SCIENCE CHINA
Information Sciences

• **Supplementary File** •

A Unified Pruning Framework for Vision Transformers

Hao YU[1] & Jianxin WU[1*]

[1]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023*, China*

Appendix A The Components of ViT

The standard transformer receives an input as a 1D sequence of token embeddings. Following this design, ViT reshapes an image into flattened sequential patches and linearly projects each patch into an embedding vector. It also concatenates the patch embeddings with a trainable classification token and adds position embeddings. Consider a ViT model with L blocks. Given an image $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$, ViT first reshapes it into flattened 2D patches $\mathbf{x}_p \in \mathbb{R}^{N\times(CP^2)}$, where $H \times W$ is the original input resolution, C is number of input channels, $P \times P$ is the resolution of each patch, $N = HW/P^2$ is the number of patches. Then, the sequence of embedding $\mathbf{x}_0$ is formed by

$$\mathbf{x}_{patch} = \mathrm{FC}_{patch}(\mathbf{x}_p)\,, \tag{A1}$$

$$\mathbf{x}_0 = [\mathbf{x}_{cls} \,|\, \mathbf{x}_{patch}] + \mathbf{x}_{pos}\,, \tag{A2}$$

where FC_{patch} is the linear projection with D output channels. $\mathbf{x}_{cls}$ and $\mathbf{x}_{patch}$ are the learnable classification token and position embedding, respectively, $[\cdot\,|\,\cdot]$ is the column-wise concatenation.

The embedding $\mathbf{x}_0$ is subsequently fed into the transformer blocks. ViT stacks several blocks to construct the whole model, and each consists of two parts: the attention layer and the FFN. In the attention layer, ViT applies LayerNorm and three linear projections (FC_q, FC_k and FC_v) in parallel to generate queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$, i.e.,

$$\mathbf{x}'_{k-1} = \mathrm{LN}(\mathbf{x}_{k-1})\,, \tag{A3}$$

$$\mathbf{Q} = \mathrm{FC}_q(\mathbf{x}'_{k-1})\,, \tag{A4}$$

$$\mathbf{K} = \mathrm{FC}_k(\mathbf{x}'_{k-1})\,, \tag{A5}$$

$$\mathbf{V} = \mathrm{FC}_v(\mathbf{x}'_{k-1})\,, \tag{A6}$$

where LN is layer normalization, $\mathbf{x}_{k-1}$ is the input embedding for $k = 1, \ldots, L$. Then, the multi-head self-attention (MHSA) and a linear projection FC_{proj} are performed:

$$\mathbf{x}''_{k-1} = \mathbf{x}_{k-1} + \mathrm{FC}_{proj}(\mathrm{MHSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})), \tag{A7}$$

where $\mathbf{x}''_{k-1}$ is the output of the attention layer in the kth block. The input and output dimensions of these linear projections are all D. Note that the MHSA module does *not* contain any parameters. It reshapes the queries, keys and values and applies scaled dot-product attention (cf. [5] for details). Then, $\mathbf{x}''_{k-1}$ will be sent into the FFN, which contains two FC layers (FC_1 and FC_2) with a GELU non-linearity:

$$\mathbf{x}_k = \mathbf{x}''_{k-1} + \mathrm{MLP}(\mathrm{LN}(\mathbf{x}''_{k-1}))\,, \tag{A8}$$

$$\mathrm{MLP}(\mathbf{x}) = \mathrm{FC}_2\left(\mathrm{GELU}(\mathrm{FC}_1(\mathbf{x}))\right)\,. \tag{A9}$$

As a whole, there are 6 learnable linear projections and 2 LayerNorm modules in each ViT block. ViT only extracts the classification token in the last block as the final representation, which is a different design from the practice in CNNs (i.e., global average pooling of all patch embeddings). It is worth noting that the parameter size of each block is $12c^2 + 4c$, which only depends on the embedding dimension.

As shown in Figure A1, we divide the block into three irrelevant structural components. Note that because of the existence of shortcut connections, the following numbers *must be the same in the entire network* after pruning: the number of channels in all components linked by the shortcut connections, the patch embedding dimensionality, and the final FC layer dimensionality. To make sure the validity of all summation and concatenation operations, the dimensionality of the classification token and the position embedding must be the same as this number, too.

Appendix B The Components of PVTv2

We also prune PVTv2 on ImageNet-1k. Our goal is to demonstrate that our method applies not only to the original transformer structures, but also to those variants that have hierarchical architecture and combine MHSA with convolution layers. PVTv2 proposes the overlapping patch embedding module to divide a model into four stages. It also introduces a convolution layer and a 3×3 depth-wise convolution into the attention layer and FFN, respectively. As shown in Figure B1, we demonstrate four uncorrelated components in a PVTv2 block:

* Corresponding author (email: wujx2001@gmail.com)

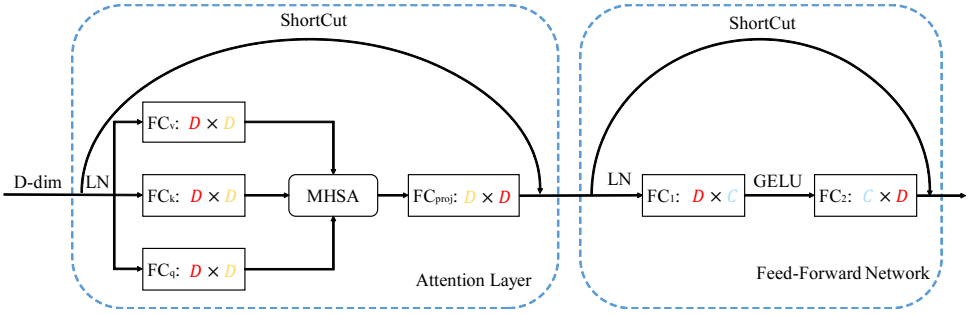

Figure A1 Illustration of three uncorrelated components in a ViT block. Our method prunes not only the channels inside the attention layer (the numbers shown in gold color) and the FFN (the numbers shown in blue), but also the channels across shortcut connections (the numbers shown in red). The two numbers in each rectangle represent the number of input and output channels of the linear projections, respectively.

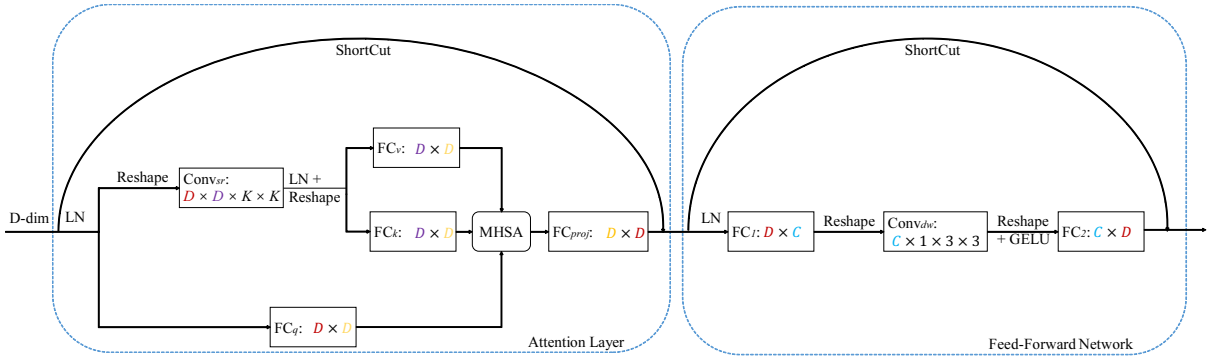

Figure B1 Illustration of four irrelevant components in a PVTv2 block.

• Component 1: The numbers shown in red, namely the shortcut connections that chain representations in every *stage*, i.e., the input channels of FC_q, Conv_{sr} and FC_1, the output channels of FC_{proj} and FC_2, and the first and third LN layers;

• Component 2: The numbers shown in gold, namely the attention embedding filters inside the attention layer in every *block*, i.e., the input channels of FC_{proj} and the output channels of FC_q, FC_k and FC_v;

• Component 3: The numbers shown in blue, namely the FFN inter-layer filters in every *block*, i.e., the input channels of FC_2, the first channel of Conv_{dw} and the output channels of FC_1;

• Component 4: The numbers shown in purple, namely the spatial reduction attention embedding filters inside the attention layer in every *block*, i.e., the input channels of FC_k and FC_v, the output channels of Conv_{sr}, and the second LN layer.

Note that because there are three additional patch embedding modules, when calculating the importance scores of component 1, we divide the whole shortcut connections into four stages and calculate importance separately. In particular, there are only three components in the final stage because the Conv_{sr} layer is removed.

Appendix C The Method of Pruning Blocks

In this section, we propose a new way to prune blocks in transformer-based models. More precisely, when pruning width, we first calculate all channels' important scores and then prune all redundant channels at once. Different from this pipeline, we construct a progressive method to compress blocks.

• First, for each block, calculate the KL-Divergence with & without it;

• Second, remove the least important block and reserve the pruned model;

• Third, if the pruned model has not reached the compression target, then use the first two steps to prune one more block; otherwise we obtain a sub-model with fewer blocks and fine-tune it.

Note that the output of one block is its next block's input, so once we remove a block, the importance of every remaining block will change, and blocks adjacent to the deleted block will become more important. Therefore, we prune one block at a time. Later we will show this progressive block pruning method is better than pruning many blocks at once.

Unlike CNNs, given a ViT model with L blocks, if we remove the FFN of the kth ($k \in \{1, 2, \ldots, L-1\}$) block and the attention layer of the next block, the remaining model is still guaranteed to be a legal ViT with $L-1$ blocks. Therefore, when compressing blocks, we actually compute $2L-1$ instead of L importance scores. The $L-1$ new *hybrid* block pruning candidates we propose here are *novel and useful*.

Appendix D Experiments

In this section, we evaluate the performance of our UP-ViTs. We first test the effectiveness on ImageNet-1k. We prune DeiT-B and present accuracy on downstream small-scale classification. We also compress the hierarchical architecture, PVTv2, which introduces convolution into MHSA. In addition, more results on downstream object detection datasets will be presented. We will show that UP-ViTs achieve better or comparable accuracies on these tasks. We also prune Transformer with adaptive input representations

Table D1 Summary of small-scale datasets.

Dataset	#Train	#Val.	#Cat.	Dataset	#Train	#Val.	#Cat.
CIFAR-100 [11]	50000	10000	100	Indoor67 [21]	5360	1340	67
CUB-200 [25]	5994	5794	200	Pets [20]	3680	3669	37
Cars [10]	8144	8041	196	DTD [2]	3760	1880	47
Aircraft [17]	6667	3333	100	iNaturalist-2019 [24]	265213	3030	1010

for the language modeling tasks. Finally, we will end this section with several analyses. All the experiments are conducted with PyTorch. Our code is open-source and available at https://github.com/yuhao318/UP-ViT.

Appendix D.1 Datasets and metrics

We first list the details of the datasets and metrics we used.

Classification. The ImageNet-1k [4] dataset consists of 1.28 million training and 50000 validation images. Those images have various spatial resolutions and come from 1000 different categories. ImageNet-1k is usually used as the benchmark for model pruning.

Besides ImageNet-1k, we also evaluate UP-ViTs on eight small-scale datasets. Table D1 summarizes the information of these datasets, including training and validation sizes, and the number of categories.

Objection Detection. We evaluate object detection performance on the MS COCO2017 [14] and the Pascal VOC07+12 [6] datasets. MS COCO 2017 contains 80 categories with 118K training and 5K validation images, respectively. Pascal VOC07+12 has 20 classes. Specifically, VOC07 contains a train-val set of 5011 images and a test set of 4952 images, and VOC12 contains a train-val set of 11540 images. We use mean Average Precision (mAP) to measure the detection accuracy.

Language Modeling. We also evaluate our approach in the WikiText-103 [18] dataset. The training data of WikiText-103 comprises about 100M tokens with 28K articles and a vocabulary of around 260K. The test data contains 245K tokens with 60 articles. The dataset is composed of shuffled Wikipedia articles where the context carries across sentences. We use perplexity to measure the performance of models. A lower perplexity indicates the probability distribution is good at predicting the sample.

Appendix D.2 Pruning DeiT-B on ImageNet-1k

As we stated previously, one of our goals is to promise the consistency of the model structure, so we first prune DeiT-B on ImageNet-1k to DeiT-S & T, and to further show the validity of our new block pruning strategy, we also prune blocks on the basis of the pruned DeiT-T. Finally we tested the compressed models' transferring ability in several downstream classification datasets.

Implementation details. First we prune DeiT-B into UP-DeiT-S & T. Note that our compressed UP-DeiT-S & T share the same structure as the original DeiT-S (for small) & DeiT-T (for tiny). After all channel scores were calculated, we removed all redundant channels at once, then fine-tuned it with knowledge distillation in 200 epochs. We initialized the learning rate as 3e-4 and set α as 0.5 when fine-tuning UP-DeiT-S. During fine-tuning UP-DeiT-T, the learning rate and α were 1e-3 and 0.2, respectively. After obtaining the compressed UP-DeiT-T model, we continue to delete 2 more blocks and fine-tune the sub-model. During fine-tuning, we set the learning rate as 3e-4 and trained 50 epochs with knowledge distillation at α being 0.4.

In the above experiments, we used the AdamW [16] optimizer and the cosine decay schedule. The weight decay was 1e-3 and the mini-batch size was 256. Random cropping, random horizontal flipping, color jittering and CutMix [31] were applied as data augmentations. The mixing ratio of CutMix was 0.5. In particular, unlike DeiT, we did *not* use mixup [32], random erasing [33] or Rand-Augment [3] in any of our experiments.

Results. Table D2 shows the pruning results of DeiT-B. We tested model accuracy on the ImageNet-1k validation dataset. During testing, the shorter side was resized as 256 by bilinear interpolation and then we cropped the 224×224 image patch in the center. The accuracy of the last epoch was reported. We also list the models' throughput in a TITAN Xp GPU with a fixed 32 mini-batch size.

Performance comparison between DeiT and UP-DeiT proves the effectiveness of our framework. Compared with DeiT-T, UP-DeiT-T-10 achieves 1.58% higher accuracy with only 10 blocks. Note that our experimental results are better than MD-DeiT [8], which is a novel distillation approach. Our method also greatly surpasses previous ViT compression methods, such as Evo-ViT and PS-ViT.

It is worth noting that those patch selecting models (e.g., Evo-ViT) have a much higher number of parameters than the original DeiT-B. This is because first, the number of parameters of ViT blocks is only related to the embedding dimension (c.f. Appendix A), while those methods only prune the patches and do not reduce the embedding dimension, and second, in order to filter out which patches should to be pruned, those methods add a patch selection module, which sometimes brings a larger number of parameters. UVC adds a weight mask to each FC layer, which doubles the model parameters. Our method can compress the dimensionality and can be combined with it.

Transferring to small datasets. We adopted mini-batch size 256 and learning rate 1e-3 when fine-tuning the DeiT-T model, and for the compressed models the learning rate was 3e-3. During fine-tuning, we applied the AdamW optimizer and the CutMix augmentation strategy. CutMix's ratio was set to 0.5. We trained models with 50 epochs.

Table D3 shows the results. The pruned model UP-DeiT-T *always* outperforms DeiT-T on all 8 datasets, which indicates that our UP-ViTs method boosts the generalization of DeiT significantly. Also, except on the iNaturalist-2019 dataset, the 10-block UP-DeiT-T-10 outperforms the 12-block DeiT-T on other datasets. UP-DeiT-T-10 even performs better than UP-DeiT-T on the Aircraft dataset.

Appendix D.3 Pruning PVTv2 on ImageNet-1k

In this section, we adopted two tasks to show our method's effectiveness, i.e., only prune depth, and prune both depth and width. Then we evaluate our model's performance on the object detection task.

Implementation details. We conducted two experiments for comparison. In the first one, we deleted blocks and pruned PVTv2-B2 into UP-PVTv2-B1; in the second, we pruned both width and depth and directly compressed PVTv2-B2 into UP-PVTv2-B0. Note that the structures of UP-PVTv2-B1 & B0 are exactly the same as those of PVTv2-B1 & B0, respectively. It is

Table D2 Results of pruning DeiT-B in ImageNet-1k. UP-DeiT-S & T and UP-DeiT-T-10 are our pruned model with 12 and 10 blocks respectively. Note that we denote the method proposed by [8] as MD-DeiT (manifold distillation DeiT). * denotes results copied from the original paper.

Model	Throughput (img/s)	#Param.	Top-1 Acc.
DeiT-B [23]	199.2	86.6M	81.84%
DeiT-S [23]	603.1	22.1M	79.85%
T2T-ViT-14 [30]	456.6	21.5M	81.38%
Swin-T [15]	384.5	28.5M	81.17%
PVT-S [26]	389.9	24.5M	79.79%
VTP* [34]	-	48.0M	80.70%
NViT-S* [28]	-	23.0M	81.22%
Evo-ViT [27]	305.7	87.3M	81.11%
MD-DeiT-S* [8]	603.1	22.1M	81.48%
UVC [29]	420.7	172.9M	80.55%
PS-ViT* [22]	282.4	-	81.50%
UP-DeiT-S	603.1	22.1M	**81.56%**
DeiT-T [23]	1408.5	5.7M	72.20%
T2T-ViT-10 [30]	857.9	5.9M	75.00%
PVT-T [26]	691.7	13.2M	75.00%
NViT-T* [28]	-	6.4M	73.91%
MD-DeiT-T* [8]	1408.5	5.7M	75.06%
UP-DeiT-T	1408.5	5.7M	**75.79%**
UP-DeiT-T-10	1674.2	4.8M	73.97%

Table D3 Accuracy (%) on different small-scale classification datasets.

Dataset	DeiT-T	UP-DeiT-T	UP-DeiT-T-10
CIFAR-100	85.01	**86.52**	85.98
CUB-200	76.75	**81.34**	80.93
Cars	85.24	**87.34**	86.27
Aircraft	72.58	75.08	**75.81**
Indoor67	74.70	**78.21**	77.99
Pets	88.93	**92.20**	91.69
DTD	71.28	**73.83**	72.50
iNaturalist-2019	69.54	**70.86**	69.17

Table D4 Results of pruning PVTv2 on ImageNet-1k. We report the accuracy of the last epoch on ImageNet-1k validation. UP-PVTv2-B1 & B0 are our compressed models.

Method	Throughput (img/s)	#Param.	Top-1 Acc.
PVTv2-B2	80.3	25.36M	82.08%
PVTv2-B1	139.9	14.01M	78.62%
UP-PVTv2-B1			**79.48%**
PVTv2-B0	249.9	3.67M	70.47%
UP-PVTv2-B0			**75.30%**

Table D5 mAP of UP-PVTv2 on Pascal VOC07 test. We trained RetinaNet on the Pascal VOC07+12 train-val dataset.

Backbone	#Param.	FLOPs	mAP	Backbone	#Param.	FLOPs	mAP
PVTv2-B1	11.7M	85.0G	81.9	PVTv2-B0	22.5M	108.9G	80.5
UP-PVTv2-B1			**83.0**	UP-PVTv2-B0			**82.0**

worth noting that PVTv2 introduces a convolutional layer into the attention module to reduce the computational cost, so we added one extra component when calculating the importance scores.

During training, we followed the data augmentation and learning rate setting of pruning DeiT-B. In particular, when pruning PVTv2-B2 to UP-PVTv2-B0, we used a learning rate of 1e-3 and an α of 0.2. The sub-model was fine-tuned for 100 epochs. When we pruned PVTv2-B2 to UP-PVTv2-B1, the learning rate were initialized as 3e-4 and we trained 50 epochs. Especially, we found that directly distilling the features in the penultimate layer (after the global average pooling and before the final classifier) is advantageous, so we applied the MSE instead of the KL-divergence as the distillation loss. The distillation ratio α was set to 20.

Results. Table D4 shows the results on PVTv2, and similar to the previous experiments, our UP-PVTv2 models obtain significantly better results when compared with the original models. Especially, UP-PVTv2-B0 improves the accuracy of PVTv2-B0 by 4.83% for ImageNet classification.

Transferring to object detection. To further validate the effectiveness of our method on a larger object detection dataset, we investigate UP-PVTv2's performances on object detection with Mask R-CNN [7] and RetinaNet [13]. First, we adopted both PVTv2-B0/B1 and our UP-PVTv2-B0/B1 as backbones of RetinaNet on Pascal VOC07+12.

As reported in Table D5, the compressed UP-PVTv2 models significantly outperform original models on Pascal VOC detection. For example, UP-PVTv2-B0 achieves 82.0 mAP, which surpasses the original PVTv2-B0 model by 1.5 AP points, and is on par with the performance of the original PVTv2-B1.

We also investigated the performance of (UP-)PVTv2-B1 backbone on both one-stage and two-stage object detectors, i.e., Mask R-CNN and RetinaNet. From the results shown in Table D6 on the larger MS-COCO dataset, our method significantly outperforms the original PVTv2-B1.

Note that the previous patch slimming methods cannot be used in such pyramidal structures, because these hierarchical architectures introduce convolution to aggregate all the patch information, and the token slimming methods randomly discard the unwanted patches. Similarly, the object detection models need all the patch to determine the target bounding box, so the previous token slimming methods do not compress the pyramidal structures (e.g., PVT or Swin Transformer), and they do not transfer their compression models on the object detection tasks. In the original Evo-ViT paper, the authors also admit that their current method cannot be used in dense prediction tasks, such as object detection and instance segmentation.

Appendix D.4 Pruning Transformer on WikiText-103

We also prune the standard Transformer with adaptive input representations on the WikiText-103 dataset. UP-Transformer achieved lower perplexity compared with training from scratch or other pruning strategies.

Implementation details. The original Transformer model follows the architectural choice described in [1], which includes sinusoidal position embeddings in the input layer and stacks 16 decoder blocks for the WikiText-103 dataset. Each MHSA module has 16 heads. For adaptive input representations, we have three bands of size 20K, 40K and 200K. The embedding layer and FFN's hidden-state have dimensions of 1024 and 4096, respectively. We reduce those dimensions to 512 and 2048 after pruning.

During fine-tuning, we use a layer dropout of 0.1 and train on 4 GPUs. We limit the number of tokens per GPU to a maximum threshold 1536 per GPU, which means each GPU processes 1536 tokens using the same model parameters. We accumulate gradient updates over two batches before committing a parameter update [19]. We use the AdamW optimizer and the weight decay is 0.05. The learning rate is linearly warmed up from 1e-5 to 1e-3 for 2K steps and then annealed using a cosine learning rate schedule. We renormalize gradients if their norm exceeds 0.1.

Results. Table D7 shows the results on WikiText-103. Training from scratch entails creating a new Transformer model with the same architecture as the pruned one and directly training it. Random and ℓ_1-norm mean that we select channels randomly

Table D6 mAP of (UP-)PVTv2-B1 on the MS COCO2017 validation dataset.

Backbone	Method	#Param.	FLOPs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
PVTv2-B1	RetinaNet 1x	23.75M	235.9G	40.5	61.0	43.3	24.0	43.9	53.1
UP-PVTv2-B1				**41.1**	**62.0**	**43.5**	**25.1**	**44.2**	**54.6**
PVTv2-B1	Mask R-CNN 1x	33.66M	252.1G	40.8	63.1	44.5	25.3	44.5	53.0
UP-PVTv2-B1				**41.6**	**64.1**	**45.5**	**25.9**	**45.0**	**54.0**

Table D7 Perplexity of Transformer on WikiText-103 validation dataset.

Backbone	#Param.	Iterations	Perplexity
Baseline [1]	291M	-	19.00
Training from Scratch	95M	60k	24.38
Random	95M	60k	22.36
ℓ_1-norm [12]			22.52
UP-Transformer			**22.00**
UP-Transformer + KD	95M	60k	**20.52**
		300k	**19.88**

Table D8 Results on the ImageNet-1k validation set with & without knowledge distillation.

Model	Distillation	Epochs	Train Times (h)	Top-1 Acc.
UP-DeiT-T	✔	200	28 (+73)	75.79%
	✗	200	14 (+73)	76.11%
UP-PVTv2-B0	✔	100	21 (+106)	75.30%
	✗	100	15 (+106)	74.33%
DeiT-T	✗	500	35	72.65%
DeiT-T	✔	200	28 (+73)	72.14%
DeiT-T (pre-trained)	✔	200	28 (+94)	75.45%
PVTv2-B0	✔	300	63 (+106)	75.08%

or based on their ℓ_1-norm values, respectively. When compared to those methods, our UP-Transformer model outperforms them. UP-Transformer also achieves comparable performance to the original Transformer model, especially when training iterations are increased and knowledge distillation is used.

Appendix D.5 Analyses

To explore the impact of different modules of our method, we performed five analyses in this section.

Knowledge distillation. We then evaluate the influence of knowledge distillation when fine-tuning UP-ViTs. For a fair comparison, we adopt the same training strategies as Appendix D.2 and Appendix D.3. We further elongate the training process for random initialization, without distillation for DeiT-T and with PVTv2-B2 distillation for PVTv2-B0. We also fine-tune the randomly initialized and pre-trained DeiT-T with the distillation of DeiT-B for 200 epochs.

We illustrate the results in Table D8, which shows that our method still obtains competitive accuracy when training without knowledge distillation, and UP-DeiT-T even performs better without distillation. We also report the time consumed during the entire training process on eight 3090 GPUs. The content in parentheses represents the required training time of the original model. For example, if we compress DeiT-B into UP-DeiT-T, the content in parentheses represents the time required to train the original DeiT-B model first. In particular, when we fine-tune the pre-trained DeiT-T by the distillation of DeiT-B, the contents of the brackets contain all the time for both pre-training DeiT-B and DeiT-T.

It is worth noting that our method includes the calculation of importance scores, so the final full training time is still slightly longer than distilling the randomly initialized DeiT-T directly by the DeiT-B's distillation. Nevertheless, we can still conclude that compared with directly distilling the small model with the large model, it is easier to achieve higher accuracy by first compressing the large model into the small model and then distilling the small model.

We also compare several different distillation strategies. Besides soft distillation, we consider the following two methods.

• Hard distillation. Let $y_t = \arg\max(q)$ be the hard decision of the teacher, where q is the teacher's output probabilities. The loss function of hard-label distillation is

$$\mathcal{L}_{\mathrm{CE}}\left(p, y\right) + \alpha \mathcal{L}_{\mathrm{CE}}\left(p, y_t\right) . \tag{D1}$$

• Soft + Patch distillation. Inspired by LV-ViT [9], we distill both the classification token and the patch tokens. Assume the student's output patch tokens are $t_s^1, \ldots, t_s^N$ and the corresponding outputs of the teacher are $t_t^1, \ldots, t_t^N$. Note that the number of embedding dimensions in t_s^i and t_t^i are different, so we introduce a new linear projection FC_{token} into the loss. The soft + patch distillation objective is:

$$\mathcal{L}_{\mathrm{CE}}\left(p, y\right) + \alpha \mathcal{L}_{KL}(p, q) + \beta \sum_{i=1}^{N} \mathcal{L}_{\mathrm{MSE}}\left(\mathrm{FC}_{token}(t_s^i), t_t^i\right) . \tag{D2}$$

Based on the three strategies, we investigate the performance of distilling UP-DeiT-T with DeiT-B. During fine-tuning, we set α of hard distillation as 2.0. When applying soft + patch distillation, we set α and β as 0.2 and 1e-3, respectively. We sampled the original ImageNet-1k to a smaller subset with one-tenth of the total training images, which is named SImageNet-1k. We trained the models in SImageNet-1k. As the distillation results in Table D9 clearly show, soft distillation achieves the highest accuracy.

Pruning multi-head self-attention. We first evaluate the influence of our attention pruning strategy when pruning component 2 by conducting an experiment of only pruning MHSA from DeiT-B. In particular, DeiT-B contains 768 dimensions with 12 heads in its attention layer, and DeiT-T has 192 dimensions with only 3 heads. After calculating the attention importance score of DeiT-B, we compare three strategies of pruning the component 2:

Table D9 Results on the ImageNet-1k validation set with 3 distillation strategies.

Distillation	Soft	Hard	Soft + Patch
Accuracy	**51.51%**	48.95%	51.49%

Table D10 Results on the ImageNet-1k validation set with different MHSA pruning strategies.

	Strategy 1	Strategy 2	Strategy 3
Accuracy	**74.54%**	74.36%	73.44%

Figure D1 Pipeline of progressively pruning DeiT-B into DeiT-T. We use pruning one block as example. Attn. means the attention layer and the corresponding FFN is the feed-forward network.

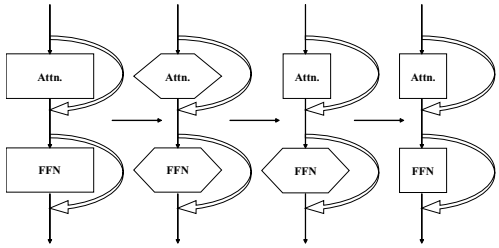

Table D11 Results of the progressive pipeline when pruning to DeiT-T on ImageNet-1k. In every step we fine-tuned model with 50 epochs. After the fourth step we prune DeiT-B into DeiT-S.

Model	Acc.	Throughput	#Param.
DeiT-B	81.84%	58.1	86.57M
Step 1	79.90%	139.4	21.69M
+Step 2	79.60%	188.2	16.36M
+Step 3	75.05%	296.7	5.72M
+Step 4	74.51%	404.5	5.72M

• Strategy 1: Our original strategy of pruning the attention layer. For the 12 heads, we remove 192 dimensions from every four heads.

• Strategy 2: Do not consider the structure difference and delete 576 dimensions across all heads;

• Strategy 3: For the 12 heads, we remove 48 dimensions for each attention head;

After deleting those redundant dimensions in component 2, we merged the remaining filters into 3 heads and keep other components unchanging. Then we extracted three different sub-models and fine-tuned them 50 epochs on SImageNet-1k. The other training settings were the same as those Appendix D.2 and we tested on the ImageNet-1k validation set. We show the results in Table D10, and strategy 1 achieves the highest accuracy.

Progressive vs. One-time pruning width. We then investigate the performance of progressively pruning width. UP-ViT is to reduce the channels by estimating the importance scores of channels. Each step is to select one channel to remove and we do not consider the correlation between different channels. However, selecting a set of channels to remove is a hard combinatorial optimization problem. In this section, we analyze the effect of progressive calculating the importance and pruning width. In particular, we devises two sets of comparative experiments.

First, we compare two pipelines during compressing DeiT-B. In the first pipeline, after calculating all importance scores, we remove all redundant channels at once and fine-tune the model. In the second pipeline, we divide the whole pruning process into 4 steps:

• Step 1: Calculate the importance scores of component 1 in DeiT-B. There are 768 channels and we throw away 576 of them and fine-tune the sub-model.

• Step 2: Continue to calculate the importance scores of component 2 per block and prune it. In particular, we throw away 48 dimensions for every attention head and keep 12 attention heads, then fine-tune the sub-model.

• Step 3: Similar to the second step, we prune component 3 and fine-tune the model. Especially, the dimensionalities of the FFN hidden layer are reduced from 3072 to 768.

• Step 4: Adjust the attention heads from 12 to 3 and fine-tune the model.

We demonstrate the first three steps of progressively pruning in Figure D1. Note that during fine-tuning, we apply knowledge distillation and the teacher model is always DeiT-B. The results are shown in Table D11. We can observe that the performance of progressive pruning (74.51%) is not as good as pruning at once (75.79%).

We also compare two pipelines of compressing the component 1 of DeiT-B. In the first, we remove 576 redundant filters at once after calculating the importance of all filters in component 1. In the second pipeline, we construct a loop that only throws away 192 filters in component 1 every time, and repeat this loop three times. Then we fine-tune the two sub-models.

The results are shown in Table D12. We fine-tuned the sub-models on SImageNet-1k and tested on the original ImageNet-1k validation set. We notice the first pipeline clearly works better, which indicates pruning all redundant filters at once is most effective when compressing the width of ViTs.

Progressive vs. One-time pruning depth. We then demonstrate the effectiveness of progressively pruning blocks. Based on the compressed UP-DeiT-T, we compare two strategies for generating UP-DeiT-T-10. In the first we directly remove the two least unimportant blocks at one time, and in the second we use our progressive block pruning strategy. Once we get the two sub-model candidates, we apply the same training policy to fine-tune model on ImageNet-1k and show the results in Table D13. The progressive block pruning strategy obtains higher accuracy.

The proxy dataset size. We then further illustrate the influence of the number of training samples in proxy dataset $\mathcal{D}$. We adopt the same training strategies as the previous section. We include Table D14 to further illustrate the influence of proxy dataset $\mathcal{D}$'s size. In particular, we also set the size of $\mathcal{D}$ as 0 and randomly select part of channels to prune.

Table D14 shows that with 5k images UP-DeiT will achieve better accuracy. However, it will bring more computation burden when calculating importance, so we sampled 2k images as the proxy dataset. In particular, when the size of $\mathcal{D}$ is greater than 1k, our method consistently outperforms random pruning.

Appendix E Limitations

Although our method can achieve high accuracy and can be applied to various Transformer architectures, we still have some shortcomings. First, our method takes a long time to fine-tune, and we also need to calculate the importance scores of all channels, which is time-consuming when the size of proxy dataset $\mathcal{D}$ is relatively large. Then, we have to manually distinguish the uncorrelated components existing in ViTs, so how to automatically divide and prune the irrelevant components is an interesting future direction. Besides this, we focus on pruning the heavy model into the existing lightweight model, such as compressing DeiT-B into DeiT-T. Therefore, continuing to prune ViTs to make them more attractive for embedded systems is another interesting direction.

Table D12 Results of two pipelines when pruning component 1 on SImageNet-1k. We fine-tuned models with 50 epochs and knowledge distillation.

Component 1	At once	Progressively
Accuracy	**66.95%**	65.05%

Table D13 Results of two pipelines when pruning UP-DeiT-T into UP-DeiT-T-10 on ImageNet-1k. We fine-tune models with 30 epochs and knowledge distillation.

Blocks	At once	Progressively
Accuracy	73.20%	**73.78%**

Table D14 Results on the ImageNet-1k validation set with different number of training samples in the proxy dataset.

Samples	0	0.5k	1k	2k	5k
Accuracy	74.09%	73.65%	74.62%	75.79%	**75.94%**

References

1 Baevski, A., Auli, M.: Adaptive input representations for neural language modeling. In: International Conference on Learning Representations (ICLR) (2018)
2 Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., , Vedaldi, A.: Describing textures in the wild. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3606–3613 (2014)
3 Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: RandAugment: Practical automated data augmentation with a reduced search space. In: The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 702–703 (2020)
4 Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 248–255 (2009)
5 Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (ICLR) (2021)
6 Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. International Journal of Computer Vision **88**(2), 303–338 (2010)
7 He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2961–2969 (2017)
8 Jia, D., Han, K., Wang, Y., Tang, Y., Guo, J., Zhang, C., Tao, D.: Efficient vision transformers via fine-grained manifold distillation. arXiv preprint arXiv:2107.01378 (2021)
9 Jiang, Z., Hou, Q., Yuan, L., Zhou, D., Shi, Y., Jin, X., Wang, A., Feng, J.: All tokens matter: Token labeling for training better vision transformers. arXiv preprint arXiv:2104.10858 (2021)
10 Jonathan, K., Michael, S., Jia, D., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: International IEEE Workshop on 3D Representation and Recognition. pp. 554–561 (2013)
11 Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
12 Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. In: International Conference on Learning Representations (ICLR) (2017)
13 Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2980–2988 (2017)
14 Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: The European Conference on Computer Vision (ECCV), LNCS, vol. 8693, pp. 740–755. Springer (2014)
15 Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin Transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10012–10022 (2021)
16 Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (ICLR) (2018)
17 Maji, S., Kannala, J., Rahtu, E., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013)
18 Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models. In: International Conference on Learning Representations (ICLR) (2017)
19 Ott, M., Auli, M., Grangier, D., Ranzato, M.A.: Analyzing uncertainty in neural machine translation. In: International Conference on Machine Learning. pp. 3956–3965 (2018)
20 Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Cats and dogs. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3498–3505 (2012)
21 Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 413–420 (2009)
22 Tang, Y., Han, K., Wang, Y., Xu, C., Guo, J., Xu, C., Tao, D.: Patch slimming for efficient vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12165–12174 (2022)
23 Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jegou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. pp. 10347–10357 (2021)
24 Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., Belongie, S.: The inaturalist species classification and detection dataset. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8769–8778 (2018)
25 Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011)
26 Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 568–578 (2021)
27 Xu, Y., Zhang, Z., Zhang, M., Sheng, K., Li, K., Dong, W., Zhang, L., Xu, C., Sun, X.: Evo-ViT: Slow-Fast token evolution for dynamic vision transformer. arXiv preprint arXiv:2108.01390 (2021)

28 Yang, H., Yin, H., Molchanov, P., Li, H., Kautz, J.: NViT: Vision transformer compression and parameter redistribution. arXiv preprint arXiv:2110.04869 (2021)
29 Yu, S., Chen, T., Shen, J., Yuan, H., Tan, J., Yang, S., Liu, J., Wang, Z.: Unified visual transformer compression. In: International Conference on Learning Representations (ICLR) (2021)
30 Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.H., Tay, F.E., Feng, J., Yan, S.: Tokens-to-Token ViT: Training vision transformers from scratch on imagenet. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 558–567 (2021)
31 Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6023–6032 (2019)
32 Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: Mixup: Beyond Empirical Risk Minimization. In: International Conference on Learning Representations (ICLR) (2018)
33 Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 13001–13008 (2020)
34 Zhu, M., Han, K., Tang, Y., Wang, Y.: Visual transformer pruning. In: KDD 2021 Workshop on Model Mining (2021)