# Variance-Based Pruning for Accelerating and Compressing Trained Networks

Uranik Berisha[12], Jens Mehnert[1] and Alexandru Paul Condurache[12]
[1]Automated Driving Research, Robert Bosch GmbH, 70469 Stuttgart, Germany
[2]Institute for Signal Processing, University of Lübeck, 23562 Lübeck, Germany
{Uranik.Berisha,JensEricMarkus.Mehnert,AlexandruPaul.Condurache}@de.bosch.com

## Abstract

*Increasingly expensive training of ever larger models such as Vision Transfomers motivate reusing the vast library of already trained state-of-the-art networks. However, their latency, high computational costs and memory demands pose significant challenges for deployment, especially on resource-constrained hardware. While structured pruning methods can reduce these factors, they often require costly retraining, sometimes for up to hundreds of epochs, or even training from scratch to recover the lost accuracy resulting from the structural modifications. Maintaining the provided performance of trained models after structured pruning and thereby avoiding extensive retraining remains a challenge.*

*To solve this, we introduce Variance-Based Pruning, a simple and structured one-shot pruning technique for efficiently compressing networks, with minimal finetuning. Our approach first gathers activation statistics, which are used to select neurons for pruning. Simultaneously the mean activations are integrated back into the model to preserve a high degree of performance. On ImageNet-1k recognition tasks, we demonstrate that directly after pruning DeiT-Base retains over 70% of its original performance and requires only 10 epochs of fine-tuning to regain 99% of the original accuracy while simultaneously reducing MACs by 35% and model size by 36%, thus speeding up the model by 1.44×.*

## 1. Introduction

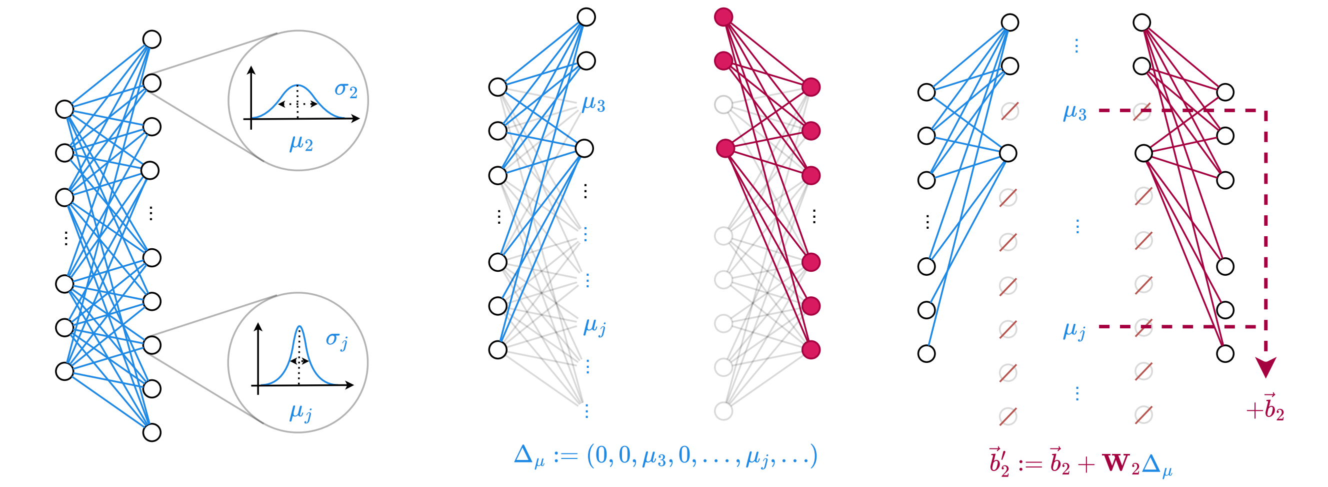
In both Computer Vision (CV) and Natural Language Processing (NLP), Transformers have become the dominant architecture [8, 21]. In CV, Vision Transformers (ViTs) [8] and their derivatives, such as Data-efficient image Transformers (DeiTs) [21], Swin-Transformers (Swins) [14] and ConvNeXts [15], achieve remarkable results but at significant costs in three main areas: training, storage, and inference. These large models require long training times, often spanning hundreds of epochs to achieve their performance, extensive memory to store their parameters, and high computational resources for inference causing high latency [21].

Fixing all three issues simultaneously remains especially challenging. For example, using the available collection of trained networks provided by the research community is a straightforward way to solve the first problem of costly training. However, this does not address the storage and inference costs, which can still make the use of these trained networks impractical for deployment on resource-constrained hardware [1]. To address these demands, various model pruning techniques have been proposed. These range from removing individual weights within layers in unstructured pruning [12] to removing entire neurons, layers, or blocks in structured pruning [1].

While unstructured pruning can often retain a high degree of accuracy in trained networks, it is difficult to translate these theoretical gains into real-world speed-ups [1]. Since modern hardware is optimized for dense matrix multiplications, they cannot exploit the resulting sparsity effectively and therefore cannot solve the inference costs [1, 12].

In contrast, structured pruning leads to straightforward reductions in both inference costs and memory, but again requires extensive retraining to recover the lost accuracy resulting from these major structural modifications. For instance, Global Vision Transformer Pruning with Hessian-Aware Saliency (NViT) [27] requires 300 epochs of finetuning despite using trained networks, highlighting how addressing memory and inference speeds alone can still demand large additional training overhead.

On the other hand, dynamic pruning approaches, such as token pruning, avoid permanently modifying the model architecture and instead dynamically reduce the number of features processed by the network [3, 19]. This minimizes the accuracy drop after pruning, indicating that the model preserved most important features and representation, thus providing a better starting point for subsequent fine-tuning. Token Merging (ToMe) [3] embraces this idea by merging tokens to reuse the pruned information, thus reducing the accuracy drop even further and allowing off-the-shelf deployment without any fine-tuning. While this solves the inference costs as well as the training costs, since the methods do not modify the model structure itself, the memory foot-


$\sigma_2$
$\mu_2$
$\sigma_j$
$\mu_j$
$\mu_3$
$\mu_j$
$\Delta_\mu := (0, 0, \mu_3, 0, \ldots, \mu_j, \ldots)$
$\mu_3$
$\mu_j$
$+\vec{b}_2$
$\vec{b}'_2 := \vec{b}_2 + \mathbf{W}_2\Delta_\mu$

(a) We compute neuron-wise mean and variance of activations at the hidden layer of each MLP.

(b) We prune neurons with the smallest variances and replace their activation with the mean.

(c) We shift the mean activation of the removed neurons into the bias of the next layer.

Figure 1. Schematic visualization of the steps of Variance-Based Pruning. (a) Activation Statistics Computation, (b) Variance-Based Pruning, (c) Mean-Shift Compensation

print remains unchanged [3]. This again is a limitation for hardware-constrained settings [1]. Fully resolving all three problems of training costs, storage, and inference speed simultaneously remains a challange.

In this work, we fill this gap by introducing **Variance-Based Pruning (VBP)**, a simple and structured pruning approach that provides speed-ups and memory savings with minimal fine-tuning. In order to keep the method simple and allow deployment across a wide variety of architectures, we prune only the heavy Multi-Layer Perceptron (MLP) layers within the transformer blocks. In the first step, our approach leverages activation statistics gathered in the hidden layers of the MLP using Welford's algorithm [24] (see Fig. 1a). In the second step, these statistics are used to identify and remove the least impactful hidden neurons (see Fig. 1b). In the last step, which we refer to as **Mean-Shift Compensation**, we use the mean activation values of pruned neurons to mitigate accuracy drop by redistributing these mean contributions into the bias of the output layer (see Fig. 1c). This retains high levels of model performance and requires only minimal fine-tuning to then regain most of the original accuracy.

**Our contributions are summarized as follows:**

- We introduce VBP, a simple low-cost structured pruning method broadly applicable across various architectures to significantly reduce computational costs while maintaining performance.
- We apply our method to multiple architectures, including ViTs, Swin-Transformers, and ConvNeXts, demonstrating its generalizability.
- We compare VBP to similarly straightforward neuron-importance measures adapted for structured pruning, as well as to state-of-the-art (SoTA) pruning approaches. We benchmark against NViT pruning, highlighting the benefits of our accuracy retention in constrained fine-tuning settings. We also apply VBP on top of ToMe demonstrating orthogonality and achieving 2× speed-ups using our hybrid approach.
- We perform an extensive ablation study, demonstrating the efficacy of each component of our method and provide a sensitivity analysis for the variance as a pruning criterion.

## 2. RELATED WORK

### 2.1. Transformer Architectures

Initially introduced in NLP by [22], Transformers have largely replaced Convolutional Neural Network (CNN), since their adaption to CV by [8] with the ViT. ViTs processes images as a sequence of non-overlapping patches, and have since been further improved. DeiT [21] refined the training procedure to decrease the training costs and reduce the data reliance. More recent architectures [7, 9, 10], such as Swin [14] and the transformer-inspired ConvNeXt [15], which use hierarchical designs, have built upon this architecture to improve efficiency and scalability. However, despite these advances, these large models remain computationally demanding, in part due to their dense MLP blocks, which significantly impacts inference speed and memory requirements [22, 30].

### 2.2. Model Compression

To reduce the costs of large models, model compression efforts focus on scaling down different components of a network. Traditionally pruning aims to identify and remove redundant structures within the network [12, 26]. For example, magnitude pruning eliminates weights with the smallest absolute values, while SNIP and related methods [13, 25] identify crucial connections using gradient-based sensitivities. Structured pruning approaches, remove entire neurons, layers or filters producing smaller matrices that allow direct computational savings in current hardware [1, 5, 27–29, 32]. Most closely related to our method are works exploring variance-based criteria for structured pruning, such as Molchanov et al. [18], which performs dropout at the neuron level and can introduce sparsity in neurons with high dropout rates. Weinstein et al. [23] use these statistics in the initial training phase to remove neurons during training. These approaches operate on models before training, improving training time but still requiring the models to be trained from scratch. Contrary to that, NViT [27] utilizes trained ViTs. They apply iterative structured dimensionality reductions across the entire ViT architecture and significantly reduce costs. However, these structured modifications still have a substantial impact on performance, requiring high pruning iterations of 50 epochs and retraining costs of 300 epochs to regain accuracy.

An alternative to structurally modifying the network is dynamic pruning, which adaptively selects and processes only a subset of features or combines features during inference to reduce costs. For CNNs, these have shown promising speed-accuracy trade-offs [4, 16]. Analogously for transformer-like models, token pruning reduce the number of tokens to be processed [3, 17, 19, 20]. For instance, Bolya et al. [3] merges similar tokens rather than pruning them, thereby conserving most of the information and retaining high accuracy before the fine-tuning. However, by design these methods cannot reduce the memory footprint.

### 2.3. Our Approach

Our work builds upon these ideas by combining (i) the advantages of structured pruning by removing low-variance activation neurons with (ii) the advantages of dynamic pruning by incoorperating their mean contribution into the bias of the last MLP layer. Our approach is based on the observation that neurons within MLP layers exhibit varying levels of contribution to the overall representation, as established in works on emerging modularity in trained Transformers for both NLP and CV [2, 31]. By applying our variance-based pruning approach to the widespread MLP layers, we provide a structured method applicable across a broad range of models. Our method is particularly beneficial for transformer-like architectures, where MLP layers heavily contribute to the computational overhead [22, 30].

## 3. Methodology

In this section, we detail the components involved in our proposed VBP approach and provide a mathematical derivation, justifying our pruning criterion and compensation strategy.

Our method consists of three major steps:

1. **Activation Statistics Computation** The neuron activation statistics are computed using Welford's algorithm.
2. **Variance-Based Pruning** The neurons to be pruned are selected based on the lowest activation variance.
3. **Mean-Shift Compensation** The mean activations of the selected neurons are added back into the output bias.

### 3.1. Step 1: Activation Statistics Computation

As we aim for a simple and widely adoptable method, we prune only the MLPs. This requires reducing the hidden layers of the network while keeping the input and output layers intact to ensure seamless integration with the rest of the model structure by maintaining dimensional consistency in the input and output.

W.l.o.g. we therefore consider a MLP with a single hidden layer of dimension $D_{\text{hid}}$, that maps an input vector $\mathbf{x} \in \mathbb{R}^{D_{\text{in}}}$ to an output vector $\mathbf{y} \in \mathbb{R}^{D_{\text{out}}}$. Using a pointwise nonlinear activation function $\sigma(\cdot)$, this MLP can then be described as a function

$$\mathbf{h} = \sigma\big(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1\big), \quad \mathbf{y} = \mathbf{W}_2 \mathbf{h} + \mathbf{b}_2, \tag{1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{D_{\text{hid}} \times D_{\text{in}}}$, $\mathbf{b}_1 \in \mathbb{R}^{D_{\text{hid}}}$, $\mathbf{W}_2 \in \mathbb{R}^{D_{\text{out}} \times D_{\text{hid}}}$, $\mathbf{b}_2 \in \mathbb{R}^{D_{\text{out}}}$. Since we deal with trained networks, these parameters $\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2$ remain fixed during pruning, which will become important in the Mean-Shift Compensation in Sec. 3.3.

For the hidden layer, we now want to calculate the per-neuron mean and variance vectors

$$\boldsymbol{\mu} = \big(\mu_1, \ldots, \mu_{D_{\text{hid}}}\big)^\top \quad \text{and} \quad \boldsymbol{\sigma}^2 = \big(\sigma_1^2, \ldots, \sigma_{D_{\text{hid}}}^2\big)^\top$$

To compute these statistics, we capture the activations $\mathbf{h}$ for every batch and use Welford's algorithm to compute a running mean $\boldsymbol{\mu}^{(j)}$ and running second moment $\mathbf{m}_2^{(j)}$ for an efficient online calculation of streaming data*. Given a total of $N$ data samples, let the $j$-th sample $\mathbf{h}^{(j)} \in \mathbb{R}^{D_{\text{hid}}}$ be the activation vector after the nonlinearity. Welford's algorithm ensures numerically stable calculations by updating $\boldsymbol{\mu}^{(j)}$ and $\mathbf{m}_2^{(j)}$ after observing the $j$-th sample as follows:

$$\boldsymbol{\mu}^{(j)} = \frac{j-1}{j}\, \boldsymbol{\mu}^{(j-1)} + \frac{1}{j}\, \mathbf{h}^{(j)} \tag{2}$$

$$\mathbf{m}_2^{(j)} = \mathbf{m}_2^{(j-1)} + \big(\mathbf{h}^{(j)} - \boldsymbol{\mu}^{(j-1)}\big) \odot \big(\mathbf{h}^{(j)} - \boldsymbol{\mu}^{(j)}\big) \tag{3}$$

*Our method does not require a minimum number of data samples for calculating activation statistics, but using an efficient online computation allows the use of as many data samples for pruning as available.

where $\odot$ denotes elementwise multiplication. Once all $N$ samples are processed, the variance is given by:

$$\boldsymbol{\sigma}^2 \;=\; \frac{\mathbf{m}_2^{(N)}}{N-1}. \tag{4}$$

## 3.2. Step 2: Variance-Based Pruning

Using this variance vector $\boldsymbol{\sigma}^2 \in \mathbb{R}^{D_{\text{hid}}}$, we rank neurons by their variance values, $\sigma_i^2$. Intuitively, if an activation rarely deviates from its mean $\mu_i$, the corresponding neuron contributes less to the expressiveness of the network. The pruning decision is formed across all hidden neurons in the network, to capture the neuron with the least variance across all layers.

Formally, for each MLP $l$, let $D_{\text{hid}}^{(l)}$ denote the number of hidden neurons. For each neuron $i$ in MLP $l$, we then simply select its activation variance $\sigma_{l,i}^2$ as the corresponding pruning score. Given a pruning ratio $p \in (0,1)$, we then only need to form the global set of scores

$$S = \bigcup_{l \in \mathcal{L}} \{\sigma_{l,i}^2 \mid i = 1, \ldots, D_{\text{hid}}^{(l)}\}.$$

from which we select the $p\%$ neurons with the smallest scores for pruning.

**Optimality from a Mean-Replacement Perspective** To retain the contribution of the pruned network, we approximate the activation of a pruned neuron $h_i$ by replacing it with its mean $\mu_i$. This introduces an error proportional to $|h_i - \mu_i|$, which, by definition of the sample variance for any distribution, has an expected value:

$$\mathbb{E}[(h_i - \mathbb{E}[h_i])^2] = \mathbb{E}[(h_i - \mu_i)^2] = \sigma_i^2.$$

Thus, pruning the neurons with the lowest variance results in the least reconstruction error, making variance the optimal metric in this context. This is intuitively confirmed by the fact that when the variance is zero, all activations are exactly equal to the mean.

## 3.3. Step 3: Mean-Shift Compensation

The intuitive approach of removing neurons and replacing their activations with $\mu_i$[†] at inference time already reduces the output dimension of $\mathbf{W}_1$, but the reconstruction of the original embedding dimension $D_{hid}$ through replacement of the pruned activations still requires carrying out the full matrix multiplication for $\mathbf{W}_2$.

Instead, we perform an equivalent transformation by shifting these mean values directly into the bias of the final linear layer $\mathbf{b}_2$, allowing us to reduce both the output dimension of $\mathbf{W}_1$ as well as the input dimension of $\mathbf{W}_2$, doubling the cost savings.

We start from the usual two-layer linear mapping in the MLP block:

$$\mathbf{y} = \mathbf{W}_2\,\mathbf{h} + \mathbf{b}_2,$$

where $\mathbf{h}$ is the hidden activation after pruning. For pruned neurons, we replace $h_j$ with its mean $\mu_j$, which we collect into a vector $\Delta_\mu \in \mathbb{R}^{D_{\text{hid}}}$

$$(\Delta_\mu)_j = \begin{cases} \mu_j, & j \in \mathcal{P}, \\ 0, & j \notin \mathcal{P}, \end{cases}$$

where $\mathcal{P}$ is the set of pruned neuron indices.

Instead of explicitly inserting these means into $\mathbf{h}$ and then multiplying by $\mathbf{W}_2$, we exploit the linearity of the mapping:

$$\mathbf{W}_2\,\mathbf{h} = \mathbf{W}_2\big(\mathbf{h} - \Delta_\mu\big) \;+\; \mathbf{W}_2\,\Delta_\mu.$$

The second term is a constant vector, so we can shift it into the bias:

$$\mathbf{b}_2' \;=\; \mathbf{b}_2 \;+\; \mathbf{W}_2\,\Delta_\mu,$$

which results in an equivalent output:

$$\mathbf{y} \;=\; \mathbf{W}_2\left(\mathbf{h} - \Delta_\mu\right) \;+\; \mathbf{b}_2'. \tag{5}$$

Because for each pruned index $j$, we have replaced the activations $h_i$ by their means $h_j = \mu_j$, it follows that $(\mathbf{h} - \Delta_\mu)_j = 0$. Consequently, we can both drop the row $j$ of $\mathbf{W}_1$ as well as the column $j$ of $\mathbf{W}_2$ without affecting the computation, reducing the hidden dimension by $|\mathcal{P}|$.

**Why Use Trained Networks?** Our approach uses the fact that $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2$ are constants in trained networks. This allows the mean to be easily shifted into the bias of the next layer, which in turn allows the low-variance neurons to be removed without requiring extensive additional training. Only minor fine-tuning is sufficient to adjust the remaining weights for best performance.

# 4. Experiments

In this section, we apply our VBP method to various transformers and similar architectures, namely DeiT, Swin, and ConvNeXt in the Tiny, Small, and Base sizes, all trained on the ImageNet-1k [6] dataset.

We evaluate these models twice. Once immediately after pruning (before fine-tuning) to measure how much accuracy is retained, and the second time after a brief fine-tuning to measure the final accuracy. Fine-tuning is performed over merely 10 epochs using knowledge-distillation from the unmodified base model. We use the AdamW optimizer with an initial learning rate of 1.5e-5, decayed by a cosine-annealing scheduler, and a batch size of 32. As regularization, we include a weight decay of 0.01. All experiments are performed on NVIDIA H200 Tensor Core GPUs.

[†] We omit the MLP indices for clarity.

| **Model** | **MACs (G)** | | **Parameters (M)** | | **Top-1 Acc. (%)** | | |
|---|---|---|---|---|---|---|---|
| (Pruning Rate) | **Full** | **VBP** | **Full** | **VBP** | **Full** | **Ret.** | **VBP** |
| DeiT-T (45%) | 1.26 | 0.94 (-25.16%) | 5.72 | 4.12 (-27.97%) | 72.02 | 49.77 (69.13%) | 70.08 (97.33%) |
| DeiT-S (50%) | 4.61 | 3.21 (-30.37%) | 22.05 | 14.96 (-32.15%) | 79.70 | 64.44 (80.85%) | 78.62 (98.64%) |
| DeiT-B (55%) | 17.58 | 11.44 (-34.93%) | 86.57 | 55.40 (-36.01%) | 81.73 | 57.58 (70.48%) | 80.67 (98.74%) |
| DeiT-B (20%) | 17.58 | 15.35 (-12.68%) | 86.57 | 75.24 (-13.09%) | 81.73 | 80.87 (98.98%) | 81.76 (100.07%) |
| Swin-T (45%) | 4.50 | 3.23 (-28.22%) | 28.29 | 21.31 (-24.67%) | 80.91 | 55.12 (68.13%) | 79.41 (98.15%) |
| Swin-S (50%) | 8.76 | 5.63 (-32.19%) | 49.61 | 35.02 (-29.41%) | 83.04 | 67.01 (80.70%) | 81.86 (98.58%) |
| Swin-B (55%) | 15.46 | 10.22 (-33.89%) | 87.77 | 56.29 (-35.87%) | 84.71 | 62.61 (73.91%) | 83.61 (98.70%) |
| Swin-B (20%) | 15.46 | 13.58 (-12.16%) | 87.77 | 76.42 (-12.93%) | 84.71 | 83.90 (99.04%) | 84.67 (99.95%) |

Table 1. Results comparing the **full** baseline model with our **VBP** model using four metrics: **MACs**: computational operations, measured in billions of operations; and **Parameters**: the total model size in millions of parameters; **Accuracy Retention (Ret.)**: retained accuracy after pruning, before fine-tuning; and **Final Accuracy**: accuracy after fine-tuning. Our method achieves competitive accuracy with significant reductions in MACs and parameters after fine-tuning, and even right after pruning, when reducing the pruning rate to 20%.

### 4.1. Main Results

As is common with various pruning techniques, the pruning rate is adapted for the size of the models. Larger models typically exhibit higher redundancy for similar tasks, and therefore allow for more pruning while maintaining accuracy. We report our results for DeiT and Swin in Tab. 1 and Tab. 2. Note that the pruning rate is applied to the MLPs only and does not indicate the total reduction in model size.

- For the base-sized models, we prune all MLPs globally using a pruning rate of $55\%$ keeping $99\%$ of the original performance after fine-tuning, reducing the total model size of DeiT-Base, Swin-Base both by $36\%$. This corresponds to $35\%$, and $34\%$ fewer MACs respectively and provides a speed-up of $1.44\times$ and $1.30\times$ times.
- The small-sized models have notably fewer parameters than the base models and are thus pruned $50\%$ again reaching $99\%$ of the original performance. This leads to a DeiT-Small size reduction of $32\%$ and Swin-Small reduction of $29\%$ with $30\%$ and $32\%$ MACs savings respectively for a total speed-up factor of $1.34\times$ and $1.29\times$.
- Finally, for the DeiT- and Swin-Tiny models, we apply a lower pruning rate of $45\%$, achieving $97\%$ and $98\%$ final performance with $28\%$ and $25\%$ reductions in model size. This translates to $25\%$ and $28\%$ fewer MACs, along with speed-ups of $1.17\times$ and $1.20\times$ respectively.
- For the Base-Models, we demonstrate that when globally pruning the MLPs by $20\%$ the accuracy retention directly after the one-shot structured pruning reaches $99\%$ of the unpruned performance without any fine-tuning. This allows for an off-the-shelf $13\%$ size and MACs reduction and a speed-up of $1.11\times$ for DeiT-Base and a similar reductions for Swin-Base

All further pruning rates ranging from $5\%$ to $50\%$ for all DeiT Models can be found in Appendix B.

| **Model**<br>(Pruning Rate) | **Tiny**<br>(45%) | **Small**<br>(50%) | **Base**<br>(55%) | **Base**<br>(20%) |
|---|---|---|---|---|
| DeiT | 1.17 | 1.34 | 1.44 | 1.11 |
| Swin | 1.20 | 1.27 | 1.30 | 1.10 |

Table 2. Speed-ups of the pruned models relative to the baseline for different sizes and pruning rates.

| **Model** | **MACs** | **Param.** | **Top-1 Acc. (%)** | |
|---|---|---|---|---|
| | **(G)** | **(M)** | **Ret.** | **Final** |
| DeiT-T | 1.26 | 5.72 | – | 72.02 |
| Magnitude | 0.91 | 3.94 | 3.56 | 68.49 |
| SNIP | 0.91 | 3.94 | 24.89 | 69.10 |
| VBP (ours) | 0.91 | 3.94 | **39.58** | **70.61** |
| DeiT-S | 4.61 | 22.05 | – | 79.70 |
| Magnitude | 3.21 | 14.96 | 4.05 | 76.55 |
| SNIP | 3.21 | 14.96 | 52.39 | 77.27 |
| VBP (ours) | 3.21 | 14.96 | **64.44** | **78.62** |
| DeiT-B | 17.58 | 86.57 | – | 81.73 |
| Magnitude | 12.0 | 58.24 | 0.37 | 78.88 |
| SNIP | 12.0 | 58.24 | 53.24 | 80.40 |
| VBP (ours) | 12.0 | 58.24 | **66.40** | **80.99** |

Table 3. Comparison of different one-shot pruning scores in structured pruning to our **VBP** approach at fixed 50% pruning rates. For constant **MACs** and **parameters**, our approach retains more accuracy (**Ret.**) and reaches a higher **final** accuracy compared to other pruning approaches.

| **Model** | **MACs (G)** | **Param. (M)** | **Speed-Up** | **Top-1 Acc. (%)** | |
|---|---|---|---|---|---|
| | | | | **Ret.** | **Final** |
| DeiT-B | 17.58 | 86.57 | – | – | 81.73 |
| ToMe-24 | 6.02 | 86.57 | 2.15 | 35.85 | 75.74 |
| ToMe-14 & VBP | 6.08 | **58.24** | 2.05 | 60.53 | **80.09** |
| ToMe-20 | 7.14 | 86.57 | 1.84 | 66.31 | 78.66 |
| ToMe-12 & VBP | 6.90 | **58.24** | 1.83 | 62.09 | **80.29** |
| ToMe-18 | 7.88 | 86.57 | 1.75 | 73.79 | 79.97 |
| ToMe-10 & VBP | 7.73 | **58.24** | 1.70 | 63.72 | **80.45** |

Table 4. Comparison of ToMe for different token reduction rates of 24, 20, and 18, with the hybrid combination of ToMe using token reduction rates of 14, 12, and 10, and **VBP** using 50% pruning rates on top. Our hybrid method achieves significantly higher **final** accuracies throughout while also providing reductions in **parameters**, for similar **MACs** and **speed-ups**. The hybrid model thereby reaches speed-ups of up to 2.05× while maintaining competitive performance.

| **Model** | **Epochs** | **Param.** | **Top-1 Acc. (%)** | |
|---|---|---|---|---|
| | | (M) | **Ret.** | **Final** |
| NViT (50 Ep.) | 50 | 56.37 | 81.13 | 82.18 |
| NViT (1 Ep.) | 1 | 56.85 | 69.10 | 81.92 |
| VBP (ours) | 1 | 56.18 | **72.37** | **82.32** |

Table 5. Comparison with NViT using two pruning durations of 50 epochs (as originally proposed) and 1 epoch (similar to our VBP approach), both followed by 10 epochs of fine-tuning. Our method achieves higher accuracy immediately after pruning, before fine-tuning (**Ret.**) compared to NViT with the same pruning duration, and reaches a higher **final** accuracy than the 50-epoch NViT model after fine-tuning for 10 epochs.

## 4.2. Performance Comparison

To facilitate direct comparisons of all models with different sizes across all experiments, we fix the pruning rate at $50\%$ for all further experiments.

We demonstrate the effectiveness of our selection criterion, by benchmarking against two other common and well-established one-shot-pruning baselines, namely Magnitude and SNIP, adapted for structured pruning. For a fixed pruning rate of $50\%$ using all selection criteria, this produces identical parameter counts of 3.94, 14.96, and 58.24 million parameters, as well as fixed $0.91$, $3.21$, and $12.00$ GMACs for the three model sizes. Our results indicate that VBP provides higher accuracy retention compared to the baselines, improving DeiT-Base by $13.16\%$, DeiT-Small by $12.05\%$, and DeiT-Tiny by $14.69\%$ percentage points compared to SNIP. This difference in accuracy retention yields a final fine-tuned performance that is $0.59\%$ to $1.51\%$ percentage points higher after fine-tuning.

## 4.3. SoTA Pruning Methods

To further validate our approach, we compare against the SoTA structured transformer pruning method NViT (CVPR'23). Specifically, we apply our method to the NViT Base model (DeiT-Base distilled from RegNetY-160) and run NViT with a latency target of $70\%$, pruning once for 50 epochs (as per the published results) and once for one epoch (similar to our setup), with both scenarios followed by ten epochs of fine-tuning using the NViT settings. In a similar pruning setting, VBP achieves a $0.4\%$ percentage points higher final accuracy, for similar number of parameters.

Finally, we demonstrate the orthogonality of VBP to token-pruning methods by applying our method on top of the SoTA token-merging approach, ToMe (ICLR'23). We observe that the accuracy retention drops relative to the baseline remains consistent for both the VBP-pruned base DeiT model in Tab. 1 and the VBP- & ToMe-pruned model. This relative consistency also persists throughout fine-tuning, indicating that our method operates orthogonally to ToMe in terms of performance.

Regarding the MAC savings it is important to note that ToMe progressively reduces the throughput of tokens processed across the layers. Later layers therefore process less tokens, making pruning less effective. The relative MAC savings from our hybrid method therefore decrease as the network depth increases. Nonetheless, VBP additionally reduces the total parameter count, which cannot be achieved by simply increasing the ToMe token reduction rate.

In fact, increasing the reduction rate to get an approximate 2× speed-up yields similar MAC savings but reduces both the accuracy retention as well as the final accuracy significantly. Instead applying VBP on top of ToMe with a smaller reduction rate, allows more than 2× speed-ups while still maintaining $98\%$ of the original accuracy with additional parameter savings.

| Variance-Based | Mean-Shift | Top-1 Acc. (%) | |
|---|---|---|---|
| | | Ret. | Final |
| ✗ | ✓ | 55.19 | 80.23 |
| ✓ | ✗ | 26.04 | 80.62 |
| ✓ | ✓ | **66.40** | **80.99** |

Table 6. Ablation study results showing the impact of Variance-Based Pruning, and Mean-Shift Compensation on the retained accuracy after pruning (**Ret.**), as well as the **final**: accuracy after fine-tuning. Both steps together yield the highest accuracy of 80.99%.

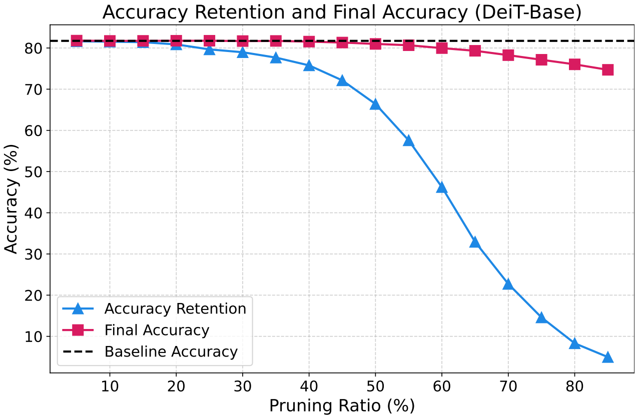


Figure 2. Accuracy retention before fine-tuning and final accuracy after 10 epochs of fine-tuning for different pruning rates applied to DeiT-Base.

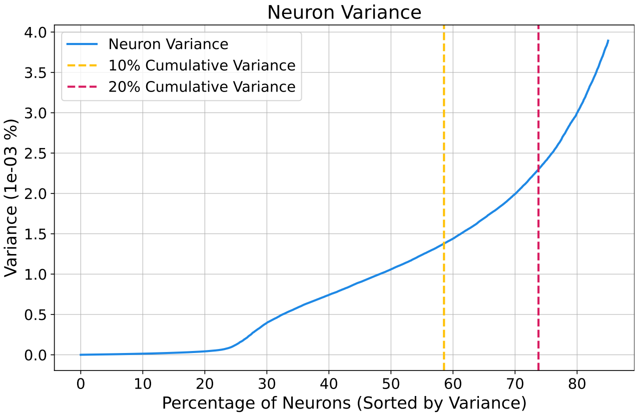


Figure 3. Activation variance for all neurons in the hidden MLP layers throughout the network and marked cumulative variances.

# 5. Analysis

To assess the contribution of each individual component of our approach, we perform an ablation study in which we systematically modify the pruning criterion and the Mean-Shift Compensation. We report our results in Tab. 6.

| Model | MACs (G) | Param. (M) | Top-1 Acc. (%) | |
|---|---|---|---|---|
| | | | Ret. | Final |
| Pre-Act. (eq. [23]) | 12.0 | 58.24 | 0.43 | 77.92 |
| Post-Act. (ours) | 12.0 | 58.24 | **66.40** | **80.99** |

Table 7. Comparison of different locations for the activation statistics computation in the network: before (**Pre-Act.**) as well as after the activation function (**Post-Act.**). For the same **MACs** and **parameters**, using pruning rates of 50%, gathering statistics post-activation significantly outperforms pre-activation in both accuracy retention (**Ret.**) as well as the **final** accuracy.

We first remove the variance-based pruning selection while still shifting the mean activation to the next bias. Removing the Variance-Based Pruning alone reduces the final accuracy by 0.76% percentage points.

We also remove the Mean-Shift Compensation while pruning using the variance-based criterion. This improves the final accuracy by 0.39% compared to using only the Mean-Shift Compensation.

However, the best results are achieved when both Variance-Based Pruning and Mean-Shift Compensation are used together. As discussed in Sec. 3.2, replacing the removed activations with their means is optimal when pruning based on variance. When both components are combined, the immediate accuracy retention increases by 11.21% percentage points, ultimately improving the fine-tuned accuracy by an additional 0.37%. This highlights the significance of both components.

## 5.1. Sensitivity of Variance Thresholding

We further examine the stability of the activation variance criterion by applying different pruning rates to the DeiT-Base model, and plotting the accuracy retention and final accuracy in Fig. 2. We observe that for smaller pruning rates (up to around 25%), the accuracy retention is high enough to allow for off-the-shelf deployment without any fine-tuning.

Beyond these rates, the accuracy begins to degrade more noticeably, reflecting the non-uniform distribution of neuron variances, as seen in Fig. 3. Notably, to account for 10% of the cumulative variance across all neuons, nearly 60% of the lowest-variance neurons are needed. This is a disproportionately large share compared to the next 15% of neurons that also account for 10% of the total variance in a layer. Consequently, more aggressive pruning leads to an increasingly faster reduction in the total variance and thus expressiveness of the network. Nevertheless, significant performance issues arise only at very high pruning levels, validating the feasibility of our method over a broad range of pruning rates (see Appendix B).

| Model | MACs (G) | | Parameters (M) | | Top-1 Acc. (%) | | |
|---|---|---|---|---|---|---|---|
| | Full | VBP | Full | VBP | Full | Retention | VBP |
| ConvNeXt-T | 4.47 | 2.96 (-33.8%) | 28.59 | 12.61 (-55.9%) | 82.90 | 16.80 (20.3%) | 81.30 (98.1%) |
| ConvNeXt-S | 8.71 | 5.11 (-41.3%) | 50.22 | 23.50 (-53.2%) | 84.57 | 30.92 (36.6%) | 82.82 (97.9%) |
| ConvNeXt-B | 15.38 | 8.91 (-42.1%) | 88.59 | 41.32 (-53.4%) | 85.51 | 57.10 (66.8%) | 83.40 (97.6%) |

Table 8. Results comparing the **full** ConvNeXt baseline models with the **VBP** models using a pruning rate of 50%. The **MACs** are reduced by up to 42% and **parameters** by over 50% while reaching competitive **final** accuracies of 98% original performance.

| Model | Tiny | Small | Base |
|---|---|---|---|
| ConvNeXt | 1.28 | 1.42 | 1.49 |

Table 9. Speed-ups of the pruned models relative to the baseline for different-sized ConvNeXt models.

## 5.2. Application Pre- vs. Post-Activation

While Sec. 3.2 shows that our mean-replacement strategy can be grounded in theory for any distribution, the question remains whether the variance should be measured before or after the activation function. The Central Limit Theorem confirms that pre-activation sums (composed of independent weight contributions) tend to approximate a normal distribution. However, once the nonlinearity $\sigma$ is applied, this distribution changes. Consequently, the pre-activation variance does not necessarily correlate with the neuron importance for retaining accuracy in an already trained model.

For instance, highly varying negative activations are compressed into range of about $(-0.2, 0)$ by GeLU, reducing their variance. As a result, such a neuron would be less likely to be pruned in a pre-activation setting and more likely to be pruned post-activation (see Appendix E).

To validate this experimentally, we tested an alternative design in which pruning decisions are made based on the pre-activation variance in Tab. 7. While this approach has been successfully applied when pruning during training [23], we find that in a trained network setting, it significantly degrades performance. Therefore, measuring variance *post*-activation better captures neuron importance in already trained models, and helps maintain model performance despite structural modifications.

## 5.3. Other Transformer-Like Architectures

While our method is broadly applicable to any model containing MLP layers, network architectures that have more and larger MLPs benefit more from VBP. We therefore apply our pruning on ConvNeXt as a transformer-like architecture, that relies heavily on MLP layers, to further evaluate the generilizability of our method.

We summarize our performance comparisons for ConvNeXt in Tab. 8. We achieve significant parameter reductions of over $50\%$ throughout all three sizes and reduced MACs by $34\%$ for ConvNeXt-Tiny and over $40\%$ for the larger sizes, while maintaining $98\%$ of the original accuracy after 10 epochs of fine-tuning across the board. The corresponding speed-ups are listed in Tab. 9, and reach up to $1.49\times$ for ConvNeXt-Base.

Noteably, we observe significantly less accuracy retention throughout the smaller sized models compared to the true transformers. We attribute this to the fact that the ConvNeXt architecture is built upon convolutions, which have significantly higher costs compared to their parameters. Consequently, since the MLPs take over a higher overall percentage of the total capacity, removing a similar fraction of neurons causes a relatively higher drop in accuracy retention. By contrast, transformers typically have less parameters in their MLP relatively speaking, therby allowing for higher pruning rates while maintaining capacity.

## 6. Conclusion

We introduce VBP, a one-shot pruning method, designed to remove neurons based on their statistical importance in a single pruning operation while minimizing accuracy loss by integrating their mean activations in a Mean-Shift Compensation step.

Our experiments demonstrate that VBP retains model performance significantly better than existing structured pruning methods. By taking advantage of trained networks, our approach provides a favourable starting point for the subsequent finetuning, eliminating the need for extensive retraining and thereby making it practical for low-cost deployment.

Furthermore, we show that VBP operates orthogonally to SoTA token pruning methods, allowing it to be seamlessly combined with approaches like ToMe to achieve even greater computational savings.

We believe that the simplicity and efficiency of our approach contributes to the democratization of deep learning by enabling a wider reuse of trained networks and hope it inspires further research in the CV community toward efficient model compression techniques.

# References

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *J. Emerg. Technol. Comput. Syst.*, 13(3), 2017. 1, 2, 3

[2] Uranik Berisha, Jens Mehnert, and Alexandru Condurache. Efficient data driven mixture-of-expert extraction from trained networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. 1, 2, 3

[4] Rinor Cakaj, Jens Mehnert, and Bin Yang. Cnn mixture-of-depths. In *Computer Vision ACCV 2024: 17th Asian Conference on Computer Vision, Hanoi, Vietnam, December 812, 2024, Proceedings, Part VII*, page 148166, Berlin, Heidelberg, 2024. Springer-Verlag. 3

[5] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 19974–19988, 2021. 3

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 4, 3

[7] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 12114–12124. IEEE, 2022. 2

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1, 2

[9] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 6804–6815. IEEE, 2021. 2

[10] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 12239–12249. IEEE, 2021. 2

[11] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 4

[12] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 598–605. Morgan Kaufmann, 1989. 1, 3

[13] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 3

[14] 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 *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9992–10002. IEEE, 2021. 1, 2

[15] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11966–11976, 2022. 1, 2

[16] Lukas Meiner, Jens Mehnert, and Alexandru Condurache. Data-free dynamic compression of cnns for tractable efficiency. In *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 2: VISAPP*, pages 196–208. INSTICC, SciTePress, 2025. 3

[17] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 12299–12308. IEEE, 2022. 3

[18] Dmitry Molchanov, Arsenii Ashukha, and Dmitry P. Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2498–2507. PMLR, 2017. 3

[19] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 13937–13949, 2021. 1, 3

[20] David Raposo, Samuel Ritter, Blake A. Richards, Timothy P. Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *CoRR*, abs/2404.02258, 2024. 3

[21] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference*

*on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 10347–10357. PMLR, 2021. 1, 2

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. 2, 3

[23] Berry Weinstein and Yonatan Belinkov. Variance pruning: Pruning language models via temporal neuron variance, 2025. Unpublished manuscript, OpenReview. 3, 7, 8

[24] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962. 2

[25] Paul Wimmer, Jens Mehnert, and Alexandru Condurache. Freezenet: Full performance by reduced storage costs. In *Computer Vision - ACCV 2020 - 15th Asian Conference on Computer Vision, Kyoto, Japan, November 30 - December 4, 2020, Revised Selected Papers, Part VI*, pages 685–701. Springer, 2020. 3

[26] Paul Wimmer, Jens Mehnert, and Alexandru Condurache. Interspace pruning: Using adaptive filter representations to improve training of sparse cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12527–12537, 2022. 3

[27] Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with hessian-aware saliency. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 18547–18557. IEEE, 2023. 1, 3

[28] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 3143–3151. AAAI Press, 2022.

[29] Lu Yu and Wei Xiang. X-pruner: explainable pruning for vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 24355–24363. IEEE, 2023. 3

[30] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 1204–1213. IEEE, 2022. 2, 3

[31] Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xiaozhi Wang, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Emergent modularity in pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4066–4083, Toronto, Canada, 2023. Association for Computational Linguistics. 3

[32] Mingjian Zhu, Yehui Tang, and Kai Han. Vision transformer pruning, 2021. 3