Structured Pruning Learns Compact and Accurate Models

Anonymous ACL submission

Abstract

The growing size of neural language models has led to increased attention in model compression. Pruning methods start from a large model and gradually remove model weights—they can significantly reduce the model size but hardly achieve impressive runtime efficiency. On the other hand, distillation methods start from a shallower, compact model and can obtain large speedups—however, they are costly to train on large amounts of unlabeled data. In this work, we show that structured pruning can match the distillation counterparts in both latency ($>10\times$) and accuracy ($>92\%$) and result in highly compact and efficient subnetworks. Unlike distillation, our task-specific pruning approach, MixedPruning, does not need to pre-specify the model architecture nor rely on unlabeled data. Our solution is to jointly prune layers and sub-modules such as heads and hidden units in Transformer models through l_0 regularization while ensuring that the resulting model is parallelizable. We also propose a layerwise distillation approach to further guide pruning. Finally, our pruned structures reveal interesting patterns—for example, more than 70% of feed-forward and 50% of self-attention layers can be easily pruned, while the first and last 1-2 layers are likely to remain for highly compressed models.

1 Introduction

Fine-tuning pre-trained language models (Devlin et al., 2019; Liu et al., 2019a; Raffel et al., 2020, *inter alia*) has become the mainstay in natural language processing. These models have high costs in terms of storage, memory, and computation time, which has motivated a large body of work to make them smaller and faster to use in real-world applications (Ganesh et al., 2021).

The two predominant approaches to model compression are pruning and distillation. Pruning methods start from a large, pre-trained model and progressively remove redundant weights of the network while updating the remaining weights. In particular, structured pruning removes groups of weights or entire subnetworks from the network and can gain actual speedups at inference time. Recent work has investigated how to structurally prune a Transformer networks, from removing entire layers (Fan et al., 2020; Sajjad et al., 2020), to pruning heads (Michel et al., 2019; Voita et al., 2019), intermediate dimensions (McCarley et al., 2019; Wang et al., 2020c) and blocks in weight matrices (Lagunas et al., 2021). However, current pruning methods still cannot obtain large speedups, and most reported results have 2-3$\times$ improvement at most. Pushing the pruning rate higher for a highly compressed model with these methods leads to significant performance drops.

	Speedup	# Params	MNLI
Distillation			
DistillBERT$_6$ ♠	2.0×	43M	82.2
TinyBERT$_6$	2.0×	43M	84.0
MobileBERT‡ ♠	2.3×	20M	83.9
DynaBERT	6.3×	11M	76.3
AutoTinyBERT‡	9.1×	3.3M	78.2
TinyBERT$_4$	11.4×	4.7M	78.8
Pruning			
Movement Pruning	1.0×	9M	81.2
Block Pruning	2.7×	25M	83.7
MixedPruning (ours)	2.7×	26M	84.9
MixedPruning (ours)	12.1×	4.4M	80.6

Table 1: A comparison of state-of-the-art distillation and pruning methods. All the models use BERT$_{\text{base}}$ as the baseline model except for those labeled as ‡ and we evaluate all the models on an NVIDIA V100 GPU by ourselves (§4.1). Green models have large speed-ups and are one order of magnitude smaller. ♠: task-agnostic distillation. We exclude task-specific data augmentation for a fair comparison.[1]

[1]We run the TinyBERT experiments by ourselves but noticed that there is a discrepancy between our result and the number reported in the original paper (TinyBERT$_4$, 80.5 on MNLI). The only difference is that we use our own teacher

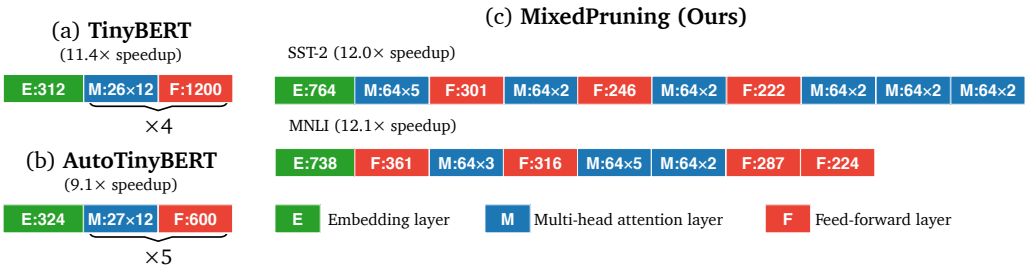

Figure 1: Comparison of model structures between (a) TinyBERT (Jiao et al., 2020): a manually-specified distillation model; (b) AutoTinyBERT (Yin et al., 2021): an automatically searched model using Neural Architecture Search; (c) the structure discovered by our pruning approach MixedPruning. `E:d` represents an embedding layer with d dimensions; `M:v×h` represents a multi-head attention layer with h heads and each head consists of v dimensions; `F:m` represents a feed-forward layer with m intermediate dimensions.

On the contrary, distillation methods (Sanh et al., 2019; Sun et al., 2019) first specify a shallower and compact model architecture and directly distill knowledge from the teacher model. For example, TinyBERT$_4$ (Jiao et al., 2020) consists of 4 layers with a 312 hidden size (the teacher model BERT$_{\text{base}}$ has 12 layers with a hidden size of 768). Due to the compact structure, these models can easily obtain $10\times$ times speedup at inference time. Deciding the structure of the student model requires careful architecture search (Yin et al., 2021), and the distillation process is usually slow, requiring large amounts of unlabeled data or data augmentation.

In this work, we propose MixedPruning and aim to close the gap between pruning and distillation. We show a surprising finding: structured pruning can also achieve highly compact sub-networks and obtain large speedups while achieving competitive accuracy as the distillation approaches without using unlabeled data (Table 1). Our intuition is straightforward: we allow for pruning bigger units (e.g., entire self-attention or feed-forward layers) and smaller units (e.g., heads or intermediate units) simultaneously. The pruning process learns masking variables of different granularity jointly through l_0 regularization. Despite its simplicity, we show that this mixed-pruning strategy is the key to large compression, significantly outperforming counterparts that only prune entire layers or submodules.

We also find that distillation objectives can provide useful signals to pruning and further achieve better performance. Since the structure of the student model changes during the course of training, we design a layerwise distillation approach, which can dynamically learn the layer mapping between the student (pruned) model and the teacher (unpruned) model during training.

We show that MixedPruning delivers more accurate models at every level of speedup and model size on GLUE tasks (Wang et al., 2019) and SQuAD (Rajpurkar et al., 2016), compared to strong pruning and distillation baselines. In particular, it can achieve $>10\times$ speedups and a 95% pruning rate across all the datasets while preserving $> 92\%$ of accuracy. The results suggest that task-specific pruning can be a very appealing solution to produce smaller and faster models without requiring additional unlabeled data for pre-training. We also discover interesting patterns in pruned structures (Figure 1)—for example, we find that for highly compressed models, most hidden dimensions ($>$70%) in FFN layers are removed through pruning. Also, the pruning process favors preserving upper and lower layers and pruning middle layers most of the time.

2 Background

2.1 Transformers

A Transformer network (Vaswani et al., 2017) is composed of L blocks and each block consists of a multi-head self-attention (MHA) layer, and a feed-forward (FFN) layer. An MHA layer with N_h heads takes an input $X \in \mathbb{R}^{\text{len}\times d}$ and outputs:

$$\text{MHA}(X) = \sum_{i=1}^{N_h} \text{Att}(W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}, W_O^{(i)}, X)$$

where $W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}, W_O^{(i)} \in \mathbb{R}^{d\times d_h}$ denote the query, key, value and output matrices respectively and $\text{Att}(\cdot)$ is an attention function. Here d denotes the hidden size (e.g., 768) and $d_h = d/N_h$ denotes the output dimension of each head (e.g., 64).

The output from the self-attention layer is then fed into a feed-forward layer, which consists of

model for distillation, which is used for all the experiments.

an up-projection and a down-projection layer, parameterzied by $W_U \in \mathbb{R}^{d \times d_f}$ and $W_D \in \mathbb{R}^{d_f \times d}$:

$$\text{FFN}(X) = \text{gelu}(XW_U) \cdot W_D.$$

Typically, $d_f = 4d$ (e.g., 3072). There is also a residual connection and a layer normalization operation after each MHA and FFN layer.

MHAs, FFNs account for $1/3$ and $2/3$ of the model parameters in Transformers.[2] According to the analysis in Ganesh et al. (2021), both MHAs and FFNs take roughly similar time on GPUs while the FFNs become the bottleneck on CPUs.

2.2 Pruning

Pruning approaches aim to remove redundant parameters from a pre-trained model without sacrificing the model performance. The pruning decisions can be made in a greedy, iterative way—e.g., removing the weights with lowest magnitude (Han et al., 2015) or other importance proxy scores (Michel et al., 2019), or modeled with mask variables as a global optimization problem (Louizos et al., 2018; Sanh et al., 2020). Since we consider pruning units of different sizes, we choose the latter solution in this work. Generally, the pruning units can vary from individual weights to sub-modules of networks (e.g., layers, heads). Although pruning individual weights can lead to a very high pruning rate (called "unstructured pruning"), it is difficult to accelerate the computation with the current hardware.

Layer pruning Several works explored strategies to drop entire layers from a pre-trained Transformer model (Fan et al., 2020; Sajjad et al., 2020)—in these cases, pruning one layer indicates a joint decision of pruning both the MHA layer and FFN layer. Empirical evidence suggested that 50% of the layers can be dropped without big loss in accuracy, resulting in a $2\times$ speedup.

Head pruning Head pruning techniques have been widely studied—Voita et al. (2019), Michel et al. (2019) showed that only a subset of heads are important and the majority can be pruned. We follow these works to mask heads by introducing variables $\mathbf{z}_{\text{head}}^{(i)} \in \{0, 1\}$ to multi-head attention:

$$\text{MHA}(X) = \sum_{i=1}^{N_h} \mathbf{z}_{\text{head}}^{(i)} \text{Att}(W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}, W_O^{(i)}, X).$$

Only removing heads cannot lead to considerable latency improvement—e.g., Li et al. (2021) demonstrated $1.4\times$ speedup with only 1 remaining head per layer. However, head pruning can be combined with pruning other components to achieve a smaller model (McCarley et al., 2019).

FFN pruning Apart from head pruning, the other major part of Transformer models—feed-forward layers (FFNs)—are also known to be overparameterized. Strategies to prune an FFN layer include pruning an entire FFN layer (Prasanna et al., 2020) and on a more fine-grained level, pruning intermediate dimensions (McCarley et al., 2019; Hou et al., 2020) by introducing mask variables $\mathbf{z}_{\text{int}} \in \{0, 1\}^{d_f}$:

$$\text{FFN}(X) = \text{gelu}(XW_U) \cdot \text{diag}(\mathbf{z}_{\text{int}}) \cdot W_D.$$

Pruning intermediate dimensions leads to speedup, as the resulting weight matrices are still fully dense.

Block pruning More recently, pruning on a smaller unit—blocks—from MHAs and FFNs have been explored (Lagunas et al., 2021). However, it is hard to optimize models with blocks pruned on the current hardware. Yao et al. (2021) attempted optimizing block-pruned models with the block sparse MatMul kernel provided by Triton (Tillet et al., 2019), but the reported results are not competitive.

2.3 Distillation

Knowledge distillation (Hinton et al., 2015; Sanh et al., 2019) is an approach that directly transfers knowledge from the pre-trained model to a student model. Previous work has designed layer mapping strategies to effectively transfer knowledge to a pre-specified model structure (Sun et al., 2019, 2020; Jiao et al., 2020). More recently, Hou et al. (2020) attempted to distill to a model with a more dynamic structure by specifying a set of widths and heights, and each pair determines a model structure. However, each layer's structure is pre-specified, so the architecture is still more constrained than pruning-based approaches. In this work, apart from designing a new pruning strategy, we propose a layerwise distillation objective tailored to model pruning and

[2] Following previous work, we exclude the embedding matrix in this calculation.

show that combining the two leads to the best empirical performance.

3 Method

In this section, we introduce our pruning method MixedPruning. Our method is based on two simple yet under-explored ideas: (1) we prune both layers and sub-modules (heads and hidden units) in a unified framework (§3.1); (2) we propose a layerwise distillation strategy to transfer knowledge from a full model to the pruned model (§3.2).

3.1 Mixed Pruning of Layers and Submodules

We propose a pruning strategy that 1) couples pruning decisions over different units; 2) leads to highly parallelizable models with flexible structures.

Previous works usually prune MHAs and FFNs with one single type of modules, e.g., McCarley et al. (2019) prune heads for MHAs and intermediate dimensions for FFNs; Prasanna et al. (2020); Lin et al. (2020) prune heads for MHAs and entire FFN layers. Intuitively, pruning smaller modules has the advantage of enabling more flexible model structures—in effect, it subsumes structures pruned with larger modules. However, empirically we find that granular pruning rarely eliminates all the heads or intermediate dimensions in any particular layer, meaning that the resuting speedup is often smaller compared to models with fewer layers, e.g. TinyBERT.

To remedy the problem, we present a very simple solution: besides pruning heads and intermediate dimensions as introduced in §2.2, we introduce two additional masks z_{MHA} and z_{FFN} for every MHA and FFN layer, with each controlling whether the corresponding MHA layer or FFN layer should be pruned or not. Now the multi-head self-attention and feed-forward layer become:

$$\text{MHA}(X) = z_{\text{MHA}} \sum_{i=1}^{N_h} (\mathbf{z}_{\text{head}}^{(i)} \cdot \text{Att}(W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}, W_O^{(i)}, X)$$

$$\text{FFN}(X) = z_{\text{FFN}} \cdot \text{gelu}(XW_U) \cdot \text{diag}(\mathbf{z}_{\text{int}}) \cdot W_D$$

With these layer masks introduced, we allow the model to directly prune an entire layer, instead of pruning all the heads in one MHA layer (or all the intermediate dimensions in one FFN layer). Different from the layer dropping strategies in Fan et al. (2020); Sajjad et al. (2020), our pruning strategy allows the model to drop MHA and FFN layers separately, instead of pruning them as a whole.

Furthermore, we also consider pruning the output dimensions of $\text{MHA}(X)$ and $\text{FFN}(X)$ (we call it the hidden dimension, which is always equal to d in Transformers) to allow for more flexibility in the final model structure. We define a set of masks $\mathbf{z}_{\text{hidn}} \in \{0, 1\}^d$ shared across all the layers, because each coordinate in a hidden representation is connected to the same coordinate in the next layer by a residual connection. Essentially, these mask variables apply to all the weight matrices in the model, e.g., $\text{diag}(\mathbf{z}_{\text{hidn}})W_Q$. Empirically, we find that only a small number of dimensions are pruned ($768 \rightarrow 760$), but it still helps improve performance (§4.3).

To learn these mask variables, we use l_0 regularization with the hard concrete distribution from Louizos et al. (2018). We also follow Wang et al. (2020c) to augment the objective with a Lagrangian multiplier to better control the desired pruning rate of pruned model[3]. We adapt the Lagrangian constraint accordingly to accommodate mixed levels of pruning masks—for example, the i-th head in an MHA layer is pruned if z_{MHA} is 0 or $\mathbf{z}_{\text{head}}^{(i)}$ is 0.

3.2 Distillation to Pruned Models

Previous work has shown that combining distillation with pruning can improve performance. Earlier works apply the distillation objective only to the output of the final layer (Sanh et al., 2020; Lagunas et al., 2021):

$$\mathcal{L}_{\texttt{pred}} = D_{\text{KL}}(\mathbf{p}_s \parallel \mathbf{p}_t),$$

where $\mathbf{p}_s$ and $\mathbf{p}_t$ are softmax probabilities from flattened logits from the pruned student and the full teacher model respectively.

In addition to prediction layer distillation, recent work show benefits by also adding a distillation objective in feed-forward layers by minimizing the mean squared error between the hidden representations of the teacher and student models. In the context of distillation approaches, the architecture of the student model is pre-specified, and it is straightforward to define layer mapping between the student and teacher model. For example, the 4-layer TinyBERT_4 model distills from the 3, 6, 9

[3]We also tried a straight-through estimator as proposed in Sanh et al. (2020) and find the performance comparable. We decide to stick to l_0 regularization because it is easier to control the precise pruning rate.

and 12-th layer of the teacher model. However, distilling intermediate layers during the pruning process is challenging as the model structure changes throughout training.

We propose a layer distillation approach for pruning to best utilize the signals from the teacher model. Instead of pre-defining a fixed layer mapping to compute the layer distillation loss, we dynamically learn the layer mapping between the full teacher model and the pruned student model. Specifically, let $\mathcal{T}$ denote a set of teacher layers that we use to distill knowledge to the student model. We define a layer mapping function $m(\cdot)$, i.e., $m(i)$ represents the student layer that distills from the teacher layer i. The hidden layer distillation loss is defined as

$$\mathcal{L}_{\texttt{layer}} = \sum_{i \in \mathcal{T}} \texttt{MSE}(W_{\texttt{layer}} \mathbf{H}_s^{m(i)}, \mathbf{H}_t^i),$$

where $W_{\texttt{layer}} \in \mathbb{R}^{d \times d}$ is a linear transformation matrix and $\mathbf{H}_s^{m(i)}$, $\mathbf{H}_t^i$ are hidden representations of the student FFN layer $m(i)$ and teacher FFN layer i. The layer mapping function $m(\cdot)$ is dynamically determined during pruning to match a teacher layer to its closest layer in the student model:

$$m(i) = \underset{j:\mathbf{z}_{\text{FFN}}^{(j)}>0}{\arg\min} \texttt{MSE}(W_{\texttt{layer}} \mathbf{H}_s^j, \mathbf{H}_t^i).$$

We combine the hidden layer distillation with prediction layer distillation to form our final distillation strategy.

$$\mathcal{L}_{\texttt{distil}} = \lambda \mathcal{L}_{\texttt{pred}} + (1-\lambda)\mathcal{L}_{\texttt{layer}}.$$

where λ controls the contribution of each loss.

4 Experiments

4.1 Setup

Datasets We evaluate our approach on GLUE tasks (Wang et al., 2019) including SST-2 (Socher et al., 2013), MNLI (Williams et al., 2018), QQP, QNLI and SQuAD 1.1 (Rajpurkar et al., 2016).

Training setup For each experiment, we first finetune the model with the distillation objective on the dataset for one epoch, then we continue training the model with the pruning objective with a scheduler to linearly increase the pruning rate to the target value within two epochs. We finetune the pruned model until convergence[4]. We train models with a target sparsities of $\{60\%, 70\%, 75\%, 80\%, 85\%, 90\%, 95\%\}$ on each dataset. For all experiments, we start from the $\text{BERT}_{\text{base}}$ model and freeze embedding weights following Sanh et al. (2020). Excluding word embeddings, the full model size is 84M and we use it to calculate prunnig rates throughout this paper. We report results on development sets of all datasets.

Baselines We compare against several strong pruning and distillation models, including 1) **DistillBERT**$_6$ (Sanh et al., 2019); 2) **TinyBERT**$_6$ and **TinyBERT**$_4$ (Jiao et al., 2020) both include general distillation for pretraining and task-specific distillation; 3) **DynaBERT** (Hou et al., 2020): a method that provides dynamic-sized models by specifying width and depth; 4) **Block Pruning** (Lagunas et al., 2021): a pruning method coupled with prediction-layer distillation. We choose their strongest approach "Hybrid Filled" as our baseline. We also compare to FLOP (Wang et al., 2020c), LayerDrop (Fan et al., 2020), MobileBERT (Sun et al., 2020) and AutoTinyBERT (Yin et al., 2021) in Appendix B[5].

For TinyBERT and DynaBERT, the released models are trained with task-specific data augmentation. For a fair comparison, we train these two models with the released code without data augmentation.[6] For Block Pruning, we train models with their released checkpoints on GLUE tasks and use SQuAD results from the paper.

Speedup evaluation Speedup rate is a primary measurement we use throughout the paper as compression rate does not necessarily reflect the actual improvement in inference latency. We use an unpruned $\text{BERT}_{\text{base}}$ as the baseline. We evaluate all the models with the same setup on an Nvidia V100 GPU to measure inference speedup. The input size is 128 for GLUE and 384 for SQuAD, and we use a batch size of 128. Note that the results might be different from the original papers as the environment for each platform is different.

4.2 Main Results

Overall performance In Figure 2, we compare the accuracy of MixedPruning models as a function of inference speedup and model size. MixedPrun-

[4]Please refer to Appendix A for more training details.

[5]We decided to put these results in Appendix B as they are not directly comparable to MixedPruning.

[6]For TinyBERT, the augmented data is 20× larger than the original data, which makes the training process significantly slower. Our approach can also benefit from data augmentation, which we will investigate in future work.

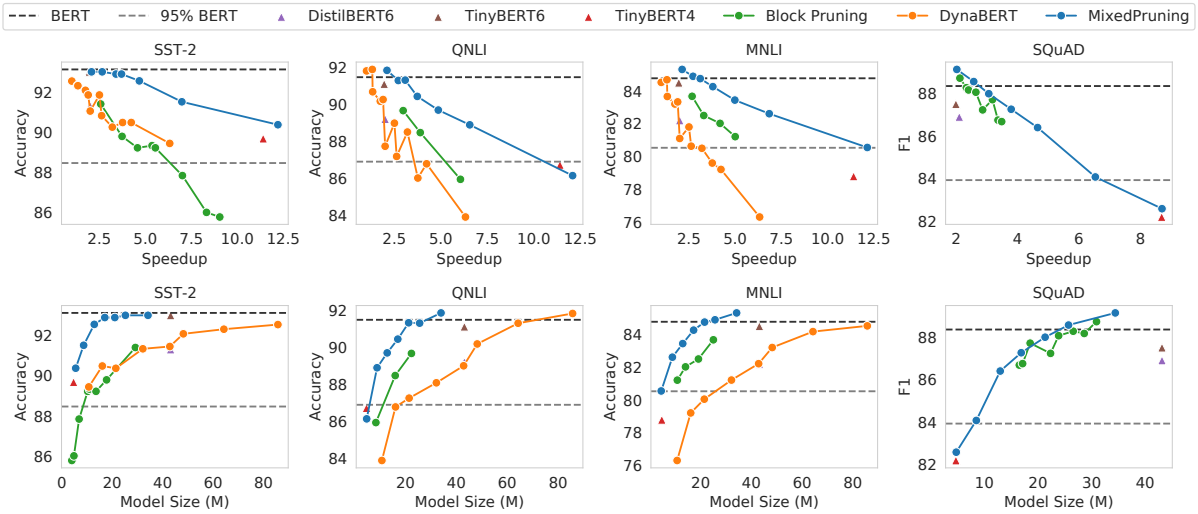

Figure 2: Accuracy v.s speedup (top) or model size (bottom). We compare MixedPruning against state-of-the-art distillation and pruning baselines. Note that we exclude embedding size when calculating model size following Lagunas et al. (2021) as forwarding through the embedding layer has little effect on inference time.

	MixedPruning		TinyBERT_4		
	speedup	accuracy	speedup	w/o GD	w/ GD
SST-2	12.0×	**90.6**	11.4×	87.7	89.7
QNLI	12.1×	86.1	11.4×	81.8	**86.7**
MNLI	12.1×	**80.6**	11.4×	78.7	78.8
QQP	11.0×	**90.1**	11.4×	89.5	90.0
SQuAD	8.7×	**82.6**	8.7×	-	82.1

Table 2: MixedPruning v.s. TinyBERT_4 models with a $\sim 10\times$ speedup. GD: general distillation (Jiao et al., 2020), which distills the student model on a large unlabeled corpus. The number of parameters of the MixedPruning models and TinyBERT_4 are around 5M (equivalently of a 95% pruning rate).

ing delivers more accurate models than distillation and pruning baselines at every speedup level and model size. Block Pruning (Lagunas et al., 2021), a recent work that shows strong performance against TinyBERT_6, is unable to achieve comparable speedups as TinyBERT_4. Instead, MixedPruning has the option to prune both layers and heads & intermediate units and can achieve a model with a comparable or higher performance compared to TinyBERT_4 and all the other models. Additionally, DynaBERT performs much worse speed-wise because it is restricted to remove at most half of the MHA and FFN layers.

Comparison with TinyBERT_4 If the final product is a compressed model, expected to reach a $10\times$ speedup compared to a full-sized model and maintain accuracy as much as possible, what would be the most effective and economical way to achieve such a model? In Table 2, we show that MixedPruning produces such models and achieves comparable or even better performance than TinyBERT_4. General distillation, which distills information from a large text corpus, is essential for training distillation models like TinyBERT. From Table 2, we observe that general distillation is indispensable in maintaining model accuracy, especially for relatively smaller-sized datasets like SST-2 and QNLI. While general distillation could take up to days for training, MixedPruning trains for tens of epochs on a task-specific dataset with a single GPU. We argue that pruning approaches —trained with distillation objectives like MixedPruning— are more economical and efficient in achieving compressed models with a high speedup.

4.3 Ablation Study

Pruning units We first conduct an ablation study to investigate how different pruning units in MixedPruning affect model performance and inference speedup. We show results in Table 3 for models of similar sizes. Only removing the option to prune hidden dimensions ($\mathbf{z}_{\text{hidn}}$) leads to a slightly faster model with a performance drop across the board. We find that it removes more layers than MixedPruning and does not lead to optimal performance under the pruning rate constraint. In addition, dropping pruning layers (z_{MHA}, z_{FFN}) brings a significant drop in terms of both model performance and speedup on highly compressed models (95%, 5M). This result shows that even with the same amount

	QNLI (60%)		QNLI (95%)		MNLI (60%)		MNLI (95%)		SQuAD (60%)		SQuAD (95%)	
MixedPruning	2.1×	**91.8**	12.1×	**86.1**	2.1×	85.1	12.1×	**80.6**	2.0×	**89.1**	8.7×	**82.6**
−hidden	2.2×	91.3	13.3×	85.6	2.1×	**85.2**	13.7×	79.8	2.0×	88.7	9.7×	80.8
−hidden & layer	2.2×	91.3	7.2×	84.6	2.1×	84.8	7.0×	78.4	2.1×	88.5	6.4×	74.1

Table 3: Ablation studies on pruning units on QNLI and SQuAD. The pruned models of a pruning rate 60% and 95% have a model size of 34M and 5M respectively. −hidden and −layer denote the variants that we remove the mask variables corresponding to hidden units ($\mathbf{z}_{\text{hidn}}$) and entire layers (z_{MHA}, z_{FFN}).

	SST-2	QNLI	MNLI	QQP	SQuAD
MixedPruning	90.6	**86.1**	**80.6**	**90.1**	**82.6**
$-\mathcal{L}_{\text{layer}}$	**91.1**	85.1	79.7	89.8	82.5
$-\mathcal{L}_{\text{pred}}, \mathcal{L}_{\text{layer}}$	86.6	84.2	78.2	88.1	75.8
Fixed Hidn Distil.	90.0	85.8	80.5	90.0	80.9

Table 4: Ablation study of different distillation objectives on pruned models with a 95% pruning rate. Fixed hidden distillation: simply matching each layer of the student and the teacher model, see §4.3 for more details.

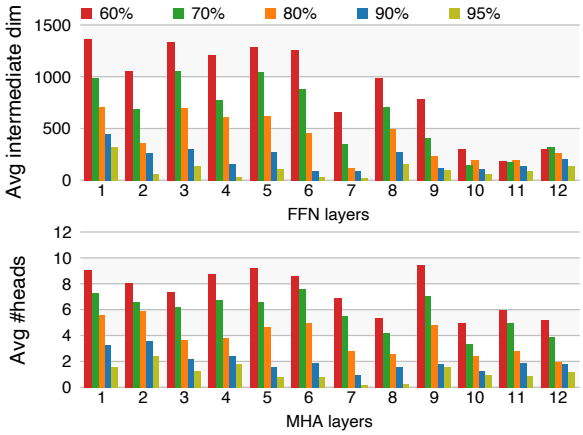

Figure 3: The average intermediate dimensions at each FFN layer and the average number of heads at each MHA layer in the pruned models across five datasets (SST-2, MNLI, QQP, QNLI, and SQuAD 1.1). We study different pruning rates $\{60\%, 70\%, 80\%, 90\%, 95\%\}$.

of parameters, different configurations for a model could lead to drastically different speedups. Removing the layer masks does not affect the lower pruning rate regime (60%, 34M), as keeping all the layers would still be the optimal choice at this pruning rate. In short, by placing masking variables at different levels, the optimization procedure is incentivized to prune units accordingly under the pruning rate constraint while maximizing the model performance.

Distillation objectives We also ablate on distillation objectives to see how each part contributes to the high performance of MixedPruning in Table 4. We first observe that removing distillation entirely leads to a performance drop up to 2-7 points across various datasets, showing the necessity to combine pruning and distillation for maintaining performance. The proposed hidden layer distillation objective dynamically matches the layers from the teacher model to the student model. A simple and intuitive alternative (named "Fixed Hidden Distillation") would be to match the layers from the teacher model to the existing layers in the student model— if the layer is pruned, the distillation to that layer stops. We find that fixed hidden distillation underperforms the dynamic layer matching objective used for MixedPruning. Interestingly, the proposed dynamic layer matching objective consistently converges to a specific alignment between the layers of the teacher model and student model. For example, we find that on QNLI the training process dynamically matches the 3, 6, 9, 12 layers in the teacher model to 1, 2, 4, 9 layers in the student model[7]. Moreover, as shown in the table, removing it hurts the performance for the majority of the datasets except SST-2.

4.4 Structures of Pruned Models

In this section, we study the model structures that are pruned from MixedPruning. We characterize the pruned models with pruning rates $\{60\%, 70\%, 80\%, 90\%, 95\%\}$ on all the five datasets that we consider. For each setting, we run MixedPruning three times. Figure 3 demonstrates the number of remaining heads and intermediate dimensions of the pruned models for different pruning rates.[8] Interestingly, we discover common structural patterns in the pruned models: (1) Feed-forward layers are significantly pruned across all pruning rates. For example, for the pruning rate 60%, the average number of intermediate dimensions in FFN layers after pruning is reduced by 71% ($3,072 \rightarrow 884$), and the average number of heads in MHA is reduced by 39% ($12 \rightarrow 7.3$). This

[7] Please refer to Appendix C for more details.
[8] We show more layer analysis in Appendix D.

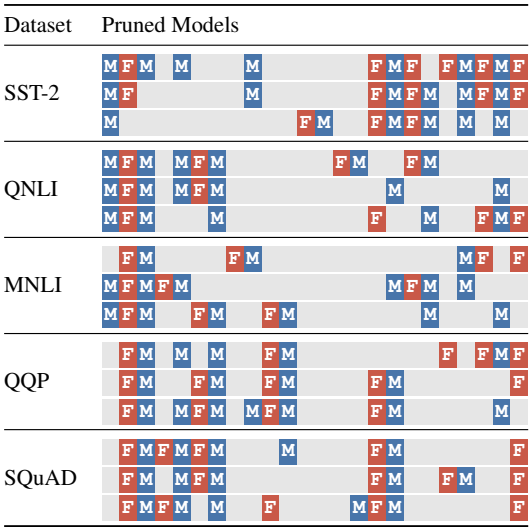

Table 5: Remaining layers in the models pruned by MixedPruning on different datasets. All models are pruned at a pruning rate of 95%. For each setting, we run the experiments three times to obtain three different pruned models. M represents a remaining MHA layer and F represents a remaining FFN layer.

suggests FFN layers are more redundant than MHA layers. (2) MixedPruning tends to prune submodules more from upper layers than lower layers. For example, upper MHA layers have fewer remaining heads than lower layers on average.

Furthermore, we study the number of remaining FFN and MHA layers and visualize the results in Table 5 for highly compressed models (pruning rate $= 95\%$). Although all the models are roughly of the same size, the remaining layers present different patterns for different datasets. We find that on SST-2 and QNLI, the first MHA layer is preserved but can be removed on QQP and SQuAD. We also observe that some layers are particularly important across all datasets. For example, the first MHA layer and the second MHA layer are preserved most of the time, while the middle layers are often removed. Generally, the pruned models contain more MHA layers than FFN layers (see Appendix D), which suggests that MHA layers are more important for solving the downstream tasks. The model structures discovered by different runs on one particularly dataset do not vary much, indicating that there exists task-specific model structures with a specific pruning rate. Similar to Press et al. (2020), we find that although standard Transformer networks have interleaving FFN layers and MHA layers, in our pruned models, adjacent FFN/MHA layers could possibly lead to a better performance.

5 Related Work

Structured pruning is more widely explored in computer vision, where channel pruning (He et al., 2017; Luo et al., 2017; Liu et al., 2017, 2019c,b; Molchanov et al., 2019) is a standard structured pruning method for convolution models. The techniques can be adapted to Transformer based models for pruning units like layers (Fan et al., 2020; Sajjad et al., 2020), feed-forward layers (Prasanna et al., 2020), heads (Michel et al., 2019; Voita et al., 2019), blocks (Lagunas et al., 2021; Yao et al., 2021) and dimensions (McCarley et al., 2019). Unstructured pruning is another major research direction, especially gaining popularity in the theory of Lottery Ticket Hypothesis (Frankle and Carbin, 2019; Zhou et al., 2019; Renda et al., 2020; Frankle et al., 2020). Several works explore pruning at initialization with or without using training data (Lee et al., 2019; Wang et al., 2020a; Tanaka et al., 2020; Su et al., 2020).

Besides pruning, many other techniques have been explored and proposed to improve the inference speed-up of Transformer models, including distillation (Turc et al., 2019; Wang et al., 2020b; Sanh et al., 2019; Jiao et al., 2020), quantization (Shen et al., 2020; Fan et al., 2020), dynamic inference acceleration (Xin et al., 2020) and so on. We refer the readers to Ganesh et al. (2021) for a comprehensive survey.

6 Conclusion

We proposed MixedPruning, a pruning strategy that incorporates all levels of pruning, including layers, heads, intermediate dimensions, and hidden dimensions for Transformer-based models. Coupled with a distillation objective tailored to structured pruning, we show that MixedPruning compresses models into a rather different structure from standard distillation models but still achieves competitive results with more than $10\times$ speedup. We conclude that task-specific structured pruning from large-sized models could be an appealing replacement for distillation to achieve extreme model compression without resorting to expensive pre-training or data augmentation. We hope that future research continues this line of work by investigating structured pruning for task-agnostic models, given that pruning from a large pre-trained model could save computation from general distillation and results in compressed models with a more flexible structure.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing Transformer depth on demand with structured dropout. In *International Conference on Learning Representations (ICLR)*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *icml*, pages 3259–3269. PMLR.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association of Computational Linguistics (TACL)*, 9:1061–1080.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems (NIPS)*, 28.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 4163–4174.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. 2021. Block pruning for faster transformers. *arXiv preprint arXiv:2109.04838*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2019. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021. Differentiable subset pruning of Transformer heads. *Transactions of the Association of Computational Linguistics (TACL)*.

Zi Lin, Jeremiah Liu, Zi Yang, Nan Hua, and Dan Roth. 2020. Pruning redundant mappings in transformer models via spectral-normalized identity prior. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 719–730.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. 2019b. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3296–3305.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019c. Rethinking the value of network pruning. In *International Conference on Learning Representations (ICLR)*.

C Louizos, M Welling, and DP Kingma. 2018. Learning sparse neural networks through l0 regularization. In *International Conference on Learning Representations (ICLR)*.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066.

JS McCarley, Rishav Chakravarti, and Avirup Sil. 2019. Structured pruning of a BERT-based question answering model. *arXiv preprint arXiv:1910.06360*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT plays the lottery, all tickets are winning. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229.

Ofir Press, Noah A Smith, and Omer Levy. 2020. Improving transformer models by reordering their sublayers. In *Association for Computational Linguistics (ACL)*, pages 2996–3005.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text Transformer. *The Journal of Machine Learning Research (JMLR)*, 21(140).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations (ICLR)*.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's BERT: Smaller and faster transformer models. *arXiv preprint arXiv:2004.03844*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian based ultra low precision quantization of BERT. In *Conference on Artificial Intelligence (AAAI)*, pages 8815–8821.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jingtong Su, Yihang Chen, Tianle Cai, Tianhao Wu, Ruiqi Gao, Liwei Wang, and Jason D Lee. 2020. Sanity-checking pruning methods: Random tickets can win the jackpot. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 20390–20401. Curran Associates, Inc.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 4314–4323.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic bert for resource-limited devices. In *Association for Computational Linguistics (ACL)*, pages 2158–2170.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems (NeurIPS)*, 33.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NIPS)*, 30:5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Association for Computational Linguistics (ACL)*, pages 5797–5808.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020a. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations (ICLR)*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020b. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained Transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020c. Structured pruning of large language models. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Association for Computational Linguistics (ACL)*, pages 2246–2251.

Zhewei Yao, Linjian Ma, Sheng Shen, Kurt Keutzer, and Michael W Mahoney. 2021. Mlpruning: A multilevel structured pruning framework for transformer-based models. *arXiv preprint arXiv:2105.14636*.

Yichun Yin, Cheng Chen, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2021. AutoTinyBERT: Automatic hyper-parameter optimization for efficient pre-trained language models. In *Association for Computational Linguistics (ACL)*, pages 5146–5157.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32. Curran Associates, Inc.

A Reproducibility & Hyperparameters

We report the hyperparameters that we use in our experiments in Table 6. We will be releasing codes and running scripts in the final version.

Hyperparameter	Value
λ	0.1, 0.3, 0.5
temperature t	2
training learning rate	3e-5
training epochs	20
finetuning epochs	20
finetuning learning rate	1e-5, 2e-5, 3e-5
batch size	32 (GLUE), 16 (SQuAD)

Table 6: Hyperparemeters in the experiments.

B Pruning & Distillation Methods Comparison

We show additional pruning and distillation methods that are not directly comparable to our method in this section.

Method	speedup	SST-2	QNLI	MNLI	SQuAD
Wang et al. (2020c)†‡	1.5X	92.1	89.1	-	85.4
Sajjad et al. (2020)†	2.0X	90.3	-	81.1	-
Fan et al. (2020)†‡	2.0X	93.2	89.5	84.1	-
Sun et al. (2020)	2.3X	92.1	91.0	83.9	90.3
Yin et al. (2021)♠	4.3X	91.4	89.7	82.3	87.6
MixedPruning (ours)	2.0X	93.0	91.8	85.3	89.1
MixedPruning (ours)	4.6X	92.6	89.7	83.4	86.4

Table 7: More pruning and distillation baselines. † denotes that it is a pruning method without distillation training. ‡ denotes that the model prunes from a $\text{RoBERTa}_{\text{base}}$ model. ♠ denotes that the model is distilled from an $\text{Electra}_{\text{base}}$ model. MobileBERT (Sun et al., 2020) has special architecture designs and distills from a $\text{BERT}_{\text{large}}$ model.

C Alignment from Dynamic Layer Matching

We find that the alignment between the layers of the student model and the teacher model shifts during the course of training. To take SST-2 for an example, at the beginning of training, the last layer of the pretrained model matches to all four layers of the teacher model. As the training goes on, the model learns the alignment to match the $7, 9, 10, 11$ layers of the student model to the $3, 6, 9, 12$ layers of the teacher model. For QQP, the model eventually learns to map $2, 5, 8, 11$ layers to the four layers of the teacher model. The final alignment shows that our dynamic layer matching distillation objective is able to find task-specific alignment and improves performance.

D FFN/MHA Layers in Pruned Models

Figure 4 shows the average number of FFN layers and MHA layers in the pruned models by MixedPruning. We study different pruning rates $\{60\%, 70\%, 80\%, 90\%, 95\%\}$. It is clear that when the pruning rate increases, the pruned models become shallower (i.e., the number of layers becomes fewer). Furthermore, we find that the pruned models usually have more MHA layers than FFN layers. This may indicates that MHA layers are more important for solving these downstream tasks compared to FFN layers.

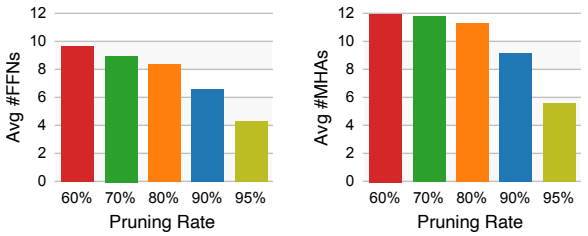

Figure 4: The average number of FFN layers and MHA layers in the pruned models at different pruning rates.