Vision Transformer Pruning Via Matrix Decomposition

Tianyi Sun
Committee of Computational and Applied Mathematics, University of Chicago
tianyisun@uchicago.edu

ABSTRACT

This is a further development of Vision Transformer Pruning [33] via matrix decomposition. The purpose of the Vision Transformer Pruning is to prune the dimension of the linear projection of the dataset by learning their associated importance score in order to reduce the storage, run-time memory, and computational demands. In this paper we further reduce dimension and complexity of the linear projection by implementing and comparing several matrix decomposition methods while preserving the generated important features. We end up selected the Singular Value Decomposition as the method to achieve our goal by comparing the original accuracy scores in the original Github repository and the accuracy scores of using those matrix decomposition methods, including Singular Value Decomposition, four versions of QR Decomposition, and LU factorization.

KEYWORDS

Computer Vision, Transformer Pruning, Matrix Computation

1 INTRODUCTION

Transformer [26] is widely used in Natural Language Processing tasks and now it has attracted much attention on computer vision applications [3], [4], [10], and [18], such as image classification [7], [11], and [25], object detection [2], and [34], and image segmentation [13], [27], and [29]. However, most of the transformer used in the set of computer vision tasks are highly storage, run-time memory, and computational resource demanding. Convolutional neural networks (CNNs) are also sharing the same issues. Developed ways of addressing this issues in CNNs' settings are including low-rank decomposition [6], [15], [16], [32], and [22], quantization [1], [9], [20], and [31], and network pruning [12], [5], [23], [24], and [30].

Inspired by those methods, several methods has been developed in the setting of vision transformer. Star-Transformer [8] changes fully-connected structure to the star-shaped topology. There are some methods focus on compressing and accelerating the transformer for the natural language processing tasks. With the emergence of vision transformers such as ViT [7], PVT [28], and TNT [11] an efficient transformer is urgently need for computer vision applications. Vision Transformer Pruning (VTP) [33], inspired by the pruning shceme in network slimming [17], has proposes an approach of pruning the vision transformer according to the learnable important scores. We add the learning importance scores before the components to be prune and sparsify them by training the network with L_1 regulation. The dimension with smaller importance scores will be pruned and the compact network can be obtained. The (VTP) method largely compresses and accelerates the original ViT models.

Final Project of TTIC-31230: Fundamental of Deep Learning , Pytorch, 2021.

Inspired by the low-rank decomposition [6], [15], [16], [32], and [22] methods for compressing and accelerating CNNs, we proposed a matrix decomposition method building up on the VTP to further reduce the storage, run-time memory, and computational resource demanding.

2 METHOD

In this section, we first review the key components of Vision Transformer (ViT) [7] and Vision Transformer Pruning according to important features [33]. Then we introduce matrix decomposition methods that is fit particularly for our case.

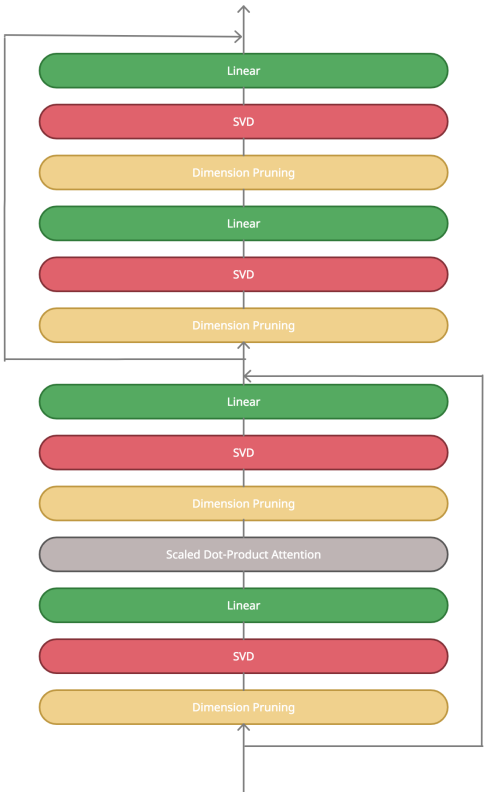

Figure 1: Further prune Vision Transformer via Singular Value Decomposition (SVD)

2.1 Vision Transformer (ViT)

The ViT model includes fully connected layer, Multi-Head Self-Attention (MHSA), linear transformer, Multi-Layer Perception (MLP), normalization layer, activation function, and shortcut connection.

- *Fully connected layer*: A fully connected layer is used to transform the input into characters that can be used in the next step, i.e. MHSA. Specifically, it can be denoted as:

$$I \rightarrow (Q_I, K_I, V_I).$$

The *I*nput $I \in \mathbb{R}^{n\times d}$ is transformed to *Q*uery $Q \in \mathbb{R}^{n\times d}$, and *V*alue $V \in \mathbb{R}^{n\times d}$ via fully-connected layers, where n is the number of patches d in the embedded dimension.

- *MHSA*: The self-attention mechanism is utilized to model the relationship between patches. It can be denoted as:

$$Attention(Q_I, K_I, V_I) = Softmax\left(\frac{Q_I K_I^T}{\sqrt{d}}\right)V.$$

- *Linear Transformer*: A linear transformer is used to transform the output of MHSA. It can be denoted as:

$$Y = I + f_{out}(Attention(Q_I, K_I, V_I)),$$

where the details of normalization layer and activation function are included in it.

- *Two-layer MLP*: Two feed forward MLP layers can be denoted as:

$$Z = Y + MLP_2(MLP_1(Y)).$$

2.2 Vision Transformer Pruning

The approach of Vision Transformer Pruning is to preserve the generated important features and remove the useless ones from the components of linear projection.

The pruning process is depending on the score of feature importance. We denote the matrix of feature importance scores is $\text{Diag}(a) \in \mathbb{R}^{n\times n}$, where $a \in \mathbb{R}^d$ represents each feature importance score. In order to enforce the sparsity of importance scores, we apply l_1 regularization on the importance scores a, which is denoted as $\hat{a}$, where $\hat{a} = \lambda||a||_1$ and λ is the sparsity hyper-parameter. And optimize it by adding the training objective. We, therefore, obtain the transformer with some important scores near zero. We rank all the values of regularized importance scores in the transformer and obtain a threshold τ according to the predefined pruning rate. With the threshold, we obtain the discrete $a^* \in \mathbb{R}^{n\times n}$ score, which is denoted as

$$a^* = \begin{cases} 1 & \text{if } \hat{a} \geq \tau \\ 0 & \text{if } \hat{a} < \tau. \end{cases}$$

The matrix X after applying the pruning processes, denoted by

$$X^* = X\ \text{Diag}(a^*),$$

and removing features with a zero score is transformed to $X^* \in \mathbb{R}^{r\times d}$. This pruning procedure is denoted as $\text{Prune}(X)$.

2.3 Matrix Decomposition

We will introduce an experiential study of Matrix Decomposition methods particularly applied in our case. And a comparison of run-time complexity and the costs of those decomposition.

2.3.1 Eigenvalue Value Decomposition (EVD) of X^*. Note that a matrix has an EVD if and only if the matrix is diagonalizable. However, $X^* \in \mathbb{R}^{r\times d}$. When $r \neq d$, the X^* has no EVD. So EVD has limitations while applying in our cases.

2.3.2 Singular Value Decomposition (SVD) of X^*. SVD is computed not depending on the dimension of the matrix. SVD can be applied to any matrix. Assume that $X^* \in \mathbb{R}^{m\times n}$ is the matrix after we have done the pruning procedure. Then X^* has a decomposition that

$$X^* = U\Sigma, V$$

where the left singular vectors $U \in \mathbb{R}^{m\times m}$ and right singular vectors $V \in \mathbb{R}^{n\times n}$ are both unitary matrices, which means $U^*U = I_m = UU^*$ and $V^*V = I_n = VV^*$. $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix of X^* in the sense that

$$\sigma_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ \mathbb{R}_+ & \text{if } i = j \end{cases},$$

where the diagonal values are the set of singular values of the matrix X^*. This is the full SVD. We can also remove those singular values that are equal to zero from the matrix, which means $\sigma_{ii} = 0$ for $i \in \max\{m, n\}$. Then we get the form of X^* that

$$X^* = U_r\Sigma_r V_r,$$

where U_r, Σ_r, and $V_r \in \mathbb{R}^{n\times n}$. This is the condensed SVD. r is the rank of matrix X^*. The columns of U_r forms an orthonormal basis for the image of X^*. The columns of V_r forms an orthonormal basis for the coimage of X^*.

2.3.3 QR Decomposition of X^*. The QR decomposition method can solve many problems on the SVD list. The QR decomposition is cheaper to compute in the sense that there are direct algorithms for computing QR and complete orthogonal decomposition in a finite number of arithmetic steps. There are several versions of QR decomposition.

- *Full QR Decomposition of X^**: For any matrix of $X^* \in \mathbb{R}^{m\times n}$, where $n \neq m$, there exits an unitary matrix $Q \in \mathbb{R}^{m\times m}$, $Q^*Q = QQ^* = I_n$ and an upper-triangular matrix $R \in \mathbb{R}^{m\times n}$, where $r_{ij} = 0$ whenever $i > j$, such that

$$X^* = QR = Q\begin{bmatrix} R_1 \\ 0 \end{bmatrix}.$$

In which, $R_1 \in \mathbb{R}^{n\times n}$ is an upper-triangular square matrix in general. If X^* has full column rank, which is $rank(X^*) = n$ then R_1 is non-singular.

- *Reduced QR Decomposition of X^**: For any matrix of $X^* \in \mathbb{R}^{m\times n}$, where $n \neq m$, there exits an orthonormal matrix $Q \in \mathbb{R}^{m\times n}$, $Q_1^*Q_1 = I_n$ but $Q_1Q_1^* \neq I_m$ unless $m = n$ and an upper-triangular square matrix $R \in \mathbb{R}^{n\times n}$, such that

$$X^* = Q_1R_1.$$

In which, Q_1 is the first n columns of Q in previous Full QR decomposition. $Q = [Q_1, Q_2]$, where $Q_2 \in \mathbb{R}^{m\times(m-n)}$ is the last $m - n$ columns of Q.
In fact, we can obtain reduced QR decomposition of X^* from the full QR decomposition of X^* by simply multiplying out

$$X^* = QR = [Q_1, Q_2]\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1R_1 + Q_20 = Q_1R_1.$$

If X^* has full column rank, which means $rank(A) = n$, then R_1 is non-singular.

Merits	Evaluation
Accuracy	Normal Equation < QR Decomposition < SVD
Speed	Normal Equation > QR Decomposition > SVD

Table 1: A comparison of the relative merits among Normal Equations, QR Decomposition, and SVD.

- *Rank-Retaining QR Decomposition of X^**: For matrix of $X^* \in \mathbb{R}^{m\times n}$, where $rank(X^*) = r$, there exists a permutation matrix $\Pi \in \mathbb{R}^{n\times n}$, a unitary matrix $Q \in \mathbb{R}^{m\times m}$, and a non-singular, upper-triangular square matrix $R_1 \in \mathbb{R}^{r\times r}$, such that
$$X^*\Pi = Q\begin{bmatrix} R_1 & S \\ 0 & 0 \end{bmatrix}.$$
In which, $S \in \mathbb{R}^{r\times(n-r)}$ is just some matrix with no special properties. We can also write it as the form that
$$X^* = QR\Pi^T = Q\begin{bmatrix} R_1 & S \\ 0 & 0 \end{bmatrix}\Pi^T.$$
- *Complete Orthogonal Decomposition of X^**: For matrix of $X^* \in \mathbb{R}^{m\times n}$, where $rank(X^*) = r$, there exists an unitary matrix $Q \in \mathbb{R}^{m\times m}$, an unitary matrix $U \in \mathbb{R}^{n\times n}$, and a non-singular, lower-triangular square matrix $L \in R^{r\times r}$, such that
$$X^* = Q\begin{bmatrix} L & 0 \\ 0 & 0 \end{bmatrix}U^*.$$
It can be obtained from a full QR decomposition of $\begin{bmatrix} R_1^* \\ S^* \end{bmatrix} \in \mathbb{R}^{m\times r}$, which has full column rank,
$$\begin{bmatrix} R_1^* \\ S^* \end{bmatrix} = Z\begin{bmatrix} R_2 \\ 0 \end{bmatrix},$$
where $Z \in \mathbb{R}^{m\times m}$ in unitary and $R_2 \in \mathbb{R}^{r\times r}$ in non-singular, upper-triangular square matrix. Observe the Rank-Retaining QR Decomposition of X^* that

$$X^* = Q\begin{bmatrix} R_1 & S \\ 0 & 0 \end{bmatrix}\Pi^T = Q\begin{bmatrix} R_2^* & 0 \\ 0 & 0 \end{bmatrix}Z * \Pi^T = Q\begin{bmatrix} L & 0 \\ 0 & 0 \end{bmatrix}U^*.$$

2.3.4 LU Decomposition of X^*. For unstructured matrix X^*, the most common method is to obtain a decomposition $X^* = LU$, where L is lower triangular and U is upper triangular. LU decomposition is a sequence of constant LU decomposition. We are not just interested in the final upper triangular matrix, we are also interested in keeping track of all the elimination steps. We will omit this method, because it is intuitively expensive to storage and keep track of the complete L and U factors.

2.3.5 Cholesky Decomposition of X^*. The Cholesky Decomposition can be computed directly from the matrix equation $A = R^T R$ where R is upper-triangular.

The matrix X^* in our vision is dense, the Table 1 compares the relative merits of normal equations, QR decomposition, and SVD.

By Table 1 and an experimental implementation. We finally selected SVD as the method to further prune Vision Transformer. As shown in Figure 1, we apply the SVD operation after each steps of Dimension Pruning operation, which is after pruning operation on all the MHSA and MLP blocks.

3 EXPERIMENTAL EVALUATION

In this section, we verify the effectiveness of the purposed methods to further prune the Vision Transformer via Matrix Decomposition on the CIFAR-10 dataset [14].

3.1 Dataset

The CIFAR-10 dataset [14] is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

3.1.1 CIFAR-10. The CIFAR-10 is labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset consists of 60000 images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

3.2 Implementation Details

We modified the code book on

https://github.com/Cydia2018/ViT-cifar10-pruning by adding Matrix Decomposition Methods we mentioned about into the places after doing the Vision Transformer Pruning operation. Since the code book is using the CIFAR-10, we do not modify other components of the code book. Figure 2 shows the result of those implementation. We will discuss it in the next section.

3.2.1 Baseline. We evaluate developed pruning method via matrix decomposition on a popular vision transformer implementation. In our experiments, a 12-layer transformer with 10 heads and 768 embedding dimensions is evaluated on CIFAR-10. For a fair comparison, we utilize the official implementation of CIFAR-10. On the CIFAR-10, we take the released model of Vision Transformer Pruning as the baseline. We fine-tune the model on the CIFAR-10 using batch size 64 for 30 epochs. The initial learning rate is set to 6.25 × 107. Following Deit [25], we use AdamW [19] with cosine learning rate decay strategy to train and fine-tune the models.

3.2.2 Training with Pruning. Based on the baseline model, we train the vision transformer with l1 regularization using different sparse regularization rates. We select the optimal sparse regularization rate (i.e. 0.0001) on CIFAR-10 and apply it on CIFAR-10. The learning rate for training with sparsity is 6.25 × 106 and the number of epochs is 100. The other training setting follows the baseline model. After sparsity, we prune the transformer by setting different pruning thresholds and the threshold is computed by the predefined pruning rate. Which is very similar with the original dataset.

3.2.3 Fine-tuning. We finetune the pruned transformer with the same optimization setting as in training, except for removing the l1 regularization.

	VTP	VTPSVD	VTPQRD	VTPLUD
ViT patch=2	80%	85.0%	71.1%	63.7%
ViT patch=4	80%	62.8%	68.3%	57.1%
ViT patch=8	30%	53.2%	53.5%	52.3%
CIFAR-10	93%	53.2%	53.5%	49.2%

Table 2: A comparison of accuracy score among Vision Transformer Pruning (VTP), Vision Transformer Pruning with Singular Value Decomposition (VTPSVD), Vision Transformer Pruning with QR Decomposition (VTPQRD), and Vision Transformer Pruning with LU Decomposition (VTPLUD) with different patches. Note that

3.3 Results and Analysis

In Figure 2 shows a result of comparison of accuracy after implemented matrix decomposition methods, including SVD, QR decomposition, and LU decomposition. It is easy to see that the SVD performs pretty well comparing with the other two matrix decomposition methods including QR decomposition, and LU decomposition. By experience, the computation is not time consuming. The times of computation is nearly the same except the LU factorization. However, while comparing VTPSVD and VTP, the accuracy score of VTPSVD is not lower than the original VTP. which means that we need to further modify and develop our mathod to achieve a better performance in our case.

4 CONCLUSION

In this paper, we build up on and further develop the method of Vision Transformer Pruning by adding a matrix decomposition operation to meet the same goal of reduce the storage, run-time memory, and computational demands. The experiments conducted on CIFAR-10 demonstrate that the pruning with Singular Value Decomposition method can largely reduce the computation costs and storage while maintaining a comparingly high accuracy with some patches. However, this method needs to be further develop to obtain a much robust optimal performance.

5 FUTURE WORK

There are several future works that I am planning to do:

- Testing the method using another datasets, i.e. ImageNet [21], which is the one used in the original paper of Vision Transformer Pruning [33].
- Taking Non-linear Optimization and Convex Optimization courses in Winter Quarter to further develop and fine-tune this method.

REFERENCES

[1] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. 2017. Deep Learning with Low Precision by Half-wave Gaussian Quantization. arXiv:cs.CV/1702.00953

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. arXiv:cs.CV/2005.12872

[3] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. 2021. Pre-Trained Image Processing Transformer. arXiv:cs.CV/2012.00364

[4] Mark Chen, Alec Radford, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. 2020. Generative Pretraining From Pixels. In *ICML*.

[5] Yann Le Cun, John S. Denker, and Sara A. Solla. 1990. *Optimal Brain Damage*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 598–605.

[6] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation. arXiv:cs.CV/1404.0736

[7] 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. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:cs.CV/2010.11929

[8] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-Transformer. arXiv:cs.CL/1902.09113

[9] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep Learning with Limited Numerical Precision. arXiv:cs.LG/1502.02551

[10] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. 2021. A Survey on Vision Transformer. arXiv:cs.CV/2012.12556

[11] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. 2021. Transformer in Transformer. arXiv:cs.CV/2103.00112

[12] Babak Hassibi and David G. Stork. 1992. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon. In *NIPS*.

[13] Jie Hu, Liujuan Cao, Yao Lu, ShengChuan Zhang, Yan Wang, Ke Li, Feiyue Huang, Ling Shao, and Rongrong Ji. 2021. ISTR: End-to-End Instance Segmentation with Transformers. arXiv:cs.CV/2105.00637

[14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2010. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html* 5 (2010), 4.

[15] Dongsoo Lee, Se Jung Kwon, Byeongwook Kim, and Gu-Yeon Wei. 2019. Learning Low-Rank Approximation for CNNs. arXiv:cs.LG/1905.10145

[16] Shaohui Lin, Rongrong Ji, Chao Chen, Dacheng Tao, and Jiebo Luo. 2019. Holistic CNN Compression via Low-Rank Decomposition with Knowledge Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 12 (2019), 2889–2905. https://doi.org/10.1109/TPAMI.2018.2873305

[17] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning Efficient Convolutional Networks through Network Slimming. arXiv:cs.CV/1708.06519

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. arXiv:cs.CV/2103.14030

[19] Ilya Loshchilov and Frank Hutter. 2018. Fixing Weight Decay Regularization in Adam. https://openreview.net/forum?id=rk6qdGgCZ

[20] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. arXiv:cs.CV/1603.05279

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. arXiv:cs.CV/1409.0575

[22] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, and Weinan E. 2016. Convolutional neural networks with low-rank regularization. arXiv:cs.LG/1511.06067

[23] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. 2021. SCOP: Scientific Control for Reliable Neural Network Pruning. arXiv:cs.CV/2010.10732

[24] Yehui Tang, Shan You, Chang Xu, Jin Han, Chen Qian, Boxin Shi, Chao Xu, and Changshui Zhang. 2020. Reborn Filters: Pruning Convolutional Neural Networks with Limited Data. In *AAAI*.

[25] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers distillation through attention. arXiv:cs.CV/2012.12877

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:cs.CL/1706.03762

[27] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. 2021. MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers. arXiv:cs.CV/2012.00759

[28] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. arXiv:cs.CV/2102.12122

[29] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. 2021. End-to-End Video Instance Segmentation with Transformers. arXiv:cs.CV/2011.14503

[30] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning Structured Sparsity in Deep Neural Networks. arXiv:cs.NE/1608.03665

[31] Zhaohui Yang, Yunhe Wang, Kai Han, Chunjing Xu, Chao Xu, Dacheng Tao, and Chang Xu. 2020. Searching for Low-Bit Weights in Quantized Neural Networks. arXiv:cs.CV/2009.08695

[32] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. 2017. On Compressing Deep Models by Low Rank and Sparse Decomposition. In *2017 IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR). 67–76. https://doi.org/10.1109/CVPR.2017.15

[33] Mingjian Zhu, Yehui Tang, and Kai Han. 2021. Vision Transformer Pruning. arXiv:cs.CV/2104.08500

[34] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2021. Deformable DETR: Deformable Transformers for End-to-End Object Detection. arXiv:cs.CV/2010.04159