# Structured Pruning Learns Compact and Accurate Models

**Mengzhou Xia  Zexuan Zhong  Danqi Chen**
Princeton University
`{mengzhou,zzhong,danqic}@cs.princeton.edu`

## 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 propose a task-specific structured pruning method CoFi[1] (**Co**arse- and **Fi**ne-grained Pruning), which delivers highly parallelizable subnetworks and matches the distillation methods in both accuracy and latency, without resorting to any unlabeled data. Our key insight is to jointly prune coarse-grained (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. We also devise a layerwise distillation strategy to transfer knowledge from unpruned to pruned models during optimization. Our experiments on GLUE and SQuAD datasets show that CoFi yields models with over 10× speedups with a small accuracy drop, showing its effectiveness and efficiency compared to previous pruning and distillation approaches.[2]

## 1 Introduction

Pre-trained language models (Devlin et al., 2019; Liu et al., 2019a; Raffel et al., 2020, *inter alia*) have become the mainstay in natural language processing. These models have high costs in terms of storage, memory, and computation time and it has motivated a large body of work on model compression 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 (Table 1).

| | $U$ | $T$ | ↗ | Params | MNLI |
|---|---|---|---|---|---|
| BERT$_{\text{base}}$ (teacher) | ✗ | ✗ | 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 |
| CoFi Pruning (ours) | ✗ | ✓ | 2.7× | 26M | 84.9 |
| CoFi Pruning (ours) | ✗ | ✓ | 12.1× | 4.4M | 80.6 |

Table 1: A comparison of state-of-the-art distillation and pruning methods. $U$ and $T$ denote whether $U$nlabeled and $T$ask-specific are used for distillation or pruning. The inference speedups (↗) are reported against a BERT$_{\text{base}}$ model and we evaluate all the models on an NVIDIA V100 GPU (§4.1). The models labeled as ‡ use a different teacher model and are not a direct comparison. Models are one order of magnitude faster.[3]

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., 2020b) and blocks in weight matrices (Lagunas et al., 2021). The trend of structured pruning leans towards removing fine-grained units to allow for flexible final structures. However, thus far, pruned models rarely achieve large speedups (2-3× improvement at most).

By contrast, distillation methods usually first

[1]CoFi is pronounced as ☕.

[2]Our code and models are publicly available at `https://github.com/princeton-nlp/CoFiPruning`.

[3]Following previous work, we exclude embedding matrices in calculating the number of parameters. We exclude task-specific data augmentation for a fair comparison. More results with data augmentation can be found in Table 3.

specify a fixed model architecture and perform a *general distillation* step on an unlabeled corpus, before further fine-tuning or distillation on task-specific data (Sanh et al., 2019; Turc et al., 2019; Sun et al., 2019; Jiao et al., 2020). Well-designed student architectures achieve compelling speedup-performance tradeoffs, yet distillation to these randomly-initialized student networks on large unlabeled data is prohibitively slow.[4] For instance, TinyBERT (Jiao et al., 2020) is first trained on 2,500M tokens for 3 epochs, which requires training 3.5 days on 4 GPUs (Figure 1).[5]

In this work, we propose a task-specific, structured pruning approach called CoFi (**Co**arse and **Fi**ne-grained Pruning) and show that structured pruning can achieve highly compact subnetworks and obtain large speedups and competitive accuracy as distillation approaches, while requiring much less computation. Our key insight is to jointly prune coarse-grained units (e.g., self-attention or feed-forward layers) and fine-grained units (e.g., heads, hidden dimensions) simultaneously. Different from existing works, our approach controls the pruning decision of every single parameter by multiple masks of different granularity. This is the key to large compression, as it allows the greatest flexibility of pruned structures and eases the optimization compared to only pruning small units.

It is known that pruning with a distillation objective can substantially improve performance (Sanh et al., 2020; Lagunas et al., 2021). Unlike a fixed student architecture, pruned structures are unkown prior to training and it is challenging to distill between intermediate layers of the unpruned and pruned models (Jiao et al., 2020). Hence, we propose a layerwise distillation method, which dynamically learns the layer mapping between the two structures. We show that this strategy can better lead to performance gains beyond simple prediction-layer distillation.

Our experiments show that CoFi delivers more accurate models at all levels of speedups and model sizes on the GLUE (Wang et al., 2019) and SQuAD v1.1 (Rajpurkar et al., 2016) datasets, compared to strong pruning and distillation baselines. Concretely, it achieves over $10\times$ speedups and a 95% sparsity across all the datasets while preserving more than 90% of accuracy. Our results suggest that task-specific structured pruning is an appealing solution in practice, yielding smaller and faster models without requiring additional unlabeled data for general distillation.

## 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 $X$ and outputs:

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

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 $\mathrm{Att}(\cdot)$ 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 $W_D \in \mathbb{R}^{d_f \times d}$:

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

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 (embeddings excluded). According to Ganesh et al. (2021), both MHAs and FFNs take similar time on GPUs while FFNs become the bottleneck on CPUs.

### 2.2 Distillation

Knowledge distillation (Hinton et al., 2015) is a model compression approach that transfers knowledge from a larger teacher model to a smaller student model. *General distillation* (Sanh et al., 2019; Sun et al., 2020; Wang et al., 2020a) and *task-specific distillation* (Sun et al., 2019) exploit unlabeled data and task-specific data respectively for knowledge transfer. A combination of the two leads to increased performance (Jiao et al., 2020). General distillation or pre-training the student network on unlabeled corpus is essential for retaining performance while being computationally expensive (Turc et al., 2019; Jiao et al., 2020).

Different distillation objectives have been also explored. Besides standard distillation from the

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

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

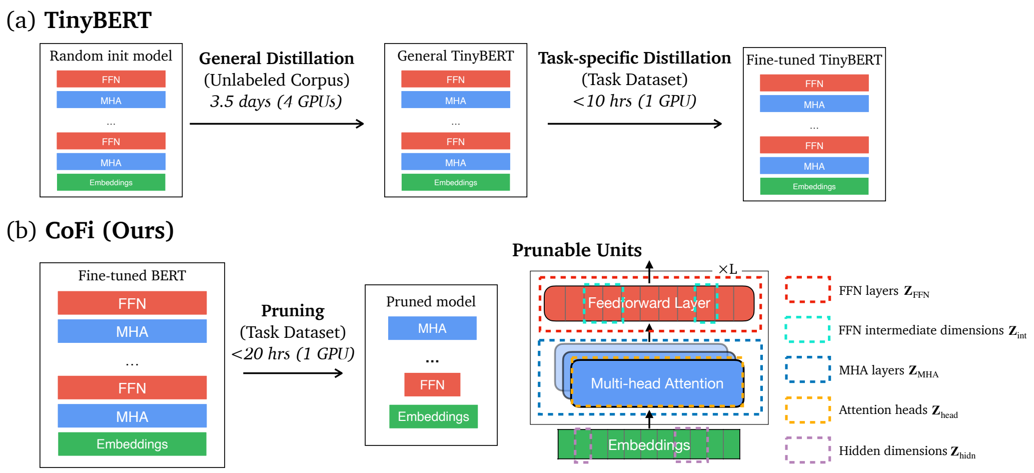

Figure 1: Comparison of (a) TinyBERT (Jiao et al., 2020) and (b) our pruning approach CoFi. TinyBERT trains a randomly-initialized network through two-step distillation: (1) general distillation on a large unlabeled corpus, which takes 3.5 days to finish on 4 GPUs, and (2) task-specific distillation on the task dataset. CoFi directly prunes the fine-tuned BERT model and jointly learn five types of mask variables (i.e., $\mathbf{z}_{\text{FFN}}, \mathbf{z}_{\text{int}}, \mathbf{z}_{\text{MHA}}, \mathbf{z}_{\text{head}}, \mathbf{z}_{\text{hidn}}$) to prune different types of units (Section 3.1). CoFi takes at most 20 hours to finish on 1 GPU on all the GLUE datasets (smaller datasets need $<$ 3 hour).[6]

prediction layer (Hinton et al., 2015), transferring knowledge layer-by-layer from representations (Jiao et al., 2020; Sun et al., 2020) and multi-head attention matrices (Wang et al., 2020a; Jiao et al., 2020; Sun et al., 2020) lead to significant improvements. Most distillation approaches assume a fixed student structure prior to training. Hou et al. (2020) attempt to distill to a dynamic structure with specified widths and heights. Yin et al. (2021) adopt a one-shot Neural Architecture Search solution to search architectures of student networks.

## 2.3 Pruning

Pruning gradually removes redundant parameters from a teacher model, mostly producing task-specific models. Previous works focus on pruning different components in Transformer models, from coarse-grained to fine-grained units.

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

**Head pruning** Voita et al. (2019); Michel et al. (2019) show that only a small 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 does not lead to large latency improvement—Li et al. (2021) demonstrate a 1.4$\times$ speedup with only one remaining head per layer.

**FFN pruning** The other major part—feed-forward layers (FFNs)—are also known to be overparameterized. Strategies to prune an FFN layer for an inference speedup include pruning an entire FFN layer (Prasanna et al., 2020; Chen et al., 2020b) 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 and unstructured 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 thus far: Yao et al. (2021) attempt 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. Similarly, *unstructured pruning* aims to remove individual weights and has been extensively studied in the literature (Chen et al., 2020a; Huang et al., 2021). Though the sparsity reaches up to 97%

[6]CoFi requires slightly longer training time compared to the task-specific distillation of TinyBERT, as CoFi searches model structures and learns parameters simultaneously.

(Sanh et al., 2020), it is hard to obtain inference speedups on the current hardware.

**Combination with distillation** Pruning is commonly combined with a prediction-layer distillation objective (Sanh et al., 2020; Lagunas et al., 2021). Yet it is not clear how to apply layerwise distillation strategies as the pruned student model's architecture evolves during training.

## 3 Method

We propose a structured pruning approach CoFi, which jointly prunes **Co**arse-grained and **Fi**ne-grained units (§3.1) with a layerwise distillation objective transferring knowledge from unpruned to pruned models (§3.2). A combination of the two leads to highly compressed models with large inference speedups.

### 3.1 Coarse- and Fine-Grained Pruning

Recent trends in structured pruning move towards pruning smaller units for model flexibility. Pruning fine-grained units naturally entails pruning coarse-grained units—for example, pruning $N_h$ (e.g., 12) heads is equivalent to pruning one entire MHA layer. However, we observe that this rarely happens in practice and poses difficulty to optimization especially at a high sparsity regime.

To remedy the problem, we present a simple solution: we allow pruning MHA and FFN layers explicitly along with fine-grained units (as shown in §2.3) by introducing two additional masks $z_{\text{MHA}}$ and $z_{\text{FFN}}$ for each layer. Now the multi-head self-attention and feed-forward layer become:

$$\begin{aligned}\text{MHA}(X) = z_{\text{MHA}} \cdot \sum_{i=1}^{N_h} (\mathbf{z}_{\text{head}}^{(i)} \cdot \\ \text{Att}(W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}, W_O^{(i)}, X), \\ \text{FFN}(X) = z_{\text{FFN}} \cdot \text{gelu}(XW_U) \cdot \text{diag}(\mathbf{z}_{\text{int}}) \cdot W_D.\end{aligned}$$

With these layer masks, we explicitly 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), we 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 layers because each dimension in a hidden representation is connected to the same dimension 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).

CoFi differs from 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 is pruned when the entire FFN layer, or its corresponding intermediate dimension, or the hidden dimension is pruned. As a comparison, a recent work Block Pruning (Lagunas et al., 2021) adopts a hybrid approach which applies a pruning pruning strategy on MHAs and FFNs separately.

To learn these mask variables, we use $l_0$ regularization modeled with hard concrete distributions following Louizos et al. (2018). We also follow Wang et al. (2020b) to replace the vanilla $l_0$ objective with a Lagrangian multiplier to better control the desired sparsity of pruned models.[7] We adapt the sparsity function accordingly to accommodate pruning masks of different granularity:

$$\begin{aligned}\hat{s} = \tfrac{1}{M} \cdot 4 \cdot d_h \cdot \textstyle\sum_i^L \sum_j^{N_h} \sum_k^d \mathbf{z}_{\text{MHA}}^{(i)} \cdot \mathbf{z}_{\text{head}}^{(i,j)} \cdot \mathbf{z}_{\text{hidden}}^{(k)} \\ + \tfrac{1}{M} \cdot 2 \cdot \textstyle\sum_i^L \sum_j^{d_f} \sum_k^d \mathbf{z}_{\text{FFN}}^{(i)} \cdot \mathbf{z}_{\text{int}}^{(i,j)} \cdot \mathbf{z}_{\text{hidden}}^{(k)},\end{aligned}$$

where $\hat{s}$ is the expected sparsity and M denotes the full model size. All masking variables are learned as real numbers in $[0,1]$ during training and we map the masking variables below a threshold to 0 during inference and get a final pruned structure where the threshold is determined by the expected sparsity of each weight matrix (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 and the teacher's output probability distributions $\mathbf{p}_s$ and $\mathbf{p}_t$ (Sanh et al., 2020; Lagunas et al., 2021):

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

[7]We also tried a straight-through estimator as proposed in Sanh et al. (2020) and found the performance comparable. We choose $l_0$ regularization because it is easier to control the sparsity precisely.

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

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

where $W_{\texttt{layer}} \in \mathbb{R}^{d \times d}$ is a linear transformation matrix, 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:

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

Calculating the distance between two sets of layers is highly parallelizable and introduces a minimal training overhead. To address the issue of layer mismatch, which mostly happens for small-sized datasets, e.g., RTE, MRPC, we add a constraint to only allow matching a teacher layer to a lower student layer than the previously matched student layer. When pruning with larger sized datasets, layer mismatch rarely happens, showing the superiority of dynamic matching—layers between student and teacher models match in a way that benefits the pruning process the most.

Finally, we combine layer distillation with the prediction-layer distillation:

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

where $\lambda$ controls the contribution of each loss.

## 4 Experiments

### 4.1 Setup

**Datasets** We evaluate our approach on eight GLUE tasks (Wang et al., 2019) and SQuAD v1.1 (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 (see Appendix D for dataset sizes and metrics).

**Training setup** In our experiments, *sparsity* is computed as the number of pruned parameters divided by the full model size (embeddings excluded). Following Wang et al. (2020b); Lagunas et al. (2021), we first finetune the model with the distillation objective, 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 (see Appendix A for more training details). 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 BERT$_{\text{base}}$ model[8] and freeze embedding weights following Sanh et al. (2020). We report results on development sets of all datasets.

**Baselines** We compare CoFi 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., 2020b), 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 F.[9]

For TinyBERT and DynaBERT, the released models are trained with task-specific augmented data. For a fair comparison, we train these two models with the released code without data augmentation.[10] For Block Pruning, we train models

[8]We also experiments CoFi on RoBERTa models (Liu et al., 2019a). Please refer to Appendix I for details.

[9]We show these results in Appendix F as they are not directly comparable to CoFi.

[10]For TinyBERT, the augmented data is $20\times$ larger than the original data, making the training process significantly slower.

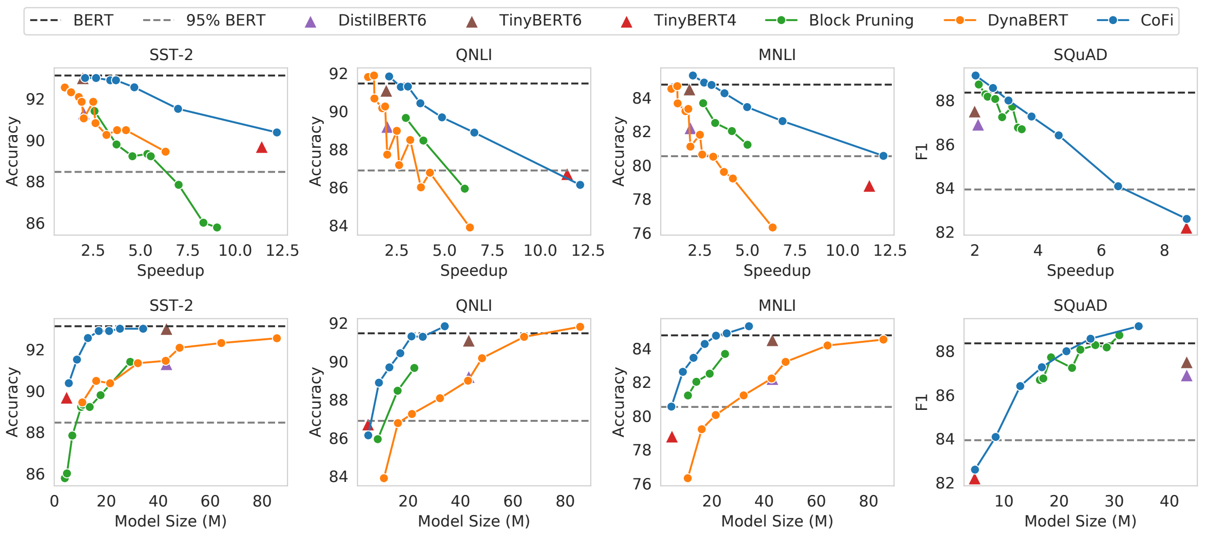

Figure 2: Accuracy v.s. speedup (top) or model size (bottom). We compare CoFi against state-of-the-art distillation and pruning baselines. Note that we exclude embedding size when calculating model size following previous work, as forwarding through the embedding layer has little effect on inference 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 |
|---|---|---|---|---|---|---|---|---|---|---|
| BERT$_{base}$ (teacher) | 93.1 | 91.5 | 84.8 | 91.2 | 61.2 | 70.0 | 88.7 | 85.0 | 88.4 | - |
| TinyBERT$_4$ w/o GD | 87.7 | 81.8 | 78.7 | 89.5 | 16.6 | 47.3 | 17.8 | 68.9 | - | $\leq 10$ |
| TinyBERT$_4$ | 89.7 | **86.7** | 78.8 | 90.0 | 32.5 | 63.2 | **85.0** | 81.4 | 82.1 | $\sim 350$ |
| Speedup $\nearrow$ | 11.4× | 11.4× | 11.4× | 11.4× | 11.4× | 11.4× | 11.4× | 11.4× | 8.7× | - |
| CoFi Pruning (ours) | **90.6** | 86.1 | **80.6** | **90.1** | **35.6** | **64.7** | 83.1 | **82.6** | **82.6** | $\leq 20$ |
| Speedup $\nearrow$ | 12.0× | 12.1× | 12.1× | 11.0× | 11.5× | 11.9× | 12.9× | 11.9× | 8.7× | - |

Table 2: CoFi 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). CoFi 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 3 for experiments with augmented data.

| Task | TinyBERT$_4$ | CoFi (ours) |
|---|---|---|
| SST-2 | 89.7 → 91.6 | 90.6 → **92.4** |
| QNLI | 86.7 → **87.6** | 86.1 → 86.8 |
| RTE | 63.2 → 62.5 | 64.7 → **67.5** |
| MRPC | 81.4 → 83.6 | 82.6 → **84.6** |

Table 3: CoFi v.s. TinyBERT$_4$ trained with task-specific data augmentation introduced in Jiao et al. (2020). All models have around 5M parameters (95% sparsity) and achieve similar speedups (11-12×). The numbers before → are without data augmentation.

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 the compression rate does not necessarily reflect the actual improvement in inference latency [11]. We use an unpruned BERT$_{base}$ as the baseline and evaluate all the models with the same hardware setup on a single NVIDIA V100 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 CoFi models to other methods in terms of both inference speedup and model size. CoFi delivers more accurate models than distillation and pruning baselines at every speedup level

[11]Models with the same compression rate could have considerably different speedups.

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, CoFi 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 CoFi produces models with over $10\times$ 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, CoFi 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 CoFi—are more economical and efficient in achieving compressed models.

We further compare CoFi with TinyBERT$_4$ under the data augmentation setting in Table 3. As the augmented dataset is not publicly released, we follow its GitHub repository to create our own augmented data. We train CoFi with the same set of augmented data and find that it still outperforms TinyBERT$_4$ on most datasets.[12]

## 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 CoFi affect model performance and inference speedup beyond the standard practice of pruning heads and FFN dimensions. We show results in Table 4 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 CoFi 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 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 performance of CoFi in Table 5. We first observe that removing distillation entirely leads to a performance drop up to 1.9-6.8 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. We also experiment with a simple alternative i.e., "fixed Hidden Distillation", which matches each layers from the teacher model to the corresponding layer in the student model – if a layer is already pruned, the distillation objective will *not* be added. We find that fixed hidden distillation underperforms the dynamic layer matching objective used for CoFi. 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.[13] Moreover, as shown in the table, removing it hurts the performance for all the datasets except SST-2.

## 4.4 Structures of Pruned Models

Finally, we study the pruned structures produced by CoFi. We characterize the pruned models of sparsities $\{60\%, 70\%, 80\%, 90\%, 95\%\}$ on five datasets. For each setting, we run CoFi three times. Figure 3 demonstrates the number of remaining heads and intermediate dimensions of the pruned models for different sparsities.[14] Interestingly, we discover common structural patterns in the pruned models: (1) Feed-forward layers are significantly pruned

[12]We only conduct experiments with data augmentation on four datasets because training on augmented data is very expensive. For example, training on the augmented dataset for MNLI takes more than 200 GPU hours in total. See more details in Appendix E.

[13]Please refer to subsection G.1 for more details.

[14]We show more layer analysis in Appendix H.

| | QNLI (60%) | | QNLI (95%) | | MNLI (60%) | | MNLI (95%) | | SQuAD (60%) | | SQuAD (95%) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\nearrow$ | acc | $\nearrow$ | acc | $\nearrow$ | acc | $\nearrow$ | acc | $\nearrow$ | F1 | $\nearrow$ | F1 |
| CoFi | 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 |
| CoFi | 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 |

Table 4: Ablation studies on pruning units on QNLI, MNLI and SQuAD. $\nearrow$: 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 $z_{\text{MHA}}$ or $z_{\text{FFN}}$), 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 | SQuAD |
|---|---|---|---|---|
| CoFi | 90.6 | **86.1** | **80.6** | **82.6** |
| $-\mathcal{L}_{\texttt{layer}}$ | **91.1** | 85.1 | 79.7 | 82.5 |
| $-\mathcal{L}_{\texttt{pred}}, \mathcal{L}_{\texttt{layer}}$ | 86.6 | 84.2 | 78.2 | 75.8 |
| Fixed Hidn Distil. | 90.0 | 85.8 | 80.5 | 80.9 |

Table 5: 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 G.2, we show that the dynamic layer distillation objective improves model performance more significantly on lower sparsity rates.

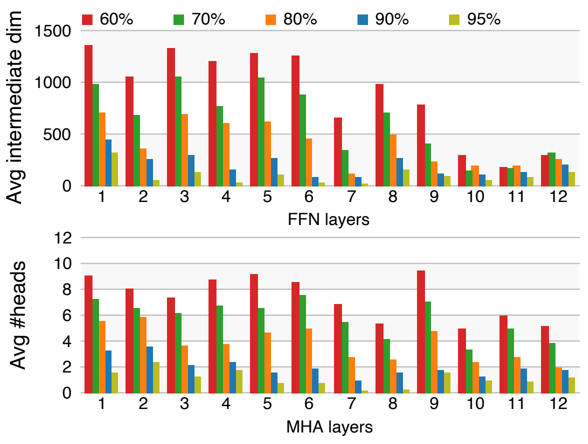

Figure 3: The average intermediate dimensions at each FFN layer and the average number of heads at each MHA layer in the pruned models across five datasets (SST-2, MNLI, QQP, QNLI, and SQuAD 1.1). We study different sparsities $\{60\%, 70\%, 80\%, 90\%, 95\%\}$.

across all sparsities. For example, at the 60% sparsity level, 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) CoFi 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 6 for highly compressed models (sparsity = 95%). Although all the models are roughly of the same size, they present different patterns across datasets, which suggests that there exist different optimal sub-networks for each dataset. 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 H), which suggests that MHA layers are more important for solving downstream tasks. 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 has been 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; Guo et al., 2020) is a standard approach for convolution neural networks. The techniques can be adapted to Transformer-based models as introduced in subsection 2.3. 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

| Dataset | Pruned Models | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SST-2 | M | F | M | | M | | | | M | | | | | | | F | M | F | | F | M | F | M | F |
| | M | F | | | | | | | M | | | | | | | F | M | F | M | | M | F | M | F |
| | M | | | | | | | | | | | F | M | | | F | M | F | M | | M | | M | |
| QNLI | M | F | M | | M | F | M | | | | | | | F | M | | | F | M | | | | | |
| | M | F | M | | M | F | M | | | | | | | | | | M | | | | | | M | |
| | M | F | M | | | | M | | | | | | | | | F | | | M | | | F | M | F |
| MNLI | | F | M | | | | | F | M | | | | | | | | | | | | M | F | | F |
| | M | F | M | F | M | | | | | | | | | | | | M | F | M | | M | | | |
| | M | F | M | | | F | M | | | F | M | | | | | | | | M | | | | M | |
| QQP | | F | M | | M | | M | | | F | M | | | | | | | | | F | | F | M | F |
| | | F | M | | | F | M | | | F | M | | | | | F | M | | | | | | | F |
| | | F | M | | M | F | M | | M | F | M | | | | | F | M | | | | | | M | |
| SQuAD | | F | M | F | M | F | M | | | | M | | | | | F | M | | | | | | | F |
| | | F | M | | M | F | M | | | | | | | | | F | M | | | F | M | | | F |
| | | F | M | F | M | | M | | | F | | | M | F | M | | | | | | | | | F |

Table 6: Remaining layers in the models pruned by CoFi 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.

et al., 2020; Chen et al., 2020a). Unstructured pruning produces models with high sparsities (Sanh et al., 2020; Xu et al., 2021; Huang et al., 2021) yet hardly bring actual inference speedups. Developing computing platform for efficient sparse tensor operations is an active research area. DeepSparse[15] is CPU inference engine that leverages unstructured sparsity for speedup. Huang et al. (2021) measure the real inference speedup induced by unstructured pruning on Moffett AI's latest hardware platform ANTOM. We do not directly compare to these methods because the evaluation environments are different.

While all the aforementioned methods produce task-specific models through pruning, several works explore upstream pruning where they prune a large pre-trained model with the masked language modeling task. Chen et al. (2020a) show a 70%-sparsity model retains the MLM accuracy produced by iterative magnitude pruning. Zafrir et al. (2021) show the potential advantage of upstream unstructured pruning against downstream pruning. We consider applying CoFi for upstream pruning as a promising future direction to produce task-agnostic models with flexible structures.

Besides pruning, many other techniques have been explored to gain inference speedups for Transformer models, including distillation as introduced in subsection 2.2, quantization (Shen et al., 2020; Fan et al., 2021), dynamic inference acceleration (Xin et al., 2020) and matrix decomposition (Noach and Goldberg, 2020). We refer the readers to Ganesh et al. (2021) for a comprehensive survey.

[15]https://github.com/neuralmagic/deepsparse

## 6 Conclusion

We propose CoFi, a structured pruning approach that incorporates all levels of pruning, including MHA/FFN layers, individual heads, and hidden dimensions for Transformer-based models. Coupled with a distillation objective tailored to structured pruning, we show that CoFi 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. Though CoFi can be directly applied to structured pruning for task-agnostic models, we frame the scope of this work to task-specific pruning due to the complexity of the design choices for upstream pruning. We hope that future research continues this line of work, given that pruning from a large pre-trained model could possibly incur less computation compared to general distillation and leads to more flexible model structures.

## Acknowledgements

The authors thank Tao Lei from Google Research, Ameet Deshpande, Dan Friedman, Sadhika Malladi from Princeton University and the anonymous reviewers for their valuable feedback on our paper. This research is supported by a Hisashi and Masae Kobayashi *67 Fellowship and a Google Research Scholar Award.

## References

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael

Carbin. 2020a. The lottery ticket hypothesis for pre-trained BERT networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15834–15846.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. 2020b. Early-bert: 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 *International Conference on Machine Learning (ICML)*, pages 3259–3269.

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.

Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. 2020. Dmcp: Differentiable markov channel pruning for neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33.

Shaoyi Huang, Dongkuan Xu, Ian EH Yen, Sung-en Chang, Bingbing Li, Shiyang Chen, Mimi Xie, Hang Liu, and Caiwen Ding. 2021. Sparse progressive distillation: Resolving overfitting under pretrain-and-finetune paradigm. *arXiv e-prints*, pages arXiv–2110.

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*.

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 *International Conference on Computer Vision (ICCV)*, 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 *International Conference on Computer Vision (ICCV)*, 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 *International Conference on Computer Vision (ICCV)*, 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.

Matan Ben Noach and Yoav Goldberg. 2020. Compressing pre-trained language models by matrix decomposition. In *Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 884–889.

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)*.

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.

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)*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020a. 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. 2020b. 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. In *Transactions of the Association of Computational Linguistics (TACL)*, volume 7, pages 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.

Dongkuan Xu, Ian En-Hsu Yen, Jinxi Zhao, and Zhibin Xiao. 2021. Rethinking network pruning–under the pre-train and fine-tune paradigm. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 2376–2382.

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.

Ofir Zafrir, Ariel Larey, Guy Boudoukh, Haihao Shen, and Moshe Wasserblat. 2021. Prune once for all: Sparse pre-trained language models. *arXiv preprint arXiv:2111.05754*.

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 7.

| Hyperparameter | |
|---|---|
| $\lambda$ | $0.1, 0.3, 0.5$ |
| 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 7: Hyperparemeters in the experiments.

For four 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. (2020b), 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 four 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 epochs and prune till the target sparsity within the 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 $\lambda$, batch size, and learning rate do not generally affect performance much.

## B Optimization Details

Louizos et al. (2018) propose $l_0$ 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 $(l, r)$. $\beta$ 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. (2020b) to add a Lagrangian term, which imposes an equality constraint $\hat{s} = t$ by introducing a violation penalty:

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

where $\hat{s}$ is the expected model sparsity calculated from $\mathbf{z}$ and $t$ is the target sparsity.

## 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 8.

| 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 8: Data statistics of GLUE and SQuAD datasets.

## E TinyBERT$_4$ w/ Data Augmentation

We conduct task-specific distillation with the script provided by the TinyBERT repository.[16] However, our reproduced results are slightly lower than the reported results in (Jiao et al., 2020). The difference between these two sets of scores may stem from augmented data or teacher models. Note that the authors of TinyBERT did not release the augmented dataset. We run their codes to obtain augmented datasets. We compare CoFi and TinyBERT under the same setting where we use the same teacher model and the same set of augmented data.

| | SST-2 | QNLI | RTE | MRPC |
|---|---|---|---|---|
| TinyBERT$_4$ reimpl. | 91.6 | 87.6 | 62.5 | 83.6 |
| Jiao et al. (2020) | 92.7 | 88.0 | 65.7 | 85.7 |

Table 9: Re-implemented (TinyBERT$_4$ reimpl.) results and the results reported in Jiao et al. (2020).

## F Additional Comparisons

### F.1 Comparison to Movement Pruning

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

### F.2 Comparison to Block Pruning

In Figure 6, we compare CoFi with Block Pruning while unifying the distillation objective. Without the layer distillation objective, CoFi 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 CoFi (SST-2), backing up our argument that pruning layers for high sparsity models is the key to high speedups.

[16]https://github.com/huawei-noah/Pretrained-Language-Model/tree/master/TinyBERT

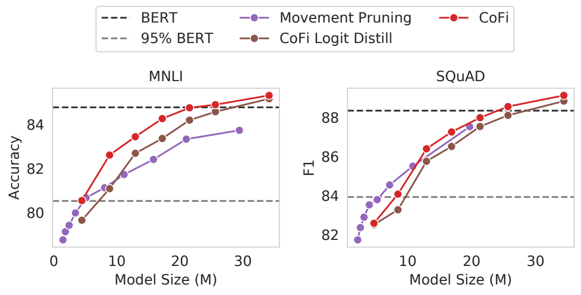

Figure 4: CoFi v.s. Movement Pruning (unstructured pruning). CoFi Logit Distill denotes that we run CoFi with prediction-layer distillation only as Movement Pruning.

### F.3 More Baselines

We show additional pruning and distillation methods that are not directly comparable to CoFi in Table 10. CoFi still largely outperforms these baselines even though these methods hold an inherent advantage due to a stronger teacher or base model.

| | $\nearrow$ | SST-2 | QNLI | MNLI | SQuAD |
|---|---|---|---|---|---|
| Wang et al. (2020b)$^\ddagger$ | 1.5× | 92.1 | 89.1 | - | 85.4 |
| Sajjad et al. (2020) | 2.0× | 90.3 | - | 81.1 | - |
| Fan et al. (2020)$^\ddagger$ | 2.0× | 93.2 | 89.5 | 84.1 | - |
| Sun et al. (2020)$^\diamond$ | 2.3× | 92.1 | 91.0 | 83.9 | 90.3 |
| Yin et al. (2021)$^\spadesuit$ | 4.3× | 91.4 | 89.7 | 82.3 | 87.6 |
| CoFi (ours) | 2.0× | 93.0 | 91.8 | 85.3 | 89.1 |
| CoFi (ours) | 4.6× | 92.6 | 89.7 | 83.4 | 86.4 |

Table 10: More pruning and distillation baselines. $\nearrow$: speedup. $\ddagger$ denotes that the model prunes from a RoBERTa$_{\text{base}}$ model. $\spadesuit$: AutoTinyBERT is distilled from an ELECTRA$_{\text{base}}$ model. $\diamond$: MobileBERT (Sun et al., 2020) has specialized architecture designs and trains their own teacher model from the scratch. CoFi models have a model size of 34M and 13M respectively, corresponding to a $60\%$ and $85\%$ sparsity.

## G More Analyses on Layer Distillation

### G.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, 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 can find task-specific alignment and improve performance.

### G.2 Ablation on Distillation Objectives

In Table 11, 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 sparsity rates and different tasks.

## H FFN/MHA Layers in Pruned Models

Figure 5 shows the average number of FFN layers and MHA layers in the pruned models by CoFi. 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 indicate that MHA layers are more important for solving these downstream tasks than FFN layers.

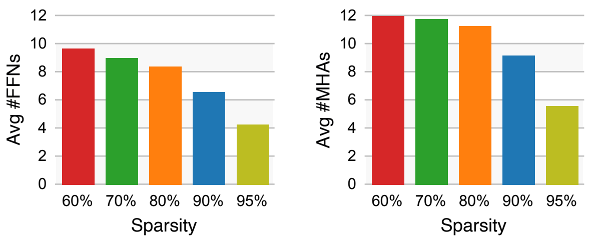

Figure 5: The average number of FFN layers and MHA layers in the pruned models at different sparsities.

## I RoBERTa Pruning

We show CoFi results with RoBERTa in Figure 7 across sparsities from 60% to 95%. Similar to BERT, models with 60% weights pruned are able to maintain the performance 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 epoch 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.

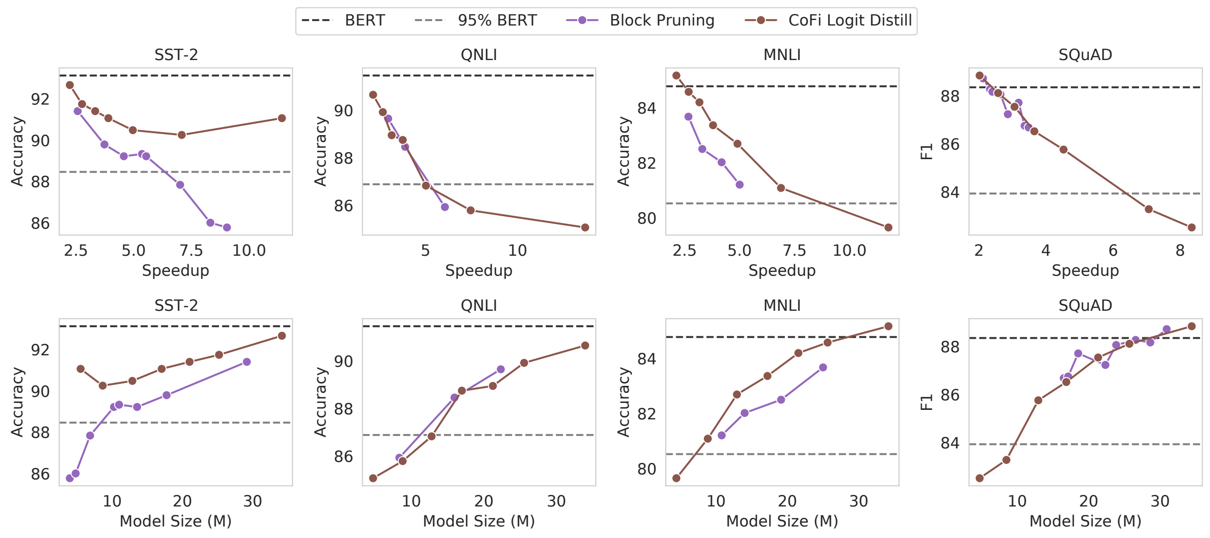

Figure 6: CoFi v.s. Block Pruning with the same distillation objective – prediction-layer distillation (Logit Distill). CoFi still outperforms or is on par with Block Pruning.

| | 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 11: Ablation study on the proposed layer distillation objective across all sparsities.

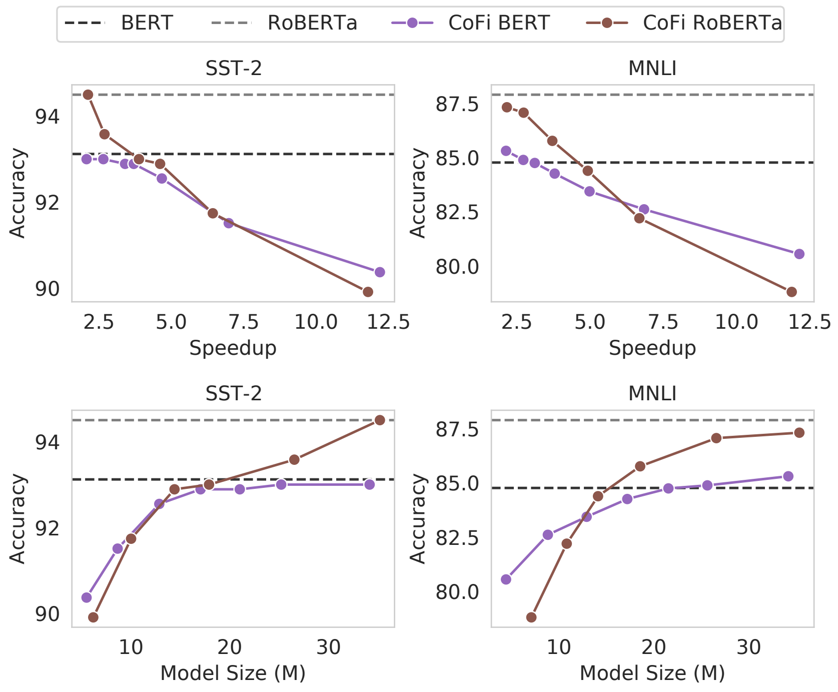

Figure 7: CoFi with BERT and RoBERTa on SST-2 and MNLI.