POLITECNICO DI TORINO

Master degree course in Data Science and Engineering

Master Degree Thesis

# Structured Pruning of Vision Transformers at Training Time

**Supervisors**
Prof. Daniele Jahier Pagliari
Dr. Matteo Risso
Dr. Alessio Burello
Dr. Beatrice Alessandra Motetti

**Candidate**
Leonardo Tredese

Academic year 2022-2023

# Abstract

Attention-based transformers have emerged as a powerful paradigm achieving state-of-the-art results on tasks such as natural language processing and computer vision. However, transformers typically present higher computational costs and parameter count compared to convolutional networks. This inefficiency impedes deploying transformers to resource constrained devices such as edge devices. Structured pruning techniques present a promising direction to compress transformers for the edge computing scenario.

This thesis investigates pruning techniques to induce structured sparsity in vision transformers, thereby reducing computational requirements while minimizing accuracy degradation. The goal is developing methodologies for efficient vision transformer inference.

Structured pruning learns importance scores for individual network components at training time by solving an optimization problem that tries to maximize task performance while minimizing the number of parameters in the model. The importance scores are then transformed into binary masks that prune unimportant structures such as specific linear layers output dimensions or entire attention heads.

To promote regularity in the induced sparsity patterns, various mask sharing strategies are proposed that couple pruning decisions across related architectural elements. Regularity is crucial because complete independence precludes the removal of some masked components due to the specific connectivity pattern of transformers, leading to lower compression rates when the model is actually deployed on hardware.

Empirical results demonstrate that in image classification tasks completely independent masking of components outperforms sharing strategies in terms of balancing accuracy and sparsity. None the less, experiments show that through a mix of shared and independent masks, the proposed pruning scheme successfully compresses vision transformers as much as 90% with an accuracy drop of just 4% or 70% compression rate with less than 1% in accuracy drop.

# Acknowledgments

I would like to express my sincere appreciation to Professor Daniele Jahier Pagliari, Matteo Risso, Alessio Burello, and Beatrice Alessandra Motetti for their invaluable guidance and support throughout this thesis project. I am deeply grateful for their mentorship.

In particular, I want to thank them for always making themselves available to discuss research directions and review drafts. Their diverse perspective enabled me to overcome obstacles and refine technical aspects of the work. Or as mr. La Rochelle would say: "thank you for being not so Italian".

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Transformers have emerged as a disruptive architecture for sequence modeling tasks. First introduced for language translation in 2017, the transformer utilizes multi-headed self-attention(MHSA) rather than recurrent or convolutional layers [1]. This allows the model to learn contextual relationships between all elements in an input sequence, leading to highly performing representations compared to prior sequence models like LSTMs. The transformer architecture obtained state-of-the-art results on large-scale language tasks and became the de-facto choice for Natural Language Processing (NLP).

Given this immense success in NLP, there has been significant interest in adapting transformers to other fields like computer vision. Convolutional neural networks (CNNs) have been the dominant model architecture for computer vision tasks for many years, achieving state-of-the-art results on image classification until the end of 2020 when they have been surpassed by transformers [2] (Figure 1.1). While CNNs capture local spatial relationships effectively in images, they lack mechanisms for modeling global context. Transformers are capable of capturing long-range dependencies in vision tasks with no regard to spatial proximity. Early vision transformers like ViT [3] obtained state of the art results on image classification by applying transformers to sequenced image patches. However, self-attention computation scales quadratically with sequence length, resulting in very high memory and computational costs for these architectures. For example, Vision Transformers like ViT have billions of floating point operations due to their self-attention mechanism applied to high-resolution images. A number of techniques have been proposed to reduce this complexity such as sparse attention patterns and model distillation, though efficient transformer design remains an active area of research [4].

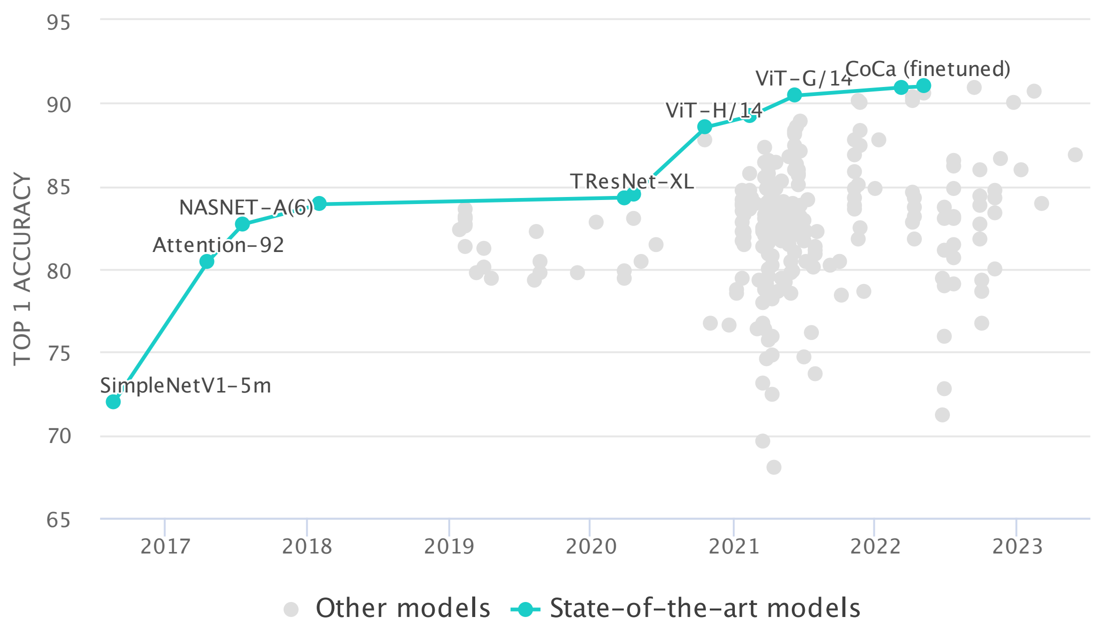

Figure 1.1: Graph that shows the evolution of accuracy for SoTA methods on ImageNet. Before ViT all the best results where achieved by convolutional networks, then the best results are obtained by transformers [2].

While Vision Transformers have achieved excellent accuracy on visual tasks, their immense computational and memory requirements make deployment challenging. For many applications, running inference solely in the cloud is not ideal due to factors like privacy concerns, latency constraints, and connection status. There has been growing interest in executing models directly on edge devices like smartphones, autonomous vehicles, and smart home assistants. Edge computing provides benefits such as data privacy, low latency, energy efficiency, and robust operation without the need of perpetual connectivity [5]. However, the memory capacity on most edge devices is limited, ranging from the order of kilobytes to few megabytes. For example, the GAP-8 microcontroller has around 512KB of state retentive memory[6] while Coral Dev Board Micro reaches 128MB of flash memory[7]. In contrast, a ViT model may have over 100 million floating point parameters requiring over 400MB of storage. This vast gap makes it infeasible to run most transformer models on current edge hardware.

One approach to enable deployment on resource-constrained edge devices is model compression techniques like pruning. Pruning techniques aim to address transformer inefficiencies by reducing model size while minimally impacting performance. Structured pruning directly removes entire structures like filters, blocks or attention heads to improve efficiency. This thesis explores structured pruning techniques applied to the transformer architecture and vision domain.

In particular, this thesis considers structured pruning methods based on differentiable masks that are learned by minimizing a loss function that combines model error on a task and the number of parameters in it. This approach enables compressing vision transformers without sacrificing their representational power. The main contributions of this work are the following:

- Design of a transformer specific structured pruning algorithm to remove low-importance structures at different granularity. Namely: attention heads, head dimension, fully connected layers and convolution layers.
- Experimental evaluation of approach on image classification benchmarks including CIFAR-10 and Tiny ImageNet.
- Analysis of trade-offs between efficiency and accuracy for varying pruning compression rates, and granularity and pruning strategies taken from literature.

Our proposed method proved to be effective with the following results:

- 90% model compression with an accuracy drop of 4% or 70% compression rate with less than 1% in accuracy drop on CIFAR-10.
- 90% model compression with an accuracy drop of 25% , 25% compression rate with 7% accuracy drop or 5% model compression for a 2% accuracy drop on Tiny ImageNet.

Analysing all the outcomes of this work will inform best practices for compressing vision transformers in resource-constrained environments.

The remainder of this thesis is organized as follows: Chapter 2 provides a comprehensive review of related work on transformer architectures, vision transformers, and neural network pruning. Chapter 3 introduces the proposed structured pruning algorithm. Chapter 4 describes the experimental setup and results. Finally, Chapter 5 discusses conclusions and directions for future work.

# Chapter 2

# Related Work

This chapter provides background on transformer architectures, vision transformers, and neural network pruning techniques. First, we introduce the transformer architecture and key components like MHSA. Next, we discuss adaptations of transformers to computer vision. Finally, we review prior work on pruning methods.

## 2.1 Transformers

The transformer architecture was introduced by Vaswani et al. in 2017 [1] for neural machine translation. It pioneered a self-attention only approach , differently from previous works based on recurrent layers [8]. Using only self-attention allows for more parallelization and gives the ability to capture long range dependencies in inputs. The core components include MHSA, feed forward networks (FFN), layer normalization, and residual connections as shown in figure 2.1.

MHSA allows relating different elements of an input sequence to compute representations reflecting context. Scaled dot-product attention computes compatibility scores between such sequence elements. Multiple heads learn distinct representation subspaces focused on different contextual interactions. Moreover computation for different heads can be parallelized to improve performances.

Formally, given an input sequence $X = (x_0, ..., x_L) \in \mathbb{R}^{L \times d}$ of length $L$ and dimension $d$, attention is defined as:

$$\text{Attention}(Q, K, V) = \text{Softmax}_{\text{row}}\left(\frac{QK^\top}{\sqrt{d_{QK}}}\right)V \tag{2.1}$$

where $Q$, $K$, $V$ stand for for the queries, keys, and values respectively, and they are all computed from the input $X$. $d_{QK}$ is the dimension of the keys $K$ and queries $Q$.

The MHSA operation can be formally described by:

$$\text{MHSA}(X) = \text{Concat}(\text{head}_0, \ldots, \text{head}_h)W_O \tag{2.2}$$

where each head $h$ is computed as:

$$\text{head}_h = \text{Attention}(XW_Q^h, XW_K^h, XW_V^h) \tag{2.3}$$

and $W_Q^h$, $W_K^h$, $W_V^h$ and $W_O$ are learnable parameters of the model.

The self-attention mechanism, utilized in the transformer architecture, exhibits inherent permutation invariance across the input sequence. Specifically, the attention scores calculated between elements do not depend on their positional order. This invariance emerges due to the reliance on inner products between token embeddings without regard for sequence order.

To incorporate information about element positions, the transformer architecture incorporates dedicated positional embeddings that are summed with the input token embeddings. These positional embeddings are vector representations associated with each position in the input sequence. Prior work has explored both fixed positional encodings [1] as well as learned positional embeddings tuned during training [3]. The integration of positional information via summation enables the self-attention layers to jointly leverage representations of content from the token embeddings and representations of position from the positional embeddings. Overall, this provides the transformer with a global receptive field.

The original architecture is then divided into encoder and decoder stack. The encoder maps an input sequence to a continuous representation. The decoder then generates an output sequence from this representation.

Specifically, the encoder stack consists of N identical modules. Each layer has two sub-modules: a MHSA module and a FFN module. Residual connections and layer normalization are employed around each sub-module.While the decoder modules also include an additional attention layer that attends to the encoder outputs to focus on relevant context.

Transformers achieved state-of-the-art results on large-scale language tasks, outperforming prior sequence models like recurrent networks [1, 8]. Follow-up work expanded transformer applications from machine translation to language modeling, question answering and other NLP tasks [1, 9, 10].

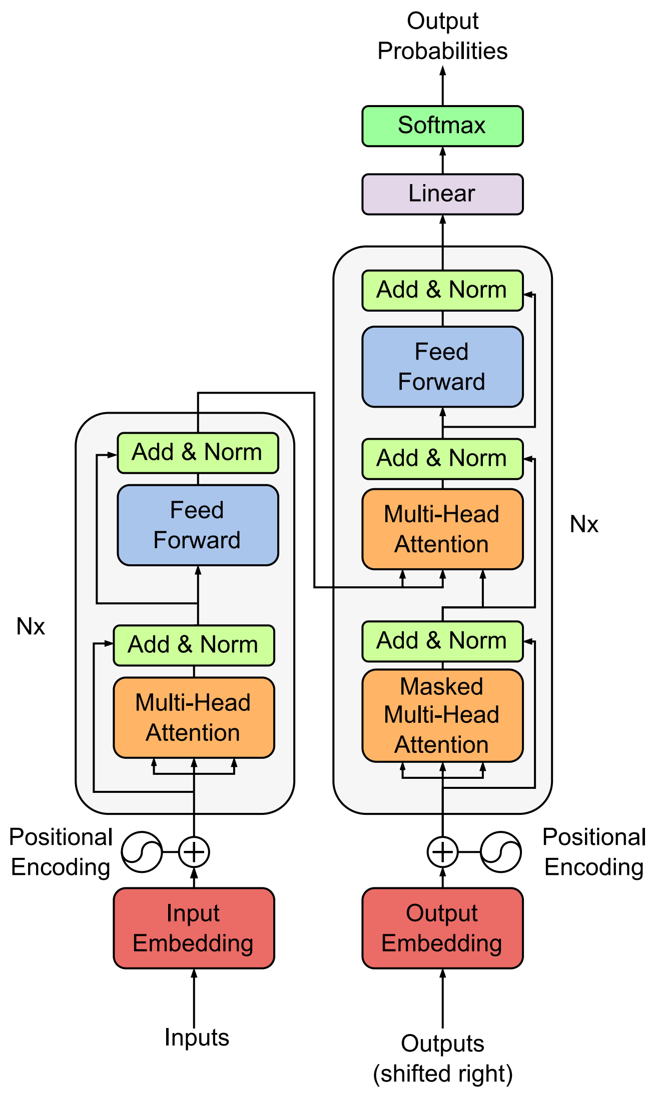

Figure 2.1: Graphical representation of the original transformer architecture.

## 2.2 Vision Transformers

Given the immense success of transformers in NLP, researchers began exploring possible adaptations to computer vision tasks. The stand-alone vision transformer (ViT) directly applied transformers to sequences of image patches [3].

As shown in figure 2.2, ViT splits an input image into fixed-size non-overlapping patches which are treated as token sequences. Then, a standard transformer encoder-only architecture is applied to these patch tokens. To represent the entire image, an extra learnable classification token is added to the sequence. Learnable positional embeddings are also injected to retain spatial information. The output corresponding to the class token is fed into a linear head to make predictions. Without the use of convolutional layers, ViT demonstrated the potential of pure transformer models for image tasks.

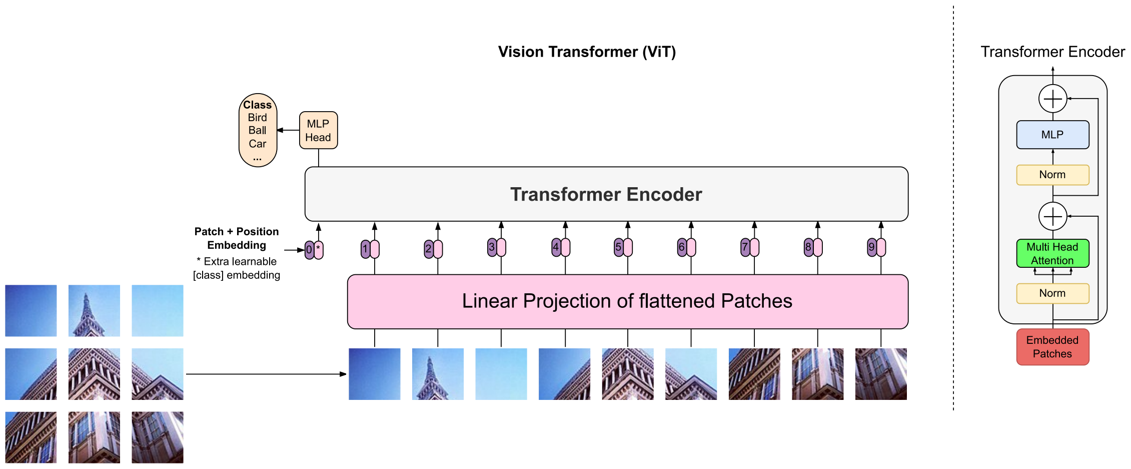

Figure 2.2: Graphical representation of the ViT architecture.

| **Model** | **Top-1 ImageNet Accuracy** | **Parameters** |
|---|---|---|
| ViT | 87.76% | 300M |
| EfficientNet B7 | 87.1% | 66M |
| DeiT | 84.2% | 87M |

Table 2.1: Comparison of accuracy and size for models trained on ImageNet

Although ViT achieves SoTA results it is less parameter-efficient then CNNS for image classification. As shown in table 2.1, ViT obtained an accuracy of 87.76% on ImageNet compared to 87.1% for EfficientNet B7 [3, 11]. However, ViT contained over 300 million parameters, while EfficientNet B7 contained only 66 million. This highlights the inefficiency of standard transformers for computer vision. Moreover ViT architecture requires extensive complex training protocols, hyperparameter tuning and optimization to reach convergence.

Follow-up vision transformers like DeiT closed this gap through distillation techniques. For instance, DeiT leveraged knowledge transfer from a teacher CNN during pretraining. DeiT reached 84.2% accuracy on ImageNet with just 87 million parameters, highlighting improved efficiency[12].

## 2.3 Neural Network Pruning

Pruning techniques aim to reduce model size by eliminating non-critical parameters from the network architecture. A simple approach is magnitude-based pruning, which removes weights if their absolute values are below a defined threshold [13]. However, this produces unstructured sparsity that cannot directly improve computational efficiency on traditional hardware. A possible benefit is that storage of the compressed pruned model parameters is actually reduced.

Sharper techniques prune entire structures within the network . For example, the L1 or L2 norm of the weights within each filter, channel or head can be calculated. Then structures with the lowest overall magnitude according to this metric are pruned away entirely. This creates structured sparsity aligned with efficiency gains in computation.

More advanced methods incorporate first-order information to determine pruning importance. For example, the authors of [14] employ the gradient magnitude of masking variables of attention heads with respect to the loss as a saliency metric. Attention heads with minimal impact on the loss are pruned. While pruning at the head-level provides some structure, it may not align with computational efficiency gains. Specifically, standard attention heads parallelization schemes may prevent exploiting this sparsity to increase inference speed.

The previously introduced pruning techniques operate post-training, removing parameters from an already optimized model. However, recent work has explored pruning during training. Movement pruning [15] associates a learnable score $S_{i,j}$ to each weight $W_{i,j}$. Such score is used to compute binary masks $M_{i,j}$ setting low scoring weights to zero. So the activation values $a_i$ in the forward pass are calculated as:

$$a_i = \sum_{k=1}^{n} W_{i,k} M_{i,k} x_k$$

And the scores are updated with the approximation of the gradient with respect to the loss $\mathcal{L}$:

$$\frac{\partial \mathcal{L}}{\partial S_{i,j}} = \frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial a_i}{\partial S_{i,j}} = \frac{\partial \mathcal{L}}{\partial a_i} W_{i,j} xj$$

This is an approximation because the binary conversion from score to mask using threshold $\tau$ is computed as $M_{i,j} = S_{i,j} \geq \tau$ , which is non-differentiable.

To overcome this issue, the straight-through estimator [16] is adopted. In the forward pass the function is computed normally, in our case the binary conversion function. I In the backward pass it approximates the gradient function by substituting it with the identity. This allows to prune and learn weights during training by propagating gradients through the discrete masking operation. A final key factor of this method is that the loss function is given by the sum of 2 components: a task loss $\mathcal{L}_{\text{task}}$ and a regularization $R(S)$ component corresponds to the sum of all the scores.

Movement pruning was originally proposed as an unstructured method, but then it was adapted to structured pruning and even to transformers [17]. In such case a single score is associated to multiple weights. In this way weights belonging to the same group are masked all together, therefore providing structured pruning.

This thesis aims to advance efficient vision transformer design through structured movement pruning. I propose techniques to prune attention heads and other components based on importance scores optimized during training. The next chapter details this methodology which was implemented using the PLiNIO [18] library. This library provides ready to use high level components for linear and convolutional layers, and also lower level components that allow the creation of new pruning strategies for unsupported layers such as MHSA.

# Chapter 3

# Proposed Pruning Methodology

This chapter introduces the proposed structured pruning methodology for vision transformers. First, we review the transformer architecture and discuss structured pruning opportunities within the model. Next, we analyze possible combinations of pruning strategies to the multi-headed self-attention layers. Then, we present techniques to prune other components like MLP and Patch Embedding layers in a structured manner.

## 3.1 Transformer Architecture and Structured Pruning

As discussed in chapter 2, the key components of a transformers are MHSA, FFN, and layer normalization. Structured pruning aims to remove either entire modules or substructures to improve efficiency.

For vision transformers like ViT and DeiT, the focus is on the encoder which applies self-attention to image patches. Structured pruning can remove entire attention heads based on importance scores. In FFN layers, whole hidden units can be pruned. Layer normalization blocks can be pruned as well.

Structured pruning can provide direct acceleration by reducing computations. For example, pruning attention heads reduces the dimensionality of the key, value, and query projections. This decreases the overall scaled dot product attention costs.

Here structured pruning is implemented through a training scheme with learnable binary masks. Specifically, in addition to the standard model weights W, binary mask importance scores S are introduced for each prunable structure. The overall optimization problem becomes minimizing both the task loss $\mathcal{L}_{task}(W;S)$ and a masking cost $\mathcal{L}_{cost}(S)$ over the weights W and masks S jointly:

$$\min_{W,S} \mathcal{L}_{task}(W;S) + \lambda \mathcal{L}_{cost}(S) \tag{3.1}$$

where $\lambda$ controls the sparsity regularization strength. The masking cost $\mathcal{L}_{cost}$ encourages sparsity in $S$, which in turn zeros out structures in $W$. This prunes structures based on importance learned during training itself.

## 3.2 Block Movement Pruning for Linear Layers

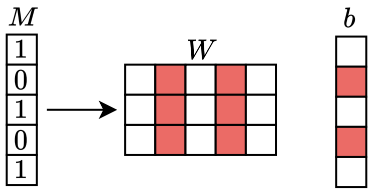

Figure 3.1: Graphical representation of Linear layer pruning

Consider a standard linear layer parameterized by a weight matrix $W \in \mathbb{R}^{n\times m}$ and bias vector $b \in \mathbb{R}^m$. The layer maps an input $x \in \mathbb{R}^n$ to an output $y \in \mathbb{R}^m$:

$$y = Wx + b \tag{3.2}$$

To enable structured sparsity, we partition $W$ into its $m$ columns $W^i \in \mathbb{R}^{n\times 1}$, and $b$ into its components $b^i \in \mathbb{R}$. Let $M^i \in \{0,1\}$ be a binary mask selecting whether column $W^i$ and component $b^i$ are pruned:

$$M^i = \mathbb{1}_{(S^i \geq \tau)} \tag{3.3}$$

where $S^i$ is a learned importance score for the $i^{\text{th}}$ component of $y$ and $\tau \in [0,1]$ is the masking threshold. If $S^i$ is below the threshold $\tau$, multiplying by the mask $M^i = 0$ zeros out the entire column $W^i$ and the component $b^i$ to induce sparsity. A regularization penalty on $S^i$ encourages masking less important output channels. Thanks to the straight-through estimator, gradients relative to $W$ and $b$ are propagated through the discrete masks.

In this setting the deletion of all channels is possible, in such case then the layer would output only the vector $0^m$. If this happens than the network graph becomes disconnected, so all the inputs would lead to the same result and the gradients would propagate only through the part of the network connected to the outputs. To avoid such possibility, we fix $S^i = 1$ for an arbitrary number of output channels, in this way we force $M^i = 1$ therefore impeding the layer from outputting just zeros.

All the masks $M^i$ associated to the layer can be stacked into a vector $M = [M^1; ...; M^m] \in \{0,1\}^m$, so the output is:

$$y = M \odot (Wx + b) \tag{3.4}$$

where $\odot$ represent the element-wise multiplication. This formulation enables block structured pruning of linear layers by masking output channels based on importance. Such importance is learned through a regularization loss $R(n_{in}, S)$, where $n_{in}$ is the number of input features of a layer and $S = [S^1; ...; S^m]$ is the vector of scores for the output of such layers. The loss can be defined as

$$R(n_{in}, S) = (n_{in} + 1) \sum_{i=1}^{m} S^i \tag{3.5}$$

## 3.3 Block Movement Pruning for Convolutional Layers

In the ViT architecture, the first operation is a patch embedding layer consisting of a convolutional projection. This convolution has a kernel size of $k \times k$ and a stride of $k \times k$, mapping the input image into non-overlapping patches which are then flattened into a sequence. As the patch embedding represents the first transform in ViT, it is an important target for pruning to induce sparsity in the model. In this section, we describe a structured pruning approach based on block movement pruning to sparsify the patch embedding convolution in a manner that aligns with computational efficiency.

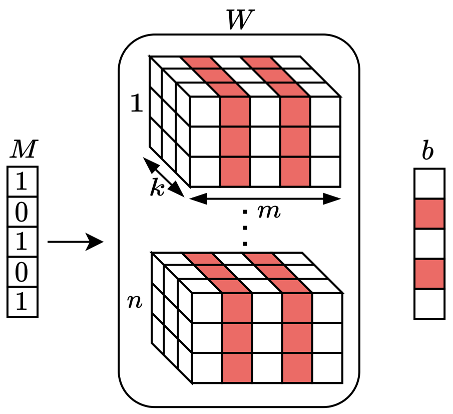

Figure 3.2: Graphical representation of Convolutional layer pruning

Consider a standard convolutional layer parameterized by a weight tensor $W \in \mathbb{R}^{n\times m\times k\times k}$ and bias vector $b \in \mathbb{R}^m$. The layer maps an input feature map $X \in \mathbb{R}^{n\times h\times w}$ to an output $Y \in \mathbb{R}^{m\times h'\times w'}$ (in ViT $h' = \frac{h}{k}, w' = \frac{w}{k}$) via:

$$Y = W * X + b \tag{3.6}$$

$$Y_{i,p,q} = \sum_{l=1}^{n}\sum_{r=1}^{k}\sum_{s=1}^{k} W_{l,i,r,s}X_{l,p+r-1,q+s-1} + b_i \tag{3.7}$$

where $*$ denotes the convolution operation, and the addition of the bias $b$ is broadcasted to $h'$ and $w'$ dimensions.

To enable structured sparsity, we can partition the $m$ output channels of $W$ into blocks. Let $M^c \in \{0,1\}$ be a binary mask selecting whether output channel $c$ is pruned:

$$M^c = \mathbb{1}_{(S^c \geq \tau)} \tag{3.8}$$

where $S^c$ is a learned channel importance score and $\tau \in [0,1]$ is a threshold. Applying $M^c$ allows to zero out output channel $c$ in $W$ and the $c$ component of $b$ to induce sparsity. A regularization penalty on $S^c$ encourages masking unimportant channels. Also in this case the STE, is used to propagate gradients through the discrete masks.

To avoid masking all channels, we fix $S^c = 1$ for some arbitrary channels, forcing $M^c = 1$.

Stacking the masks into $M = [M^1; ...; M^m] \in \{0,1\}^m$, the is computed as:

$$Y = M \odot (W * X + b) \tag{3.9}$$

$$Y_{i,p,q} = \sum_{l=1}^{n} \sum_{r=1}^{k} \sum_{s=1}^{k} M^i (W_{l,i,r,s} X_{l,p+r-1,q+s-1} + b_i) \tag{3.10}$$

where $\odot$ denotes element-wise multiplication broadcasted to dimensions $h'$ and $w'$. This formulation enables block structured sparsity for convolutional layers. Sparsity is learned through a regularization loss $R(n_{in}, S)$, where $n_{in}$ is the number of input features of a layer and $S = [S^1; ...; S^m]$ is the vector of scores for the output of such layers. The loss can be defined as

$$R(n_{in}, S) = (n_{in}k^2 + 1) \sum_{i=1}^{m} S^i \tag{3.11}$$

## 3.4 Block Movement Pruning for MHSA

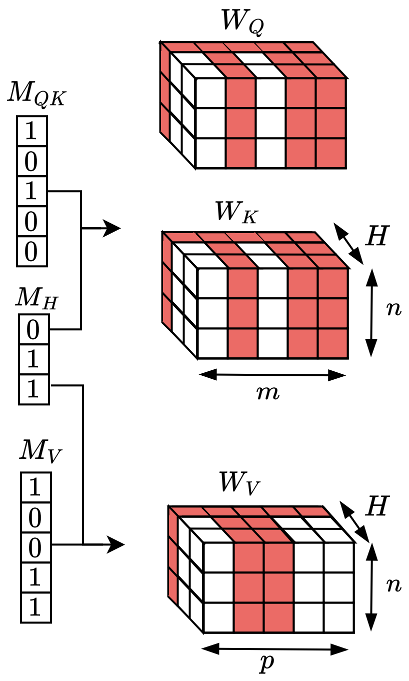

Figure 3.3: Graphical representation of MHSA layer pruning

A core component of this approach involves extending the described pruning of linear layers to transformer self-attention layers. I propose assigning an importance score $S_H^i$ to each attention head $i \in \{1, ..., H\}$. Additionally, importance scores $S_{QK}^j$ and $S_V^z$ are assigned to column of the query/key projections and each column $z$ of the value projection for all heads. The heads and columns with the lowest importance are pruned by masking out the corresponding query, key, and value transformations. Query and key dimensions must match because they are combined with a dot product to calculate attention scores, on the other hand value dimensions are independent from both.

Formally, consider a self-attention layer as defined in equation (2.2) with $H$ heads, query and key dimension $m$ and value dimension $p$. Let $W_Q^h$, $W_K^h \in \mathbb{R}^{n \times m}$ and $W_V^h \in \mathbb{R}^{n \times p}$ represent the query, key, and value projections for an attention head $h \in 1, ..., H$.

$M_H^i \in [0,1]$ is the binary pruning mask for head $i$. $M_{QK}^j \in [0,1]$ is the pruning mask for the $j^{\text{th}}$ column of $W_Q^i, W_K^i \forall i \in \{1, ..., H\}$. $M_V^z$ is the pruning mask for the $z^{\text{th}}$ column of $W_V^i \forall i \in \{1, ..., H\}$

The binary pruning masks $M_H^i, M_{QK}^j, M_V^z$ are derived from continuous importance scores $S_H^i, S_{QK}^j, S_V^z$:

$$M_H^i = \mathbb{1}_{(S_H^i \geq \tau)} \tag{3.12}$$

$$M_{QK}^j = \mathbb{1}_{(S_{QK}^j \geq \tau)} \tag{3.13}$$

$$M_V^z = \mathbb{1}_{(S_V^z \geq \tau)} \tag{3.14}$$

where $\tau \in [0,1]$ is a threshold. Thanks to the straight-through estimator, gradients are propagated through the discrete masks.

Formally attention head $i$ is defined as:

$$Head_i(X) = M_H^i \text{Softmax}_{\text{row}} \left( \frac{(XW_Q^i \odot M_{QK})(XW_K^i \odot M_{QK})^\top}{\sqrt{\sum_{i=1}^m M_{QK}^i}} \right) XW_V^i \odot M_V \tag{3.15}$$

where $M_H = [M_H^1; ...; M_H^H], M_{QK} = [M_{QK}^1; ...; M_{QK}^m], M_V = [M_V^1; ...; M_V^m] \in \{0,1\}^m$ and $\odot$ denotes element-wise multiplication.

A sparsity penalty $R(n_{in}, S_H, S_{QK}, S_V)$ encourages pruning uninformative dimensions. Please note that $S_H = [S_H^1; ...; S_H^m], S_{QK} = [S_{QK}^1; ...; S_{QK}^m], S_V = [S_V^1; ...; S_V^m]$ and $n_{in}$ is the dimension of the input. The penalty is defined as:

$$R(n_{in}, S_H, S_{QK}, S_V, S_O) = n_{in}\sum_{i=1}^{h} S_H^i(2\sum_{i=1}^{m} S_{QK}^i + \sum_{i=1}^{p} S_V^i) + (\sum_{i=1}^{h} S_H^i \sum_{i=1}^{p} S_V^i + 1)\sum_{i=1}^{d} S_O^i \tag{3.16}$$

where $S_O \in \mathbb{R}^d$ are the scores for the output linear projection of the MHSA.

## 3.5 Movement Pruning for Layer Normalization

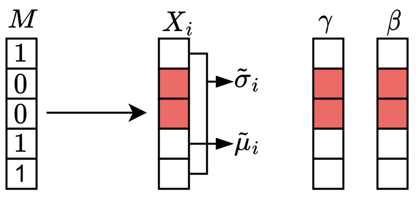

Figure 3.4: Graphical representation of Layer Normalization pruning

Layer normalization (LN) normalizes token representations across channels independently per token. Consider input $X \in \mathbb{R}^{L\times d}$ with $L$ tokens of dimension $d$ .

LN is applied token-wise:

$$\text{LN}(X)_i = \frac{X_i - \mu_i}{\sigma_i}\gamma + \beta \tag{3.17}$$

where $\mu_i$ and $\sigma_i$ are the mean and standard deviation statistics calculated per token $i$ across its $d$ dimensions. While $\gamma, \beta \in \mathbb{R}^d$ are learnable parameters.

To prune dimensions, each has a mask $M^c \in \{0, 1\}$ derived from a score $S^c$:

$$M^c = \mathbb{1}_{(S^c \geq \tau)} \tag{3.18}$$

where $\tau \in$ [0,1] is a threshold. Applying $M_c$ zeros out dimension $c$ in $X$, and must be excluded when calculating the new LN statistics: $\tilde{\mu}, \tilde{\sigma}$. Therefore layer Normalization becomes:

$$\mathrm{LN}(X)_i = \frac{X_i - \tilde{\mu_i}}{\tilde{\sigma_i}} \gamma \odot M + \beta \odot M \tag{3.19}$$

Where $\odot$ is the element-wise multiplication and $M_V = [M^1; ...; M^d] \in \{0,1\}^d$. A sparsity penalty $R(S)$, where $S = [S^1; ...; S^m]$ encourages pruning uninformative dimensions.

$$R(S) = 2 \sum_{i=1}^{d} S^i \tag{3.20}$$

## 3.6 Transformer Architecture pruning strategies

When applying block movement pruning to components like MHSA layers and FFNs in transformers, several approaches can be taken regarding sharing or independence of pruning masks across layers and blocks.

### 3.6.1 Unshared Masking

The first strategy, shown in figure 3.5 is to have all pruning masks independent. Each attention head, projection row, FFN unit, etc. has its own dedicated importance score and mask. This provides maximal flexibility to prune any individual component's output based on importance. However, due to residual connections, output dimensions of attention layers, FFN layers, and patch embedding must match otherwise they cannot be reduced. That is because channels pruned in some layers may be retained in others, preventing removal of pruned weights from the network. Therefore this method is used only as an upper bound for the following strategies.

### 3.6.2 Shared Residual Masking

To enable reduction of layers connected by residuals, their pruning can be coupled for structural consistency (figure ). The patch embedding output, class token, positional embeddings, attention outputs, and FFN outputs share a common mask. Residual connections then carry zeros in masked

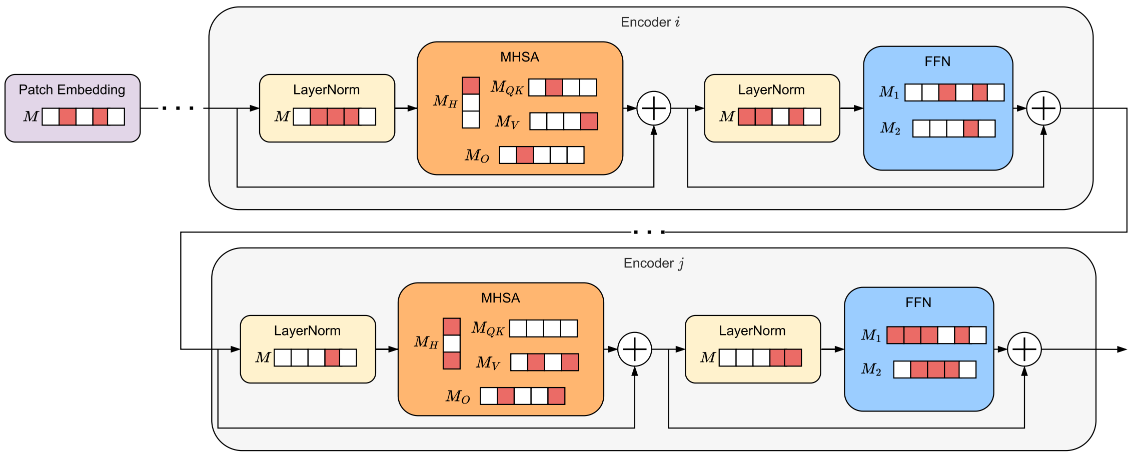

Figure 3.5: Example of unshared pruning, boxes in red are pruned weights.

channels, allowing downstream layers to reduce size by removing them. Internal layers of MHSA and FFNs can still be masked independently for flexibility. However, in shared masking a mask affects many more parameters, therefore masking a single channel will affect much more the cost loss. To avoid pruning only residual-connected layers we need to rebalance the optimization problem properly.

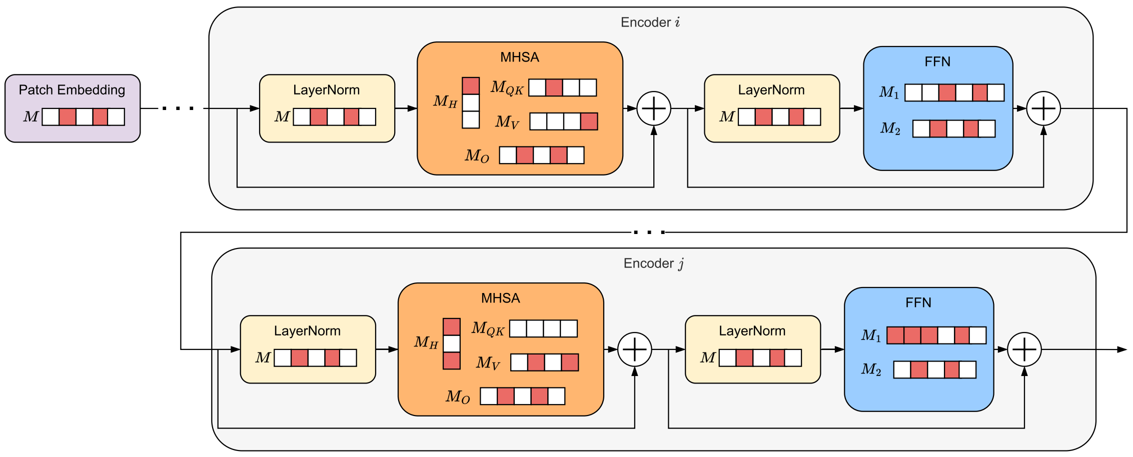

Figure 3.6: Example of shared residual dimension pruning, boxes in red are pruned weights.

### 3.6.3 Full Sharing

An even more constrained approach is to share all pruning masks (figure 3.7), including those of internal transform layers in each block. This means the same mask would be applied to the query, key, and value projections within MHSA as well as internal FFN layers across multiple encoder blocks. While this maximizes structural consistency, it leaves no flexibility for pruning components independently.

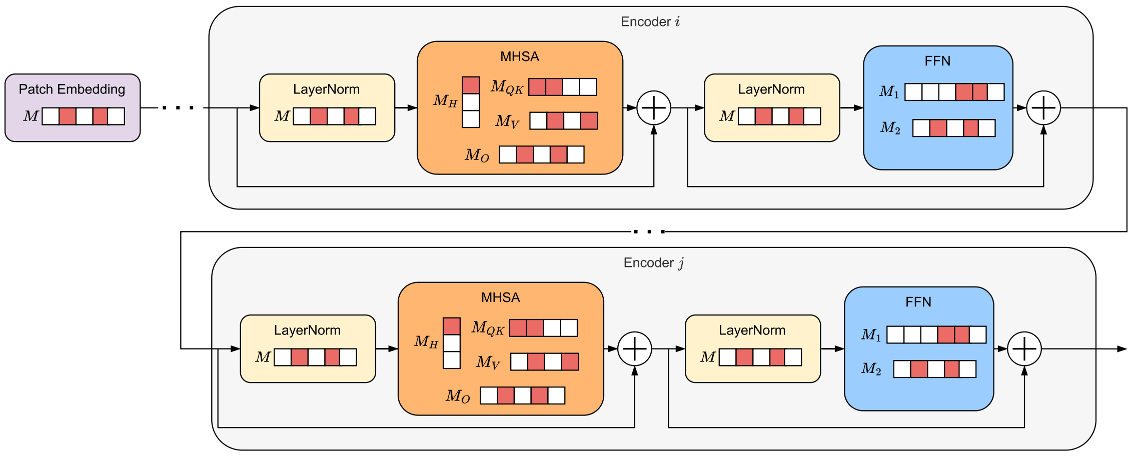

Figure 3.7: Example of full sharing pruning, boxes in red are pruned weights.

### 3.6.4 To learn or not to learn

With each mask potentially being learnable or fixed, there is a combinatorially large space of possible masking configurations to search. Trying all combinations was intractable. As an initial study, I fix either: heads masks, head dimension masks, FFN masks or hidden dimension masks. This reduces the search space to a tractable size. Further work could expand the search space to combinations of different kinds of masking. However, exploring deeply such space would surely require a significant amount of computational time or resources.

# Chapter 4

# Experimental Setup

This chapter details the experimental setup used for evaluating the proposed structured pruning techniques on vision transformer models. The experiments specifically aim to characterize the accuracy-size trade-off attained under different configurations, such as independent versus shared masking. The metrics measured include Top-1 classification accuracy on Tiny ImageNet and CIFAR-10 and number of parameters. This chapter describes the adopted ViT models, training methodology and datasets used in the experiments.

## 4.1 Experimental Setup

### 4.1.1 Datasets

Two image classification datasets are utilized to comprehensively evaluate the proposed pruning techniques - CIFAR-10 for initial analysis and ablation study, and Tiny ImageNet for assessing technique effectiveness on larger-scale data.

CIFAR-10 [19] consists of 60,000 32x32 low resolution color images evenly distributed over 10 classes, with 50,000 images for model training and 10,000 for validation and testing each. The small dataset size allows rapid iteration on architectural decisions . The balance across classes in training/validation/testing prevents skew in performance measurements. CIFAR-10 is adopted to evaluate pruning techniques prior to scaling up to larger datasets.

Tiny ImageNet [20] comprises 100,000 64x64 resolution images evenly distributed across 200 classes. The data is divided into training (80,000 images), validation (10,000 images), and testing (10,000 images) sets. The higher resolution and increased number of classes better represents real-world visual recognition challenges then CIFAR-10. Tiny ImageNet is a subset of the full ImageNet database designed for this intermediate stage of testing before tackling the full ImageNet benchmark.

ImageNet, containing over 1 million diverse high resolution images across 1000 classes, is the ultimate computer vision dataset benchmark. However, its massive scale poses challenges due to computational resource limitations, and therefore it was not adopted here but it will considered in future works.

### 4.1.2 Model Architectures

The Vision Transformer (ViT) [3] forms the core model architecture for evaluating the proposed pruning techniques. Specifically, the ViT-Tiny and ViT-Small configurations are utilized, which trade-off accuracy and computational requirements.

ViT-Tiny uses an input resolution of 384x384 pixels with 16x16 non-overlapping image patches as input to the Transformer encoder. The encoder comprises 12 blocks. ViT-Tiny has 3 attention heads per block and a hidden state dimension of 192, resulting in approximately 5.8 million parameters.

ViT-Small increases capacity with 6 attention heads per block and hidden dimension of 384, yielding approximately 22.2 million parameters. Both models contain 12 encoder blocks and 16x16 input patches. The larger ViT-Small aims to provide higher accuracy at the cost of greater compute requirements and so it will be trained on Tiny ImageNet, while ViT-Tiny will be trained on CIFAR-10.

Models are implemented in Python 3.10 using PyTorch 2.0, and the PLiNIO [18] library provides tools for implementing the proposed structured pruning techniques, including both ready-made components for pruning CNN and linear layers along with lower-level tools to construct custom structured pruning of components like MHSA.

As vision transformers are difficult to train from scratch, model weights are initialized from the timm checkpoints pretrained on ImageNet and fine-tuned during training. This transfers learned feature representations to enable training transformer models on limited compute resources.

### 4.1.3 Training Methodology

The training data consists of the full CIFAR-10 or Tiny ImageNet training sets, augmented on the fly via multiple random transformations. Images are first resized to match the transformer input dimensions using bicubic interpolation. RandAugment [21] is applied with two operations and a magnitude of 9, followed by image normalization. Additional augmentations include MixUp [22] with $\alpha_{\text{MixUp}} = 0.8$ and CutMix [23] with $\alpha_{\text{CutMix}} = 1$ and 0.1 label smoothing. Finally, pixel-level [24] random erasing is applied with 25% probability. This aggressive augmentation regime aims to improve model generalization, prevent overfitting and reach SoTA results [25].

The training process involves two key phases - initial structured pruning to learn importance scores and induce sparsity, followed by fine-tuning of the sparse model.

For structured pruning, the AdamW optimizer is utilized with a cosine annealing learning rate schedule employing cyclical warm restarts [26]. The learning rate decays from an initial value of 1e-4 down to 1e-8 every 4 epochs, with the cycle repeated throughout training. Warm restarts help unmasked weights compensate for recently masked parameters. A batch size of 32 for CIFAR-10 and 512 for Tiny ImageNet are used to maximize available GPU utilization. Weight decay is set to 5e-2 for model weights and 0 for importance scores, preventing the optimizer from masking all possible components. Furthermore, the learning rate for shared masks is scaled by $1 - \frac{\text{number of times shared}}{\text{total masks}}$ to avoid over-representing shared masks. This pruning phase lasts up to 500 epochs, although training stops early if validation accuracy has degraded over the past 20 epochs while model size did not improve for 20 epochs.

Next, the masked structures are removed and the model is fine-tuned to recover accuracy. Learning rate and batch sizes are unvaried. The learning is now scheduled without warm restarts. Fine-tuning is performed for at most 100 epochs, although training can stop early if the validation accuracy has not improved in the last 10 epochs.

## 4.2 Ablation Study

An ablation study is performed to systematically determine optimal configurations of the proposed structured pruning techniques. Isolating and evaluating each factor provides insights into how it affects the accuracy-size tradeoff.

Experiments are conducted by trying different combinations of regularization $\lambda$, mask sharing schemes and mask fixing decisions. Studying these parameters in a controlled setting informs best practices when applying the techniques. Guidelines for optimally configuring the techniques are provided based on the empirical results to maximize the accuracy and efficiency benefits of structured pruning.

The ablation study experiments are performed on the CIFAR-10 dataset using the ViT-Tiny architecture. The following list shows how the key parameters are chosen:

- Pruning hyperparameter $\lambda$ - Tested values of 5e-7, 1e-7, 5e-8, 1e-8.
- Mask sharing methodology - Compared shared residual masking and full sharing across all layers.
- Mask fixation schemes - Explored no fixing, whole heads, residual masks, internal layer of FFNs, and head dimensions.

This comprises a total search space of 40 possible combinations . Each combination is then tested on 3 random seeds, so a total of 120 individual training experiments are conducted for thorough analysis.

In addition to the 40 configurations previously described, we conduct further experiments using the unshared masking scheme to provide an upper bound on performance. The unshared scheme assigns independent masks to each component as described in Chapter 3, giving maximum flexibility but no structural consistency.

For the unshared masking experiments, we evaluate the same 4 values of $\lambda$ {5e-7, 1e-7, 5e-8, 1e-8} with 3 random seeds for each $\lambda$. This comprises an additional 12 training runs to complement the 120 already tested.

The motivation is that unshared masking provides an oracle upper bound on the attainable accuracy, since it can freely prune any individual component. Comparing shared masking techniques to this upper bound quantifies the representational capacity sacrificed for structural consistency.

## 4.3 Evaluation on Tiny ImageNet

Based on the thorough ablation study, we select the optimal structured pruning configuration and compare it against baselines on the larger Tiny ImageNet benchmark.

Specifically, we analyze the ablation study outcomes to determine which configuration obtained the most Pareto optimal results on CIFAR-10. A setting is Pareto optimal if no other configuration had both better accuracy and lower complexity. We select a pruning scheme with many pareto optimal results across the hyperparameter sweeps for each of the random seeds. This Pareto optimal configuration used on Tiny ImageNet using the ViT-Small architecture to evaluate how the method transfers to larger-scale data.

For comparison, we train some baseline vision transformer models:

- Unshared masking: Assign independent masks to each component
- $\ell_1$ regularization: Induce sparsity via LASSO penalty on weights associated to the same channel.
- $\ell_2$ regularization: Limit model capacity via Ridge penalty on weights associated to the same channel.

The unshared masking provides an upper bound on accuracy by freely pruning any individual component. The $\ell_1$ and $\ell_2$ pruning baselines provide a simple comparison baseline to compare against our method. $\ell_1$ and $\ell_2$ methods where implemented with the

These techniques are tuned to match the computational complexity of our method, as measured by number of parameters, for fair comparison. This evaluates whether structured pruning provides benefits over independent norm pruning and magnitude pruning approaches when controlling for efficiency.

Outcomes on Tiny ImageNet can demonstrate if our structured pruning method transfers to larger scale vision tasks. Comparisons to the baselines reveal whether our approach efficiently maximizes accuracy.

# Chapter 5

# Results

This chapter presents detailed experimental results and analysis evaluating the proposed structured pruning techniques for efficient vision transformer design. Outcomes of the extensive ablation study on CIFAR-10 are quantified, including impacts of key factors on accuracy and efficiency tradeoffs. The optimal configuration is selected and evaluated against baselines on Tiny ImageNet to demonstrate technique viability on larger-scale data.

## 5.1 Ablation Study Outcomes

Table 5.1 presents a summary of accuracy and number of parameters for the 40 masking configurations in the ablation study on CIFAR-10 described in Chapter 4. The table organizes results by the hyperparameter $\lambda$, sharing scheme, and fixed components.

The ablation study results table provides insights into how the key factors of pruning regularization strength, sharing scheme, and fixed components affect the accuracy-efficiency tradeoff. Several notable trends emerge from the data reported in table 5.1. For example, we notice that keeping the hidden dimension consistently yields higher accuracy across configurations up to 18%, though at the expense of increased model size of 389K parameters. In contrast, making all masks learnable generally results in the smallest models, reaching as little as 165K parameters, but with significant loss in accuracy from the most accurate configuration of up to 21%. While table 5.1 is useful for tracking scores across configurations, but it does not clearly highlight optimal solutions in the accuracy against size space. For this reason we perform a Pareto analysis. The accuracy-size Pareto frontier reveals optimal balancing of the tradeoffs. This Pareto analysis provides clearer guidelines than raw scores for identifying techniques to attain high performance at small size. This analysis is resumed in table 5.2 and is visually represented in figures 5.1, 5.2, and 5.3. Each figure represents the results obtained using a different random seed, this approach helps to evaluate the proposed method performance independently of the randomness involved in training.

The ablation study results demonstrate that two pruning strategies substantially underperform across configurations: pruning all structures except the FFN layer, and full sharing of masks. Pruning every component besides the FFN consistently leads to zero Pareto optimal solutions with respect to the other methods. This indicates that the FFN alone lacks representational capacity for the task. Full sharing also exhibits significant accuracy drops for most settings, only achieving Pareto optimal solutions below roughly 600K parameters with an accuracy up to 86% that is excessively low given the simplicity of CIFAR-10. Based on these clear deficiencies observed, both non-FFN pruning and full sharing will be excluded from evaluation on the more complex Tiny ImageNet benchmark.

The pareto frontier illustrates that some structural fixing combined with shared residual masking prove to be superior. In particular, the sharing of masks only between residual-connected layers while fixing heads or hidden dimensions results in Pareto optimality in 9 and 8 out of the 12 experiments respectively, with accuracy as high as 96% with less that 50% of the original parameters. This indicates that these two structured pruning schemes effectively balance induced sparsity and retention of representation power with respect to the other combinations. The head and hidden dimension likely capture the most important information. By retaining them while pruning the remaining redundant structures, compression is achieved with minimal accuracy loss.

Additionally, the configuration with shared residual masking and no fixed structures exhibits Pareto optimality in half of the experiments. While less consistent than fixing heads or dimensions, its performance remains competitive with roughly 2% less accuracy and 200k less parameters than the selected configurations. This is also visually depicted in Figures 5.1, 5.2, and 5.3. With no constraints on prunable structures, this scheme provides maximum flexibility in compressing any component. Despite the slight reduction in Pareto efficiency, this fully flexible pruning is worth further exploration given its strong flexibility and overall performance . The Tiny ImageNet experiments will also evaluate this technique alongside the consistently optimal head and hidden dimension fixing schemes. These insights inform the stronger-performing techniques to focus on in subsequent Tiny ImageNet

| $\lambda$ | wiring | keep heads | keep hidden dim. | keep ffn | keep head dim. | Accuracy Mean ± Std | Size Mean ± Std |
|---|---|---|---|---|---|---|---|
| 1E-08 | full sharing | False | False | False | False | 96.96% ± 0.24% | 3.86M ± 288.02K |
| 1E-08 | full sharing | False | False | False | True | 96.67% ± 0.25% | 3.21M ± 617.42K |
| 1E-08 | full sharing | False | False | True | False | 96.81% ± 0.21% | 3.86M ± 337.15K |
| 1E-08 | full sharing | False | True | False | False | 97.79% ± 0.12% | 5.55M ± 7.07K |
| 1E-08 | full sharing | True | False | False | False | 97.22% ± 0.07% | 4.15M ± 150.99K |
| 1E-08 | shared residual | False | False | False | False | 97.18% ± 0.68% | 2.98M ± 135.29K |
| 1E-08 | shared residual | False | False | False | True | 96.88% ± 0.25% | 3.07M ± 70.25K |
| 1E-08 | shared residual | False | False | True | False | 96.78% ± 0.32% | 3.56M ± 399.23K |
| 1E-08 | shared residual | False | True | False | False | 97.23% ± 0.26% | 3.98M ± 26.60K |
| 1E-08 | shared residual | True | False | False | False | 97.33% ± 0.07% | 3.38M ± 90.26K |
| 5E-08 | full sharing | False | False | False | False | 95.01% ± 0.76% | 2.36M ± 240.50K |
| 5E-08 | full sharing | False | False | False | True | 93.65% ± 0.38% | 1.32M ± 145.31K |
| 5E-08 | full sharing | False | False | True | False | 94.22% ± 0.52% | 2.08M ± 112.05K |
| 5E-08 | full sharing | False | True | False | False | 96.10% ± 0.10% | 4.05M ± 16.67K |
| 5E-08 | full sharing | True | False | False | False | 96.17% ± 0.10% | 2.78M ± 108.31K |
| 5E-08 | shared residual | False | False | False | False | 94.26% ± 1.86% | 1.49M ± 60.28K |
| 5E-08 | shared residual | False | False | False | True | 95.07% ± 0.09% | 1.51M ± 243.25K |
| 5E-08 | shared residual | False | False | True | False | 95.03% ± 0.31% | 1.73M ± 6.85K |
| 5E-08 | shared residual | False | True | False | False | 96.68% ± 0.05% | 2.48M ± 33.78K |
| 5E-08 | shared residual | True | False | False | False | 96.28% ± 0.21% | 1.73M ± 83.71K |
| 1E-07 | full sharing | False | False | False | False | 92.33% ± 0.56% | 1.42M ± 340.36K |
| 1E-07 | full sharing | False | False | False | True | 91.46% ± 1.85% | 1.15M ± 332.41K |
| 1E-07 | full sharing | False | False | True | False | 92.38% ± 0.47% | 1.52M ± 168.42K |
| 1E-07 | full sharing | False | True | False | False | 96.02% ± 0.26% | 3.09M ± 217.98K |
| 1E-07 | full sharing | True | False | False | False | 94.17% ± 0.61% | 1.88M ± 282.96K |
| 1E-07 | shared residual | False | False | False | False | 93.34% ± 0.53% | 931.04K ± 45.65K |
| 1E-07 | shared residual | False | False | False | True | 93.59% ± 0.79% | 1.16M ± 249.59K |
| 1E-07 | shared residual | False | False | True | False | 92.19% ± 0.09% | 1.10M ± 21.46K |
| 1E-07 | shared residual | False | True | False | False | 95.64% ± 0.33% | 1.92M ± 30.62K |
| 1E-07 | shared residual | True | False | False | False | 95.15% ± 0.33% | 1.06M ± 48.18K |
| 5E-07 | full sharing | False | False | False | False | 73.49% ± 1.00% | 165.60K ± 2.25K |
| 5E-07 | full sharing | False | False | False | True | 79.46% ± 0.42% | 501.48K ± 437.32K |
| 5E-07 | full sharing | False | False | True | False | 56.48% ± 1.35% | 225.95K ± 9.80K |
| 5E-07 | full sharing | False | True | False | False | 86.08% ± 2.54% | 536.66K ± 43.92K |
| 5E-07 | full sharing | True | False | False | False | 73.92% ± 2.09% | 228.58K ± 26.46K |
| 5E-07 | shared residual | False | False | False | False | 71.32% ± 3.35% | 202.71K ± 12.87K |
| 5E-07 | shared residual | False | False | False | True | 74.06% ± 1.42% | 232.75K ± 8.91K |
| 5E-07 | shared residual | False | False | True | False | 55.88% ± 2.03% | 210.18K ± 10.77K |
| 5E-07 | shared residual | False | True | False | False | 92.71% ± 0.23% | 621.23K ± 4.33K |
| 5E-07 | shared residual | True | False | False | False | 70.63% ± 0.83% | 188.91K ± 27.63K |

Table 5.1: Ablation study results on CIFAR-10. The table shows Top-1 validation accuracy (%), and number of parameters for the 40 shared masking experiments.

experiments.

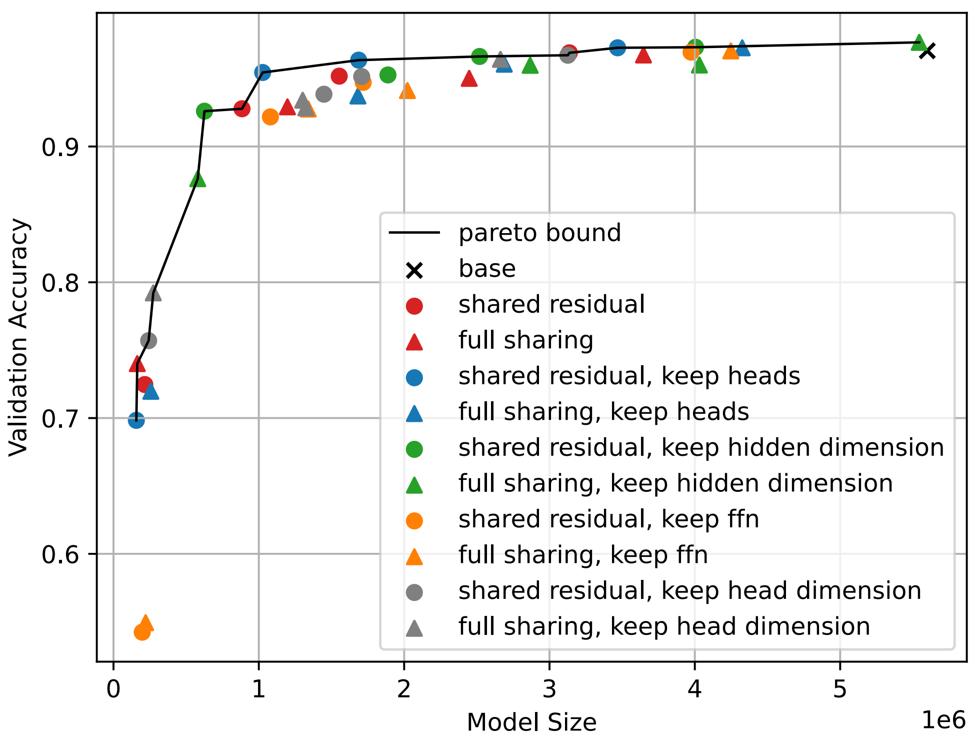

Figure 5.1: Seed 7
Pareto efficiency analysis of structured pruning configurations on CIFAR-10. The plots show the accuracy versus number of parameters for the 40 shared masking experiments on three different seeds. The Pareto optimal configurations are on the black line, and they balance model complexity while maximizing accuracy compared to other settings from the ablation study

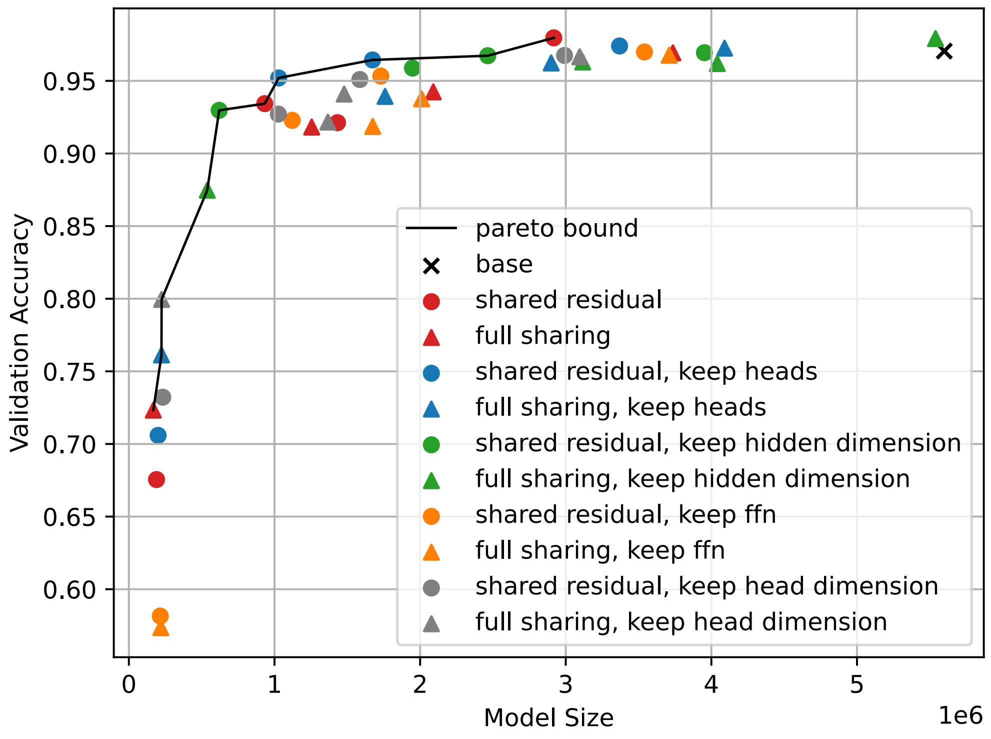

Figure 5.2: Seed 13

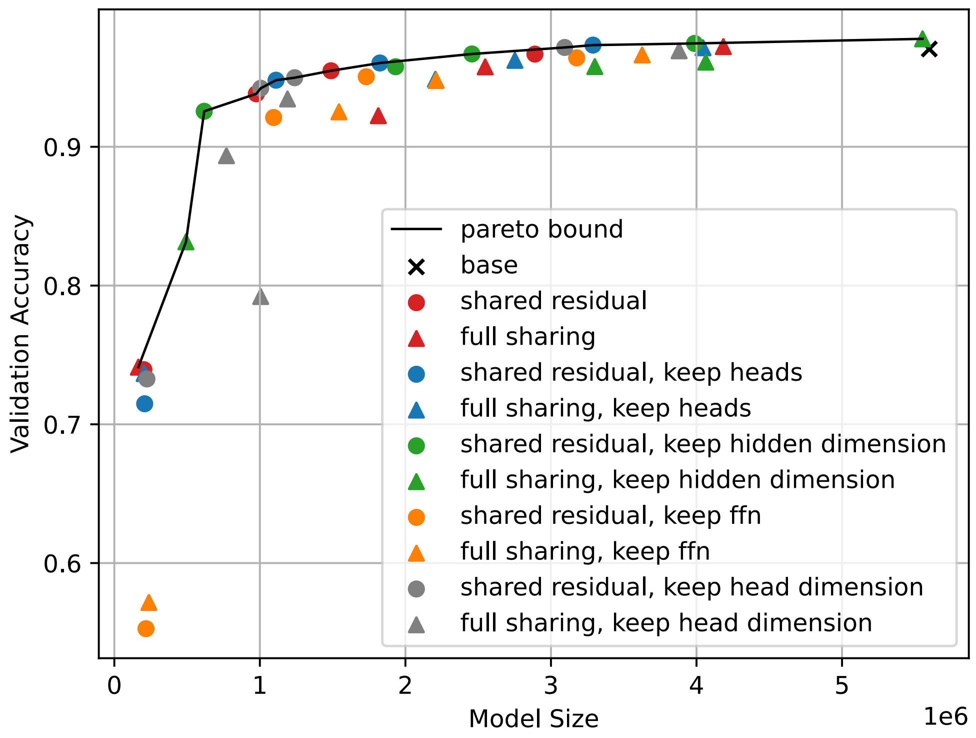

Figure 5.3: Seed 42

The ablation study results are compared against two baseline strategies: unshared pruning and regularization-based pruning. As shown in figures 5.4, 5.5 and 5.6, unshared pruning demonstrates similar accuracy (on average 2% drop) to the Pareto optimal schemes for larger model sizes, and shows better performance for highly compressed smaller models with 86% accuracy at just 120K parameters, compared to 75% accuracy for models with mask sharing of similar size. This indicates being extremely flexible in pruning out performs representational capacity of shared schemes at high sparsity levels. Unfortunately the ability to prune structures freely comes at the cost of limited practical model compression, as the sparsity does not follow the layout of transformer networks. In contrast, the L1 and L2 norm regularization baselines , measured at 75%, 50% and 25% sparsity, consistently underperform compared to the Pareto frontier with a loss in accuracy of respectively 22%, 13% and 3% . Their naive pruning fails to match the importance-based masking effectiveness. Overall, the structured ablation study schemes outperform these commonly employed techniques tuned for an equivalent level of sparsity. The benefits of selective structural fixing combined with importance-based pruning are quantitatively demonstrated.

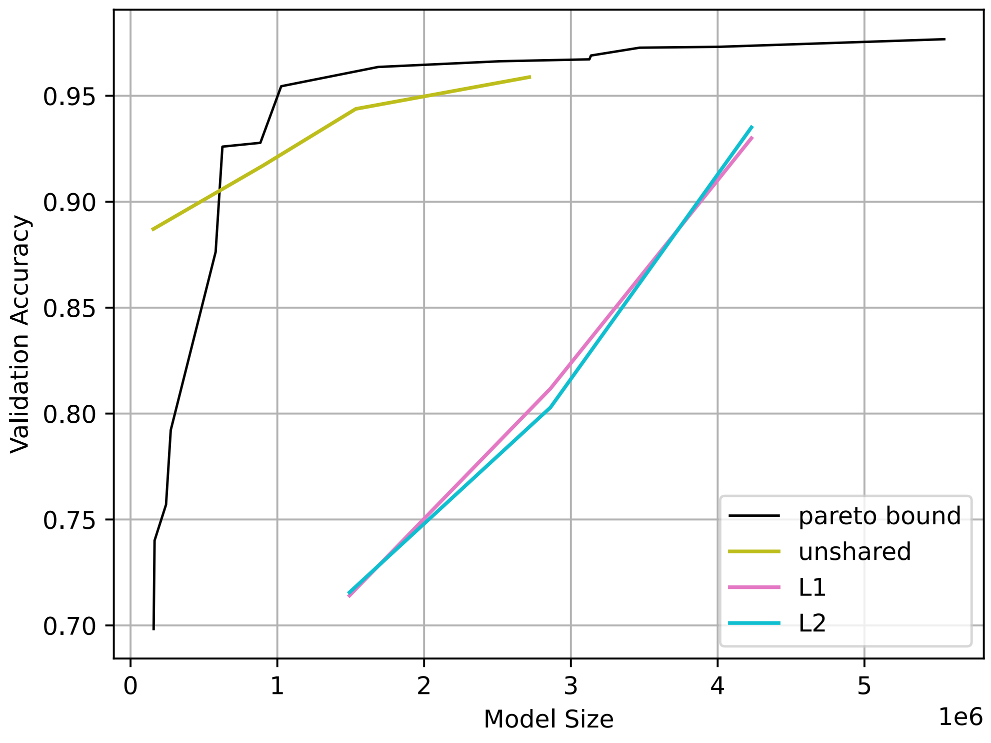

Figure 5.4: Seed 7
Comparison of the pareto bound of shared pruning experiments against unshared pruning, l1 and l2 norm pruning on CIFAR-10.

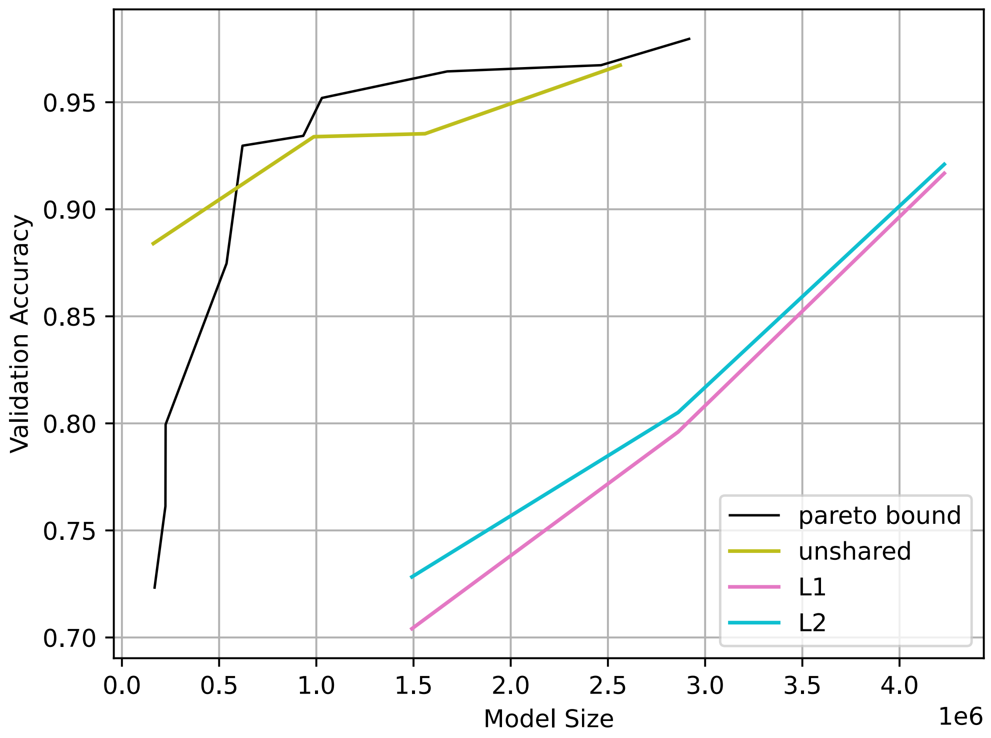

Figure 5.5: Seed 13

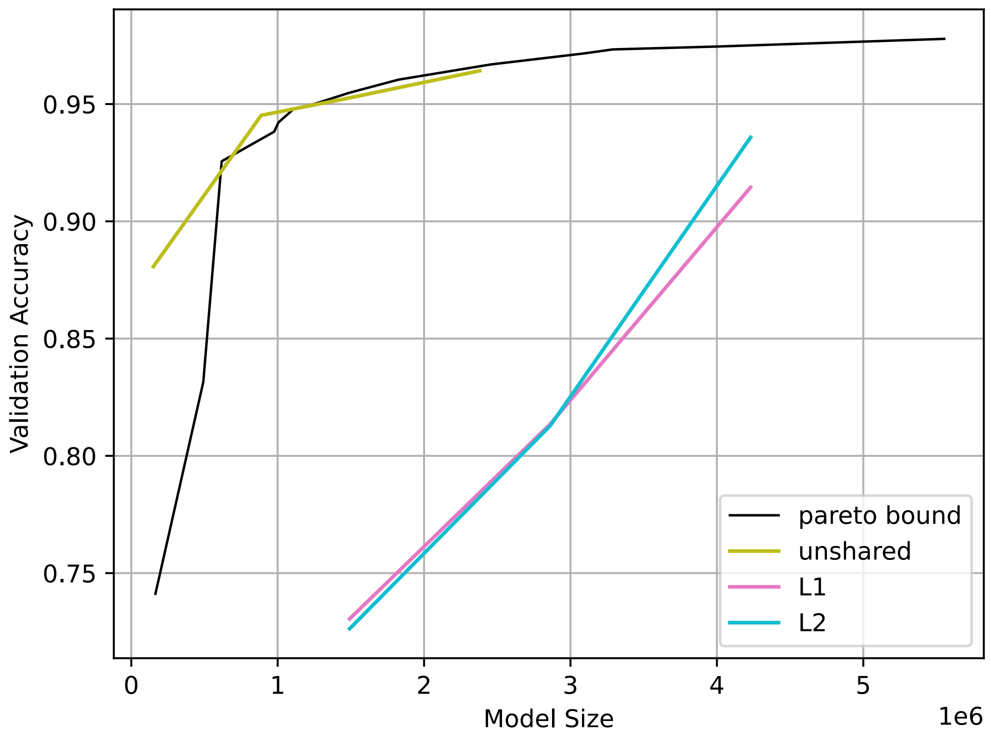

Figure 5.6: Seed 42

## 5.2 Results on Tiny ImageNet

The Tiny ImageNet dataset presents a more challenging testbed for vision transformers, comprising 200 classes of higher resolution 64x64 images. Experiments assess the top performing techniques identified on CIFAR-10, providing insights into their generalization. All shared tested residual masking schemes lead to various Pareto optimal results.

Differently from before, here accuracy degrades more rapidly with increased sparsity compared to CIFAR-10. At 70% sparsity (roughly 15M parameters) the accuracy already lowers to 80% which represents a drop of 7%. The smallest accuracy loss we could achieve was 4% at 18M parameters, 3M less than the original 21M. This highlights the greater complexity in learning representations for Tiny ImageNet. However, the relative performance trends between tested techniques remain consistent with CIFAR-10, further validating the importance of structured and shared masking.

As before, in figures 5.7,5.8 and 5.9 show that the gap between the pareto optimal results and unshared structured pruning remains similar. Differently than CIFAR-10, naive L1 and L2 pruning perform quite similar our method around 75% sparsity, but despite being finetuned their accuracy degrades way faster than the proposed method, with over 30% accuracy drop at 25% sparsity .

Overall, the experiments confirm the efficacy of sharing residual masks transfers from smaller to larger vision datasets, and that they still perform better than naive methods. The structured schemes offer decent trade-offs, though higher model capacity is needed to maintain accuracy on Tiny ImageNet. This results provide an insightful direction in the field of pruning transformers to deploy on resource-constrained platforms.

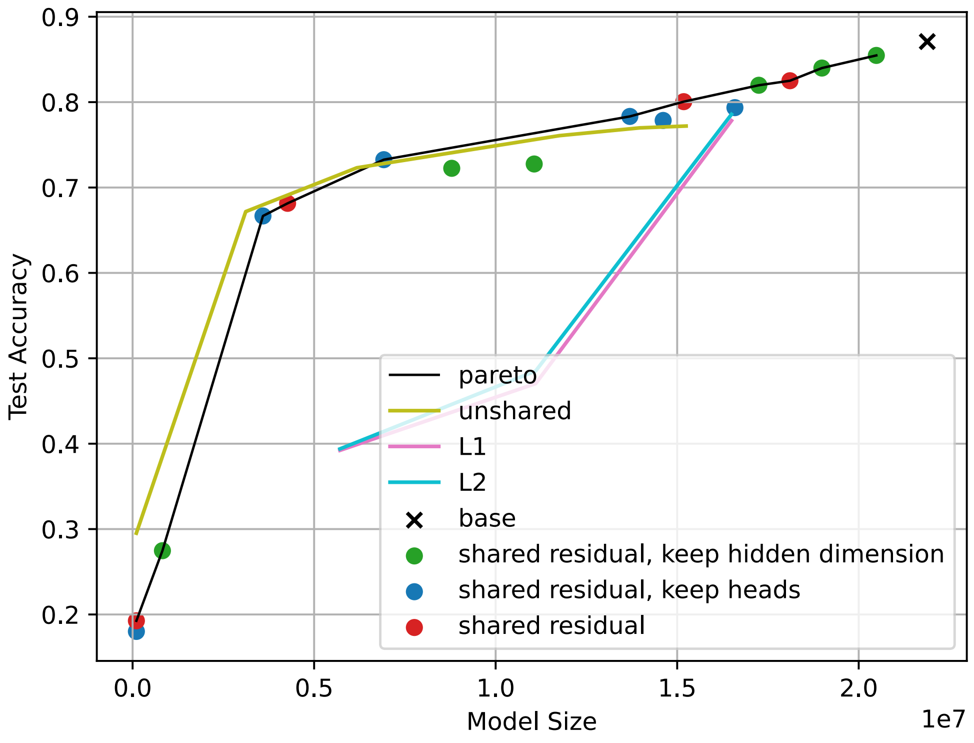

Figure 5.7: Seed 7
Comparison of the pareto bound of shared pruning experiments against unshared pruning, l1 and l2 norm pruning on Tiny ImageNet.

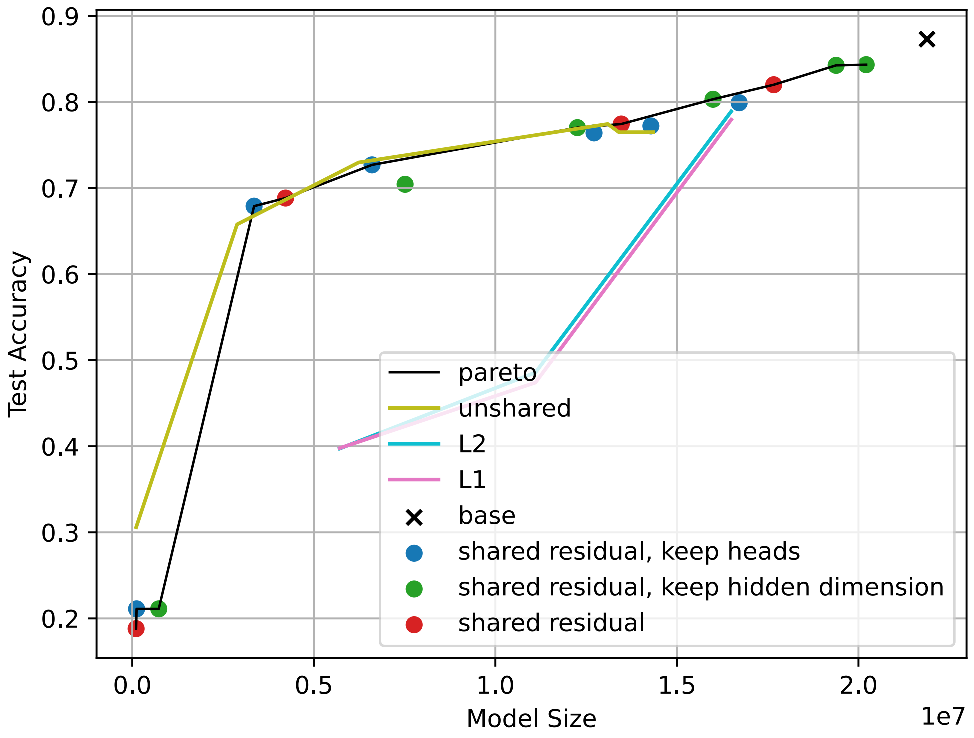

Figure 5.8: Seed 13

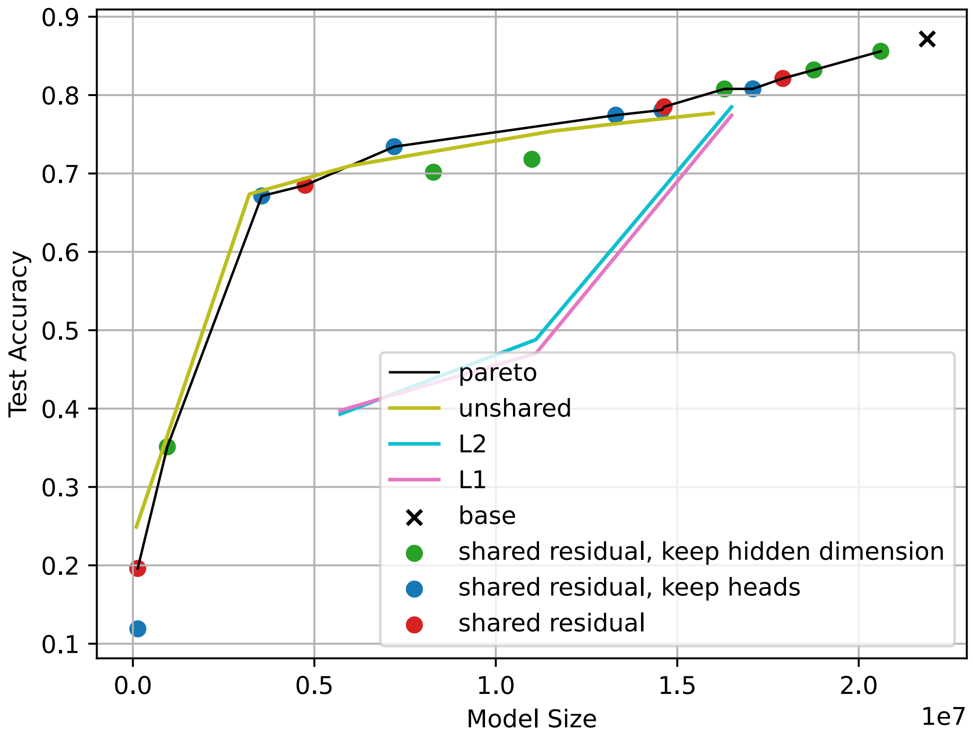

Figure 5.9: Seed 42

| wiring | keep heads | keep hidden dim. | keep ffn | keep head dim. | pareto optimal experiments |
|---|---|---|---|---|---|
| full sharing | False | False | False | False | 3 / 12 |
| full sharing | False | False | False | True | 2 / 12 |
| full sharing | False | False | True | False | 0 / 12 |
| full sharing | False | True | False | False | 5 / 12 |
| full sharing | True | False | False | False | 1 / 12 |
| shared residual | False | False | False | False | 6 / 12 |
| shared residual | False | False | False | True | 5 / 12 |
| shared residual | False | False | True | False | 0 / 12 |
| shared residual | False | True | False | False | 8 / 12 |
| shared residual | True | False | False | False | 9 / 12 |

Table 5.2: Number of times a configuration was Pareto with respect to other experiments performed on the same seed.

Chapter 6

# Conclusion and Future Works

This thesis introduced structured pruning techniques tailored to compress transformers for improved efficiency and thoroughly evaluated them on vision transformers . An extensive experimental methodology was followed to provide insights into optimal configurations and trade-offs.

The proposed approach of learned binary masks enables directly removing structures like heads and dimensions based on importance scores. Comprehensive ablation studies on CIFAR-10 determined that shared residual masking with selective fixing performs best. The benefits transferred to larger Tiny ImageNet data as well.

These techniques make deploying computationally expensive vision transformers possible on resource-constrained edge devices with only MB of memory. Structured pruning halved model size with minimal accuracy loss of 1-2% in some settings. However, inherent trade-offs exist between higher compression rates and degraded accuracy.

While extremely promising, there remain many open questions and avenues for future work. The search space of masking strategies could be expanded by combining different types of fixed and learned masks. Additional tasks beyond image classification could be investigated as well, such as objection detection and segmentation. Or also new domains such as NLP could be explored.

Finally, the ultimate assessment would be deployment on actual embedded systems. The pruned models should be converted to formats amenable to edge devices and executed to measure real-world throughput, latency, and power consumption improvements compared to unoptimized transformers.

In conclusion, this thesis introduced a methodology for structured pruning of vision transformers guided by extensive experiments. Further research can build upon these insights to continue advancing efficient transformer design.

# Acronyms

| | |
|---|---|
| CNN | Convolutional Neural Network |
| DL | Deep Learning |
| FFN | Feed Forward Network |
| MHSA | Multi-Headed Self-Attention |
| ML | Machine Learning |
| NLP | Natural Language Processing |
| STE | Straight Through Estimator |
| ViT | Vision Transformer |

# Bibliography

[1] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[2] Meta AI. *SoTA Image Classification on ImageNet.* Accessed: 14 November 2023. 2023. URL: https://paperswithcode.com/sota/image-classification-on-imagenet?tag_filter=4%2C17.

[3] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[4] Krishna Teja Chitty-Venkata et al. *A Survey of Techniques for Optimizing Transformer Inference.* 2023. arXiv: 2307.07982 [cs.LG].

[5] Harmke de Groot. "IoT and the cloud: A hacked personality and an empty battery head-ache or an intuitive environment to make our lives easier?" In: *2015 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S).* 2015, pp. 1–5. DOI: 10.1109/S3S.2015.7333501.

[6] Eric Flamand et al. "GAP-8: A RISC-V SoC for AI at the Edge of the IoT". In: *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP).* 2018, pp. 1–4. DOI: 10.1109/ASAP.2018.8445101.

[7] Google. *Coral Dev Board Micro.* 2023. URL: https://coral.ai/products/dev-board-micro.

[8] Sainbayar Sukhbaatar et al. "Weakly Supervised Memory Networks". In: *CoRR* abs/1503.08895 (2015). arXiv: 1503.08895. URL: http://arxiv.org/abs/1503.08895.

[9] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[10] Yang Liu and Mirella Lapata. "Text Summarization with Pretrained Encoders". In: *CoRR* abs/1908.08345 (2019). arXiv: 1908.08345. URL: http://arxiv.org/abs/1908.08345.

[11] Hugo Touvron et al. "Fixing the train-test resolution discrepancy: FixEfficientNet". In: *CoRR* abs/2003.08237 (2020). arXiv: 2003.08237. URL: https://arxiv.org/abs/2003.08237.

[12] Hugo Touvron et al. "Training data-efficient image transformers & distillation through attention". In: *CoRR* abs/2012.12877 (2020). arXiv: 2012.12877. URL: https://arxiv.org/abs/2012.12877.

[13] Trevor Gale, Erich Elsen, and Sara Hooker. "The State of Sparsity in Deep Neural Networks". In: *CoRR* abs/1902.09574 (2019). arXiv: 1902.09574. URL: http://arxiv.org/abs/1902.09574.

[14] Paul Michel, Omer Levy, and Graham Neubig. "Are Sixteen Heads Really Better than One?" In: *CoRR* abs/1905.10650 (2019). arXiv: 1905.10650. URL: http://arxiv.org/abs/1905.10650.

[15] Victor Sanh, Thomas Wolf, and Alexander M. Rush. "Movement Pruning: Adaptive Sparsity by Fine-Tuning". In: *CoRR* abs/2005.07683 (2020). arXiv: 2005.07683. URL: https://arxiv.org/abs/2005.07683.

[16] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". In: *CoRR* abs/1308.3432 (2013). arXiv: 1308.3432. URL: http://arxiv.org/abs/1308.3432.

[17] François Lagunas et al. "Block Pruning For Faster Transformers". In: *CoRR* abs/2109.04838 (2021). arXiv: 2109.04838. URL: https://arxiv.org/abs/2109.04838.

[18] Daniele Jahier Pagliari et al. *PLiNIO: A User-Friendly Library of Gradient-based Methods for Complexity-aware DNN Optimization*. 2023. arXiv: 2307.09488 [cs.LG].

[19] Alex Krizhevsky. *CIFAR-10*. URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[20] Stanford University. *Tiny ImageNet*. URL: http://cs231n.stanford.edu/tiny-imagenet-200.zip.

[21] Ekin D. Cubuk et al. "RandAugment: Practical data augmentation with no separate search". In: *CoRR* abs/1909.13719 (2019). arXiv: 1909.13719. URL: http://arxiv.org/abs/1909.13719.

[22] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *CoRR* abs/1710.09412 (2017). arXiv: 1710.09412. URL: http://arxiv.org/abs/1710.09412.

[23] Sangdoo Yun et al. "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features". In: *CoRR* abs/1905.04899 (2019). arXiv: 1905.04899. URL: http://arxiv.org/abs/1905.04899.

[24] Zhun Zhong et al. "Random Erasing Data Augmentation". In: *CoRR* abs/1708.04896 (2017). arXiv: 1708.04896. URL: http://arxiv.org/abs/1708.04896.

[25] Xiangning Chen et al. *Symbolic Discovery of Optimization Algorithms*. 2023. arXiv: 2302.06675 [cs.LG].

[26] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Restarts". In: *CoRR* abs/1608.03983 (2016). arXiv: 1608.03983. URL: http://arxiv.org/abs/1608.03983.