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. The two predominant approaches are *pruning*, which gradually removes weights from a pre-trained model, and *distillation*, which trains a smaller compact model to match a larger one. Pruning methods can significantly reduce the model size but hardly achieve large speedups as distillation. However, distillation methods require large amounts of unlabeled data and are expensive to train. In this work, we aim to close this gap and propose a structured pruning method—MixedPruning—which matches the distillation counterparts in both latency and accuracy and only incurs 5% of training cost without using unlabeled data. Our key insight is to jointly prune coarse (e.g., layers) and fine-grained (e.g., heads and hidden units) modules, which controls the pruning decision of each parameter with masks of different granularity. This pruning strategy eases optimization and delivers highly competitive and parallelizable subnetworks that were not demonstrated before. We also propose a novel layer-wise distillation approach to further guide pruning. We evaluate MixedPruning extensively on SQuAD and GLUE datasets and demonstrate its effectiveness and efficiency over state-of-the-art pruning and distillation methods.

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 search for an accurate subnetwork in a larger, pre-trained model. Recent work has investigated how to structurally prune Transformer networks (Vaswani et al., 2017), 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× improvement at most.

On the contrary, distillation methods (Sanh et al., 2019; Sun et al., 2019) first specify a shallow model architecture and directly distill knowledge from the teacher model (e.g., TinyBERT$_4$ (Jiao et al., 2020) consists of 4 layers with a 312 hidden size). Due

	Speedup	# Params	MNLI
BERT$_{\text{base}}$	1.0×	85M	84.8
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. The speed-ups are all reported against the BERT$_{\text{base}}$ model. Embeddings are excluded in calculating # of parameters. All the models except for those labeled as ‡ use BERT$_{\text{base}}$ as the teacher model, and we evaluate all the models on an NVIDIA V100 GPU (§4.1). Green models have large speed-ups and are one order of magnitude smaller. 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 their reported number in the paper (TinyBERT$_4$, 80.5 on MNLI). The only difference is that we use our own teacher model for distillation, which is used for all the experiments.

to the compact structure, these models can easily obtain $10\times$ times speedup at inference time. However, these randomly-initialized student networks[2] require large amounts of unlabeled data to learn the knowledge in pre-trained models (namely "general distillation") and hence make the distillation process prohibitly slow. For instance, TinyBERT$_4$ is first trained on 2,500M tokens for 3 epochs, which requires training 3.5 days on 4 GPUs.[3]

In this work, we propose a structured pruning approach MixedPruning, which aims to close the gap between pruning and distillation. We show a surprising finding: *structured pruning can achieve highly compact subnetworks and obtain large speedups and competitive accuracy as distillation approaches* (Table 1). Unlike distillation, our approach only learns from downstream task data and incurs less than 5% of training cost (Figure 1). Our key insight is to jointly prune bigger units (e.g., entire self-attention or feed-forward layers) and smaller units (e.g., heads, intermediate and hidden dimensions) simultaneously. Compared to existing approaches which prune a single submodule at a time, our approach controls the pruning decision of every single parameter by multiple masks of different granularity and all the masking variables (along with model parameters) are jointly optimized. Despite its simplicity, we show that this mixed-pruning strategy is the key to large compression—It allows the greatest flexibility of pruned structures and eases the optimization compared to only pruning smaller units.

Additionally, we 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 devise 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 the GLUE tasks (Wang et al., 2019) and SQuAD (Rajpurkar et al., 2016) compared to strong pruning and distillation baselines. In particular, it achieves $>10\times$ speedups and a 95% sparsity across all the datasets while preserving $>90\%$ 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 general distillation. Finally, we also discover interesting patterns in pruned structures (Table 5). For example, we find that for highly compressed models, the pruning process favors preserving upper and lower layers and pruning middle layers most of the time.

[2]There are exceptions like DistillBERT (Sanh et al., 2020) which initializes the student from the teacher by taking one layer out of two. It is unclear how to generalize this initialization to other compact structures though.

[3]See training time measurement details in Appendix J.

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).

Next comes a feed-forward layer, which consists of an up-projection and a down-projection layer, parameterized 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, $d_f = 4d$. 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.[4] 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 remove redundant parameters from a large model for compression.

Layer pruning Fan et al. (2020) and Sajjad et al. (2020) explored strategies to drop entire Transformer blocks (a pair of MHA and FFN layer) from a pre-trained model. Empirical evidence suggested that 50% of the layers can be dropped without a big accuracy loss, resulting in a $2\times$ speedup.

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

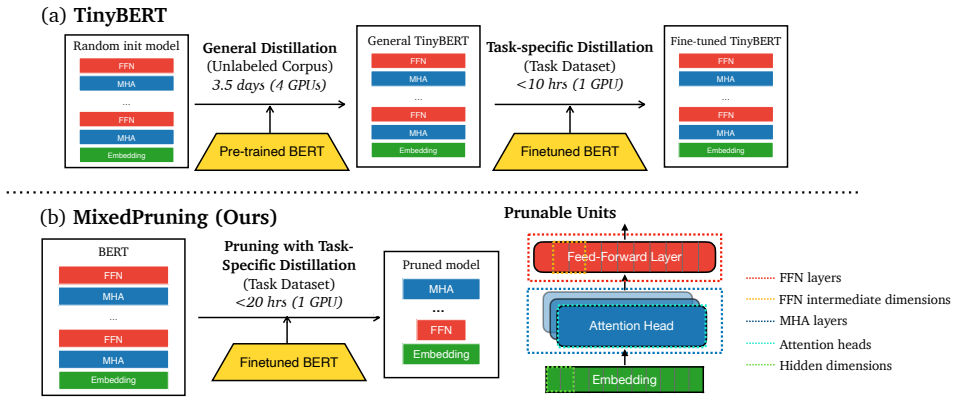

Figure 1: Comparison of (a) TinyBERT (Jiao et al., 2020) and (b) our pruning approach MixedPruning. TinyBERT trains a randomly-initialized network through two-step distillation: (1) general distillation on a large-scale corpus, which takes 3.5 days to finish on 4 GPUs, and (2) task-specific distillation on the task dataset. MixedPruning directly prunes the fine-tuned BERT model and jointly prunes five different types of units (Section 3.1) and takes at most 20 hours to finish on 1 GPU on all the GLUE datasets (smaller datasets need $<$ 3 hour). Note that for clarity, we do not show the prunable hidden dimensions in Attention Head and Feed-Forward Layer.

Head pruning Head pruning Voita et al. (2019), Michel et al. (2019) show that 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× 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 As the other major part—feed-forward layers (FFNs)—are also known to be over-parameterized. Strategies to prune an FFN layer for an inference speedup include pruning an entire FFN layer (Prasanna et al., 2020; Chen et al., 2020) and at a more fine-grained level, pruning intermediate dimensions (McCarley et al., 2019; Hou et al., 2020) by introducing $\mathbf{z}_{\text{int}} \in \{0, 1\}^{d_f}$:

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

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 to optimize 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 transfers knowledge from a large teacher model to a student model with a pre-defined compact structure. Previous work has designed layer mapping strategies to distill layer-wise signals to a fixed student model (Sun et al., 2019, 2020; Jiao et al., 2020). More recently, Hou et al. (2020) attempted to distill to a more dynamic structure with specified widths and heights. Yin et al. (2021) adopt a one-shot Neural Architecture Search solution to automatically search architecture of student networks. However, these resulting models still have a constrained structure compared to pruning-based approaches. In this work, apart from designing a new pruning strategy, we propose a layerwise distillation objective tailored to structured pruning and show that combining the two leads to the best empirical performance.

3 Method

Our pruning method, MixedPruning, involves two simple yet effective ideas: (1) we prune each single parameter through multiple masking decisions of different granularity (§3.1); (2) we propose a layerwise distillation strategy to effectively transfer knowledge from a full model to a student model with constantly changing structures (§3.2).

3.1 Mixed Pruning of Different Granularity

Our pruning strategy couples pruning decisions over different-sized and nested units and delivers

parallelizable models with flexible structures.

We observed from existing work that pruning only large modules (e.g., FFN layers) constrains the flexibility of pruned model structure; however, pruning only small units (e.g., individual heads) never leads to removing a larger submodule as it naturally entails. More specifically, existing head or intermediate dimension pruning strategies rarely eliminate all the units in one layer, prohibiting highly compressed models from achieving significant speedups.

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 $\text{FFN}(X)$, referred to as 'hidden dimensions' in this paper, 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, as each coordinate in a hidden representation is connected to the same coordinate in the next layer through a residual connection. These mask variables are applied 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 (e.g., $768 \rightarrow 760$), but it still helps improve performance significantly (§4.3).

Although our mixed-pruning strategy is straightforward, it greatly differs from the previous pruning approaches in that multiple mask variables jointly control the pruning decision of one single parameter. For example, a weight in an FFN layer may be pruned because the entire FFN layer is pruned, or its corresponding intermediate dimension is pruned, or the hidden dimension is pruned. As a comparison, the most recent work—Block Pruning (Lagunas et al., 2021) adopts a hybrid approach which simply prunes a different type of unit from MHAs and FFNs separately and thus is significantly different from MixedPruning.

To learn these mask variables, we use l_0 regularization with the hard concrete distribution from Louizos et al. (2018). We follow Wang et al. (2020c) to replace the vanilla l_0 objective with a Lagrangian multiplier to better control the desired sparsity of pruned model[5]. 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 (see Appendix B for more details).

3.2 Distillation to Pruned Models

Previous work has shown that combining distillation with pruning improves performance, where the distillation objective only involves a cross-entropy loss between the pruned student's output and the teacher's probability distribution $\mathbf{p}_s$ and $\mathbf{p}_t$ (Sanh et al., 2020; Lagunas et al., 2021): $\mathcal{L}_{\texttt{pred}} = D_{\texttt{KL}}(\mathbf{p}_s \parallel \mathbf{p}_t)$. In addition to prediction layer distillation, recent works show great benefits in distillation of intermediate layers (Sun et al., 2019; Jiao et al., 2020). In the context of distillation approaches, the architecture of the student model is pre-specified, and it is straightforward to define a layer mapping between the student and teacher model. For example, the 4-layer TinyBERT$_4$ model distills from the 3, 6, 9 and 12-th layer of a 12-layer teacher model. However, distilling intermediate layers during the pruning process is challenging as the model structure changes throughout training.

We propose a layerwise distillation approach for pruning to best utilize the signals from the teacher model. Instead of pre-defining a fixed layer mapping, we dynamically search a 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

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

[5]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 sparsity.

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

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

Though an upper student layer would possibly match a lower teacher layer or more than one student layer could match a teacher, empirically such scenarios rarely happen. This shows the superiority of dynamic matching – layers between student and teacher models match in a way that benefits the pruning process the most.

We combine layer distillation with logit 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 8 GLUE tasks (Wang et al., 2019) and SQuAD 1.1 (Rajpurkar et al., 2016). GLUE tasks include SST-2 (Socher et al., 2013), MNLI (Williams et al., 2018), QQP, QNLI, MRPC (Dolan and Brockett, 2005), CoLA (Warstadt et al., 2019), STS-B (Cer et al., 2017) and RTE. Please refer to Appendix D for dataset size and metrics.

Training setup In our experiments, *sparsity* is computed as the percentage of the number of pruned parameters out of the full model size (excluding embeddings). Following Wang et al. (2020c) and Lagunas et al. (2021), we first finetune the model with the distillation objective for a number of steps, then we continue training the model with the pruning objective with a scheduler to linearly increase the sparsity to the target value. We finetune the pruned model until convergence[6]. We train models with target sparsities of $\{60\%, 70\%, 75\%, 80\%, 85\%, 90\%, 95\%\}$ on each dataset. For all the experiments, we start from the $\text{BERT}_{\text{base}}$ model[7] and freeze embedding weights following Sanh et al. (2020). Excluding word embeddings, the full model size is 85M and we use it to calculate sparsity rates throughout this paper. We report results on development sets of all datasets.

Baselines We compare MixedPruning against several baselines: **DistillBERT**$_6$ (Sanh et al., 2019), **TinyBERT**$_6$ and **TinyBERT**$_4$ (Jiao et al., 2020), **DynaBERT** (Hou et al., 2020), and **Block Pruning** (Lagunas et al., 2021) (see Appendix C for details). We also compare to other pruning methods such as FLOP (Wang et al., 2020c), LayerDrop (Fan et al., 2020), Movement Pruning (Sanh et al., 2020), and distillation methods such as MobileBERT (Sun et al., 2020) and AutoTinyBERT (Yin et al., 2021) in Appendix E[8].

For TinyBERT and DynaBERT, the released models are trained with augmented dataset. For a fair comparison, we train these two models with the released code without data augmentation (additional results with data augmentation can be found in Appendix H).[9] 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 and evaluate all the models with the same hardware setup on a single GPU to measure inference speedup. The input size is 128 for GLUE tasks 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 to other methods in terms of both inference speedup and model size. MixedPruning 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 compara-

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

[7] We also apply MixedPruning to RoBERTa. Please refer to Appendix I for details.

[8] We show these results in Appendix E as they are not directly comparable to MixedPruning.

[9] For TinyBERT, the augmented data is 20× larger than the original data, making the training process significantly slower.

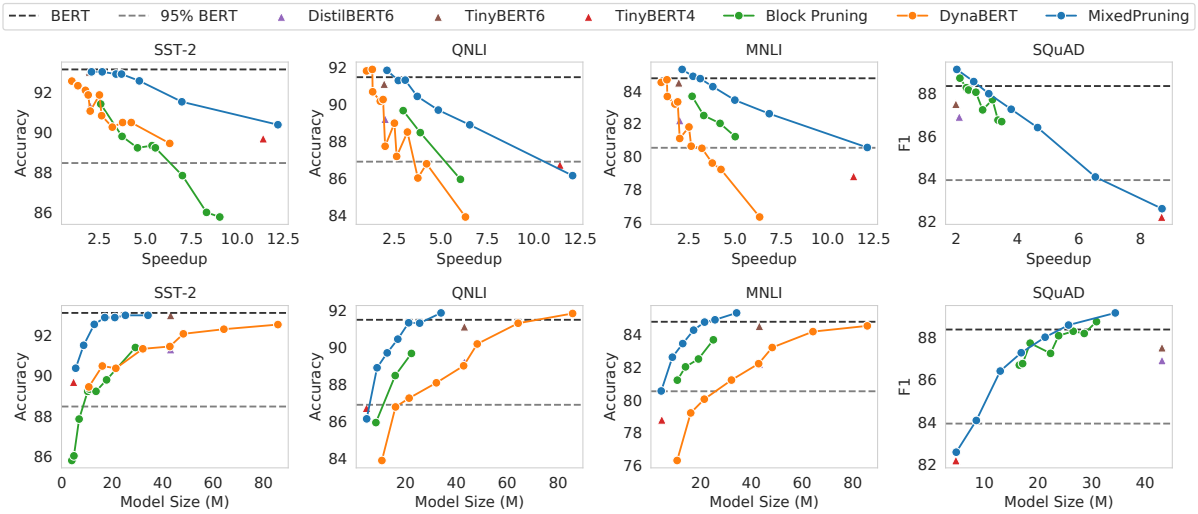

Figure 2: Accuracy v.s speedup (top) or model size (bottom). We compare MixedPruning against state-of-the-art distillation and pruning baselines. Note that we exclude embedding size when calculating model size following Lagunas et al. (2021) as forwarding through the embedding layer has little effect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

Task	SST-2 (67k)	QNLI (105k)	MNLI (393k)	QQP (364k)	CoLA (8.5k)	RTE (2.5k)	STS-B (7k)	MRPC (3.7k)	SQuAD (88k)	Train Time
TinyBERT_4 w/o GD	87.7	81.8	78.7	89.5	16.6	47.3	17.8	68.9	-	≤ 10
TinyBERT_4	89.7	**86.7**	78.8	90.0	32.5	63.2	**85.0**	81.4	82.1	~ 350
Speedup	11.4×	11.4×	11.4×	11.4×	11.4×	11.4×	11.4×	11.4×	8.7×	-
MixedPruning	**90.6**	86.1	**80.6**	**90.1**	**35.6**	**64.7**	83.1	**82.6**	**82.6**	≤ 20
Speedup	12.0×	12.1×	12.1×	11.0×	11.5×	11.9×	12.9×	11.9×	8.7×	-

Table 2: MixedPruning v.s. TinyBERT_4 (Jiao et al., 2020) models with a $\sim$10× speedup. GD: general distillation, which distills the student model on a large unlabeled corpus. Train time is measured in GPU hours (see Appendix J for details). The number of parameters for both models are around 5M (around 95% sparsity). MixedPruning closes the gap between distillation and pruning with significantly less computation. Note that we remove data augmentation from TinyBERT for a fair comparison, see Table 9 for experiments with augmented data.

ble 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 In Table 2, we show that MixedPruning produces models with over 10× inference speedup and achieves comparable or even better performance than TinyBERT_4. General distillation (GD), which distills information from a large corpus, is essential for training distillation models, especially for small-sized datasets (e.g., TinyBERT_4 w/o GD performs poorly on CoLA, RTE and STS-B). While general distillation could take up to hundreds of GPU hours for training, MixedPruning trains for a maximum of 20 hours 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 large speedup.

4.3 Ablation Study

Pruning units We first conduct an ablation study to investigate how additional pruning units such as MHA layers, FFN layers and hidden units in MixedPruning affect model performance and inference speedup beyond the standard practice of pruning heads and FFN dimensions. We show results in Table 3 for models of similar sizes. Removing the option to prune hidden dimensions ($\mathbf{z}_{\text{hidn}}$) leads to a slightly faster model with a performance drop across the board and we find that it removes more layers than MixedPruning and does not lead to optimal performance under a specific sparsity constraint. In addition, removing the layer masks ($\mathbf{z}_{\text{MHA}}$, $\mathbf{z}_{\text{FFN}}$) brings a significant drop in terms

	QNLI (60%)		QNLI (95%)		MNLI (60%)		MNLI (95%)		SQuAD (60%)		SQuAD (95%)	
	↗	acc	↗	acc	↗	acc	↗	acc	↗	F1	↗	F1
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
−layer	2.1×	91.5	8.3×	86.7	2.1×	85.4	8.4×	80.6	2.0×	89.1	7.9×	80.5
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
−layer & hidden	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, MNLI and SQuAD. ↗: speedup. The pruned models of a sparsity 60% and 95% have a model size of 34M and 5M respectively. −layer: When we do not prune entire layers (no z_{MHA} or z_{FFN}), the speed-ups are greatly reduced for a high sparsity e.g., 95%; −hidden: when we remove the mask variables corresponding to hidden units ($\mathbf{z}_{\text{hidn}}$), we observe a significant drop in accuracy.

	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 sparsity = 95%. Fixed hidden distillation: simply matching each layer of the student and the teacher model, see §4.3 for more details. In subsection F.2, we show that the dynamic layer distillation objective improves model performance more significantly on lower sparsity rates.

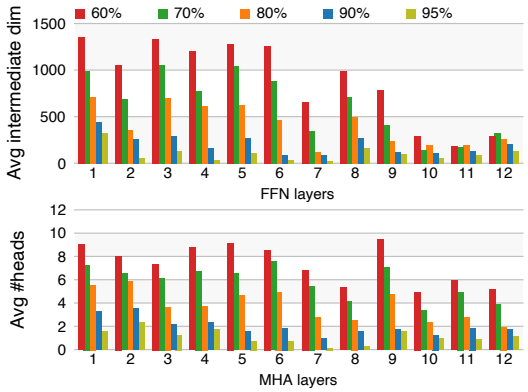

Figure 3: The average intermediate dimensions at each FFN layer and the average number of heads at each MHA layer in the pruned models across five datasets (SST-2, MNLI, QQP, QNLI, and SQuAD 1.1). We study different sparsities $\{60\%, 70\%, 80\%, 90\%, 95\%\}$.

of both model performance and speedup on highly compressed models (95%, 5M). This result shows that even with the same amount of parameters, different configurations for a model could lead to drastically different speedups. However, it does not affect the lower sparsity regime (60%, 34M). In short, by placing masking variables at different levels, the optimization procedure is incentivized to prune units accordingly under the sparsity 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[10]. 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 sparsities $\{60\%, 70\%, 80\%, 90\%, 95\%\}$ 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 sparsities.[11] Interestingly, we discover common

[10] Please refer to subsection F.1 for more details.
[11] We show more layer analysis in Appendix G.

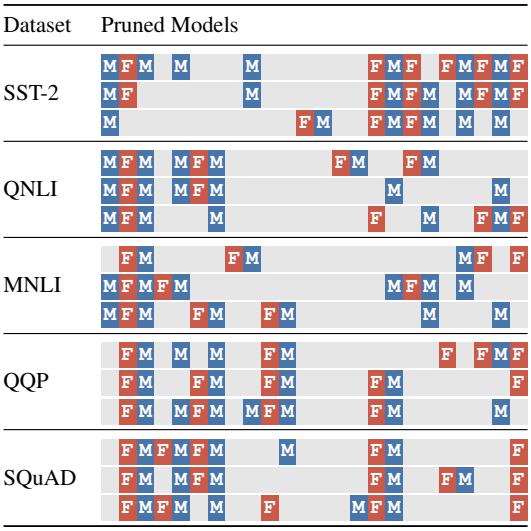

Table 5: Remaining layers in the models pruned by MixedPruning on different datasets. All models are pruned at a sparsity of 95%. For each setting, we run the experiments three times to obtain three different pruned models. M represents a remaining MHA layer and F represents a remaining FFN layer.

structural patterns in the pruned models: (1) Feed-forward layers are significantly pruned across all sparsities. For example, for the sparsity 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 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 (sparsity $= 95\%$). 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 G), 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 sparsity. 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 approach for convolution models. The techniques can be adapted to Transformer based models as introduced in subsection 2.2. 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 to gain 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., 2021), dynamic inference acceleration (Xin et al., 2020). 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

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. 2020. Earlybert: Efficient bert training via early-bird lottery tickets. *arXiv preprint arXiv:2101.00063*.

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.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

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

Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. 2021. Training with quantization noise for extreme model compression. 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.

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)*.

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.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

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

Table 6: Hyperparemeters in the experiments.

For 4 relatively larger GLUE datasets, MNLI, QNLI, SST-2 and QQP, and SQuAD, we train the model for 20 epochs in total and finetune the finalized sub-network for another 20 epochs. In the first 20 epochs, following (Lagunas et al., 2021) and (Wang et al., 2020c), we first finetune the model with the distillation objective for 1 epoch, and then start pruning with a linear schedule to achieve the target sparsity within 2 epochs. For the 4 small GLUE datasets, we train the model for 100 epochs in total and finetune for 20 epochs. We finetune the model with the distillation objective for 4 epoch and prune till the target sparsity within next 20 epochs. Note that even if the final sparsity is achieved, the pruning process keeps searching better performing structures in the rest of the training epochs. In addition, we find that finetuning the final subnetwork is essential for high sparsity models. Hyperparameters like λ, batch size and learning rate do not affect performance much.

B Optimization Details

Louizos et al. (2018) proposes l_0 optimization for model compression where the masks are modelled with hard concrete distributions as follows:

$$\begin{aligned}
\mathbf{u} &\sim U(0,1) \\
\mathbf{s} &= \text{sigmoid}\left(\frac{1}{\beta}\log\frac{\mathbf{u}}{\mathbf{1}-\mathbf{u}} + \log\alpha\right) \\
\tilde{\mathbf{s}} &= \mathbf{s}\times(r-l)+l \\
\mathbf{z} &= \min(1, \max(0, \tilde{\mathbf{s}})).
\end{aligned}$$

$U(0,1)$ is a uniform distribution in the interval $[0,1]$; $l < 0$ and $r > 0$ are two constants that stretch the sigmoid output into the interval (l, r). β is a hyperparameter that controls the steepness of the sigmoid function and $\log\alpha$ is the main learnable parameter. We learn the masks through updating the learnable parameters of the distributions from which the masks are sampled in the forward pass.

In our preliminary experiments, we find that optimizing $\lambda\|\mathbf{z}\|_0$ with different learning rates and pruning schedules may converge to models of drastically different sizes. Hence, we follow Wang et al. (2020c) to add a Lagrangian term, which imposes an equality constraint $s(\alpha) = t$ by introducing a violation penalty:

$$\mathcal{L}_c = \lambda_1 \cdot (s(\alpha) - t) + \lambda_2 \cdot (s(\alpha) - t)^2,$$

where $s(\alpha)$ is the expected model sparsity calculated from $\mathbf{z}$.

Different from (Wang et al., 2020c), where each parameter is controlled by one single mask, in MixedPruning, each parameter is controlled by three masks with the sparsity calculated as follows:

$$\begin{aligned}
s(\alpha) = &\frac{1}{M}\cdot 4 \cdot \sum_i^L \sum_j^{N_h} \sum_k^{d_{\text{model}}} \mathbf{z}^{(i)}_{\text{MHA}} \cdot \mathbf{z}^{(i,j)}_{\text{head}} \cdot \mathbf{z}^{(k)}_{\text{hidden}} \\
&+ \frac{1}{M}\cdot 2 \cdot \sum_i^L \sum_j^{4\cdot d_{\text{model}}} \sum_k^{d_{\text{model}}} \mathbf{z}^{(i)}_{\text{FFN}} \cdot \mathbf{z}^{(i,j)}_{\text{int}} \cdot \mathbf{z}^{(k)}_{\text{hidden}}
\end{aligned}$$

where M denotes the full model size.

C Details of Baseline Methods

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.

D Data Statistics

We show train sizes and metrics for each dataset we use in Table 7.

E Additional Comparisons

E.1 Comparison to Movement Pruning

We compare MixedPruning with a state-of-the-art **unstructured pruning** method, Movement Prun-

Task	Train Size	Metric
SST-2	67k	accuracy
QNLI	105k	accuracy
MNLI	393k	accuracy
QQP	364k	accuracy
CoLA	8.5k	Matthews corr.
RTE	2.5k	accuracy
STS-B	7k	Spearman corr.
MRPC	3.7k	accuracy
SQuAD	88k	F1

Table 7: Data statistics of GLUE and SQuAD datasets.

ing (Sanh et al., 2020) in Figure 4. As Movement Pruning is trained with logit distillation only, we also show results of MixedPruning trained with the same distillation objective. We observe that MixedPruning largely outperforms Movement Pruning even without layerwise distillation on MNLI and is comparable to SQuAD on models with a size over $10M$ parameters. MixedPruning, as a structured pruning method, is less performant on models of a sparsity higher than 95%, as pruning flexibility is largely restricted by the smallest pruning submodule. However, pruned models from MixedPruning achieve $2-11\times$ inference speedups while no speedup gains are achieved from Movement Pruning.

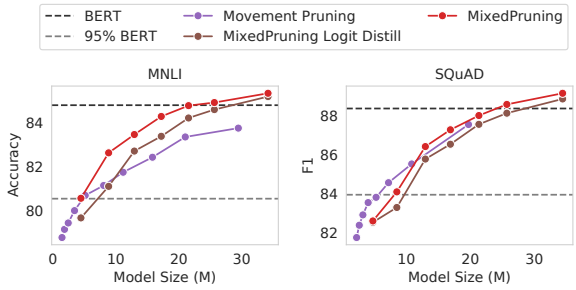

Figure 4: MixedPruning v.s. Movement Pruning (unstructured pruning). MixedPruning Logit Distill denotes that we run MixedPruning with logit distillation only as Movement Pruning.

E.2 Comparison to Block Pruning

In Figure 6, we compare MixedPruning with Block Pruning while unifying the distillation objective. Without the layer distillation objective, MixedPruning still outperforms or is on par with Block Pruning. Block Pruning never achieves a speedup of 10 even the pruned model is of a similar size as MixedPruning (SST-2), backing up our argument that pruning layers for high sparsity models is the key to high speedups.

E.3 More Baselines

We show additional pruning and distillation methods that are not directly comparable to our method in Table 8. MixedPruning still largely outperforms these baselines even though these methods hold an inherent advantage due to a stronger teacher or base model.

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

Table 8: More pruning and distillation baselines. ↗: speedup. ‡ denotes that the model prunes from a $\text{RoBERTa}_{\text{base}}$ model. ♠: AutoTinyBERT is distilled from an $\text{ELECTRA}_{\text{base}}$ model. ◇: MobileBERT (Sun et al., 2020) has specialized architecture designs and trains their own teacher model from the scratch. MixedPruning models have a model size of 34M and 13M respectively, corresponding to a 60% and 85% sparsity.

F More Analyses on Layer Distillation

F.1 Layer Alignment

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.

F.2 Ablation on Distillation Objectives

In Table 10, we show ablation studies on adding the dynamic layer distillation onto prediction distillation across all sparsities. Using the layer distillation loss clearly helps improve the performance on all sparisity rates and on different tasks.

Task	MixedPruning w/ DA speedup	MixedPruning w/ DA acc	MixedPruning w/o DA speedup	MixedPruning w/o DA acc	TinyBERT$_4$ speedup	TinyBERT$_4$ w/ DA acc†	TinyBERT$_4$ w/ DA acc‡	TinyBERT$_4$ w/o DA acc
SST-2	12.4×	**92.4**	12.0×	90.6	11.4×	91.6	92.7	89.7
QNLI.	13.6×	86.8	12.1×	86.1	11.4×	**87.6**	88.0	86.7
RTE	11.5×	**67.5**	11.9×	64.7	11.4×	62.5	65.7	63.2
MRPC	12.1×	**84.6**	11.9×	82.6	11.4×	83.6	85.8	81.4

Table 9: MixedPruning v.s. TinyBERT$_4$ (Jiao et al., 2020) trained with augmented data and without augmented data. The MixedPruning models have a sparsity of 95%. DA: data augmentation, introduced in Jiao et al. (2020). † denotes that we train TinyBERT$_4$ with the same copy of data and same teacher model as MixedPruning. ‡ denotes the numbers from released checkpoints of TinyBERT. The difference between these two sets of scores may stem from augmented data or teacher models.

G FFN/MHA Layers in Pruned Models

Figure 5 shows the average number of FFN layers and MHA layers in the pruned models by MixedPruning. We study different sparsities $\{60\%, 70\%, 80\%, 90\%, 95\%\}$. It is clear that when the sparsity 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.

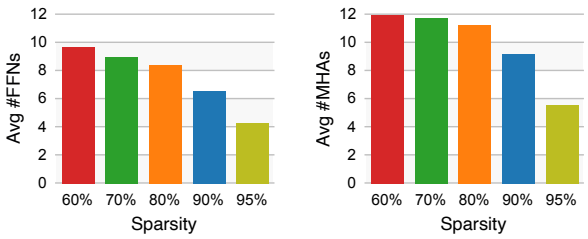

Figure 5: The average number of FFN layers and MHA layers in the pruned models at different sparsities.

H Results with Data Augmentation

We compare MixedPruning with TinyBERT$_4$ under the data augmentation setting in Table 9. As the augmented dataset is not released by TinyBERT authors, we follow its GitHub repository to create our own augmented data. We train MixedPruning with the same set of augmented dataset and find that it outperforms or is comparable to TinyBERT$_4$. Note that we only conduct experiments with data augmentation on four datasets due to the high training costs. For example, augmenting the MNLI dataset and training on the augmented dataset will take $>$ 200 GPU hours in total.

I RoBERTa Pruning

We show MixedPruning results with RoBERTa in Figure 7 across sparsities from 60% to 95%. Similar to BERT, models with 60% weights pruned are able to maintain of a full model. Pruning from RoBERTa outperforms BERT on sparsities lower than 90% but as the sparsity further increases, BERT surpasses RoBERTa. Similar patterns are observed from DynaBERT (Hou et al., 2020).

J Training Time Measurement

We use NVIDIA RTX 2080Ti GPUs to measure the training time of TinyBERT. For the general distillation step of TinyBERT, we measure the training time on a small corpus (containing 10.6M tokens) on 4 GPUs and estimate the training time on the original corpus (containing 2500M tokens) by scaling the time with the corpus size difference. Specifically, it takes 430s to finish one epochs on 10.6M tokens with 4 GPUs, and we estimate that it will take 338 GPU hours (or 3.5 days with 4 GPUs) to finish three epochs on 2500M tokens.

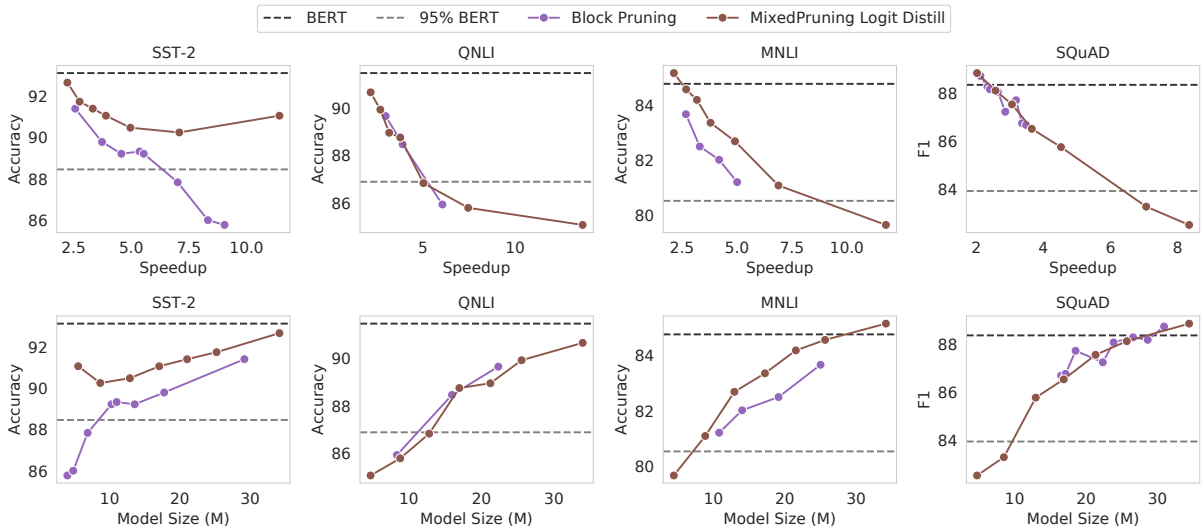

Figure 6: MixedPruning v.s. Block Pruning with the same distillation objective – logit distillation. MixedPruning still outperforms or is on par with Block Pruning.

	SST-2		QNLI		MNLI		SQuAD	
sparsity	$\mathcal{L}_{\texttt{pred}}$	$+\mathcal{L}_{\texttt{layer}}$	$\mathcal{L}_{\texttt{pred}}$	$+\mathcal{L}_{\texttt{layer}}$	$\mathcal{L}_{\texttt{pred}}$	$+\mathcal{L}_{\texttt{layer}}$	$\mathcal{L}_{\texttt{pred}}$	$+\mathcal{L}_{\texttt{layer}}$
60%	92.66	93.00 (+0.34)	90.66	91.84 (+1.18)	85.16	85.31 (+0.15)	88.84	89.13 (+0.29)
70%	91.74	93.00 (+1.26)	89.93	91.29 (+1.36)	84.57	84.89 (+0.32)	88.11	88.56 (+0.45)
75%	91.40	92.89 (+1.49)	88.96	91.31 (+2.35)	84.19	84.75 (+0.56)	87.54	87.99 (+0.45)
80%	91.06	92.89 (+1.83)	88.76	90.43 (+0.67)	83.36	84.26 (+0.90)	86.52	87.26 (+0.74)
85%	90.48	92.55 (+2.07)	86.84	89.69 (+2.85)	82.69	83.44 (+0.75)	85.76	86.40 (+0.64)
90%	90.25	91.51 (+1.26)	85.80	88.89 (+3.19)	81.09	82.61 (+1.52)	83.28	84.08 (+0.80)
95%	91.06	90.37 (−0.69)	85.08	86.14 (+1.06)	79.66	80.55 (+0.89)	82.52	82.59 (+0.07)

Table 10: Ablation study on the proposed layer distillation objective across all sparsities.

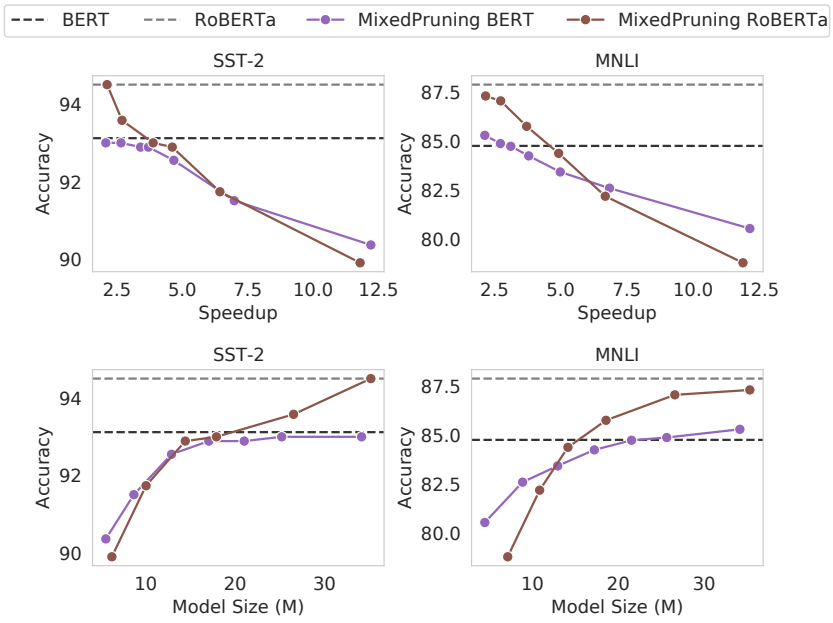

Figure 7: MixedPruning with BERT and RoBERTa on SST-2 and MNLI.