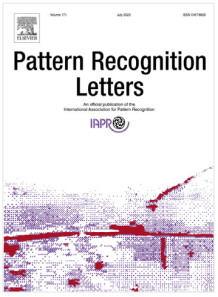
Journal Pre-proof



Multi-Dimension Compression of Feed-Forward Network in Vision Transformers

Shipeng Bai, Jun Chen, Yu Yang, Yong Liu

PII: S0167-8655(23)00286-6
DOI: https://doi.org/10.1016/j.patrec.2023.10.014
Reference: PATREC 8986

To appear in: *Pattern Recognition Letters*

Received date : 21 November 2022
Revised date : 11 April 2023
Accepted date : 18 October 2023

Please cite this article as: S. Bai, J. Chen, Y. Yang et al., Multi-Dimension Compression of Feed-Forward Network in Vision Transformers, *Pattern Recognition Letters* (2023), doi: https://doi.org/10.1016/j.patrec.2023.10.014.

Multi-Dimension Compression of Feed-Forward Network in Vision Transformers

Shipeng Bai[a,1], Jun Chen[a,1], Yu Yang[a], Yong Liu[a,*]

[a]*Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China*

Abstract

Vision Transformers (ViTs) have recently made a splash in computer vision domain and achieved state-of-the-art in many vision tasks. Nevertheless, due to their vast model size and high computational costs, rare transformer-based models are adopted in real-world applications. Since the computational costs of attention operation is the square of the input size, some compression methods for the Multi-Head Self-Attention (MHSA) module have been proposed, reducing its FLOPs successfully but almost without parameters reduction. Meanwhile, the number of parameters and computational costs in the Feed-Forward Network (FFN) module exceeds the MHSA larger, while its compression technologies have not been delved deeper. Consequently, we focus our insight on the compression of FFN layer and present a pruning method named Multi-Dimension Compression of Feed-Forward Network in Vision Transformers(MCF), which greatly reduces the model's parameters and computational costs. Firstly, we identify the critical elements in the output of the FFN module and then employ them to guide the irregular sparsity of this layer, recognizing insignificant elements of FFN layer that have less impact on the output. Successively, to discard the insignificant elements, we transform the irregular sparsity into regular sparsity and prune them, thus reducing the models' parameters and getting a substantial speed-up during inference. Extensive results on ImageNet-1K validate the effectiveness of our proposed method, which obtains significant parameters and computational costs reduction with almost unimpaired generalization. For example, we compress DeiT-Tiny with 42% reduction in FLOPs and 33% reduction in parameters, almost without losing accuracy on the ImageNet dataset. Further, we verify the effectiveness of our method in the downstream task, using the pruned DeiT-Small as the backbone for the object detection task on the COCO dataset, gaining revenue without compromising its performance.

Keywords: Vision Transformers, Feed-Forward Network, Pruning, FLOPs, Parameters

1. Introduction

With the great success of attention mechanism in natural language processing tasks, researchers have begun to introduce the attention mechanism into visual domain to model long-range dependency for better results, such as object detection[2, 18, 29], image classification[7]and semantic segmentation[11]. However, the computational costs and memory footprint of vision transformers(ViTs) block their deployment on resource limited devices such as mobile phones and various IoT devices.

In the field of model compression, many technologies have been proposed to reduce the computational costs and memory footprint of neural networks, such as quantization[6], knowledge distillation[13] and network pruning[12]. However, these works are focused on convolutional neural networks(CNNs). Owing to the substantial architecture differences between CNNs and ViTs, it is not clear whether the compression technologies of CNNs are equally applicable to ViTs.

In this case, researchers begin to explore the compression methods that are suitable for ViTs. For example,[20] lessen the computational complexity of muti-head self-attention(MHSA) by replacing the global attention with local attention,[35] employ a pyramid structure to substitute the original straight tube framework, making transformer more efficient. Other methods compress the ViTs by removing unimportant patches[25, 27, 31], trimming the insignificant heads[21], changing the dimension of the patch and dropping entire layer[34]. However, all the above methods focus their insights on the compression of the MHSA module without exploring the feed-forward network(FFN). At the same time, these methods reduce the computational costs without reducing the memory footprint. As shown in Table1, the number of parameters and FLOPs in FFN module exceeds 60% in DeiT[29] and Swin-Transformer[20], which is far more than that in the MHSA.

In this paper, we propose a pruning method named Multi-Dimension Compression of Feed-Forward Network in Vision Transformers(MCF), which not only reduces the computational costs but also the number of parameters of ViTs. Inspired by spectral clustering[16, 17], we first de-

*Corresponding author
Email addresses: shipengbai@zju.edu.cn (Shipeng Bai), junc@zju.edu.cn (Jun Chen), yu.yang@zju.edu.cn (Yu Yang), yongliu@iipc.zju.edu.cn (Yong Liu)

[1]Shipeng Bai and Jun Chen are equal contributors.

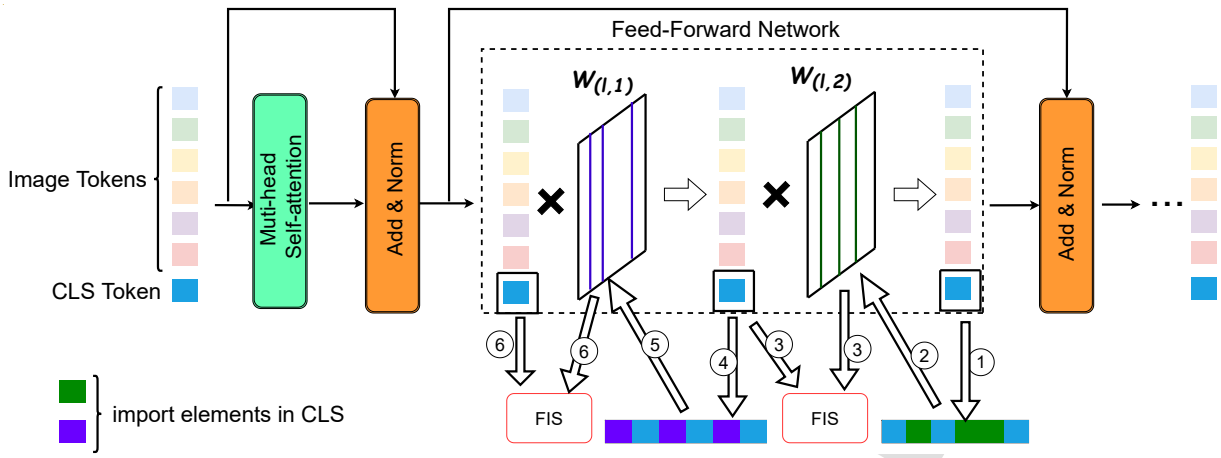


Figure 1: The general overview of MCF process. The CLS Token denotes the class token in DeiT. The FIS denotes the 'FFN Irregular Sparsity' and is described in section 3.2. The arrows and numbers denote the order of sparsity. Specifically, we first determine the significant elements(purple and green squares) in CLS Token and then feed the corresponding columns in the weights and the input of FFN into the FIS together.

How to determine the significant elements are described in section 3.2.

Table 1: FLOPs and Parameters proportion in Vision Transformers

Modle	MHSA	FFN	TYPE
DeiT-Base	6.35G (36%)	11.15G (64%)	FLOPs
	29.9M (34%)	56.6M (66%)	Params
Swin-Base	5.57G (36%)	9.83G (64%)	FLOPs
	30.9M (35%)	56M (65%)	Params

termine the important elements in the FFN module and then prune it. The general overview of MCF process is shown in Figure 1. Our contributions can be summarized as follows:

- **FFN Irregular Sparsity:** We first design a module to get the sparse weights, where the insignificant elements are changed to zero according to how much they contribute to the output of the FFN.
- **Convert to Structure Sparsity:** A new method that transforms the irregular sparsity into a structured sparsity according to the corresponding relationship between the weight of the FFN and the input. The detail is described in Section 3.3.
- **X-Transformation:** After converting to structure sparsity, different FFN modules have different transformation rules. To further accelerate the model, we use an approximation method to perform the transformation uniformly. The detail is described in Section3.4.

Extensive experiments on ImageNet[26] with Swin-Transformer[20] and DeiT[29] demonstrate the effectiveness and generality of MCF. For example, MCF yields around 34.5% FLOPs reduction and 33.9% memory reduction in DeiT-Small, with little performance degradation (only 0.5% loss compared to the uncompressed baseline).

2. Related Works

2.1. Vision Transformers

Although intuitively, the transformer model seems powerless against the particular inductive bias of spatial association for image-oriented tasks, it has proven to be as good as CNNs for visual tasks. VIT[7] first achieves similar or even superior performance on mainstream classification benchmarks as compared with traditional CNNs. DeiT[29] uses the distillation to train the transformer model. It proves that the CNN teacher can transfer its inductive bias in a soft way to the transformer student through knowledge distillation. ConViTs[8] connects a parallel convolution branch to the transformer branch to add convolutional inductive bias. Local ViTs[15] and CPVT[5] are also enhance inductive bias in different ways. Swin-Transformer[20] utilizes a shifted window along the spatial dimension to model global and boundary features. Learning from CNN, a pyramid structure replaces the straight tube structure. T2T-ViTs[35] introduces the hierarchical transformer first. Reducing image size and increasing dimension by aggregating adjacent tokens into a single token. There also have some works to improve the depth of the transformer structure[30] and transform it into self-supervised learning[4].

2.2. Vision Transformer Compression

To improve model efficiency, recent works attempt to compress the vision transformer models via various techniques. DVT[31] and ToMe[1] combine similar tokens using a general and light-eight matching algorithm. Dynamic ViTs[25] proposes a dynamic token sparsification framework to prune redundant tokens progressively and dynamically based on the input. IA-RED^2[22] and A-vit[32] all use a learned strategy network to dynamically and hierarchically remove visual tokens at different levels. PathSliming[27] identifies the effective patches in the last layer and then uses them to guide the patch selection process of the previous layer. SViTEs[3] extracts and trains sparse subnetworks dynamically while maintaining a fixed small parameter budget. UVC[34] proposes a unified building framework, combining pruning, layer skipping, and knowledge distillation to reduce the calculation of the model. Dynamic Spatial Sparsification(DSS)[24] proposes a dynamic token sparsification framework to prune the redundant tokens. Almost all VIT acceleration is token-based and reduces the calculations only. Nevertheless, the computation of the linear layer is also not negligible. So we aim to reduce the complexity of the FFN module by reducing the input dimension and hidden dimension.

3. Method

MCF has three steps in the pruning process: FFN Irregular Sparsity, Convert To Regular Sparsity, and X (input) Transformation. In this section, we first review the vision transformer briefly and then illustrate the three steps of MCF separately.

3.1. Vision Transformer Architecture

ViTs perform tokenization by dividing the input image into patches and projecting each patch to a token embedding. Meanwhile, an extra class token $[CLS]$ is added to the set of image tokens and is responsible for aggregating global image information and final classification. All of the tokens are added by a learnable vector (i.e., positional encoding) and fed into the sequentially-stacked transformer encoders consisting of a MHSA and a FFN. MHSA uses attention mechanism to learn the relationship between each patch. FFN enhances patch information through dimensional changes and non-linear module.

Notations. We use $X_l \in \mathbb{R}^{P\times C}$ to represent the input feature map of the *l-th* FFN layer. P is the patch number and C is embedding dimension. Let $W_{l,1} \in \mathbb{R}^{C\times 4C}$ and $W_{l,2} \in \mathbb{R}^{4C\times C}$ denote the weights of the *l-th* FFN layer. Based on these definitions, FFN calculation can be described as:

$$X_{(l+1)} = \sigma(X_l W_{l,1}) W_{l,2}, \tag{1}$$

where σ is the activation function. For simplicity, we omit the bias term of FFN module, which can easily be generalized to the whole equation.

3.2. FFN Irregular Sparsity(FIS)

FIS explores the irregular sparsity topology in FFN module of vision transformers. For the weights with the FFN, we first obtain its importance matrix and then change the insignificant elements to zero according to this importance matrix. For the convenience of understanding, we only use one picture to illustrate the generation of the importance matrix. Normally the size of input X of FFN module is $P \times C$, however $[CLS]$ aggregates the information from other tokens of this image, so we use $[CLS]$ instead of the whole input of FFN module (Swin-Transformer without class token we use the average of multiple patches) to facilitate our sparse computation.

As illustrated in Figure 2(a), each element in $\hat{X}_{l+1}$ is the product of the rows of X and the columns of $W_{l,1}$. In accordance with the process of matrix multiplication, the elements $\hat{X}_{l+1}(0,0)$ can be described as the sum of C elements:

$$\begin{aligned}\hat{X}_{l+1}(0,0) &= X \odot W^T_{l,1}(0) \\ &= \underbrace{X(0) * W^T_{l,1}(0,0) + \cdots + X(C) * W^T_{l,1}(0,C)}_{C}\end{aligned} \tag{2}$$

where $\odot$ indicates the Hadamard product operator. Looking at the intermediate values$(X(0)*W_{l,1}(0,0), X(1)*W_{l,1}(1,0), \cdots, X(C)*W_{l,1}(C,0))$, we observed that some values are three or four orders of magnitude different from the final value, so we can safely change them to zero without affecting the final value. Based on this, we obtain a mask of weights, where the corresponding position is zero and the rest is one. However, such a mask is closely related to the input. We use multiple images and add up the mask obtained each time to get the importance matrix $I(i)$. We formulate this process as follows:

$$\begin{aligned} I(i) &= \sum_{j}^{S} \Psi(X \odot W^T_{l,1}(i)), \quad i \in (0, 4C) \\ \Psi(z) &= \begin{cases} 0, & m < z < n \\ 1, & others \end{cases} \end{aligned} \tag{3}$$

in which S is the tiny dataset(1024 images) that we used to calculate the mask. The function of $\Psi(\cdot)$ is to set the elements that satisfied the threshold condition to zero. For a certain layer, we get two thresholds m and n based on its connection weights' average absolute value and variance.

After getting the importance matrix, we sparse the weights according to the pruning ratios. Specifically, the sparse process of the weight $W_{l,1}$ can be described as the following formula:

$$W'_{l,1} = W_{l,1} \odot h(I), \quad h(x) = \begin{cases} 0, & x < threshold \\ 1, & others \end{cases} \tag{4}$$

where $h(\cdot)$ changes some elements in each column to 0 and the rest to 1 according to the pruning ratio.

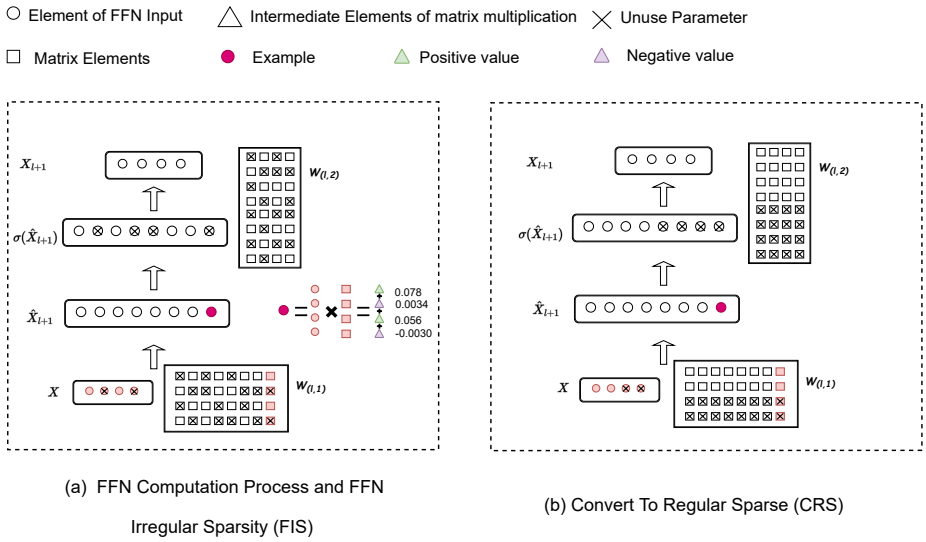


Figure 2: An example of FIS and CRS on FFN computation. (a) shows the computation of an FFN for a given input and the basis for the sparsification of the weights. (b) describes the transformation of the resulting irregular sparsity into regular sparsity with the same transformation of the input X.

Further, the size of $\hat{X}_{l+1}$ is $P \times 4C$, which means we have to perform 4C operations on a single image to get the mask of FFN weights. This is undoubtedly resource-intensive. To simplify each sparse process, we look for essential parameters and use them to guide the sparsity of FFN. Each image has a category with the highest corresponding score and based on the score categories we derive the critical elements in each layer from back to front. According to matrix multiplication, each component of the CLS corresponds to a column of the weight matrix. As shown in Figure 1, the crucial elements of CLS in the output are used to qualify the essential columns in the weight matrix. These columns and the CLS will be involved in the FIS operation.

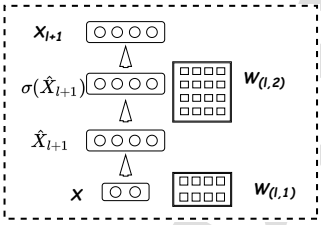


Figure 3: Final structure. Primitive structure is shown in Figure2(a).

3.3. Convert To Regular Sparsity (CRS)

After the FIS operation, we get irregular sparse weights. Such weights have no acceleration effect in inference without special hardware or library. This can be addressed with structured pruning that can achieve practical acceleration without specific software and hardware design, so irregular sparsity must be changed to regular sparsity.

In the computation of the FFN, the inputs and weights are calculated in one-to-one correspondence, i.e., if any two values in X are swapped in position, those same two values in the weights are also swapped in position, with no effect on the final result. As shown in Figure 2(b), we place the zeros in the weights at the end of the row and move the rest of the elements up in order to facilitate cropping. At the same time, X is also panned and changed. The experiments illustrate that we can safely trim these zeros without affecting the final result.

3.4. X-Transformation

There still have two problems hinder the acceleration of the model. On the one hand, each linear layer of FFN has different sparse rules, and the input of each linear layer needs to be changed accordingly. In this process, there is extra computational consumption and memory consumption. Therefore, a uniform variation is sought to replace the complex and tedious change of input in the inference. In the case of X-pruning, we look at the inference process for plenty of images. We found that almost all the cropped pixels were at the edges of the image, which is consistent with the original experience that removing background-related pixels does not affect the final result.

On the other hand, every linear layer in the FFN has to be changed, which destroys the integrity of the FFN. The function of FFN module is to enhance the information between patches and increase the non-linearity of the model, with a dimensional change of ($D \Rightarrow 4D \Rightarrow D$). It is undoubtedly somewhat tedious to perform two changes

Table 2: Comparison of the vision Transformers compressed by MCF with different benchmarks on ImageNet.

Model	Method	**Top-1** Acc. (%)	**FLOPs** (G)	**FLOPs** (G) ↓ (%)	**Params** (M)
DeiT-Tiny	Baseline	72.2	1.3	0	5.72
	UVC[34]	71.3(-0.9)	0.64	-50.8	5.72
	SCOP[28]	68.9(-3.3)	0.8	-38.4	5.72
	SVITE[3]	70.12(-2.08)	0.99	-24.0	5.72
	HVT[23]	69.7(-2.5)	0.64	-50.8	5.74(+0.02%)
	WDPruning[33]	71.1(-1.1)	0.9	-30.8	3.8(-29.6%)
	MCF(Ours)	**71.5(-0.7)**	0.72	-42.0	**3.87(-33.3%)**
DeiT-Small	Baseline	79.8	4.6	0	22.05
	UVC[34]	79.44(-0.36)	2.7	-42.4	22.05
	PathSliming[27]	79.4(-0.4)	2.6	-43.5	22.05
	SVITE[3]	79.22(-0.58)	3.14	-31.7	22.05
	HVT[23]	78.0(-1.8)	2.4	-47.8	22.09(+0.04%)
	IA-RED^2[22]	79.1(-0.7)	3.22	-30.0	22.05
	ToMe[1]	79.4(-0.4)	2.7	-42.4	22.05
	DSS[24]	79.3(-0.5)	2.9	-37.0	22.05
	MCF(Ours)	**79.4(-0.4)**	2.5	-45.9	**14.67(-34%)**
Swin-Base	Baseline	83.41	15.4	0	86.8
	SPViTs[10]	82.8 (-0.6)	12.1	-21.4	72.2(-16.8%)
	STEP[14]	80.6(-2.8)	12.1	-21.4	69.3(-20.2%)
	MCF(Ours)	**82.9(-0.5)**	11.2	-27.3	**64.5(-26.5%)**

Table 3: Ablation study on the modules implemented in MCF. X-T denotes the X-Transformation

Method			DeiT-Small			
FIS	CRS	X-T	Top-1 (%)	FLOPs(G)	Params(M)	Throughput(image/s)
✗	✗	✗	79.8	4.6	22.05	524
✓	✗	✗	79.7	4.6 (↓ 0%)	22.05(↓ 0%)	524 (↑ 0%)
✓	✓	✗	79.4	3.5 (↓ 23.9%)	16.5 (↓ 25.2%)	956 (↑ 82%)
✓	✓	✓	79.3	3.0 (↓ 34.7%)	14.67 (↓ 33.5%)	1471 (↑ 180%)
direct pruning			70.2	3.0 (↓ 34.7%)	14.67 (↓ 33.5%)	1471 (↑ 180%)

in one FFN module. The output dimension of the first module could be reduced as well, making the FFN module more uniform. As our pruning changes the output of each linear layer, its dimensionality is the same as the original. However, the FFN module consists of two linear layers and an activation function that acts as a whole. Trimming the input of each linear layer corrupts the continuum integrity of the FFN module. The output of the first linear layer is the input to the second linear layer, and the output dimension of $W_{l,1}(4D)$ is pruned so that its output dimension satisfies the input of the second linear layer. we trim each input to the linear layer first, as shown in Figure 2(b), every cross is cut out. The element in $\sigma(\hat{X}_{l+1})$ that is cropped also corresponds to the column in $W_{l,1}$, so we can cut out the column in $W_{l,1}$ that corresponds to the input. The overall structure of pruned FFN module is shown in Figure 3.

4. Experiments and Analysis

4.1. Datasets and Metrics

We conduct image classification experiments on the ImageNet[26] and object detection experiments on COCO[19]. Our method is implemented on DeiT-Tiny/Small/Base[29] and Swin-Transformer[20] model. DeiT has the same architectures as ViTs, except for an additional distillation. Swin-Transformer has a shifted window along the spatial dimension to model global and boundary features, and a backbone with a deep backbone structure. Our metrics are accuracy, parameters, and FLOPs.

4.2. Baseline Pruning Methods and Implementation.

In order to demonstrate the effectiveness of our method on compression, we compare our method

Table 4: Ablation studies of Throughput

Ablation	DeiT-Tiny	DeiT-Small	DeiT-Base
Baseline	1230	524	194
Throughput ↑	3476(182%)	1471(180%)	505(160%)

with the latest compression methods including IRRED^2[22], PathSliming[27], UVC[34], SVITE[3], HVT[23], WDPruning[33], ToMe[1], DSS[24] and SCOP[28].

Our compressed model can get in one training session. It's divided into three Small parts. First, we sparse the FFNs sequentially from back to front, performing FIS operations on one FFN per epoch.(This will go through a dozen epochs). Then the X-Transformer phenomenon we observed is applied to the inference process. The remaining rounds compensate for the loss in the compression process. As mentioned above, we mainly follow the training setup of Swin-Transformer[20], except for a relatively small learning rate, which facilitates the sparsity of the pre-trained model.

4.3. Comparison with the state-of-the-art methods

The main results are listed in Table 2. We note that the current methods are aimed at reducing the number of floating-point calculations, and there is no way to reduce the number of model parameters. MCF reduces the number of floating point calculations by the same amount as other methods while further reducing the parameters by about 40%, with guaranteed performance.

In DeiT-Tiny compression, MCF cuts 42.0% of the FLOPs and 33.3% of the parameters while losing only 0.7% accuracy. In particular, the latest SVITE reduces the FLOPs by 24%, but its accuracy drops by 2.08%. HVT reduces the FLOPs by 50% and increases parameters by 0.02M with 2.5% accuracy drop. MCF outperforms other methods both in terms of accuracy and parameters. In DeiT-Small compression, MCF has a similar performance to the PathSliming, ToMe, and DSS, with only a 0.4% accuracy drop and a 40% FLOPS drop, but they have no reduction in the number of parameters. Compared to UVC, SVITE, MCF has a significant advantage in terms of the number of parameters, while our inference speed and performance are equal to theirs. In Swin-Base, MCF outperforms SPViTs and STEP, achieving the greatest compression ratio with the least performance loss. These experiments demonstrate the effectiveness of our method.

4.4. Ablation and Throughput Study of MCF

As MCF deploys three different approaches to the model compression process in sequence, it would be natural to question whether each moving part is necessary for the pipeline and their respective contributions to the final results. Also, explore how much acceleration such compression can provide in hardware.

Table 5: DeiT-Small as the backbone of detection

Detection	Backbone	AP (%)	Params (M)	FLOPs (G)
YOLOS-S	DeiT-S(original)	36.1	30.7	194
	DeiT-S(pruned)	36.0	23.9	140

We conduct ablation study in DeiT-Small, the results are presented in Table 3. We first present the result when we only conduct FIS on DeiT-Small to get the irregularly sparse model. There is only a 0.1% accuracy drop, indicating that our sparse method has little impact on the performance of the model. However, the FLOPs, Params and Throughput of DeiT-Small have no improvement compared to the original model. So we need to transform irregular sparsity into regular sparsity and prune it to obtain substantial acceleration. Using FIS and CRS without X-Transformer, the computational effort is reduced by 25%, the number of parameters is reduced by 25%, and the Throughput is increased by 82%. For further acceleration, we propose X-Transformation to release the extra overhead caused by transforming the inputs. The reduction of FLOPs and Params in the fourth row of the table demonstrates the effectiveness of X-T. In the last row, we do not use any method to compress the model. The performance is reduced by 9% compared to the baseline.

As shown in Table 4, the Throughput has increased significantly after compression. In DeiT-Tiny, the Throughput has increased from 1230 to 3467 images/s, while DeiT-Small and DeiT-Base have also increased by almost 200%. The experiments of Throughput were carried out with a GPU Titan RTX (24GB) with a batch size of 64.

4.4.1. Downstream task: object detection

Although many pruning methods perform well in classification, there is no way to measure whether this compression affects their usage in downstream tasks. We apply the compressed model to object detection on COCO[19] dataset and estimate that it performed well in downstream tasks. The results are shown in the Table 5. The reduction in resource consumption is 30% with only 0.1% AP drop. Our training follows this YOLOS-S[9] training method precisely, except that the backbone is replaced with a pruned model.

5. Conclusion

In this work, we propose a novel pruning method to reduce the resource consumption of ViTs. MCF has three components, FIS, CRS, and X-Transformation. The effectiveness of our method is verified on classification and detection tasks with DeiT and Swin-Transformer models, and the results show that VRF significantly reduces the size and computational costs of the model without affecting its performance.

References

[1] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. 2022.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[3] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

[4] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021.

[5] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

[6] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021.

[9] Yuxin Fang, Bencheng Liao, Xinggang Wang, Jiemin Fang, Jiyang Qi, Rui Wu, Jianwei Niu, and Wenyu Liu. You only look at one sequence: Rethinking transformer in vision through object detection. In *NeurIPS*, 2021.

[10] Haoyu He, Jing Liu, Zizheng Pan, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Pruning self-attentions into convolutional layers in single path. *ArXiv*, abs/2111.11802, 2021.

[11] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. Transreid: Transformer-based object re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15013–15022, 2021.

[12] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

[13] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[14] Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459, 2021.

[15] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021.

[16] Zhihui Li, Feiping Nie, Xiaojun Chang, Liqiang Nie, Huaxiang Zhang, and Yi Yang. Rank-constrained spectral clustering with flexible embedding. *IEEE transactions on neural networks and learning systems*, 29(12):6073–6082, 2018.

[17] Zhihui Li, Feiping Nie, Xiaojun Chang, Yi Yang, Chengqi Zhang, and Nicu Sebe. Dynamic affinity graph construction for spectral clustering using multiple features. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):6323–6332, 2018.

[18] Zhihui Li, Lina Yao, Xiaojun Chang, Kun Zhan, Jiande Sun, and Huaxiang Zhang. Zero-shot event detection via event-adaptive concept relevance mining. *Pattern Recognition*, 88:595–603, 2019.

[19] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[21] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

[22] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red : Interpretability-aware redundancy reduction for vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[23] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 377–386, 2021.

[24] Yongming Rao, Zuyan Liu, Wenliang Zhao, Jie Zhou, and Jiwen Lu. Dynamic spatial sparsification for efficient vision transformers and convolutional neural networks. *arXiv preprint arXiv:2207.01580*, 2022.

[25] 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.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[27] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. *arXiv preprint arXiv:2106.02852*, 2021.

[28] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *Advances in Neural Information Processing Systems*, 33:10936–10947, 2020.

[29] 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.

[30] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021.

[31] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in Neural Information Processing Systems*, 34, 2021.

[32] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10809–10818, 2022.

[33] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 2022, 2022.

[34] Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. *arXiv preprint arXiv:2203.08243*, 2022.

[35] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021.

1. Highlights

- FFN Irregular Sparsity: A new sparse method that replaces the insignificant elements with zero.
- Convert to Structure Sparsity: It converts irregular sparsity into regular sparsity.
- X-Transformation: A unified converting method further accelerates the model.
- Extensive experiments on ImageNet with Swin-Transformer and DeiT demonstrate the effectiveness of our method.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: