NVIT: VISION TRANSFORMER COMPRESSION AND PARAMETER REDISTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers yield state-of-the-art results across many tasks. However, they still impose huge computational costs during inference. We apply global, structural pruning with latency-aware regularization on all parameters of the Vision Transformer (ViT) model for latency reduction. Furthermore, we analyze the pruned architectures and find interesting regularities in the final weight structure. Our discovered insights lead to a new architecture called *NViT* (Novel ViT), with a redistribution of where parameters are used. This architecture utilizes parameters more efficiently and enables control of the latency-accuracy trade-off. On ImageNet-1K, we prune the *DEIT-Base* (Touvron et al., 2021) model to a **2.6**$\times$ FLOPs reduction, **5.1**$\times$ parameter reduction, and **1.9**$\times$ run-time speedup with only **0.07%** loss in accuracy. We achieve more than **1%** accuracy gain when compressing the base model to the throughput of the Small/Tiny variants. NViT gains **0.1-1.1%** accuracy over the hand-designed DEIT family when trained from scratch, while being faster.

1 INTRODUCTION

Self-attention based transformer models demonstrate high model capacity, easy scalability, and superior ability in capturing long-range dependency (Vaswani et al., 2017; Devlin et al., 2018; Radford et al., 2018; Jiao et al., 2019; Brown et al., 2020). They have thus been widely applied to natural language processing (NLP) tasks, and recently received growing attention for computer vision tasks. Vision Transformer, *i.e.*, the ViT (Dosovitskiy et al., 2020), shows that embedding image patches into tokens and passing them through a sequence of transformer blocks can lead to higher accuracy compared to state-of-the-art CNN models. DEIT, recent work by Touvron et al. (2021), further presents a data-efficient training method such that acceptable accuracy can be achieved without extensive pretraining. Offering competitive performance to CNNs under similar training regimes, transformers now point to the appealing perspective of solving both NLP and vision tasks with the same architecture (Zheng et al., 2021; Kim et al., 2021; Jiang et al., 2021).

Unlike CNNs built with convolutional layers that are mainly parameterized by few dimensions like the kernel size and the number of filters, the ViT has multiple distinct components, *i.e.*, QKV projection, multi-head attention, multi-layer perceptron, etc. (Vaswani et al., 2017), each defined by independent dimensions. As a result, the dimensionality of each component in each ViT block needs to be carefully designed to achieve a decent trade-off between efficiency and accuracy. However, this is typically not the case for state-of-the-art models. Models such as ViT (Dosovitskiy et al., 2020) and DEIT (Touvron et al., 2021) mainly inherit the design heuristics from NLP tasks, *e.g., use MLP expansion ratio 4*, *fix QKV per head*, *all the blocks having the same dimensions*, etc., which may not be optimal for computer vision (Chen et al., 2021a), causing significant redundancy in the base model and a worse efficiency-accuracy trade-off upon scaling, as we show extensively in our experiments.

This work targets efficient ViTs by exploring latency-aware global structural pruning, leveraging the insights to redistribute parameters for enhanced accuracy-efficiency trade-off. Our approach, as visualized in Figure 1, starts from analyzing the blocks in the computation graph of ViT to identify all the dimensions that can be independently controlled. We apply global structural pruning over all the components in all blocks. This offers complete flexibility to explore their combinations towards an optimal architecture in a complicated design space. Our global pruning utilizes an importance score based on the first-order Taylor expansion of the pruning loss, offering comparability among all prunable components from all layers. Furthermore, we incorporate the estimated latency reduction of each neuron into its importance score. This guides the final pruned architecture to be faster on target

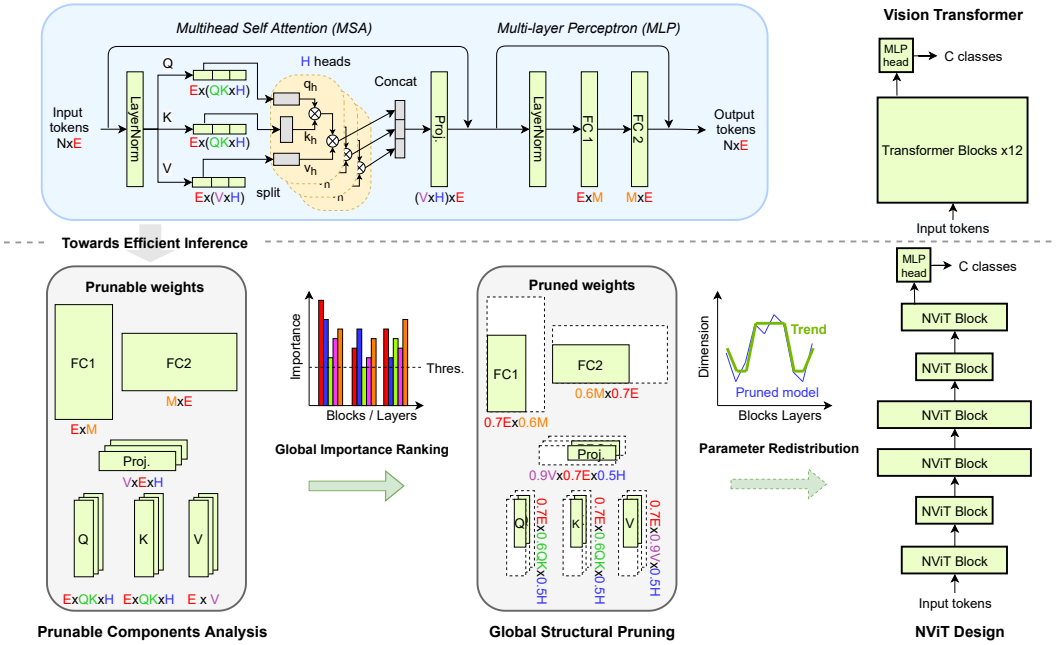

Figure 1: Towards efficient vision transformer models. Starting form ViT, specifically DEIT, we identify the design space of pruning (i) embedding size E, (ii) number of head H, (iii) query/key size QK, (iv) value size V and (v) MLP hidden dimension M in Section 3.1. Then we utilize a global ranking of importance score to perform iterative global structural pruning in Section 3.2. Finally we derive a simple NViT architecture from observing the dimension trend of all the components in the pruned model, as in Section 4.1.

devices, as we show in experiments. The pruned models are distilled utilizing the information of the ground truth labels, a pretrained CNN teacher akin to DEIT (Touvron et al., 2021), and the original full model. On the ImageNet-1K benchmark (Russakovsky et al., 2015), structural pruning enables a nearly lossless 5.14× parameter reduction, 2.57× FLOPs reduction and 1.86× speed up on V100 GPU over the DEIT-Base model. An 1% and 1.7% accuracy gain is observed over DEIT-Small and DEIT-Tiny models when we compress the base model to a similar latency.

Using structural pruning for architectural guidance, we further make an important observation that the popular uniform distribution of parameters across all layers is, in fact, not optimal. A simple redistribution of parameters can already provide stronger architectural alternatives, as we show in our experiments for both pretraining and downstream tasks. To this end, we present a new parameter distribution rule to scale ViT architectures, enabling a breed of models named as NViT. When scaling to similar FLOPs and latency, NViT architectures achieve 0.1%, 0.2% and 1.1% accuracy gains over the DEIT-Base, Small, and Tiny models respectively when trained from scratch on ImageNet-1K.

Our main contributions are as follows:

- Provide a systematic analysis on the prunable components in the ViT model. We identify the ability to perform structural pruning on the embedding dimension, number of heads, MLP hidden dimension, QK dimension and V dimension of each head separately;
- Propose a latency-aware, importance-based criteria that enables hardware-friendly global structural pruning of all the components, achieving a nearly lossless 1.9× speedup;
- Present a new architecture scaling rule that enables *NViT*, a new family of efficient vision transformer architectures that redistributes the dimensions of DEIT models to outperform them under similar FLOPs and latency. This is the first work showing the potential of discovering novel scalable architectures by pruning vision transformers;
- Demonstrate that the high performance achieved by the pruned models and NViT models transfer effectively to downstream tasks.

2 RELATED WORK

2.1 VISION TRANSFORMER MODELS

Inspired by the success of transformer models in NLP tasks, recent research proposes to use transformer models on computer vision tasks. The inspiring vision transformer (ViT) (Dosovitskiy et al., 2020) demonstrates the possibility of performing high-accuracy image classification with transformer architecture only, yet also finds that learning attention between image patches is a challenging task that requires large training set and model size. This stimulates recent works to ease extensive pre-training and improve the efficiency-accuracy tradeoff. One noticeable approach DEIT (Touvron et al., 2021) provides carefully designed training schemes and data augmentation techniques to train ViT from scratch on ImageNet only. Another line of work renovates ViT transformer blocks to better capture image features, such as changing input tokenization (Yuan et al., 2021; Graham et al., 2021), using hierarchical architecture (Liu et al., 2021; Wang et al., 2021; Graham et al., 2021), upgrading positional encoding (Chu et al., 2021), and performing localized attention (Liu et al., 2021; Han et al., 2021), bringing in further accuracy improvements under similar computation cost.

In this work we focus on the original ViT architecture (Dosovitskiy et al., 2020) amid its widespread usage, as illustrated in the top of Figure 1. ViT model first divides the input image into patches that are tokenized to embedding dimension E through a linear projection. Image tokens, together with an independently initialized *class token*, form an input $x \in \mathbb{R}^{N \times E}$. Input tokens pass through transformer blocks before classification is made from the class token output of the last block.

In its simplest form, a ViT block includes a multi-head self attention (MSA) and a multi-layer perceptron (MLP) module. The MSA module first linearly transforms the $N \times E$ tokens into queries $q \in \mathbb{R}^{N \times (QK \times H)}$, keys $k \in \mathbb{R}^{N \times (QK \times H)}$, and values $v \in \mathbb{R}^{N \times (V \times H)}$. The q, k and v are then split into H heads. Each head performs the self-attention operation defined in Equation (1) in parallel:

$$\text{Attn}(q_h, k_h, v_h) = \text{softmax}\left(\frac{q_h k_h^T}{\sqrt{d_h}}\right) v_h, \tag{1}$$

where $\frac{1}{\sqrt{d_h}}$ is the scaling factor, and $q_h \in \mathbb{R}^{N \times QK}$, $k_h \in \mathbb{R}^{N \times QK}$ and $v_h \in \mathbb{R}^{N \times V}$ are the query, key and value of head h. The output of all the heads are then concatenated prior to a fully-connected (FC) linear projection back to the original dimension of $\mathbb{R}^{N \times E}$. Note that though previous works set dimension $QK = V$ in designing the model architecture (Dosovitskiy et al., 2020; Touvron et al., 2021; Chen et al., 2021a), setting them differently will not go against the shape rule of matrix multiplication. The MLP module includes two FC layers with a hidden dimension of M. The output of the last FC layer preserves token dimension at $\mathbb{R}^{N \times E}$.

Built upon the original ViT, DEIT models (Touvron et al., 2021) further exploit a *distillation token*, which learns from the output label of a CNN teacher during the training process to incorporate some inductive bias of the CNN model, and significantly improves the DEIT accuracy.

Our work uses the DEIT model architecture as a starting point, where we explore the potential of compressing the model and better allocating of the dimensions of different blocks for enhanced efficiency-accuracy tradeoff. Our method is also applicable to other vision transformer architectures for further efficiency improvements, which will be explored in future works.

2.2 VIT COMPRESSION TECHNIQUES

Despite rapid progress of ViTs for vision tasks, ViT blocks still heavily inherit the architecture of transformers for NLP tasks (Vaswani et al., 2017), *e.g.,* the dimension of each block is determined without much optimization, and all the blocks in the stacked model architecture share the same dimension. As we will show later, such design may not be optimal for computer vision tasks.

To improve model efficiency, very recent works successfully use structural pruning techniques on vision transformer models, with trainable gate variables (Zhu et al., 2021) or Taylor importance score (Chen et al., 2021b). Both methods show the potential of compressing ViT models, yet only consider part of the prunable architecture like the number of heads or the MLP hidden dimension, use manually designed sparsity distribution, and do not take run time latency into account, thus may not lead to optimal compressed models. Our method resolves these issues through a latency-aware global structural pruning of all prunable components across all layers in a jointly manner.

Besides pruning, AutoFormer (Chen et al., 2021a) uses a neural architecture search (NAS) approach to search for efficient ViT models. AutoFormer only explores a small number of dimension choices, due to the constraint on the supernet training cost; while our method continuously explores the entire design space of ViT model with a single training process, leading to the finding of more efficient architectures with less training cost. Most importantly, our pruning scheme successfully leads to a scalable design rule, where we can easily design efficient vision transformer models at different scales. This has never been attempted in previous works.

There has been a series of research on improving the efficiency of attention-based models on NLP tasks, specifically pruning the number of heads in the MSA block. This includes utilizing stochastic binary gate (Louizos et al., 2017; Voita et al., 2019), pruning with sensitivity score (Michel et al., 2019) and exploring lottery ticket hypothesis (Frankle & Carbin, 2018; Behnke & Heafield, 2020). Our work aims on pruning all parameters including heads, and therefore, has a larger pruning space.

Other great ways of improving the efficiency of transformers include weight sharing across transformer blocks (Lan et al., 2019), dynamically controlling the attention span of each token (Chen et al., 2019; Tambe et al., 2020), and allowing the model to output the result in an earlier transformer block (Zhou et al., 2020; Schwartz et al., 2020). These techniques are orthogonal to our pruning-based method and have remained unexplored on vision models. Applying them on ViT models and combining with our pruning method would be an interesting and fruitful future direction.

3 Latency-aware global structural pruning

3.1 Identifying prunable structures

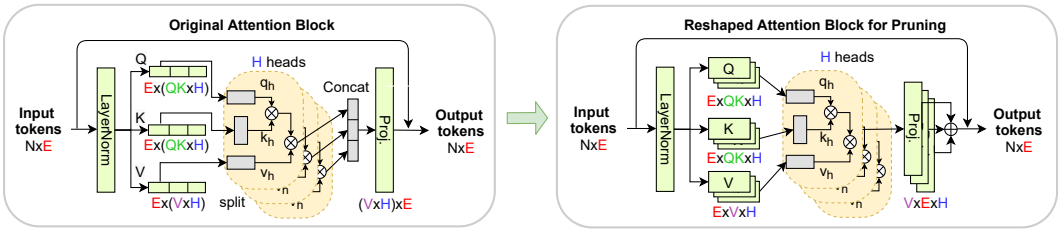

Figure 2: Attention block reshaping for latency-friendly structural pruning. We reshaped the QKV projection and final output projection in the attention block to explicitly control the number of head and align the QK & V dimensions in each head head.

In this work, we focus on pruning all the independent structural components in ViT, namely:

- The embedding dimension shared across all blocks, denoted as *EMB*;
- The number of heads in MSA for every block, denoted as H;
- The output dimension of Q and K projection per head in MSA, denoted as QK;
- The output dimension of V projection and input dimension of the PROJ per head, denoted as V;
- The hidden dimension of MLP per block, denoted as *MLP*.

Note that this is slightly different from the dimensions we showed in Section 2.1. As highlighted on the left of Figure 2, in a typical ViT implementation, the QKV projection output dimensions are a concatenation of all the attention heads (Wightman, 2019), effectively $QK \times H$ or $V \times H$. The projected tokens are then split into H heads to allow the computation of MSA in parallel. If we directly prune this concatenated dimension, then there is no control on the remaining QK and V dimension of each head. Therefore, the latency of the entire MSA will be bounded by the head with the largest dimension.

To alleviate such inconsistency between pruned head dimensions, we explicitly control the number of heads and align the QK and V dimension remaining in each head. As illustrated on the right of Figure 2, for model pruning we reshape the weight of Q, K, V and PROJ projection layers to single out the head dimension H. Performing structural pruning on the reshaped block along the H dimension will enable the removal of an entire head, while pruning along the QK/V dimension guarantees the

remained QK and V dimension of all the heads are the same. This reshaping is only applied during the pruning process, while the final pruned model is converted back to the concatenated scheme. Note that H, QK, V and MLP in different blocks can all be independently pruned. While the H dimension should be kept the same for all Q, K, V and PROJ weights within each block. Finally, EMB needs to be kept the same for all the blocks due to the shortcut connections.

A comparison of this latency-friendly pruning scheme with directly applying structural pruning on the concatenated $QK \times H$ and $V \times H$ dimension is provided in Appendix B.3, where we demonstrate lower latency and higher accuracy can be achieved with our pruning scheme.

3.2 Structural pruning criteria

3.2.1 Global importance ranking

Structural pruning aims to find a model with weights $\mathbf{W}$, under a certain model size constraint C, that leads to the minimal loss $\mathcal{L}(\mathcal{D}, \mathbf{W})$ on dataset $\mathcal{D}$. This can be solved by optimization as:

$$\min_{\mathbf{W}} \mathcal{L}(\mathcal{D}, \mathbf{W}),\ s.t.\ \left|\left|\sum_{s\in\mathcal{S}} |w_s|\right|\right|_0 \leq C, \tag{2}$$

where w_s is an element in the structural group $\mathcal{S}$ of weight $\mathbf{W}$. Unfortunately, the optimization under ℓ_0 norm constraint is known to be non-convex, NP hard and requires inefficient combinatorial search.

As an alternative, previous research relaxes the full combinatorial search with an iterative greedy search scheme (Guo et al., 2016; He et al., 2017; Molchanov et al., 2019). Under this scheme, starting with well-trained parameters $\mathbf{W}$ achieving $\min_{\mathbf{W}} \mathcal{L}(\mathcal{D}, \mathbf{W})$, we greedily remove a few structural groups at a time based on their importance scores, until the targeted constraint is achieved. The expected squared perturbation in the loss function by setting the group $\mathcal{S}$ to 0 is equal to:

$$\mathcal{I}_{\mathcal{S}} = \Big(\mathcal{L}(\mathcal{D}, \mathbf{W}|w_s = 0) - \mathcal{L}(\mathcal{D}, \mathbf{W})\Big)^2. \tag{3}$$

To efficiently evaluate $\mathcal{I}_{\mathcal{S}}$ in Equation (3), instead of calculating the loss difference for each group, we perform a first-order Taylor expansion of the loss value with pruned weights, as:

$$\mathcal{L}(\mathcal{D}, \mathbf{W}|w_s = 0) = \mathcal{L}(\mathcal{D}, \mathbf{W}) + \sum_{s\in\mathcal{S}} g_s w_s, \tag{4}$$

where $g_s = \partial\mathcal{L}/\partial w_s$ denotes the gradient of training objective with respect to w_s. Substituting this approximation into Equation (3) leads to a simplified importance score:

$$\mathcal{I}_{\mathcal{S}}(\mathbf{W}) = \left(\sum_{s\in\mathcal{S}} g_s w_s\right)^2. \tag{5}$$

Since the gradients with respect to all weight elements are already available from backpropagation, the importance score in Equation (5) can be easily calculated during the finetuning process without additional cost. Unlike the magnitude of the weight being in various ranges in different layers, the Taylor-based importance score can be compared among all layers of weight as a global pruning criteria as it reflects the contribution of the weight to the loss value, as shown to be very effective for CNN filter pruning (Molchanov et al., 2019; Ding et al., 2019; Yin et al., 2020; Chawla et al., 2021).

In this work we further show that Taylor importance score of different groups can be biased to encourage more intense pruning of some parameters. The simplest solution would be to set a multiplier per structural group. As explained in Appendix A.2, we divide importance of H by 6 to encourage removal of heads.

Pruning can be tailored towards latency reduction by penalizing the importance score with latency-aware regularization:

$$\mathcal{I}^L_{\mathcal{S}}(\mathbf{W}) = \mathcal{I}_{\mathcal{S}}(\mathbf{W}) - \eta\Big(\text{Lat}(\mathbf{W}) - \text{Lat}(\mathbf{W}\backslash\mathcal{S})\Big). \tag{6}$$

$\text{Lat}(\cdot)$ denotes the latency of the current model, which is characterized by a lookup table of attention block latency given the current EMB, H, QK, V, and MLP dimension of each block in the pruned

model. Details of the latency lookup table are provided in Appendix A.3, where we show a small lookup table can achieve accurate latency estimation throughout the pruning process. We use $\mathcal{I}_S^L$ as the pruning criteria for iterative pruning in our work, with detailed procedure in Appendix A.2. A compact and dense model can be achieved by removing pruned groups and recompiling the model.

3.2.2 AMPERE (2:4) GPU SPARSITY

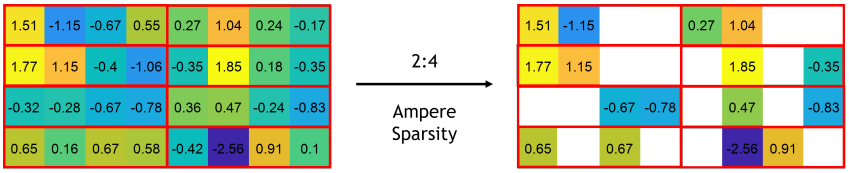

Figure 3: Illustration of 2:4 Ampere sparsity. In every 4 consecutive weight elements the two with the smallest magnitude are set to zero (denoted as blank spaces on the right).

Emerging software and hardware developments have brought support to the acceleration of a more fine-grained sparsity patterns. The recently introduced NVIDIA Ampere GPU architecture supports acceleration of sparse matrix multiplication with a specific pattern of 2:4 sparsity (2 of the 4 consecutive weight elements are zero, see Figure 3). This comes with a limitation of requiring the input and output dimensions of all linear projections to be **divisible by 16** (Mishra et al., 2021). We assure compatibility with such pattern by structurally pruning matrices to have the remaining dimension be divisible by 16 (more details in Appendix A.2). Interestingly, we find that Ampere sparsity can be performed by just magnitude pruning with **no accuracy drop** after the initial pruning is finished.

3.3 TRAINING OBJECTIVE

We next consider the training objective function that supports both pruning for importance ranking and finetuning for weight update. To start with, we inherit the CNN hard distillation training objective as proposed in DEIT (Touvron et al., 2021), which is formulated as follows:

$$\mathcal{L}_{\text{CNN}} = \mathcal{L}_{\text{CE}}\Big(\Psi(z_c^s), Y\Big) + \mathcal{L}_{\text{CE}}\Big(\Psi(z_d^s), Y^{\text{CNN}}\Big), \tag{7}$$

where $\Psi(\cdot)$ denotes softmax and $\mathcal{L}_{\text{CE}}$ the cross entropy loss. We refer to logits computed from the *class token* of the pruned model as z_c^s, and the one computed from the *distillation token* as z_d^s. Note that z_c^s is supervised by the true label Y, while z_d^s is supervised by the output label of a CNN teacher Y^{CNN} to capture the inductive bias of CNN through attention. Unless otherwise stated, we use a pretrained RegNetY-16GF model (Radosavovic et al., 2020) as the teacher, in line with DEIT.

In addition to CNN distillation, we consider *full model distillation* given the unique access to such supervision under the pruning setup. Specifically, the "full model" corresponds to the pretrained model, which serves as the starting point of the pruning process. Ideally a pruned model shall behave similar to its original counterpart. To encourage this, we distill the classification logits from both the class and distillation tokens of the pruned model from the original counterpart, forming Equation (8):

$$\mathcal{L}_{\text{full}} = \mathcal{L}_{\text{KL}}\Big(\Psi(\tau z_c^s), \Psi(\tau z_c^t)\Big) + \mathcal{L}_{\text{KL}}\Big(\Psi(\tau z_d^s), \Psi(\tau z_d^t)\Big). \tag{8}$$

Superscripts ^{t} and ^{s} represent the output of the full pretrained model and the model being pruned respectively. $\mathcal{L}_{\text{KL}}$ denotes the KL divergence loss, and τ is the distillation temperature.

The final objective is therefore composed as: $\mathcal{L} = \alpha\mathcal{L}_{\text{full}} + \mathcal{L}_{\text{CNN}}$. An ablation study of alternating the formulation of the training objective is provided in Appendix B.1.

3.4 PRUNING ANALYSIS ON IMAGENET-1K

We evaluate our pruning method on the challenging ImageNet-1K benchmark, using the DEIT-Base model pretrained with CNN distillation as the starting point. The model size, run time speedup and accuracy of the state-of-the-art methods and our method are compared in Table 1. The training and finetuning hyperparameters can be found in Appendix A.1, and details for our pruning configurations can be found in Appendix A.2.

For best insights, we conduct pruning in four configurations, where we refer to our pruning scheme as Novel ViT Pruning (NVP):

- **NVP-B** aims to match the accuracy of DEIT-B model, which achieves an 1.86× speedup and a 2.57× reduction on FLOPs over DEIT-B with neglectable 0.07% accuracy drop. It also achieves a 2.25× further FLOPs reduction over the more efficient SWIN-B transformer under same accuracy.
- **NVP-H** aims to half the run time latency of DEIT-B, with only 0.4% accuracy loss. It also achieves 1.41× further reduction on FLOPs over SWIN-S with similar accuracy.
- **NVP-S** aims to match the latency of DEIT-S model, with 1% higher accuracy.
- **NVP-T** aims to match the latency of DEIT-T model, with 1.7% higher accuracy.

The model size-accuracy tradeoff also outperforms previous model compression methods like SViTE and AutoFormer by a large margin. Since our pruning scheme supports the utilization of Ampere sparsity on advanced GPU architectures, with the help of Apex ASP (Mishra et al., 2021), an additional 5% speedup can be achieved on our pruned models without further accuracy loss.

Table 1: **Structural pruning results on ImageNet-1K.** Our pruned models are reported in Novel ViT Pruning (NVP) lines. We compare the parameters and FLOPs count (and compression ratio), run time speedup and accuracy of different models. NVP models are compared with DEIT (Touvron et al., 2021), SWIN (Liu et al., 2021), T2T-ViT (Yuan et al., 2021), AutoFormer (Chen et al., 2021a) and SViTE (Chen et al., 2021b). All compression ratios and speedups are computed with respect to that of DEIT-Base model. Latency of NVP are estimated on a single GPU with batch size 256. "ASP" stands for post-training 2:4 Ampere sparsity pruning, enabling 2× parameter reduction and 2× throughput on linear operations with TensorRT (Mishra et al., 2021).

	Size (Compression)		Speedup (×)		
Model	#Para (×)	#FLOPs (×)	V100	RTX 3080	Top-1 Acc.
DEIT-B	86M (1.00)	17.6G (1.00)	1.00	1.00	83.36
SWIN-B	88M (0.99)	15.4G (1.14)	0.95	-	83.30
NVP-B	34M (2.57)	6.8G (2.57)	**1.86**	1.75	83.29
+ ASP	17M (5.14)	6.8G (2.57)	**1.86**	**1.85**	83.29
SWIN-S	50M (1.74)	8.7G (2.02)	1.49	-	83.00
AutoFormer-B	54M (1.60)	11G (1.60)	-	-	82.40
NVP-H	30M (2.84)	6.2G (2.85)	**2.01**	1.89	82.95
+ ASP	15M (5.68)	6.2G (2.85)	**2.01**	**1.99**	82.95
DEIT-S	22M (3.94)	4.6G (3.82)	2.44	2.27	81.20
AutoFormer-S	23M (3.77)	5.1G (3.45)	-	-	81.70
T2T-ViT-14	21.5M (4.03)	6.1G (3.38)	-	-	81.50
SWIN-T	29M (2.99)	4.5G (3.91)	2.58	-	81.30
SViTE	35M (2.49)	7.5G (2.35)	-	-	81.28
NVP-S	21M (4.18)	4.2G (4.24)	**2.52**	2.35	**82.19**
+ ASP	10.5M (8.36)	4.2G (4.24)	**2.52**	**2.47**	**82.19**
DEIT-T	5.6M (15.28)	1.2G (14.01)	5.18	4.66	74.50
AutoFormer-T	5.7M (15.14)	1.3G (13.54)	-	-	74.70
NVP-T	6.9M (12.47)	1.3G (13.55)	4.97	4.55	**76.21**
+ ASP	3.5M (24.94)	1.3G (13.55)	4.97	**4.66**	**76.21**

4 Parameter Redistribution & NViT

4.1 Trends observed in ViT pruning

As observed by Liu et al. (2018), channel/filter pruning in CNN models can provide guidance on finding efficient network architectures, yet this has never been explored on ViT models. Here we show *for the first time* that our pruning method can serve as an effective architecture search tool for ViT models, and more interestingly the inferred design rules are scalable to different model sizes.

As we prune the ViT model to different sizes, we observe the following insights in the way the model architecture changes, as visualized by the gray lines of Figure 4:

- Number of heads, QK of each head and MLP scales *linearly* with the dimension of EMB; while V of each head can be largely kept the same;

- The scaling factors of head, QK and MLP are *not uniform* among all blocks: dimension is larger in the blocks in the middle and smaller towards the two ends;
- The first and last blocks require *larger* dimensions compared to neighboring blocks;
- *Reducing* dimensions related to the multi-head attention (H, QK, V) while *increasing* MLP dimension may lead to more accurate model under similar latency.

Compared to ViT, our insight shows the need to scale QK of each head with the EMB size, and more importantly to allow different transformer blocks in the model to have different dimensions. Interestingly, these trends are not observed in NLP transformer compression (Michel et al., 2019; Voita et al., 2019). Appendix C discusses intuitive hypothesis on why these trends appear in ViT, as we show a similar **less-more-less** trend in the attention score diversity among all heads of each block, which potentially reflects how much redundancy appears in each block.

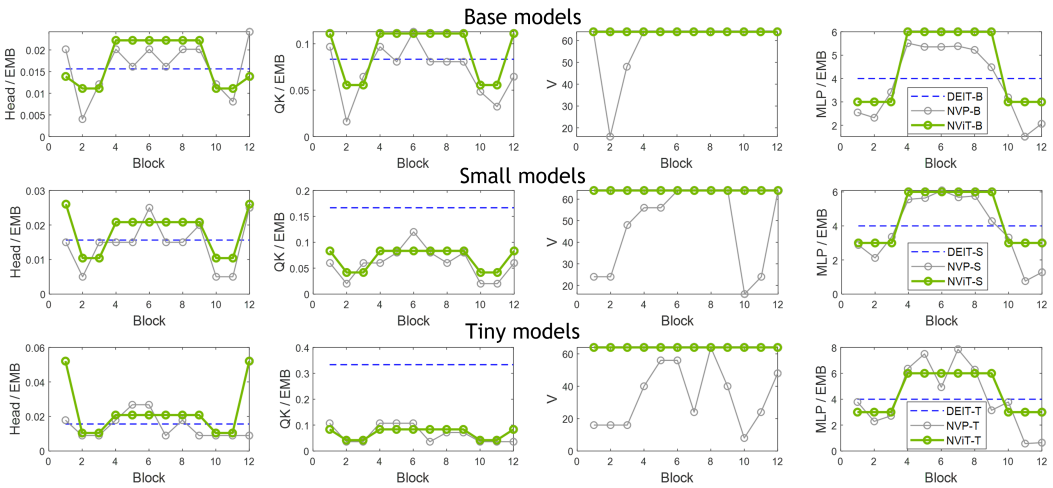

Figure 4: Model dimension comparison between NViT (green), DEIT (blue) and pruned NVP model (grey). Compared to DEIT, the pruned model has more parameters in blocks towards the middle and in MLP; but fewer parameters in blocks towards the two ends and in MSA (H and QK). We follow these insights closely in the design of our NViT models.

4.2 NViT DESIGN AND PERFORMANCE

NViT design rule. Our pruning insights lead to a redistribution of parameters and a new architecture we name *NViT*. We follow the trends in Figure 4 and smooth them out to simplify the architecture, shown with green lines. More specifically, we redistribute the parameters in ViT models in a more efficient manner, using a simplified rule in Table 2 to determine the parameter dimensions of each block based on embedding size EMB and a single scaling factor ϵ. We use EMB as the main driving factor for model scaling, as it leads to the most drastic change in size and performance compared to other structural components, as shown in the single component pruning results in Appendix B.2. Since the first and last attention blocks are typically more important, we set the dimension of them separately. For a 12-layer vision transformer model, we use $\epsilon = 2$ for block 4-9, and use $\epsilon = 1$ for other intermediate blocks. H is rounded to the nearest even number, and QK rounded to the nearest number divisible by 8 to satisfy the dimension requirement of Ampere GPUs.

Table 2: **NViT block dimensions.** For comparison the dimensions of a DEIT block are also listed.

Blocks	H	QK	V	MLP
DEIT	EMB/64	64	64	EMB×4
First/last	10	EMB/10	64	EMB×3
Intermediate	ϵ×EMB/100	ϵ×EMB/20	64	ϵ×EMB×3

Scalability. We next demonstrate the scalability of the proposed NViT architecture through providing a set of variants that match the latency of the models in DEIT family, as specified by the embedding

size in Table 3. Figure 4 compares the dimensions of NViT models vs. the DEIT models vs. our pruned model with a similar accuracy (for base) or latency (for small and tiny), where we show that the NViT architecture well approximates the trend observed in the pruned model.

Comparison. To verify that our NViT parameter redistribution is beneficial, we train all pairs of DEIT and NViT models from scratch on the ImageNet-1K benchmark with the same training objective and hyperparameter choices, which are specified in Appendix A.1.2. As shown in Table 3, our redistribution lets NViT achieve higher accuracy than DEIT with similar FLOPs and lower latency. Specifically, NViT-B, NViT-S and NViT-T achieve a Top-1 accuracy gain of 0.11%, 0.21% and 1.07%, respectively, over their DEIT counterparts.

Table 3: **Comparing NViT models with DEIT model family.** All compression ratios and speedups are computed with respected to that of the DEIT-Base model. Latency was estimated on an RTX 2080 GPU. DEIT accuracy marked with * indicates the train-from-scratch accuracy we achieve from the DEIT GitHub repo[1] using default hyperparameters[2], which is slightly lower than the reported accuracy in DEIT paper. **All pairs of models are trained from scratch with the same hyperparameters from the official DEIT-B script.**

Model	EMB	#Para (×)	#FLOPs (×)	Speedup (×)	Accuracy (%)
DEIT-B	768	86M (1.00)	17.6G (1.00)	1.00	82.99*
NViT-B	720	86M (1.00)	17.6G (1.00)	1.01	**83.10**
DEIT-S	384	22M (3.94)	4.6G (3.82)	2.29	81.01*
NViT-S	384	23M (3.75)	4.7G (3.75)	2.31	**81.22**
DEIT-T	192	5.6M (15.28)	1.2G (14.01)	4.39	72.84*
NViT-T	192	6.4M (13.34)	1.3G (13.69)	4.53	**73.91**

4.3 Downstream tasks

Table 4: **Transfer learning tasks performance with ImageNet pretraining.** We report the Top-1 accuracy of finetuning the ImageNet trained models on other datasets. DEIT-B (Touvron et al., 2021) and ViT-B (Dosovitskiy et al., 2020) results as reported by Touvron et al. (2021) are provided for reference. Speedup evaluated on RTX 2080 GPU with respect to DEIT-B.

Model	CIFAR-10 (%)	CIFAR-100 (%)	iNat-18 (%)	iNat-19 (%)	Speedup (×)
DEIT-B	99.1	91.3	73.7	78.4	1.00
ViT-B/16	98.1	87.1	-	-	1.00
DEIT-T	97.93	85.66	62.41	72.08	4.39
NViT-T	98.18	85.68	63.43	73.73	**4.53**
NVP-T	**98.31**	**85.88**	**64.78**	**74.65**	4.48

Finally, we evaluate the generalization ability of our pruned models and the derived NViTs with transfer learning. Here we finetune the ImageNet trained DEIT, NViT and pruned models on CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), iNaturalist 2018 and 2019 (Van Horn et al., 2018) dataset respectively. The detailed information of these datasets and our training configuration for the downstream tasks are available in Appendix A.4. The transfer leaning results are provided in Table 4. Pruned models consistently outperform the DEIT models on all the tasks, while NViT models achieve a better performance than DEIT with better speedup. These observations show that the efficiency demonstrated on ImageNet can be preserved on downstream tasks.

5 Conclusions

This work proposes a latency-aware global pruning framework that provides significant lossless compression on DEIT-Base model, facilitating the finding of simple design heuristics for novel efficient vision transformers (NViT) with consistent performance improvements. We hope this work opens up a new way to better understand the contribution of different components in the ViT architecture, and inspires more efficient ViT models.

[1] Available at `https://github.com/facebookresearch/deit`.

[2] As in Table 9 of Touvron et al. (2021).

REPRODUCIBILITY STATEMENT

We have included details on how to reimplement our work. For structural pruning experiments, we provide the hyperparameter choices in Appendix A.1.1, and the implementation details with link to full model checkpoints in Appendix A.2. Details of the latency lookup table are included in Appendix A.3. The NViT models have all dimensions specified in Section 4.2, with training hyperparameters available in Appendix A.1.2. The dataset details and hyperparameters used in our downstream task experiments are available in Appendix A.4.

REFERENCES

Maximiliana Behnke and Kenneth Heafield. Losing heads in the lottery: Pruning transformer attention in neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2664–2674, 2020.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Akshay Chawla, Hongxu Yin, Pavlo Molchanov, and Jose Alvarez. Data-free knowledge distillation for object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3289–3298, 2021.

Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. *arXiv preprint arXiv:2107.00651*, 2021a.

Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *arXiv preprint arXiv:2106.04533*, 2021b.

Ting Chen, Ji Lin, Tian Lin, Song Han, Chong Wang, and Denny Zhou. Adaptive mixture of low-rank factorizations for compact neural modeling, 2019. URL `https://openreview.net/forum?id=r1xFE3Rqt7`.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Xiaohan Ding, Guiguang Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, and Ji Liu. Global sparse momentum sgd for pruning very deep neural networks. *arXiv preprint arXiv:1909.12778*, 2019.

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

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Ben 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. *arXiv preprint arXiv:2104.01136*, 2021.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pp. 1379–1387, 2016.

Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074*, 2021.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. *arXiv preprint arXiv:2102.03334*, 2021.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

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. *arXiv preprint arXiv:2103.14030*, 2021.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Jiachen Mao, Huanrui Yang, Ang Li, Hai Li, and Yiran Chen. Tprune: Efficient transformer pruning for mobile devices. *ACM Transactions on Cyber-Physical Systems*, 5(3):1–22, 2021.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *arXiv preprint arXiv:1905.10650*, 2019.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428–10436, 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A Smith. The right tool for the job: Matching model and instance complexities. *arXiv preprint arXiv:2004.07453*, 2020.

Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul Whatmough, Alexander M Rush, David Brooks, et al. Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference. *arXiv preprint arXiv:2011.14203*, 2020.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021.

Alan Weiser and Sergio E Zarantonello. A note on piecewise linear and multilinear table interpolation in many dimensions. *Mathematics of Computation*, 50(181):189–196, 1988.

Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6881–6890, 2021.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *arXiv preprint arXiv:2006.04152*, 2020.

Mingjian Zhu, Kai Han, Yehui Tang, and Yunhe Wang. Visual transformer pruning. *arXiv preprint arXiv:2104.08500*, 2021.

A PRUNING AND TRAINING DETAILS

A.1 TRAINING HYPERPARAMETERS

In our experiments, we use the same data preprocessing, data augmentation, optimizer setup, and learning rate scheduling scheme as mentioned in Table 9 of the DEIT paper (Touvron et al., 2021), unless otherwise mentioned in the following sections.

A.1.1 PRUNING AND FINETUNING

For pruning and finetuning we use the training objective $\mathcal{L} = \alpha\mathcal{L}_{\text{full}} + \mathcal{L}_{\text{CNN}}$ to update the model. We set the balancing factor $\alpha = 1 \cdot 10^5$ and full model distillation temperature $\tau = 20$. The pruning process is performed starting from the pretrained DEIT-Base model, with a fixed learning rate of $0.0002 \times \frac{\text{batchsize}}{512}$. We perform the pruning experiments on the cluster of four NVIDIA V100 32G GPUs, with a batch size of 128 on each GPU. We prune the model continuously until a targeted latency is reached, which is discussed in detail in Appendix A.2. Followed by the iterative pruning we remove the pruned away dimensions of the pruned model to turn it into a small and dense model, and continue to finetune the small model to further recover accuracy. Entire finetuning is performed for 300 epochs with an initial learning rate of $0.0002 \times \frac{\text{batchsize}}{512}$, cosine learning rate scheduling and no learning rate warm up. The finetuning is performed on a cluster of 32 NVIDIA V100 32G GPUs, with a batch size of 144 on each GPU.

A.1.2 NVIT EXPERIMENTS

For the experiments on NViT models we use the CNN hard distillation objective as in Equation (7) as the training objective for all the models. We train Each pair of comparable DEIT and NViT models with the same set of hyperparameters. In all experiments, we train the model from scratch for 300 epochs with an initial learning rate of $0.0005 \times \frac{\text{batchsize}}{512}$, cosine learning rate scheduling and 5 epochs of learning rate warm up. The models are trained on a cluster of 16 V100 32G GPUs, with a batch size of 48 on each GPU for base models and a batch size of 144 on each GPU for small and tiny models.

A.2 PRUNING CONFIGURATION

Table 5: Pruning configurations and remained dimensions for models reported in Table 1. The reported dimensions are averaged across all the blocks.

			Avg. dim remained				
Model	Target speedup	Pruning steps	EMB	H	QK	V	MLP
DEIT-B	N/A	0	768	12	64	64	3072
NVP-B	1.85×	480	496	8.00	35.33	58.67	1917.3
NVP-H	2.00×	524	480	7.33	32.67	56.67	1816.0
NVP-S	2.56×	642	400	5.83	24.00	47.33	1557.3
NVP-T	5.26×	908	224	3.17	14.67	34.00	930.67

We use DEIT-Base model with CNN distillation as the starting point of our pruning process, whose pretrained model is available at `https://dl.fbaipublicfiles.com/deit/deit_base_distilled_patch16_224-df68dfff.pth`. We prune the model in an iterative manner: We compute the moving average of the latency-aware importance score $\mathcal{I}_S^L$ for all unpruned dimension groups in each training step of the pruned model. Every 100 steps, we remove a group of dimensions that has the minimum total importance. Removed dimensions will never be reactivated. We prune EMB and MLP in a group size of 16, QK and V in a group size of 8, and H in a group size of 2, so that the input and output dimensions of all the linear projection operations in the model can be divided by 16, thus satisfying the dimension requirement of the Ampere GPU.

Given that EMB dimension is shared across all 12 blocks, we divided the Taylor importance of EMB by 12 to make the importance comparable with other components. We also divide the Taylor

importance of H dimension by 6 to encourage head pruning in the model. An ablation study on the effectiveness of scaling down head importance is provided in Appendix B.4.

The pruning process will terminate once the estimated latency of the model reaches a targeted speedup ratio over that of the DEIT-base model. The pruned model will then be converted into a small dense model and finetuned to further restore the accuracy.

Table 5 reports the target speedup ratio we use to achieve NVP-B, NVP-H, NVP-S and NVP-T architectures reported in Table 1. The resulted number of pruning steps and the averaged dimension of EMB, H, QK, V and MLP among all the transformer blocks are also provided.

A.3 LATENCY LOOKUP TABLE PROFILING DETAIL

We use a latency lookup table to efficiently evaluate the latency of the pruned model given all its EMB, H, QK, V and MLP dimensions. We initialize the lookup table by profiling the latency of a single vision transformer block on a V100 GPU with batch size 576. We evaluate the latency through a grid of:

- EMB: 0, 256, 512, 768 (latency assigned as 0 at zero EMB);
- H: 1, 3, 6, 9, 12;
- QK: 1, 16, 32, 48, 64;
- V: 1, 16, 32, 48, 64;
- MLP: 1, and 128 to 3072 with interval 128;

resulting into 9375 configurations in total. We run each configuration for 100 times and record the median latency value in the lookup table. For a block with arbitrary dimensions, its latency is estimated via a linear interpolation of the lookup table, which we implement with the *RegularGridInterpolator* function from *SciPy* (Weiser & Zarantonello, 1988; Virtanen et al., 2020). The estimated latency of the entire model is computed as the sum of the estimated latency of all the blocks, while omitting the latency of the first projection layer and the final classification FC layer.

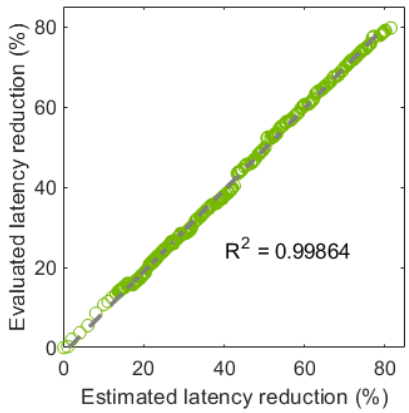

Figure 5: Estimated latency from the lookup table vs. evaluated latency on V100 GPU with batch size 256. Reduction ratio computed with respect to the latency of the full model.

To show the usefulness of the lookup table, we compare the estimated and evaluated latency of different model architectures in Figure 5. Each point represent the model achieved from a pruning step towards *NVP-T* configuration (See Appendix A.2). The estimated latency and evaluated latency of ViT demonstrate **strong** linear relationship throughout the pruning process, with $\mathcal{R}^2 = \mathbf{0.99864}$. This enables us to accurately estimate the latency improvement brought by removing each group of dimensions, and to use the estimated speedup of the pruned model as the stopping criteria of the pruning process.

A.4 DOWNSTREAM TASK DETAILS

The details of the datasets used for our downstream task transfer learning experiments are provided in Table 6.

Table 6: Datasets used for downstream task experiments.

Dataset	Train size	Test size	# Classes
CIFAR-10 (Krizhevsky & Hinton, 2009)	50,000	10,000	10
CIFAR-100 (Krizhevsky & Hinton, 2009)	50,000	10,000	100
iNaturalist 2018 (Van Horn et al., 2018)	437,513	24,426	8,142
iNaturalist 2019 (Van Horn et al., 2018)	265,240	3,003	1,010

Similar to the experiment setting of DEIT (Touvron et al., 2021), for downstream task experiments we rescale all the images to 224×224 to ensure we have the same augmentation as the ImageNet training. All models are trained for 300 epochs with a initial learning rate of $0.0005 \times \frac{\text{batchsize}}{512}$, cosine learning rate scheduling and 5 epochs of learning rate warm up. We use batch size 512 for CIFAR-10 and CIFAR-100 models, and batch size 1024 for iNaturalist models.

B ADDITIONAL ABLATION STUDIES

B.1 TRAINING OBJECTIVE

As discussed in Section 3.3, we propose to use a combination of full model distillation and CNN hard distillation as the final objective of our pruning and finetuning process. Here we ablate the validity of this choice and compare the finetuning performance achieved with removing one or both distillation loss from the objective. Specifically, we consider the following 4 objectives:

- Proposed objective: $\mathcal{L} = \alpha \mathcal{L}_{\text{full}} + \mathcal{L}_{\text{CNN}}$;
- CNN distillation only: $\mathcal{L}_{\text{CNN}}$ as in Equation (7);
- Full model distillation with cross-entropy: $\mathcal{L}_{\text{full}} + \mathcal{L}_{\text{CE}}\left(\Psi\left(\frac{z_c^s + z_d^s}{2}\right), Y\right)$;
- Cross-entropy only: $\mathcal{L}_{\text{CE}}\left(\Psi\left(\frac{z_c^s + z_d^s}{2}\right), Y\right)$.

We use each of the 4 objectives to finetune the pruned model achieved with NVP-T configuration, and report the final Top-1 accuracy in Table 7. The finetuning is performed for 50 epochs, with all other hyperparameters set the same as described in Appendix A.1.1. The proposed objective achieves the best accuracy.

Table 7: NVP-T model finetuning accuracy with different objectives.

Objective	Proposed	CNN	Full model	CE only
Top-1 Acc.	**73.55**	73.40	72.62	72.36

B.2 PRUNING INDIVIDUAL COMPONENTS

In this section we show the result of pruning EMB, MLP, QK and V component individually. The pruning procedure and objective are almost the same as described in Section 3.2, other than here we only enable the importance computation and neuron removal on a single component. The pruning interval of EMB, MLP, QK and V are set to 1000, 50, 200 and 200 respectively, in order to allow the model to be updated for similar amount of steps when pruning different components to the same percentage. 32 neurons are pruned for each pruning step. We stop the pruning process and finetune the model for 50 epochs after the targeted pruned away percentage is reached.

The compression rate and accuracy achieved by pruning each component are discussed in Table 8. Under similar pruned away ratio, we can observe that pruning EMB leads to the most significant compression on the parameter and FLOPs count, as well as the largest drop in accuracy. This implies that the embedding dimension leads to the most effective exploration on the compression-accuracy tradeoff, which motivates us to use EMB as the key driving factor in designing our NViT scaling rule in Section 4.1.

Table 8: Iterative pruning single component to targeted percentage.

Component	Pruned away	Para (×)	FLOPs (×)	Top-1 Accuracy
Base	0%	1	1	
EMB	50%	1.98	1.92	79.24
MLP	50%	1.49	1.47	82.13
QK	50%	1.09	1.10	82.98
V	50%	1.09	1.10	82.63
EMB	70%	2.95	2.77	73.15
MLP	75%	1.97	1.91	80.29
QK	75%	1.14	1.16	82.64
V	75%	1.14	1.16	81.51

B.3 Effectiveness of latency-friendly pruning scheme

We also illustrate the benefit of explicitly single out the head dimension and align the dimensions of each head in structure pruning. We show the tradeoff curve between parameter/latency reduction and the accuracy achieved with or without explicit head alignment in Figure 6. For models pruned without head alignment, we estimate their latency as if each head are padded with zeros to have the same QK and V dimensions during inference. Under a similar accuracy target, though models achieved without head alignment can have a larger model size compression rate, the associated latency speedup is smaller than that achieved with our proposed head-aligned pruning scheme.

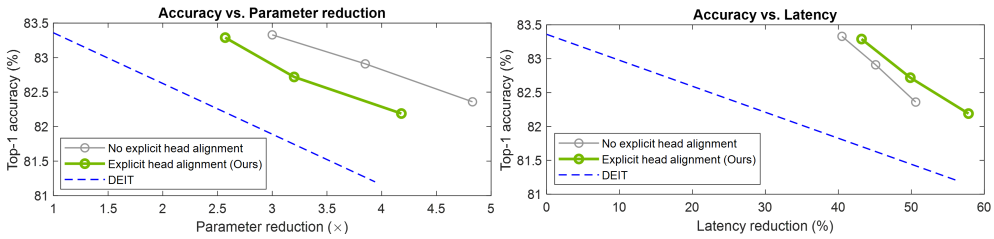

Figure 6: Comparing the parameter reduction-accuracy tradeoff and latency reduction-accuracy tradeoff of different pruning schemes. Latency estimated on RTX 2080 GPU. Model size compression rate and latency reduction rate are computed based on that of the DEIT-Base model respectively.

B.4 Encouraging head pruning in the global pruning

As discussed in the pruning procedure, we propose to divide the Taylor importance of the head dimension by 6, in order to encourage more heads to be pruned. This decision is made based on the observation that encouraging head pruning can enable having a larger pruned model under the same latency budget, which can often lead to better accuracy for the pruned model.

In this section we illustrate this observation by exploring different manipulation of head dimension importance in the global pruning process. Here we apply the proposed global latency-aware structural pruning method to prune the model until its estimated latency reaches 45% of that of the full DEIT-Base model. Table 9 shows the resulted model size, latency and accuracy of the pruned model under different head importance scaling, As we scale down the Taylor importance of the head dimension,

we encourage more heads to be pruned away, leading to the model reaching the latency target in lesser pruning steps. This in turn enables the pruned model to have a higher embedding size, and allow the model to have more parameters and FLOPs within the latency budget, potentially reaching a higher accuracy. Meanwhile, pruning the head too much will also hurt the accuracy as it largely reduces the feature diversity learned by the model. We empirically find that dividing the Taylor importance of head dimension by 6 leads to the highest accuracy under the same latency budget, thus we utilize the scaling factor 1/6 in our pruning method.

Table 9: Pruned model with different head Taylor importance manipulation. The dimension remained is averaged across all attention blocks. The latency is estimated on a single RTX 2080 GPU with batch size 256. We report the accuracy of the pruned model without any further finetuning.

		Dim remained		Compression (×)			
H scale	Pruning steps	EMB	H	#Para	#FLOPs	Top-1 Acc.	Lat. (s)
1/2	725	352	11.3	4.36	4.20	76.91	0.3123
1/3	714	368	10.3	4.18	4.05	78.11	0.3101
1/4	671	400	9.5	3.77	3.69	78.43	0.3177
1/6	573	464	6.5	3.20	3.22	**78.76**	0.3195
1/8	509	480	5.0	3.07	3.14	78.41	0.3139

C INTUITION BEHIND NVIT MODEL DESIGN

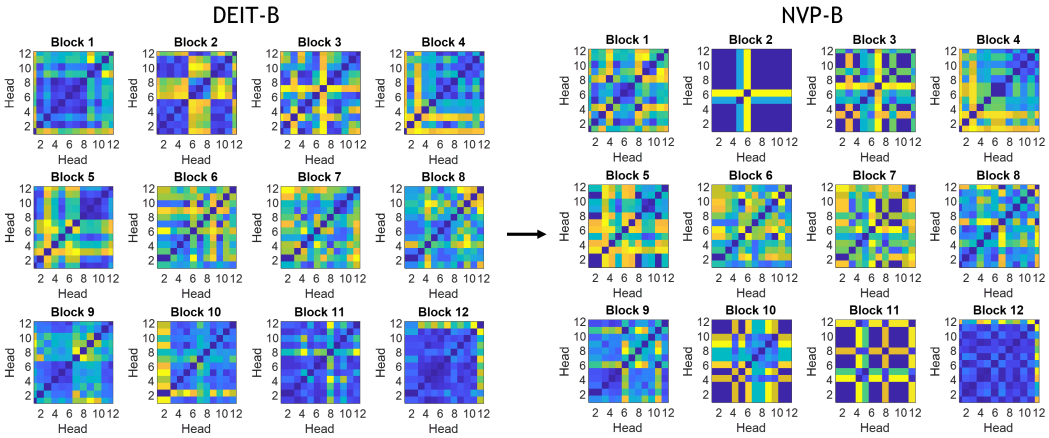

Figure 7: Pair-wise cosine distance between all heads' attention score in each transformer block. Blue indicates a smaller distance while yellow indicates a larger one. The dark blue blocks in NVP-B figures corresponds to the heads being pruned away, which have all-zero attention scores thus zero cosine distance in between.

As we observe in Figure 4 and mentioned in Section 4.1, the pruned models tend to preserve *more* dimensions in the transformer blocks towards the *middle* layers, while having *less* dimensions towards the *two ends* of the model. Here We explore an intuitive analysis on why this trend occurs by observing the diversity of features captured in each transformer blocks. Given the attention computation serves important functionality in ViT models, here we use the diversity of the attention score learned by each head as an example. Specifically, we take a random batch of 100 ImageNet validation set images, pass them through the pretrained DEIT-Base model and our NVP-B model, and record the averaged attention score $softmax\left(\frac{q_h k_h^T}{\sqrt{d_h}}\right)$ of all the images computed in each head h. We then compute the pair-wise cosine distance of the attention score from each head as a measure of diversity, and visualize the results in Figure 7.

In DEIT-B model, we can observe that in earlier blocks like block 2 and later blocks like block 11, there are clear patches of darker blue indicating a group of heads having attention scores similar to

each other. While for blocks in the middle such as block 5-8, almost all pairs of heads appear to be fairly diverse. Such difference in diversity leads to different behavior in the pruning process, where less heads are preserved in earlier and later blocks while more are preserved in the middle. Note that all remaining heads in NVP-B model appears to be diverse with each other, showing a more efficient utilization of the model capacity. Interestingly, this less-more-less trend of dimensional change across different transformer is not observed in previous works compressing BERT model for NLP tasks (Voita et al., 2019; Michel et al., 2019; Mao et al., 2021). The learning dynamic of ViT model leading to this trend is worth investigating in the future work.