Vision Transformer Slimming: Multi-Dimension Searching in Continuous Optimization Space

Arnav Chavan\textsuperscript{*1,3}, Zhiqiang Shen\textsuperscript{*2,3}, Zhuang Liu\textsuperscript{4}, Zechun Liu\textsuperscript{5}, Kwang-Ting Cheng\textsuperscript{6} and Eric Xing\textsuperscript{2,3}

\textsuperscript{1}IIT Dhanbad \hspace{1cm} \textsuperscript{2}CMU \hspace{1cm} \textsuperscript{3}MBZUAI \hspace{1cm} \textsuperscript{4}UC Berkeley \hspace{1cm} \textsuperscript{5}Reality Labs, Meta Inc. \hspace{1cm} \textsuperscript{6}HKUST

arnav.18je0156@am.iitism.ac.in {zhiqians,zechunl}@andrew.cmu.edu zhuangl@berkeley.edu

timcheng@ust.hk epxing@cs.cmu.edu

Abstract

This paper explores the feasibility of finding an optimal sub-model from a vision transformer and introduces a pure vision transformer slimming (ViT-Slim) framework. It can search a sub-structure from the original model end-to-end across multiple dimensions, including the input tokens, MHSA and MLP modules with state-of-the-art performance. Our method is based on a learnable and unified \(\ell_1\) sparsity constraint with pre-defined factors to reflect the global importance in the continuous searching space of different dimensions. The searching process is highly efficient through a single-shot training scheme. For instance, on DeiT-S, ViT-Slim only takes \(\sim 43\) GPU hours for the searching process, and the searched structure is flexible with diverse dimensionality in different modules. Then, a budget threshold is employed according to the requirements of accuracy-FLOPs trade-off on running devices, and a retraining process is performed to obtain the final model. The extensive experiments show that our ViT-Slim can compress up to 40\% of parameters and 40\% FLOPs on various vision transformers while increasing the accuracy by \(\sim 0.6\%\) on ImageNet. We also demonstrate the advantage of our searched models on several downstream datasets. Our code is available at https://github.com/Arnav0400/ViT-Slim.

1. Introduction

Transformer \cite{vaswani2017attention} has been a strong network model for various vision tasks, such as image classification \cite{he2016deep, resnet, resnetv2, densenet, EfficientNet}, object detection \cite{ssd, fasterrcnn, yolo}, segmentation \cite{U-Net, maskrcnn, FCN}, etc. It is primarily composed of three underlying modules: Multi-Head Self-Attention (MHSA), Multi-Layer Perceptron (MLP) and Image Patching Mechanism. The major limitation of ViT in practice is the enormous model sizes and excessive training and inference costs, which hinders its broader usage in real-world applications. Thus, a large body of recent studies has focused on compressing the vision transformer network through searching a stronger and more efficient architecture \cite{guzik2021saformer, dft, prp, rnet} for better deployment on various hardware devices.

However, many commonly-used searching strategies are recourse-consuming, such as the popular reinforcement learning and evolutionary searching approaches. Single-path one-shot (SPOS) \cite{pse} is an efficient searching strategy and promising solution for this task, while it still needs to train the supernet for hundreds of epochs and then evaluate thousands of subnets for finding out the optimal sub-architecture, which is still time-consuming, typically with more than tens of GPU days.

Recently, there are some works utilizing Batch Normalization (BN) scaling parameter as the indicator for the importance of operations to prune or search subnets, such as Network Slimming \cite{netslim}, SCP \cite{scp}, BN-NAS \cite{bn-nas}, etc., since BN’s parameter is an existing factor and light-weight measure metric in the network for the importance of subnets. This searching method can offer \(10\times\) speedup of training than SPOS in general. But in practice, not all networks contain BN layers, such as the transformers. Also, there are many unique properties in a transformer design like the dependency of input tokens from shallow layers to deep layers. Simply utilizing such a strategy is not necessarily practical or optimal for the newer transformer models.

Consequently, the main remaining problem is that there is no BN layer involved in conventional transformer architectures, so we cannot directly employ the scaling coefficient in BN as the indicator for searching. To address this, in this work we propose to incorporate the explicitly soft masks to indicate the global importance of dimensions across different modules of a transformer. We consider jointly searching on all three dimensions in a transformer end-to-end, including: layerwise tokens/patches, MHSA and MLP dimensions. In particular, we design additional differentiable soft masks on different modules for the in-
individual dimension of a transformer, and the $\ell_1$-sparsity is also imposed to force the masks to be sparse during searching. We only need a few epochs to finetune these mask parameters (they are initialized to 1 so as to give equal importance to all dimensions at the beginning of search) together with the original model’s parameters for completing the whole searching process, which is extremely efficient. For the token search part, we apply a $tanh$ over masks to avoid exploding mask values which is observed empirically.

We call our method ViT-Slim, a joint sparse-mask based searching method with an implicit weight sharing mechanism for searching a better sub-transformer network. This is a more general and flexible design than previous BN-based approaches since we have no requirement of BN layers in the networks. This is more friendly to transformers and a feature-by-feature comparison with other ViT compression methods is shown in Table 1. One core advantage of our method is the efficiency of searching, we can inherit the pretrained parameters and conduct a fast search upon it. Another advantage is the zero-cost subnet selection and high flexibility. In contrast to the SPOS searching that requires to evaluate thousands of subnets on validation data, once we complete the searching process, we can obtain countless subnetworks and the final structure can be determined by the requirement of accuracy-FLOPs trade-off at the real devices that we deploy our model on, without any extra evaluation. The last advantage is that we can search for a finer-grained architecture such as the different dimensionalities in different self-attention heads, as our search space is continuous in them. This characteristic allows us to find the architecture with unique individual dimensions and shapes in different layers and modules, which would always find out a better subnet than other counterparts.

Comprehensive experiments and ablation studies are conducted on ImageNet [13], which show that ViT-Slim can compress up to 40% of parameters and 40% FLOPs on various vision transformers like DeiT [47], Swin [28] without any compromising accuracy (in some circumstances our compressed model is even better than the original one). We also demonstrate the advantage of our searched models on several downstream datasets.

Our main contributions are:

- We introduce ViT-Slim, a framework that can jointly perform an efficient architecture search over all three modules - MHSA, MLP and Patching Mechanism in vision transformers. We stress that our method searches for structured architectures which can bring practical efficiency on modern hardware (e.g., GPUs).
- We empirically explore various structured slimming strategies by sharing weights in candidate structures, and provide the best performing structure by employing a continuous search space in contrast to a pre-defined discrete search space in existing works.
- Our method can perform directly over pre-trained transformers by employing a single shot searching mechanism for all possible budgets, eliminating the need of search-specific pre-training of large models and performing repeated searching for different modules/budgets.
- We achieve state-of-the-art performance at different budgets on ImageNet across a variety of ViT compression and search variants. Our proposed ViT-Slim can compress up to 40% of parameters and 40% FLOPs while increasing the accuracy by $\sim 0.6\%$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Target Arch</th>
<th>Searching Space</th>
<th>Searching Method</th>
<th>Searching Time</th>
<th>Inherit Pre-train</th>
<th>Reduce Params &amp; FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT-Slim (Ours)</td>
<td>ViT/DeiT/Swin, etc.</td>
<td>Continuous, all modules</td>
<td>One-shot w/ $\ell_1$-sparsity</td>
<td>43 g-hrs (50-ep)</td>
<td>Yes</td>
<td>Both</td>
</tr>
<tr>
<td>Dynamic-VIT [34]</td>
<td>ViT/DeiT family</td>
<td>Dynamic Patch Sel.</td>
<td>Layerwise Prediction module</td>
<td>26 g-hrs (30-ep)</td>
<td>Yes</td>
<td>FLOPs only</td>
</tr>
<tr>
<td>Patch-Slimming [45]</td>
<td>ViT/DeiT family</td>
<td>Layerwise Patch Sel.</td>
<td>Top-Down layerwise</td>
<td>31 g-hrs (36-ep)</td>
<td>Yes</td>
<td>FLOPs only</td>
</tr>
<tr>
<td>AutoFormer [7]</td>
<td>Customized ViT</td>
<td>Discrete, Pre-defined</td>
<td>Evolutionary</td>
<td>300-ep</td>
<td>No</td>
<td>Both</td>
</tr>
<tr>
<td>Customized ViT</td>
<td>Discrete, Pre-defined</td>
<td>Evolutionary</td>
<td>500-ep</td>
<td>No</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>ViTAS [12]</td>
<td>Customized ViT</td>
<td>Discrete, Pre-defined</td>
<td>Evolutionary</td>
<td>300-ep</td>
<td>No</td>
<td>Both</td>
</tr>
</tbody>
</table>
| Efficient Models and Architecture Search | Neural network compression has been recognized as an important technology for applying deep neural network models to equipment with limited resources. The compression research extends from channel pruning [17, 27, 56], quantization and binarization [1, 9, 21, 29, 33, 63], knowledge distillation [18, 32, 36, 39–41], compact neural network design [10, 22, 31, 52, 60] to architecture search [35, 65, 66]. Specifically, MobileNets [19, 37] proposed to decompose the convolution filters to depth-wise convolution and point-wise convolution for reducing the parameters in convolutional neural networks (CNNs). EfficientNet [44] proposed to search the uniform scaling ratio among all dimensions of depth/width/resolution to achieve much better accuracy and efficiency. Network Slimming [27] used the BN parameters as the scaling factors to find the best sub-structure. JointPruning [30] jointly searched the layer-wise channel number choices together with the depth and resolution for finer-grained compression. NetAdapt [55] and AMC [16] adopted the feedback loop or the reinforcement learning method to search the channel numbers for the CNNs. Besides, many neural architecture search (NAS) methods are targeting at exploring the operation choices (e.g., $3 \times 3$, $5 \times 5$, $7 \times 7$ convolutions) for architectures. For instance, SPOS [15] built a supernet that contained all the possible choices and used the evolutionary search for a subnetwork.
in the supernet. DARTS [26], FB-Net [51] and Proxyless-NAS [2] used gradient-based method to update the mask associating with each operation choices. However, these NAS methods defined on the discrete operation search spaces can hardly be generalized to tackle the problem of continuous channel number search.

**Efficient Vision Transformers.** There are several works exploring on this direction [6–8, 34, 42]. Patch-Slimming [45] explored the direction to improve the efficiency of transformers by sequentially pruning patches from top-to-bottom layers. Similarly, Dynamic-ViT [34] explored dynamic patch selection based on the input patches. They employed multiple hierarchical prediction modules to estimate the importance score of each patch. However, patch pruning did not improve parameter efficiency. ViTAS [42] used the evolutionary algorithm to search optimal architecture at a target budget. However, their search space was discrete, pre-defined and thus limited. GLIT [6] introduced a locality module to model local features along with the global features. But their method used CNNs along with the attention and performed an evolutionary search over global and local modules. BigNAS [57] introduced the single-stage method to generate highly efficient child models by slicing weight matrices. Based on this, AutoFormer [7] showed that weight entanglement was a better alternative than defining weight matrices for every possible sub module in architecture search and using an evolutionary algorithm to search for optimal sub-networks. But all of them had a limited discrete search space as [42] due to the adaptation of an evolutionary algorithm for searching. S²ViTE [8] provided an end-to-end sparsity exploration for vision transformers with an iterative pruning and growing strategy. Their structured pruning method eliminated complete attention heads and MLP neurons based on a scoring function calculated using the Taylor expansion for the loss function and $\ell_1$-norm, respectively. We argue that eliminating complete attention heads is a sub-optimal choice and limits the learning dynamics of a transformer. Allowing the model to determine optimal dimensions for every attention head (instead of eliminating complete heads) is a better alternative for pruning MHSA module.

3. Proposed Approach

3.1. Overview and Motivation

In this section, we start by working around some important questions raised by existing works, viz:

- Can we take advantage of the existence of only fully connected layers in transformers and make the search space continuous and hence much larger than existing works without larger memory or compute overhead?
- What is the optimal structure configuration that has to be searched in ViT family, and can joint search of architec-

```
Figure 1. An overview of our proposed ViT-Slim framework. $D'$ and $M'$ are the dimensions after slimming operation.
```

3.2. Achieving A Continuous Search Space

One-shot NAS methods for CNNs [11, 15, 44] explicitly define multiple decoupled candidate blocks for every layer for training a central supernet. This strategy is suitable for CNNs as the candidate blocks at every layer come from a wide variety of sub-architectures so as to maintain the property of sampling diverse subnets from the supernet while searching. This is not true for transformers, as they are internally composed of multiple fully connected layers stacked in different configurations for different modules. The fact that the core blocks are all composed of fully connected layers opens the possibility to expand the search space by sharing weights between the candidate fully connected layers inside any block.

Consider a fully connected layer which has to be searched for optimal output dimension given an input dimension $D_{in}$, conventional method is to define multiple candidate layers with output dimensions from pre-defined search space and search the optimal layer from them with a suitable searching algorithm. However, there are a couple of disadvantages - 1) Longer search time as every candidate layer needs to be at least partially trained for searching 2) Larger memory footprint contributed by weight matrices of every candidate layer. We propose to solve these issues by weight sharing between all possible candidate layers. We fix the maximum permissible output dimension to $D_{max}$.
and define a super weight matrix $W_{sup} \in D_{in} \times D_{max}$. The candidate layer weights can be easily sliced from $W_{sup}$.

To achieve a continuous search space, we adopt a single-stage method to rank the importance of every dimension of supernet weights, taking inspiration from [27] we use $l_1$-Sparsity to achieve it. We first pre-train the supernet until convergence (in practice, our method would work directly over pre-trained networks eliminating the need of supernet training). We then define masks corresponding to every dimension that has to be searched. The magnitude of mask values correspond to the importance score of respective dimension and thus we initialize all of them by 1. Once the pretrained weights are loaded we induce these masks into the model. Considering the previous example of a single fully-connected layer where the output dimension is to be searched of a layer with weight matrix $W_{sup}$, a mask $z \in D_{max}$ is defined. A dot product between $W_{sup}$ and $z$ gives the candidate weight matrix to be used in forwarding propagation. The search algorithm employs a loss function which is a combination of pre-training loss function (Cross-Entropy in classification tasks) and $l_1$-norm on masks. This combined loss drives the mask values towards 0 while minimizing the target loss function. In a way the optimization landscape implicitly drives the masks to rank themselves according to their impact on final performance.

### 3.3. Identifying Optimal Search Space

The fundamental question in NAS is defining a search space. This section presents our defined search space.

**Flexible MHSA and MLP Modules.** Recent works have followed two methodologies for defining the search space for MHSA module - 1) Searching for the number of heads in every distinct MHSA module [8, 25] and/or 2) Searching for a common feature dimension size from a pre-defined discrete sample space for all attention heads in any particular MHSA module [6, 7, 42]. These methods have shown some solid results but they are not completely flexible. A greater degree of flexibility can be achieved if every attention head can have a distinct feature dimension size. As-suming a super-transformer network with a maximum of $L$ MHSA modules each with a maximum permissible number of heads set to $H$. This gives a total of $L \times H$ unique attention heads. If we fix the maximum permissible feature dimension size to $d$, the size of the equivalent search space is equal to $(d + 1)^{L \times H}$. Searching in such a massive search space is computationally very difficult. However, such a diverse search space has a couple of advantages: 1) The search algorithm can be more flexible in adapting the super-transformer to smaller architectures while maintaining the performance; and 2) The extracted architectures would be much more efficient as the search algorithm can push the feature dimensions of least important attention heads to even zero, decreasing the FLOPs substantially. We use the complete $(d + 1)^{L \times H}$ search space in our method.

Similarly, for the MLP, there are a total of $L$ modules throughout the network with existing works using discrete and limited search space to search the optimal feature dimension. If we fix the maximum permissible feature dimension size to $M$, the search space which can be explored by our method is equal to $(M + 1)^L$. Combining MHSA and MLP together in a single search mechanism is quite straightforward. Thus, it will generate a much more diverse set of child networks compared to all the existing works.

**Patch Selection.** Patch slimming [45] showed that MHSA aggregates patches and consequently cosine-similarity between the patches increases exponentially layer-wise becoming as large as 0.9 in the final layers. This opens the possibility to eliminate a large number of deeper patches and a few unimportant shallow patches too. Inducing patch-selection with MHSA and MLP search can extract more efficient architectures with reduced FLOPs at the same parameter count. While intuitively, a dynamic way is more promising on the patch dimension as the selected patches should be aligned with the input images to reflect the importance of different regions in the image.

### 3.4. Single Shot Arch. Search with $l_1$-Sparsity

The main objective of our search method is to rank the mask values according to their impact on final performance. Once ranked, we eliminate the dimensions having the least mask values. Let $f: \mathbb{R}^x \rightarrow \mathbb{R}^y$ denote a vision transformer network which learns to transform input $x$ into target outputs $y$ with weights $W$ and intermediate activations/tensors $T \in \mathbb{R}^d$ constructed from weights $W$ and input $x$. We define $z \in \mathbb{R}^d$ as a set of sparsity masks, where $z_i \in z$ refers to the mask associated with the intermediate tensor $t_i \in T$. To apply the mask, $z_i$ is multiplied with corresponding entries of $t_i$. The optimization objective can be stated as:

$$\min_{W, z} \mathcal{L}_{CE}(f(z \odot T(W, x)), y) + ||z||_1, \quad (1)$$

We introduce uniform masking to search optimal dimension size for each distinct head in MHSA modules, dimension size for each MLP module and layewise most important patches. Consider a transformer network with $L$ layers of MHSA+MLP blocks and each block consisting of $H$ self-attention heads. The input tensor to each MHSA layer $t_{a_l} \in \mathbb{R}^{N \times D}$ where $N$ is number of patches and $D$ is global feature dimension size. Inside every head $i$ of MHSA module, $t_{a_l}$ is transformed with fully connected layers to $q_i \in \mathbb{R}^{N \times d}$, $k_i \in \mathbb{R}^{N \times d}$ and $v_i \in \mathbb{R}^{N \times d}$, $d$ denoting feature dimension of each self-attention head. We define mask $z_{a_l} \in \mathbb{R}^{L \times H}$, and corresponding vectors in $z_{a_l}$, $z_{a_{l,h}} \in \mathbb{R}^d$ corresponding to $l^{th}$ layer and $h^{th}$ head. The total possibilities for MHSA modules across the network that can be explored by our method is thus $(d + 1)^{L \times H}$. The computa-
tions inside MHSA module with sparsity masks is:

\[ A_t = \text{softmax}(\langle q_t \odot z_{a_l,h} \rangle \times \langle k_t \odot z_{a_l,h} \rangle^T \sqrt{d}) \] (2)

\[ O_{l,t} = A_t \times \langle v_t \odot z_{a_l,h} \rangle \] (3)

\[ t_{m_l} = \text{projection}([O_1, O_{l_1}, ..., O_{H_l}]) \] (4)

where \( t_{m_l} \in \mathbb{R}^{N \times D} \) is the output from MHSA block which in turn becomes the input to MLP block. Inside MLP block, \( t_{m_l} \) is projected to a higher-dimensional space through a fully connected layer \( f_1 \) to form an intermediate tensor \( t_{e_l} \in \mathbb{R}^{N \times M} \) which is again projected back to \( R^{N \times D} \) through another fully connected layer \( f_2 \). We define mask \( z_m \in \mathbb{R}^L \), and corresponding vectors in \( z_m \), \( z_{m_l} \in \mathbb{R}^M \) corresponding to \( l^{th} \) layer. The total possibilities for MLP modules across the network is thus \((M+1)^L\) by our method. The following computation shows the interaction of masks with MLP module:

\[ t_{e_l} = f_1(t_{m_l}) \odot z_{m_{l+1}}, t_{a_{l+1}} = f_2(t_{e_l}) \] (5)

Solving Patch Dependency Across Layers in Patch Dim.

For patch selection, we define a distinct mask value corresponding to each patch in every layer and eliminate the patches corresponding to a lower mask value. A problem arises due to the global single-shot search that there may be anomalous instances where the same patch is eliminated in a shallower layer but exits in the deeper layer. In practice, such anomalous instances are limited indicating that \( \ell_1 \)-Sparsity based search strategy aligns the patches as per their importance and a shallow eliminated patch implicitly forces it’s deeper counterpart to have less importance and consequently eliminates it. To counter these limited anomalous patches, once a patch is eliminated from an earlier layer, we eliminate it from further layers too, while imposing budget. Additionally, we apply a \( \text{tanh}^{1} \) activation function over patch-specific masks before taking a dot product with the patches to stop mask values from exploding.

3.4.1 Searching Time Analysis

Our method directly works over pre-trained models, eliminating the need of training a search-specific model. We induce sparsity masks on a pre-trained model and jointly optimize the masks and model weights with a combination of CE loss and \( \ell_1 \)-norm of masks. In our setup, we fix the search epochs to 50 for all searches. This translates to \( \sim 43 \) GPU hrs for DeiT-S and \( \sim 71 \) GPU hrs for DeiT-B. At the end of search, masks are ranked according to their values.

3.4.2 Retraining with Implicit Budget Imposition

Once ranked, depending upon the target budget, low-rank dimensions/patches are eliminated from the network. For MHSA+MLP joint search, budget, search人权 count space translates to FLOPs and Params budget too due to the linear relation between the number of dimensions and FLOPs/Params, and inducing patch selection further decreases FLOPs. Once the budget-specific structure is extracted, it is retrained with exactly the same setup as the pre-trained model. This allows the weights to adjust from continuous masks in searching space to binary/non-existing masks in the final structure.

4. Experiments

In this section, we first explore the contribution of each individual component in the final performance of the ViT [14]/DeiT [46] model and search for the optimal uni-dimensional searched model. We then move to joint search combining all three components and show that our method outperforms all existing architecture search and pruning methods. We also show the applicability of our method to other transformer architectures, such as Swin [28]. Finally, we further show the performance of searched models on the downstream datasets in a transfer learning setup.

4.1. Training Procedures and Settings

There are three steps in our workflow for the ViT-Slim framework, including:

(i) One-shot Searching. We use pre-trained weights to initialize existing vision transformer models and use them as our supernet. We then induce sparsity masks into the model depending upon the dimension to search and jointly train the weights and masks with the loss function given in Equation 1 for 50 epochs with a constant learning rate of 5e-04 and AdamW optimizer with a weight decay of 1e-03. The batch size is set to 1024 for DeiT-S and 512 for DeiT-B. We also employ stochastic depth [20], cutmix [58], mixup [59], randaugment [12], random erasing [62], etc., as augmentations following DeiT while searching.

(ii) Budget Selection. Once the search is complete, the masks are ranked according to their values after the searching step. Depending upon the target budget of compression, low-rank dimensions are eliminated from the supernet to extract the final structure of searched model.

(iii) Re-training. Finally, we retrain the extracted compressed structure for 300 epochs with the same setting on which it was originally pre-trained in [28, 46].

4.2. Unidimensional Search

To show the impact of MHSA and MLP modules independently in the final model’s performance, we search for optimal dimensions in both of them separately. We induce sparsity masks in respective modules of DeiT-S and search two supernets each with a sparsity weight of 1e-04 for MLP and MHSA dimensions. The total number of masks induced in MLP is \( 4 \times \) that of MHSA due to the existence of \( 4 \times \) more dimensions in MLP module. The post-search accuracy and final accuracy at different budgets are shown in Ta-
Table 2. Performance of DeiT-S [47] for MHSA and MLP dimension search. Budget indicates the % of active dimensions of respective search modules across the network.

<table>
<thead>
<tr>
<th>W1</th>
<th>W2</th>
<th>Post Search Accuracy</th>
<th>Final Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2e-04</td>
<td>1e-04</td>
<td>76.54</td>
<td>93.37</td>
</tr>
<tr>
<td>3e-04</td>
<td>1e-04</td>
<td>76.34</td>
<td>93.36</td>
</tr>
<tr>
<td>4e-04</td>
<td>1e-04</td>
<td>76.40</td>
<td>93.39</td>
</tr>
<tr>
<td>2e-04</td>
<td>4e-05</td>
<td>76.69</td>
<td>93.61</td>
</tr>
<tr>
<td>2e-04</td>
<td>5e-05</td>
<td>76.68</td>
<td>93.64</td>
</tr>
</tbody>
</table>

Table 3. W1 (MHSA sparsity weight) and W2 (MLP sparsity weight) grid search for DeiT-S [47] (ViT-Slim1). Post search indicates post search soft accuracy of supernet. Final accuracy indicates final accuracy of the compressed models at 60% budget.

At higher budgets, compressed models even perform better than pre-trained model giving as much as 1% boost in accuracy. MHSA search performs better than MLP search at the same budgets, but MLP achieves a better degree of parameter and FLOPs compression. MHSA at 40% and MLP at 60% have the same FLOPs but MLP outperforms MHSA. Similarly, MHSA at 40% and MLP at 70% having the same number of parameters, MLP outperforms MHSA by a fair margin of 1.2%. This clearly shows that it is much easier to compress MLP dimensions as compared to MHSA dimensions to achieve the same target FLOPs/Parameters, indicating the importance of MHSA to be greater than MLP.

### 4.3. Partial Combination for Parameter Search

Next, we combine MHSA and MLP in a single supernet. The most important hyperparameter to control is the sparsity weight for each of different modules. Based on the fact that MLP is easy to compress and has 4× more dimensions than MHSA, we expect the optimal sparsity weight to be in a similar ratio. We start with an equal sparsity weight of 1e-04 and do a thorough grid search to achieve the optimal performance as shown in Table 3. We search and retrain at 60% budget for a fair comparison. From the results, it is clear that post-search accuracy directly reflects the final model accuracy. As expected, the optimal sparsity weights 2e-04 and 5e-05 are in the ratio of 4:1.

We retrain the optimal searched model at multiple budgets as shown in Table 4. The performance of the model is intact and even better than the pre-trained model up to a budget of 70% after which the accuracy starts to drop. This translates to better performance with 30% FLOPs and parameter reduction. However, going from 60% to 50% budget shows a drastic drop in performance. We name these budgeted model family as ViT-SlimPS indicating direct parameter search or partial search without patch selection.

### Discussion: Budgeted FLOPs and Parameter Reduction

The budget indicates the number of active dimensions that exist in the final searched model for the respective modules. However, for MHSA and MLP joint search, budget translates to final FLOPs and parameter budget due to a linear relation between them. This can be helpful to deploy large transformer networks as per the target FLOPs/Parameter budget on specific hardware.

### 4.4. Multidimensional Joint Search

Finally, a joint search over all three dimensions - MHSA, MLP and Patch Selection is performed. Once we identify the optimal MHSA and MLP sparsity weights as 2e-04 and 5e-05, respectively, we do a grid search over patch sparsity as shown in Table 5. Optimal patch sparsity weight used in all our experiments is 1e-04. To show the effect of patch selection at different budgets, we fix the MHSA and MLP budget at 70% (as 70% keeps the performance intact as shown in Table 4) and retrain at multiple patch-
Table 7. Performance of DeiT-B [47] (ViT-Slim-B$_{JS}$) and Swin-T [28] for MLP and MHSA joint dimension search. Budget indicates the % of active MHSA and MLP dimensions across the network respectively.

<table>
<thead>
<tr>
<th>Budget</th>
<th>Model</th>
<th>#Params (M)</th>
<th>FLOPs (B)</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>DeiT-B</td>
<td>86.6</td>
<td>17.5</td>
<td>81.8</td>
<td>95.6</td>
</tr>
<tr>
<td>80</td>
<td>PS-ViT-B [45]</td>
<td>86.6</td>
<td>10.5</td>
<td>81.5</td>
<td>-</td>
</tr>
<tr>
<td>60</td>
<td>S$^2$VITE-B [6]</td>
<td>56.8</td>
<td>11.7</td>
<td>82.2</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>GLiT-B [6]</td>
<td>96.1</td>
<td>17.0</td>
<td>82.3</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>AutoFormer-B [7]</td>
<td>54.0</td>
<td>11.0</td>
<td>82.4</td>
<td>95.7</td>
</tr>
<tr>
<td>60</td>
<td>ViT-Slim-B</td>
<td>52.6</td>
<td>10.6</td>
<td>82.4</td>
<td>96.1</td>
</tr>
</tbody>
</table>

Table 8. Comparison with SOTA ViT search and compression methods on ImageNet-1k. “PS” is the partial search or parameter search, i.e., MHSA+MLP dimensions, since Patch Selection will not affect the number of parameters but only help reduce FLOPs in our models. “JS” is our final joint-dimension search.

<table>
<thead>
<tr>
<th>Model</th>
<th>#Params (M)</th>
<th>FLOPs (B)</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeiT-S [47]</td>
<td>22.0</td>
<td>4.6</td>
<td>79.9</td>
<td>95.0</td>
</tr>
<tr>
<td>GLiT-S [6]</td>
<td>24.6</td>
<td>4.4</td>
<td>80.5</td>
<td>-</td>
</tr>
<tr>
<td>DynamicViT-S [34]</td>
<td>22.0</td>
<td>4.0</td>
<td>79.8</td>
<td>-</td>
</tr>
<tr>
<td>ViT-Slim$_{PS}$</td>
<td>17.7</td>
<td>3.7</td>
<td>80.6</td>
<td>95.3</td>
</tr>
<tr>
<td>S$^2$VITE - S [6]</td>
<td>14.6</td>
<td>3.1</td>
<td>79.2</td>
<td>-</td>
</tr>
<tr>
<td>PS-ViT-S [45]</td>
<td>22.0</td>
<td>2.7</td>
<td>79.4</td>
<td>-</td>
</tr>
<tr>
<td>ViTAS - E [12]</td>
<td>12.6</td>
<td>2.7</td>
<td>77.4</td>
<td>93.8</td>
</tr>
<tr>
<td>S$^2$VITE+ - S [5]</td>
<td>14.6</td>
<td>2.7</td>
<td>78.2</td>
<td>-</td>
</tr>
<tr>
<td>ViT-Slim$_{JS}$</td>
<td>15.7</td>
<td>3.1</td>
<td>79.9</td>
<td>95.0</td>
</tr>
<tr>
<td>ViT-Slim$_{JS}$</td>
<td>15.7</td>
<td>28</td>
<td>79.5</td>
<td>94.6</td>
</tr>
</tbody>
</table>

Table 9. Transfer learning accuracy of DeiT-S [47] and ViT-Slim$_{PS}$ on CIFAR-100 (C100), CIFAR-10 (C10) [24], iNaturalist-2018 (iNat-18) and iNaturalist-2019 [48] (iNat-19) datasets. All models are searched and pre-trained on ImageNet.

Table 10. Comparison of Layerwise attention head dimensions at various budgets. Figure 2 shows the MHSA modules across DeiT-S searched model. There are a total of 12 MHSA modules each with 6 attention heads. Numbers inside the grids indicate the dimension size of that particular head. It can be seen that at low budgets, deeper layers have the least dimension sizes. Most of the dimensions are intact in the middle of the network and a moderate reduction in dimensions is done in the beginning of the network. Self-attention mech-
anism is required in the middle and to some extent at the beginning of the network when the patches are distinct and information exchange between them is required.

**Layerwise MLP dimensions at various budgets.** Similarly, Figure 3 shows the MLP module dimensions across DeiT-S searched model at various budgets. The pattern across layers is similar to that of MHSA, where deeper layers have a greater degree of reduced dimensions as compared to the earlier layers. The deeper layers have maximum dimensions removed, while the middle layers have maximum dimensions intact. This is again in agreement with the fact that the majority of features are already learnt in the earlier layers, enabling deeper layers to consist of smaller dimension sizes.

### 5.2. Attention Maps Visualization

We adopt the method presented in [4] which employs Deep Taylor Decomposition to calculate local relevance and then propagates these relevancy scores through the layers to generate a final relevancy map. Class-wise visualisation of randomly chosen ImageNet images is shown in Figure 4. ViT-Slim$_{JS}$ focuses better on class-specific important areas as compared to DeiT-S and thus achieves better performance. This also shows that ViT-Slim$_{JS}$ has better interpretability and thus helps in domains where interpretability and explainability of deep networks are important.

### 6. Conclusion

We have presented ViT-Slim, a flexible and efficient searching strategy for subnet discovery on vision transformers leveraging the model sparsity. The proposed method can jointly search on all three dimensions in a ViT, including: layerwise tokens/patches, MHSA and MLP modules end-to-end. We identified that the global importance factors are crucial, and designed additional differentiable soft masks on different modules to reflect the individual importance of dimension. Moreover, the $\ell_1$-sparsity is imposed to force the masks to be sparse during searching. Extensive experiments are conducted on ImageNet and downstream datasets using a variety of ViT architectures to demonstrate the efficiency and effectiveness of our proposed method.
References


[27] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient
convolutional networks through network slimming. In Proceedings of the IEEE international conference on computer vision, pages 2736–2744, 2017. 1, 2, 4


[53] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph

4940


