing a self-attention (SA) layer and a fully connected feed-forward network, with a residual structure [4], and a layernorm applied after the summation of the shortcut and the residual. While the feed-forward network consists simply of two fully connected layers, the self-attention layer is computed through the multi-head self-attention (MSA) mechanism [10], which is more complicated and usually requires more computational resources than the convolution operations used in CNNs. Therefore, pruning methods [16, 12, 13] have been proposed to construct efficient vision transformers. However, most of them only consider pruning DeiT on the image classification task. In this paper, we present a pruning method for transformer backbones which is valid on both image classification and object detection tasks. Since our method aims to search for the intrinsic dimensions (i.e., the possible lowest dimensions to maintain network performance) of transformers, we name it SiDT in the rest of the paper. Although SiDT is inspired by previous pruning methods like Network Slimming [7] and Vision Transformer Pruning (VTP) [16], it has its own merits:

- SiDT can prune transformers for not only classification tasks, but also other vision tasks like object detection.

- We have analyzed the computational complexity of the unpruned and the pruned models.
- The models with 20% or 40% dimensions pruned perform similarly or even better than the unpruned model.
- SiDT prunes the dimensions of linear embeddings, different from the feature pruning of VTP.

2. Related Work

2.1. Vision Transformers

Vision Transformer (ViT) [2] is among the vision models whose backbones are purely transformers. ViT has partitioned the input image into small patches to mimic the tokens in the language transformers. Instead of pixels, these patches are embedded into features of certain dimensions, serving as the input of the attention module. Since its job is to learn representations, ViT has included the encoder module only, i.e., a stack of multi-head self-attentions. Despite ViT's high accuracy on image classification, there are some concerns about its quadratic computational complexity on the number of queries n. That means the complexity is also quadratic on the input resolution $H \times W$, whereas the convolution operation has linear complexity. ViT has also been restricted to image classification since pixel-level tasks like segmentation typically need to deal with high resolution features.

A window-based transformer called Swin Transformer [8] has then been proposed for these more complicated vision tasks. Like ViT, Swin has also provided a series of backbones which are based purely on transformers, especially the transformer encoders. The first advantage of Swin is that it can generate hierarchical features so that they can be used to solve semantic segmentation and object detection tasks with suitable heads. To obtain features of different resolutions, Swin has merged $2 \times 2 = 4$ image patches into 1 patch at the end of each architecture stage. Since the size of patches is fixed, the image height and the width are both reduced by a half after merging. The overall transformer architecture is divided into one initial stage without merging and three intermediate stages with merging, and hence it can produce features of four resolution levels. Another advantage comes from the window-based multi-head self-attention (W-MSA) with shifting. Compared with the quadratic complexity of MSA, W-MSA has achieved a linear complexity from computing the attentions locally, within a small window of patches. Global information across different windows is then exchanged via shifting the window partitions.

2.2. Dimension Pruning

The dimension/channel pruning problem of CNNs can be solved by adding group sparsity to the convolutional weights [14, 11] or formulated as a neural architecture search problem [17, 6]. Among them, a method called Network Slimming (NetSlim) has been proposed based on learning the channel scaling factors [7], which is able to reduce the model complexity and computational cost, and preserve the accuracy at the same time. These channel scaling factors are simply defined to be the learnable scale parameters of the batch normalization layer, and the channels corresponding to low scales are pruned. To learn sparse scales, the ℓ_1 regularization of these scale parameters is added to the loss during training. After being trained with ℓ_1 sparsity and the channels with low scales pruned, the model is further fine-tuned to achieve better performance. We shall be aware that the regularization term is not added to the convolutional weights, but directly to the scale parameters, which play a similar role as the architecture parameters in the differentiable neural architecture search context [6]. That is why searching for dimensions is indeed dimension pruning.

Similar to the channel pruning in CNNs, there are also some studies for vision transformer pruning. Inspired by NetSlim, VTP [16] has assigned scoring parameters to the features before the linear embedding or projection layers and pruned the dimensions of these features which are corresponding to low scores. Since the dimensions of the linear layers depend on the dimensions of the input features, the parameters of these layers are also reduced. Another pruning method has been proposed in NViT [12], which is based on the scores of grouped structural parameters. The scores are different from those of VTP as they are computed directly from the weight parameters. NViT has taken pruning the number of heads and the latency on hardware into account. Moreover, it

has been pointed out that having the same dimensions across all layers in the conventional transformer design might not be optimal [12], which inspires the studies of automated transformer architecture design.

These pruning methods have obtained high pruning ratio with a very small accuracy loss for vision transformers like DeiT [9], on the image classification tasks. It would be natural to consider pruning Swin or other light transformer backbones for multiple computer vision tasks. WDPruning [13] is a direct pruning method for Swin on ImageNet classification, without the fine-tuning stage. It has also provided an option for depth pruning, and an automated learned pruning ratio based on learnable thresholds of saliency scores. However, experimental results have shown worse accuracy of the pruned models, as it has not been fine-tuned. Inspired by these previous studies, we consider pruning Swin backbone as dimension search in this paper. Before we specify the details of each stage, we summarize a general framework for searching the dimensions of operations [7, 16] (see also Fig. 1(a)):

- Specify the architecture parameters for representing the dimensions of the operations.
- Set up a loss function which involves the architecture parameters and the other learnable parameters.
- Optimize the loss via gradient descent and prune the network based on the values of the architecture parameters.
- Fine-tune the pruned network.

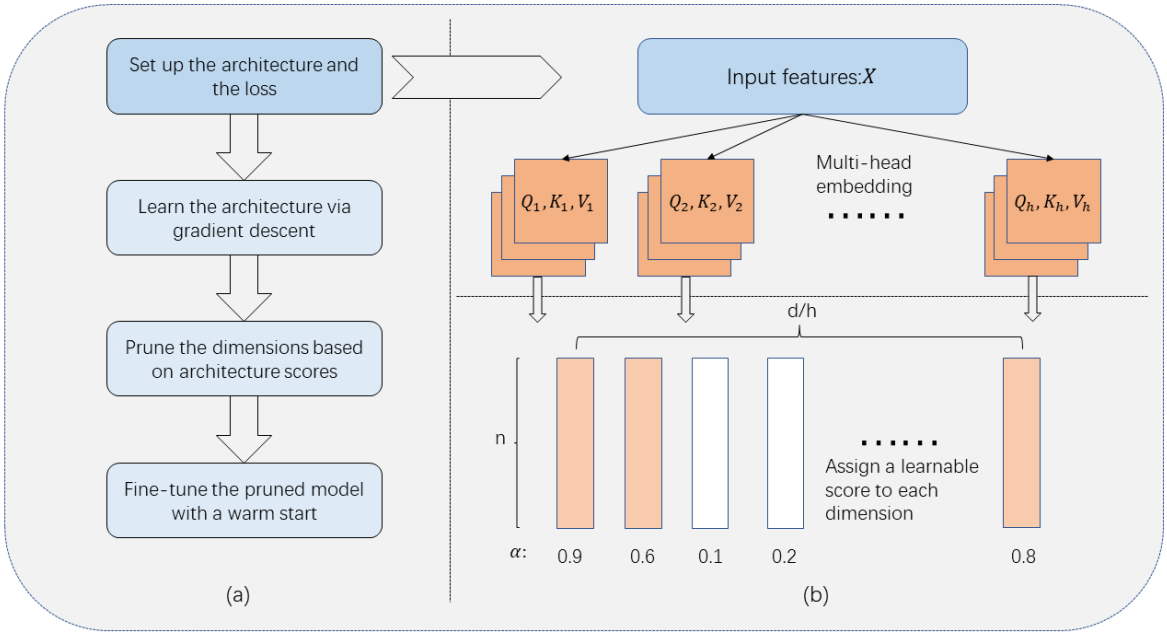

Fig. 1: (a) The stages of transformer pruning. (b) Assign the scoring matrix $A = diag(\alpha)$ to the output dimensions of multi-head queries, keys, and values.

3. Method

Architecture parameters. For the dimension search of transformers, we still follow the four stages summarized in Section 2.2. Since the searching, pruning and fine-tuning stages are similar, the key difference is how we set up the architecture parameters. Whereas we prune convolution operations in CNNs, there are a few types of operations for different transformers. So, we discuss in detail the strategies of setting up architectures parameters for MSA, W-MSA and multilayer perceptron (MLP) [8]. Suppose again the batch size is 1, $X \in R^{d \times H \times W}$ is the input feature map with H and W the resolution and d the dimension of the feature. Set $n = H \times W$, we obtain the transformed input feature $X \in R^{n \times d}$.

For SA [10], X is linearly embedded into the query Q, key K and value V of the same shapes:

$$Q = XW_Q, K = XW_K, V = XW_V,$$

where the embedding matrices $W_Q, W_K, W_V \in R^{d\times d}$, if the embedding dimensions for the query, key and value are equal to d. Then the attention map a is computed via the softmax function σ of the scaled product of the query and the key:

$$a(Q,K) = \sigma\left(QK^T/\sqrt{d}\right) \in R^{n\times n}$$

and assigned to the value to compute the output of SA:

$$SA(Q,K,V) = \sigma\left(QK^T/\sqrt{d}\right)V \in R^{n\times d}.$$

Note that the output of SA has the same shape as the input X. To set up the architecture parameters, we apply a uniform score matrix A for Q, $K$and V via matrix multiplication:

$$\tilde{Q} = QA, \tilde{K} = KA, \tilde{V} = VA,$$

where $A \in R^{d\times d}$ is a diagonal matrix, whose diagonal elements are the architecture parameters α_i for $i = 1, 2, \dots, d$. In other words, we assign a score α_i to the i-th dimension of the d-dimensional query, and to the key and value at the same i-th dimension. Then we compute the SA module based on the scored query, key, and value, and obtain $SA(\tilde{Q}, \tilde{K}, \tilde{V})$.

For MSA [10], we need to compute multiple SA modules and each of them is a head. Let h be the number of heads. For $k = 1, 2, \dots, h$, we also compute Q_k, K_k and $V_k \in R^{n\times d/h}$ through linear embedding of X via $W_{Q,k}$, $W_{K,k}$ and $W_{V,k} \in R^{d\times d/h}$ like that of SA, and obtain the heads:

$$H_k = SA(Q_k, K_k, V_k) \in R^{n\times d/h}.$$

With Q, K and V the concatenations of Q_k, K_k, and V_k, the output of the MSA module is computed by concatenating the heads and projecting linearly via $W_O \in R^{d\times d}$:

$$MSA(Q,K,V) = [H_1, H_2, \dots, H_h]W_O \in R^{d\times d}.$$

We use a stronger *scoring matrix* $A \in R^{d/h\times d/h}$ for MSA, which is not only uniform over the query, key and value, but also over all the heads:

$$\tilde{Q}_k = Q_k A, \tilde{K}_k = K_k A, \tilde{V}_k = V_k A,$$

for $k = 1, 2, \dots, h$. Then we compute the new MSA module and obtain $\tilde{H}_k = SA\left(\tilde{Q}_k, \tilde{K}_k, \tilde{V}_k\right) \in R^{n\times d/h}$ and:

$$MSA\left(\tilde{Q}, \tilde{K}, \tilde{V}\right) = \left[\tilde{H}_1, \tilde{H}_2, \dots, \tilde{H}_h\right]W_O.$$

For W-MSA [8], the input features $X \in R^{n\times d}$ are divided into a few windows of size $M \times M$, and MSA is computed locally within these windows. That is to say, we reshape X to be a tensor in $R^{n/M^2\times M^2\times d}$, and obtain Q_k, K_k, and $V_k \in R^{n/M^2\times M^2\times d/h}$ for $k = 1, 2, \dots, h$ after embedding of multi-head. Here Q_k, K_k, and V_k can be viewed as the concatenations of $Q_{k,l}$, $K_{k,l}$, and $V_{k,l} \in R^{M^2\times d/h}$ for $l = 1, 2, \dots, n/M^2$. For each window, we compute the MSA module and obtain $W_{,l} = MSA\left(Q_{,l}, K_{,l}, V_{,l}\right) \in R^{M^2\times d}$. Finally, we rearrange the outputs of these windows and obtain:

$$W - MSA(Q,K,V) = [W_{,1}, W_{,2}, \dots, W_{,n/M^2}] \in R^{n\times d}.$$

To set up the architecture parameters for W-MSA, again we use a uniform scoring matrix $A \in R^{d/h\times d/h}$ for the query, key and value, over all the heads and windows:

$$\tilde{Q}_{k,l} = Q_{k,l}A, \tilde{K}_{k,l} = K_{k,l}A, \tilde{V}_{k,l} = V_{k,l}A,$$

Then we have $\widetilde{W}_{,l} = MSA(\tilde{Q}_{,l}, \tilde{K}_{,l}, \tilde{V}_{,l})$ and

$$W - MSA\left(\tilde{Q}, \tilde{K}, \tilde{V}\right) = [\widetilde{W}_{,1}, \widetilde{W}_{,2}, \dots, \widetilde{W}_{,n/M^2}].$$

The last module to be discussed is MLP [8], which simply contains two linear layers with activation. Suppose $X \in R^{n\times d}$ is the input feature, and d_m represents the dimensions of the hidden state. Suppose further $W_1 \in R^{d\times d_m}$ and $W_2 \in R^{d_m\times d}$ are two matrices for linear embedding, g_{MLP} is the activation. Then we have:

$$MLP(X) = g_{MLP}(XW_1)W_2 \in R^{n\times d}.$$

The scoring matrix A is applied immediately after W_1 through matrix multiplication, and get $g_{MLP}(XW_1A)W_2$. Here A can be viewed as the scores for the dimensions of the hidden state.

Pruning. The pruning procedure is summarized in Fig. 1. During the searching stage, the elements in the scoring matrix A are regularized by ℓ_1 norm like NetSlim [7], and involved in the overall loss:

$$L = l(X, T; W) + \gamma l_1(A),$$

where l is the classification or detection loss, l_1 is the ℓ_1 loss, X, T and A are the input, target and the architecture parameters, and W represents the other learnable parameters. γ is a scale hyperparameter to be set up in the section of experiments. The architecture parameters A are updated via gradient descent or architecture search algorithms [6], together with the elements of the embedding matrices W. After the completion of searching, we rank the diagonal elements of the scoring matrix A according to their absolute values. The dimensions of the embedding matrices are pruned if their corresponding scores are ranked low. Suppose the remaining ratio of the dimensions after pruning is ρ. Then only ρd dimensions with higher scores are left in the pruned matrices.

For MSA, we have $W_{Q,k}$, $W_{K,k}$ and $W_{V,k} \in R^{d \times \rho d/h}$ after pruning, and hence Q_k, K_k, and $V_k \in R^{n \times \rho d/h}$. Since we have not pruned the query or key number n, the attention map still belongs to $R^{n \times n}$, and the head $H_k \in R^{n \times \rho d/h}$. This leads to the projection matrix $W_O \in R^{\rho d \times d}$, and the output of the pruned MSA in $R^{n \times d}$, with the same shape as the unpruned model. One can easily see that the original unpruned MSA module has $O(4d^2)$ parameters and a computational complexity of $O(4nd^2 + 2n^2d)$. For the pruned MSA, the number of parameters is reduced to$O(4\rho d^2)$, and the computational complexity is reduced to $O(4\rho nd^2 + 2\rho n^2 d)$. Similarly, the unpruned W-MSA module has $O(4d^2)$ parameters and a computational complexity of $O(4nd^2 + 2nM^2d)$. For the pruned W-MSA, the number of parameters is reduced to $O(4\rho d^2)$, and the computational complexity is reduced to $O(4\rho nd^2 + 2\rho nM^2 d)$. Finally, the unpruned MLP has $O(2dd_m)$ parameters and a computational complexity of $O(2ndd_m)$. For the pruned MLP, the number of parameters is reduced to $O(2\rho dd_m)$, and the computational complexity is reduced to $O(2\rho ndd_m)$. This is because $W_1 \in R^{d \times \rho d_m}$ and $W_2 \in R^{\rho d_m \times d}$ after pruning.

One shall note that our settings of architecture parameters are different from those of VTP [16]. VTP's scoring matrix A is applied directly to the input feature X, whereas ours is applied to Q, K and V. In other words, VTP prunes the features but we prune the linear embeddings. As we apply the same matrix A to the embedding dimensions of multiple heads, we have only d/h such architecture parameters, making the model easier to train. Moreover, VTP is applied to DeiT on the classification task only, whereas our method prunes Swin Transformer, which serves as a backbone for multiple vision tasks. Finally, we have also provided the complexity analysis of the unpruned and pruned operations, which is missing in previous studies.

4. Experiments

We conduct SiDT for Swin Transformer on CIFAR-100 classification [18]. We prune its tiny version (Swin-T), which has 27.53M parameters and a complexity of 4.49G FLOPS. The settings of the search stage are similar to those for training the unpruned baseline[1], with batch size = 256, patch size = 4, window size = 7, embedding dimension = 96, initial learning rate = 0.00025, momentum = 0.9, weight decay = 0.05, epochs = 160, and the sparsity scale $\gamma = 0.0001$ for ℓ_1 regularization. After searching, we obtain the scores of all the dimensions and rank them according to their absolute values. Next, the dimensions with lower scores are pruned, based on predefined pruning ratios of 20%, 40%, 60% and 80%. Finally, the pruned model is trained again with a warm start, using the same settings as the search stage. Table I shows that the number of parameters and computational costs can be greatly reduced after pruning, while preserving the accuracy at the same time, compared to the baseline [15]. After pruning 80% of the dimensions, the accuracy is only around 2% lower than the recovered baseline. The model with 20% or 40% dimensions pruned has an accuracy which is even higher

[1] When setting up the architecture parameters, we refer to the code at https://github.com/Cydia2018/ViT-cifar10-pruning

than the baseline model. This can be explained by the relatively larger size of Swin-T on easier datasets like CIFAR, as over-parameterized models can cause overfitting.

Additionally, we have also pruned the Swin-T backbone for the COCO object detection task [5], following the settings in the Swin paper [8]. That is, batch size = 16, initial learning rate = 0.0001, weight decay = 0.05, epochs = 36, and all the other settings of the backbone are the same as the Swin-T for CIFAR classification discussed above. We use Cascade Mask R-CNN [1] as the detection head, in accordance with that of the Swin-T baseline. Again, we follow the steps in Fig. 1, and prune the model with pruning ratios of 20% and 40%. During the search stage, we also start training with a pretrained Swin-T object detection model. Table II indicates that the model with 20% dimensions of the backbone pruned has a similar performance of box mAP and mask mAP as the unpruned model. Here mAP means the mean average precision over all categories. The box or mask indicates that mAP is computed over bounding boxes or masks. Even if 40% dimensions of the backbone are pruned, the loss in mAP is still less than 1.5%. This is a fair result since the detection task is more complicated than the classification task, and pruning a detection model can lead to a slightly larger accuracy decline.

TABLE I: Prune Swin-T via SiDT on CIFAR-100 classification task. PR = Pruning Ratio. Acc = accuracy. Para. = number of parameters.

PR	Acc (%)	Para. (M)	FLOPS (G)
0% (Baseline [15])	78.07	-	-
0% (Baseline[2])	81.78	27.60	4.49
20% SiDT	**82.75**	23.28	3.53
40% SiDT	82.11	17.89	2.60
60% SiDT	80.81	11.92	1.73
80% SiDT	79.35	**7.17**	**0.92**

TABLE II: Prune Swin-T backbone via SiDT on COCO object detection task. PR = Pruning Ratio.

PR	mAP		Para. (M)	
	Box	Mask	Total	Backbone
0% (Baseline [8])	**50.5**	**43.7**	86	28
20% SiDT	50.4	**43.7**	80	22
40% SiDT	49.2	42.9	**74**	**16**

5. Conclusion

We have developed SiDT, a method for searching for the intrinsic dimensions of transformers, and provided its complexity analysis. Experiments on multiple vision tasks have shown that SiDT can promote the efficiency of vision transformers with little accuracy loss. This method will be applied to more computer vision tasks in future work.

6. Acknowledgements

The work was partially supported by NSF grants DMS-1854434, DMS-1952644, and a Qualcomm Faculty Award. The authors would like to thank Dr. Shuai Zhang and Dr. Jiancheng Lyu for helpful discussions.

References

[1] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154-6162.

https://doi.org/10.1109/CVPR.2018.00644

[2] The baseline result is recovered on our machine of single RTX 3090 GPU.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*, 2020.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

https://doi.org/10.1109/CVPR.2016.90

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014, pp. 740-755.

https://doi.org/10.1007/978-3-319-10602-1_48

[6] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search," in *ICLR 2019*, 2019.

[7] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. of the IEEE International Conference on Computer Vision*, 2017, pp. 2736-2744.

https://doi.org/10.1109/ICCV.2017.298

[8] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *in Proc. of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10012-10022.

https://doi.org/10.1109/ICCV48922.2021.00986

[9] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, 2021, pp. 10347-10357.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, vol. 30, 2017.

[11] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems*, vol. 29, 2016.

[12] H. Yang, H. Yin, P. Molchanov, H. Li, and J. Kautz, "Nvit: Vision transformer compression and parameter redistribution", arXiv preprint arXiv:2110.04869.

[13] F. Yu, K. Huang, M. Wang, Y. Cheng, W. Chu, and L. Cui, "Width & Depth Pruning for Vision Transformers," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[14] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68(1), pp. 49-67, 2006.

https://doi.org/10.1111/j.1467-9868.2005.00532.x

[15] Z. Zhang, H. Zhang, L. Zhao, T. Chen, S. Arik, and T. Pfister, "Nested Hierarchical Transformer: Towards Accurate, Data-Efficient and Interpretable Visual Understanding," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[16] M. Zhu, Y. Tang, and K. Han, "Vision Transformer Pruning," arXiv preprint arXiv:2104.08500.

[17] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697-8710.

https://doi.org/10.1109/CVPR.2018.00907

[18] A. Krizhevsky, G. Hinton, et al, "Learning multiple layers of features from tiny images," Technical report, Citeseer, 2009.