SP^3: Enhancing Structured Pruning via PCA Projection

Anonymous ACL submission

Abstract

Structured pruning is a widely used technique for reducing the size of pre-trained language models (PLMs), but current methods often overlook the potential of compressing the hidden dimension (d) in PLMs, a dimension critical to model size and efficiency. This paper introduces a novel structured pruning approach, Structured Pruning with PCA Projection (SP^3), targeting the effective reduction of d by projecting features into a space defined by principal components before masking. Extensive experiments on benchmarks (GLUE and SQuAD) show that SP^3 can reduce d by 70%, compress 94% of the $\text{BERT}_{\text{base}}$ model, and maintain over 96% accuracy and outperform other methods that compress d by 6% in accuracy at the same compression ratio. SP^3 has also proven effective with other models, including OPT and LLaMa. Our data and code is available on an anonymous repo.

1 Introduction

Pre-trained language models (PLMs) such as BERT (Devlin et al., 2018), GPT-2 (Radford et al.), ChatGPT (Ouyang et al., 2022), and LLaMA (Touvron et al., 2023), which primarily utilize transformer structures, have become increasingly prominent in Natural Language Processing due to their outstanding performance. However, the significant storage requirements and lengthy inference times of these models pose practical challenges, necessitating efforts to optimize their size and computational speed.

Among various model compression techniques, distillation (Sanh et al., 2019) and, pruning (Louizos et al., 2018), particularly structural pruning (Xia et al., 2022), are prominent. Distillation is adaptable, allowing for the design of compact student models while transferring knowledge from larger teacher models. Nevertheless, it might not fully leverage the teacher model's parameters, often requiring the student model to undergo extensive training on large datasets. In contrast, structured pruning directly trims the original model, thereby utilizing its parameters more efficiently and lessening the dependency on large datasets.

The transformer architecture in PLMs is defined by weight matrices controlled by hidden dimension d, attention head size d_h in multi-head attention layers, and filter dimension d_f in feed-forward network layers. Focused efforts in structured pruning have aimed at downsizing d_h and d_f (Michel et al., 2019; Kwon et al., 2022; Voita et al., 2019; Wang et al., 2020). Yet, strategies for pruning d are underexplored, with limited compression success. For instance, CoFi (Xia et al., 2022) achieves a notable 95% sparsity and maintains over 90% accuracy but only marginally reduces d from 768 to 760. We posit that the hidden dimension d holds significant potential for compression. Recent research demonstrates that high-dimensional features produced by models can be projected into a subspace, leveraging their inherent low-rank properties (Chen et al., 2021). Our observations further reveal that the sizes of these subspaces can differ among various layers. This variability underscores the necessity of compressing d within each individual layer, as the distribution of principal components' weights changes across layers, a phenomenon depicted in Fig. 1 (a). Unfortunately, presently, distillation remains the most effective means for compressing d. This paper aims to explore a potent structured pruning method for efficiently reducing d in PLMs.

We analyze and conjecture that existing pruning methods rarely compress d effectively due to two main reasons. First, conventional mask-based pruning methods directly trim dimensions in the original feature space, which is not conducive to removing unimportant dimensions. We interpret the pruning of d from the perspective of features' principal components. Drawing inspiration from

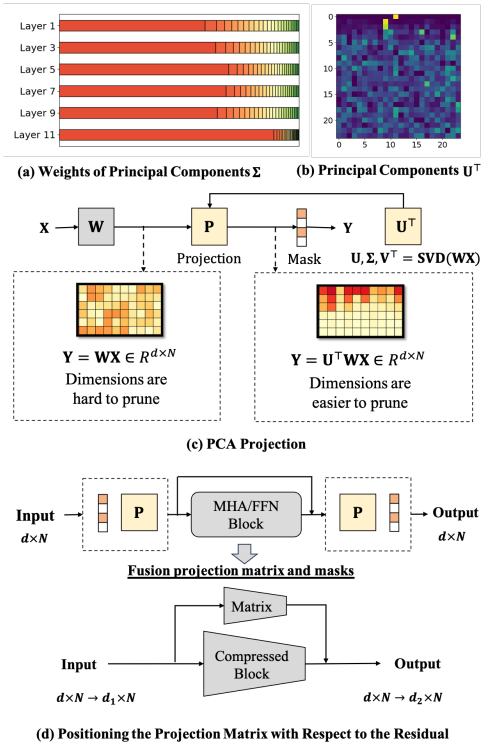


Figure 1: (a) Depiction of principal component weights within Σ. The length of each rectangle responds to the percentage of that principal component's weight among all principal components; (b) Illustration of principal component weights in U per layer. Each row represents a principal component, each column represents a dimension, and the colors reflect the correlation between the principal component and the dimension; (c) Projection of features onto principal components for easier dimension pruning; (d) Positioning the projection matrix outside the residual. In the figure, d, d_1, d_2 denote the dimensions of the features and N denotes the number of tokens.

PCA theory (Maćkiewicz and Ratajczak, 1993), effective compression should preserve the principal components of the features. Considering that most dimensions in the original feature space are linked to these principal components, as indicated by the evenly distributed values across each row representing a principal component in Fig. 1(b), traditional mask-based pruning in this space struggles to preserve the principal components' integrity. Secondly, the residual structure in each block, including both the multi-head attention (MHA) and the feed-forward network (FFN), requires the input and output dimensions of each block to be identical, hindering the pruning of different d across various layers.

To overcome these limitations, we propose an enhanced Structured Pruning approach using PCA Projection (SP^3) to reduce the hidden dimension. This method involves projecting the original features into a new space, where principal components act as the basis vectors. In this way, mask-based structured pruning becomes more effective in eliminating less important dimensions while retaining the principal components. For the projection, we select several tokens from the dataset designated for pruning and process them through the transformer model to acquire their feature matrix. Subsequently, we apply SVD to this matrix to obtain the projection matrix, which is then integrated before the mask matrix in the conventional pruning methodology. This process enables the projection of the feature matrix into a space defined by the principal components (as depicted in Fig.1 (c)). Additionally, to enable different d in different layers, we introduce additional linear transformations in the residuals, resulting in layer-adapted structured pruning (as shown in Fig.1 (d)).

The primary contributions of this work are:

- We take a fresh perspective on structured pruning by reducing the hidden dimension of the transformer model, which has been rarely explored before.
- Our SP^3 method enhances compression efficacy by projecting features into space formed by principal components before masking, which helps preserve the principal components of features. Furthermore, it integrates linear transformations in residual connections, enabling variable hidden dimensions across different layers for a more customized structured pruning approach.
- Extensive testing on the GLUE and SQuAD benchmarks reveals that SP^3 can effectively reduce the hidden dimension by 70%, compressing 94% of the $\text{BERT}_{\text{base}}$ model while still maintaining an accuracy of over 96%. When compared to other methods that also compress the hidden dimension (d), SP^3 demonstrates a 6% improvement in accuracy at the same compression ratio.

2 Background

2.1 Transformer Architecture

A typical transformer comprises a stack of transformer layers, each containing a multi-head attention (MHA) block and a feed-forward network (FFN) block. We take BERT (Devlin et al., 2018) as an example to describe this architecture.

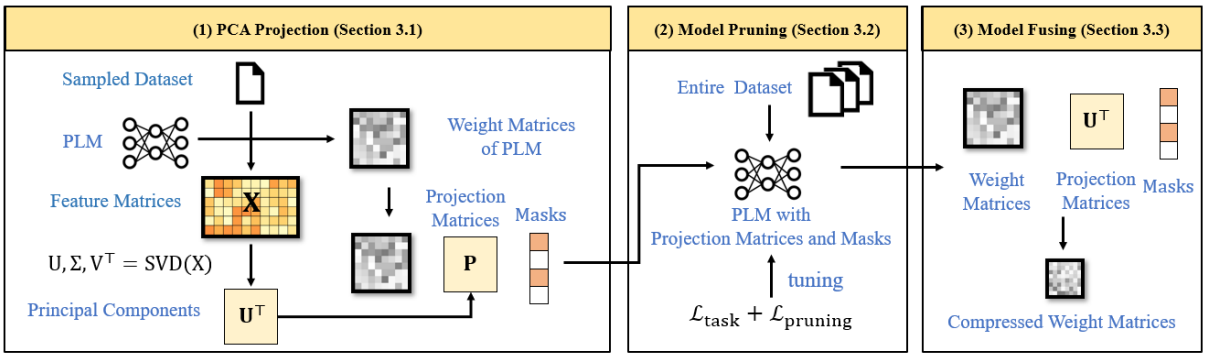


Figure 2: Illustration of the workflow of SP^3 .

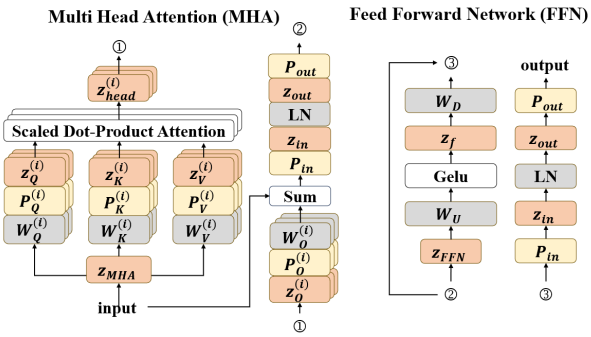


Figure 3: Illustration of the SP^3 architecture, in which the gray rectangles represent the weight matrices, the yellow rectangles signify the projection matrices, and the red rectangles indicate the masks.

Multi Head Attention (MHA). The MHA block takes $x \in \mathbb{R}^{d \times N}$ as input and consists of H attention heads that facilitate interactions between tokens, with a layer normalization (LN) (Ba et al., 2016) step.

$$
\begin{aligned}
x_{\mathrm{M}} &= \mathrm{LN}(x + \mathrm{MHA}(x)), \\
\mathrm{MHA}(x) &= \sum_{i=1}^{H} \mathrm{Att}^{(i)}(x), \\
\mathrm{Att}^{(i)}(x) &= W_O^{(i)\top} W_V^{(i)} x \cdot \mathrm{Softmax}\left(\frac{(W_K^{(i)} x)^\top (W_Q^{(i)} x)}{\sqrt{d_h}}\right),
\end{aligned} \tag{1}
$$

where $W_Q^{(i)}, W_K^{(i)} \in \mathbb{R}^{d_{\mathrm{QK}} \times d}, W_V^{(i)}, W_O^{(i)} \in \mathbb{R}^{d_{\mathrm{VO}} \times d}$ are the parameters of an MHA block, denoting the query, key, value, and output matrices, respectively. Here d denotes the hidden dimension, $d_h = d/H$ denotes the head size, N denotes the sequence length, and d_{QK} and d_{VO} denote the intermediate dimensions of the MHA block.

Feed Forward Network (FFN). The FFN block takes x_M as input and generates x_F as output:

$$
\begin{aligned}
x_{\mathrm{F}} &= \mathrm{LN}(x_{\mathrm{M}} + \mathrm{FFN}(x_{\mathrm{M}})), \\
\mathrm{FFN}(x_{\mathrm{M}}) &= W_{\mathrm{D}}^\top \mathrm{gelu}(W_{\mathrm{U}} x_{\mathrm{M}}),
\end{aligned} \tag{2}
$$

where gelu is the activation function, and $W_U, W_D \in \mathbb{R}^{d_f \times d}$ are two weight matrices of the FFN block. Here, d_f indicates the intermediate dimension of the FFN block.

2.2 Mask-based Structured Pruning

Mask-based structured pruning approaches eliminate parameters by integrating additional mask variables z into the original model's parameters, as follows:

$$
\begin{aligned}
x_{\mathrm{M}} &= \mathrm{diag}(z_{\mathrm{out}})\mathrm{LN}(\mathrm{diag}(z_{\mathrm{in}})(x + z_{\mathrm{MHA}}\mathrm{MHA}(x))), \\
\mathrm{MHA}(x) &= \sum_{i=1}^{H} z_{\mathrm{head}}^{(i)} \mathrm{Att}^{(i)}(x), \\
x_{\mathrm{F}} &= \mathrm{diag}(z_{\mathrm{out}})\mathrm{LN}(\mathrm{diag}(z_{\mathrm{in}})(x_{\mathrm{M}} + z_{\mathrm{FFN}}\mathrm{FFN}(x_{\mathrm{M}}))), \\
\mathrm{FFN}(x_{\mathrm{M}}) &= W_{\mathrm{D}}^\top \mathrm{diag}(z_{\mathrm{f}})\mathrm{gelu}(W_{\mathrm{U}} x_{\mathrm{M}}),
\end{aligned} \tag{3}
$$

where $z_{\mathrm{MHA}}, z_{\mathrm{FFN}} \in \{0,1\}, z_{\mathrm{head}} \in \{0,1\}^H, z_{\mathrm{f}} \in \{0,1\}^{d_f}, z_{\mathrm{in}}, z_{\mathrm{out}} \in \{0,1\}^d$ are used to remove entire MHA block, FFN block, attention head, intermediate dimensions of FFN block, and hidden dimensions. Then, the pruning process can be modeled as an optimization problem with sparsity constraints as follows,

$$
\begin{aligned}
&\min \mathcal{L}(\theta) \\
&\text{s.t.} \quad \mathrm{sparsity}(z) \geq t,
\end{aligned} \tag{4}
$$

where θ denotes the parameters of the model, and t denotes the pre-defined target sparsity.

3 SP^3

Workflow of SP^3. The SP^3 we propose unfolds in three phases, as depicted in Fig. 2. Initially, PCA projection is conducted by selecting a subset of the training data, processing it through the transformer model to yield the feature matrix X for each block per layer, and then determining each X's principal

components U via SVD decomposition. Following this, a projection matrix, initialized with these principal components, is integrated into the model's computational framework, coupled with a mask for structured pruning, detailed in Section 3.1. Subsequently, the model undergoes pruning based on the masks tailored to a specific task and pruning objective, as outlined in Section 3.2. The final stage involves merging these masks and projection matrices with the original model parameters, thereby creating the compressed model, which is further elaborated in Section 3.3.

3.1 PCA Projection

Current structured pruning methods typically compress PLMs by directly applying a mask to their hidden dimensions. However, this approach overlooks the relationship between each feature dimension and its corresponding principal component. Principal component analysis of these features reveals that a few principal components hold the majority of the weight in every layer (as indicated in Fig. 1(a)). Additionally, these principal components are connected to a large number of dimensions, as evidenced by the uniformly distributed values across each row representing a principal component (see Fig. 1(b)). Consequently, directly pruning dimensions can adversely affect several principal components, thereby complicating the compression of hidden dimensions.

To address this, we incorporate a projection matrix prior to each masking matrix in the computational process of PLMs, referred to as PCA Projection. This matrix is designed to project features into a space determined by their principal components. Subsequently, these principal components can be pruned directly using masks. It's crucial to ensure that these added projection matrices do not alter the output of the PLM and can be seamlessly integrated with the model's parameters. In the following section, we detail the methodology for injecting these projection matrices to meet both of these conditions.

Hidden Dimension PCA Projection. To decrease the hidden dimension (d) between layers, our initial step involves selecting calibration data from the training dataset. We then derive the corresponding feature matrix $X \in \mathbb{R}^{d \times T}$ from either the Multi-Head Attention (MHA) block or the Feed-Forward Network (FFN) block, along with their respective residuals. Here, T represents the number of sampled tokens. We choose T to be marginally greater than d to guarantee that the number of tokens exceeds the number of feature dimension. This matrix also acts as the input for the $\mathrm{LayerNorm}$ layer. Subsequently, we apply SVD to extract the principal components of the feature. Each column of the matrix U represents a principal component, obtained from the SVD of the matrix X, as described by $U, \Sigma, V^\top = \mathrm{SVD}(X - \overline{X})$, where $\overline{X}$ represents the matrix obtained by broadcasting the average values of each column.

Next, we introduce the matrix U both before and after each $\mathrm{LayerNorm}$ layer as follows. Initially, we recognize:

$$\begin{aligned} \mathrm{LN}(X) &= \mathrm{diag}(\gamma)\mathrm{norm}(X - \overline{X}) \\ &= \mathrm{diag}(\gamma)\mathrm{norm}(UU^T(X - \overline{X})), \end{aligned} \tag{5}$$

where LN signifies $\mathrm{LayerNorm}$ module (the bias term is omitted for simplicity), and norm represents the normalization function defined as $\mathrm{norm}(x) = x/\|x\| * d$. Given the following formulas,

$$\begin{aligned} \mathrm{norm}(Ux) &= Ux/\|Ux\| * d \\ &= Ux/\sqrt{(x^\top U^\top Ux)} * d \\ &= Ux/\sqrt{(x^\top x)} * d \\ &= U\mathrm{norm}(x), \end{aligned} \tag{6}$$

$$\begin{aligned} X - \overline{X} &= (I - \frac{1}{d}11^\top)X \\ &= RX, \end{aligned} \tag{7}$$

where we define $R = (I - \frac{1}{d}11^\top)$ using I to represent the identity matrix and 1 to denote a vector filled with ones. Then, we have,

$$\mathrm{LN}(X) = \mathrm{diag}(\gamma)U\mathrm{norm}(U^\top RX). \tag{8}$$

Based on the formula above, we assign projection matrices $P_{\text{in}} = U^\top R$ and $P_{\text{out}} = \gamma U$, respectively.

The computation process after introducing the hidden dimension PCA Projection is as follows,

$$\begin{aligned} x_{\mathrm{M}} &= P^{\mathrm{M}}_{\text{out}}\mathrm{diag}(z^{\mathrm{M}}_{\text{out}})\mathrm{LN}(\mathrm{diag}(z^{\mathrm{M}}_{\text{in}})P^{\mathrm{M}}_{\text{in}}(x + z_{\mathrm{MHA}}\mathrm{MHA}(x))), \\ x_{\mathrm{F}} &= P^{\mathrm{F}}_{\text{out}}\mathrm{diag}(z^{\mathrm{F}}_{\text{out}})\mathrm{LN}(\mathrm{diag}(z^{\mathrm{F}}_{\text{in}})P^{\mathrm{F}}_{\text{in}}(x_{\mathrm{M}} + z_{\mathrm{FFN}}\mathrm{FFN}(x_{\mathrm{M}}))), \end{aligned} \tag{9}$$

where the superscripts ^{M} and ^{F} denote the projection and mask matrices within the MHA and FFN blocks, respectively.

MHA Intermediate Dimension PCA Projection. While our focus is primarily on compressing the hidden dimension, our method is also applicable to compressing the intermediate dimension of the

MHA block. We begin by collecting the feature matrix $X \in R^{d \times N}$ from the output preceding the MHA block. We aim to project the features $W_Q^{(i)}X$, $W_K^{(i)}X$ and $W_V^{(i)}X$ using their principal components $U_Q^{(i)}, U_K^{(i)}$, and $U_V^{(i)}$ respectively. They should satisfy the following equations:

$$\begin{aligned} W_V^{(i)}X &= U_V^{(i)\top}U_V^{(i)}W_V^{(i)}X, \\ X^\top W_Q^{(i)\top}W_K^{(i)}X &= X^\top W_Q^{(i)\top}U_Q^{(i)\top}U_K^{(i)}W_K^{(i)}X, \end{aligned} \tag{10}$$

where $U_V^{(i)}$ in the equation can be solved directly by SVD, i.e., $U_V^{(i)} = U$, where $U, \Sigma, V^\top = \text{SVD}(W_V X)$. The solution of $U_Q^{(i)}, U_K^{(i)}$ in Eq. 10 is a more complex, and we follow the method in DRONE (Chen et al., 2021) to compute $U_Q^{(i)}, U_K^{(i)}$:

$$\begin{aligned} U_Q^{(i)} &= \Sigma_Z^{(i)\frac{1}{2}} U_Z^{(i)\top} \Sigma_1^{(i)-1} U_1^{(i)\top}, \\ U_K^{(i)} &= \Sigma_Z^{(i)\frac{1}{2}} V_Z^{(i)} \Sigma_2^{(i)-1} U_2^{(i)\top}, \end{aligned} \tag{11}$$

where

$$\begin{aligned} U_1^{(i)}, \Sigma_1^{(i)}, V_1^{(i)\top} &= \text{SVD}(W_Q^{(i)}X), \\ U_2^{(i)}, \Sigma_2^{(i)}, V_2^{(i)\top} &= \text{SVD}(W_K^{(i)}X), \\ Z^{(i)} &= \Sigma_1^{(i)\top} U_1^{(i)\top} U_2^{(i)} \Sigma_2^{(i)}, \\ U_Z^{(i)}, \Sigma_Z^{(i)}, V_Z^{(i)\top} &= \text{SVD}(Z^{(i)}). \end{aligned} \tag{12}$$

We then assign $P_Q^{(i)} = U_Q^{(i)}$, $P_K^{(i)} = U_K^{(i)}$, $P_V^{(i)} = U_V^{(i)\top}$, and $P_O^{(i)} = U_V^{(i)}$ as our target projection matrices. For the proof of Eq. 11, refer to Appendix A.

The computation process incorporating MHA intermediate dimension PCA Projection is as follows,

$$\begin{aligned} &\text{Att}^{(i)}(x) = W_O^{(i)\top} P_O^{(i)\top} \text{diag}(z_O^{(i)})^\top \text{diag}(z_V^{(i)}) P_V^{(i)} W_V^{(i)} x \cdot \\ &\text{Softmax}\left(\frac{(W_K^{(i)}x)^\top P_K^{(i)\top} \text{diag}(z_K^{(i)})^\top \text{diag}(z_Q^{(i)}) P_Q^{(i)} (W_Q^{(i)}x)}{\sqrt{d_h}}\right). \end{aligned} \tag{13}$$

Model Structure. We integrate the aforementioned PCA projections across various dimensions and present the final structure of our model in Fig. 3. Unlike mask-based pruning methods such as CoFi, which employ a shared mask in each block, our approach utilizes distinct masks for different blocks and incorporates specific projection matrices for each block (as contrasted in Eq. 3 and Eq. 9). This strategy enables us to independently compress the hidden dimension of each block. Additionally, we insert projection matrices in the MHA block to compress its intermediate dimension (as detailed in Eq. 13). The transformer's residual structure necessitates the same hidden dimension across different blocks. By introducing a linear transformation in the residuals, we effectively circumvent this constraint (as illustrated in Figure 1(d)). Readers interested in the algorithmic details can find the pseudo-code for the PCA Projection in the Appendix B.

Complexity. The computational complexity of PCA Projection primarily hinges on three factors: the number of layers n, the hidden dimensions d, and the number of sampled tokens T, resulting in an approximate complexity of $O(n * d^2 * T)$. In practice, for a model with 100M parameters, the PCA Projection process takes around 5 minutes. For a larger model with 7B parameters, we estimate that the PCA Projection would require roughly 1 hour, significantly less than the time required for model pruning.

3.2 Model Pruning

Objective Function. Pruning is governed by the masks incorporated into the model, and the model's sparsity is determined based on these masks. Throughout the training process, all mask variables are treated as continuous real numbers. Upon completion of the training, those mask variables falling below a certain threshold — set in accordance with the desired sparsity level — are converted to 0, thereby establishing the final pruned structure. We use pruning loss (Wang et al., 2020) that forces the expected sparsity of the model to be close to the desired sparsity:

$$\mathcal{L}_{\text{pruning}} = \lambda_1 \cdot (\hat{s} - t) + \lambda_2 \cdot (\hat{s} - t)^2, \tag{14}$$

where $\hat{s}$ is the expected sparsity, t is the target sparsity, and λ_1, λ_2 are two Lagrange multipliers. Please refer to Appendix C for more detail of $\hat{s}$. Additionally, we incorporate the same task-specific loss, $\mathcal{L}_{task}$, as used in CoFi (Xia et al., 2022), applied to the task-oriented dataset during the pruning process to maintain model performance on specified tasks.

Multi-stage Pruning. In SP^3, we implement masks at three hierarchical levels: dimension-level (z_{in}, z_{out}, z_{f}, z_{Q}, z_{K}, z_{V}, z_{O}), head-level (z_{head}), and layer-level (z_{MHA}, z_{FFN}). Our findings reveal that during training, the model prioritizes head-level and layer-level masks for compression, while dimension-level masks are sometimes less emphasized. Consequently, we have divided the training process into two distinct stages. In the first stage, we exclusively employ dimension-level masks, and

then, in the subsequent stage, we add head-level and layer-level masks. Empirically, the initial stage spanned 5 epochs, followed by the second stage, which lasted for 15 epochs.

3.3 Model Fusing

After training, we first prune projection matrices, attention heads, MHA blocks, and FFN blocks based on different masks. After that, we merge the projection matrices with weight matrices. Suppose function Pr denotes pruning all zero rows and columns of the matrix.

MHA Layer Fusing. For the MHA layer, we have

$$\begin{aligned}\hat{W}_Q^{(i)} &= \mathrm{Pr}(\mathrm{diag}(z_{\mathrm{Q}})P_Q W_Q^{(i)} P_{\mathrm{out}}^{\mathrm{M}} \mathrm{diag}(z_{\mathrm{out}}^{\mathrm{M}})),\\ \hat{W}_K^{(i)} &= \mathrm{Pr}(\mathrm{diag}(z_{\mathrm{K}})P_K W_K^{(i)} P_{\mathrm{out}}^{\mathrm{M}} \mathrm{diag}(z_{\mathrm{out}}^{\mathrm{M}})),\\ \hat{W}_V^{(i)} &= \mathrm{Pr}(\mathrm{diag}(z_{\mathrm{V}})P_V W_V^{(i)} P_{\mathrm{out}}^{\mathrm{M}} \mathrm{diag}(z_{\mathrm{out}}^{\mathrm{M}})),\\ \hat{W}_O^{(i)\top} &= \mathrm{Pr}(\mathrm{diag}(z_{\mathrm{in}}^{\mathrm{F}})P_{\mathrm{in}}^{\mathrm{F}} W_O^{(i)\top} P_O \mathrm{diag}(z_{\mathrm{O}})),\end{aligned} \tag{15}$$

where $\hat{W}_Q^{(i)}, \hat{W}_K^{(i)}, \hat{W}_V^{(i)}$, and $\hat{W}_O^{(i)}$ denotes the compressed weight matrices of the MHA block.

FFN Layer Fusing. For the FFN layer, we have

$$\begin{aligned}\hat{W}_U &= \mathrm{Pr}(\mathrm{diag}(z_f)W_U P_{\mathrm{out}}^{\mathrm{M}} \mathrm{diag}(z_{\mathrm{out}}^{\mathrm{M}})),\\ \hat{W}_D^{\top} &= \mathrm{Pr}(\mathrm{diag}(z_{\mathrm{in}}^{\mathrm{F}})P_{\mathrm{in}}^{\mathrm{F}} W_D^{\top} \mathrm{diag}(z_f)),\end{aligned} \tag{16}$$

where $\hat{W}_U$ and $\hat{W}_D$ denotes the compressed weight matrices of the FFN block.

Residual Modification. Since we adaptively learn different hidden dimensions in different layers, we need to add weight matrices to the residual. For example, for i-th block of Transformer, we have

$$x^{(i)} = \mathrm{LN}(W_{\mathrm{A}}^{(i)} x + \mathrm{Block}^{(i)}(x^{(i-1)}))), \tag{17}$$

where W_{A} is a matrix added to the residual, which is determined by multiplying the pruned projection matrices before and after the block (refer to Figure 1 (d)), i.e.,

$$W_{\mathrm{A}}^{(i)} = \mathrm{Pr}(\mathrm{diag}(z_{\mathrm{in}}^{(i-1)})P_{\mathrm{out}}^{(i-1)} P_{\mathrm{in}}^{(i)} \mathrm{diag}(z_{\mathrm{out}}^{(i)})). \tag{18}$$

4 Experiments

4.1 Setup

Datasets. We evaluate our approach on GLUE (Wang et al., 2018) tasks and SQuAD (Rajpurkar et al., 2016) v1.1. GLUE tasks include SST-2 (Socher et al., 2013), MNLI (Kim et al., 2019), QQP (Wang et al., 2017), QNLI, MRPC (Dolan and Brockett, 2005), STS-B, and RTE. We exclude CoLA due to their unstable behaviors, and we cannot reproduce some baseline results based on our device on the CoLA dataset. Each comparison model is evaluated on the test set of these datasets. For the GLUE benchmark, accuracy metrics are used for MNLI, QQP, QNLI, SST2, MRPC, and RTE tasks, while Spearman's correlation is reported for STS-B. In the case of the SQuAD benchmark, we present the F1 score. For comprehensive details about these datasets, please refer to Appendix D.1.

Baselines. We evaluate the proposed SP^3 against established distillation methods like TinyBERT (Jiao et al., 2020), WID (Wu et al., 2023b), and the structured pruning approach CoFi (Xia et al., 2022).

Settings. Our primary experiments focus on compressing the $\mathrm{BERT}_{\mathrm{base}}$ model. SP^3 and CoFi perform model pruning using the previously mentioned training datasets. In contrast, WID and TinyBERT additionally use more general pre-training data. For a fairer comparison, we have excluded the data augmentation component from TinyBERT in our experiments. To generate calibration data for the SVD decomposition in SP^3, we randomly choose 512 samples from each dataset and concatenate them. From this merged data, we then randomly select 4,096 tokens as our calibration dataset. For detailed information on our experimental methods, please see the Appendix D.2.

4.2 Main Results

As shown in Table 1, we compress the $\mathrm{BERT}_{\mathrm{base}}$ model and compare the performance of SP^3 with other methods under similar sparsity. First, compared to the original model, SP^3 retains 96% of the model performance while removing 94% of the model parameters. Second, compared to other baselines, SP^3 has extra degrees of freedom. Compared to the pruning method CoFi, SP^3 can additionally compress the hidden dimensions and intermediate dimensions of the MHA block. Unlike distillation methods, SP^3 can use different hidden dimensions at different layers with additional head-level and layer-level pruning. Overall, SP^3 obtains the best performance on all datasets, indicating that utilizing these extra degrees of freedom can benefit the performance during compression. Comparisons with more baselines on $\mathrm{BERT}_{\mathrm{base}}$ and experiments on OPT and LLaMa are shown in Appendix E.

We delve into the impact of specifically com-

	Params.	SST-2	QNLI	MNLI	QQP	RTE	STS-B	MRPC	SQuAD	Avg.
$\text{BERT}_{\text{base}}$	85M	93.1	91.5	84.8	91.2	70.4	89.1	85.6	88.4	86.76
TinyBERT_4	4.7M	89.7	86.7	78.8	90.0	63.2	85.0	81.4	82.1	82.11
WID	5.0M	88.8	85.4	78.4	89.5	60.3	84.5	81.9	81.2	81.25
CoFi	~5.0M	90.6	86.1	80.6	**90.1**	64.7	83.1	82.6	82.6	82.55
SP^3	~5.0M	**91.4**	**87.6**	**81.6**	**90.1**	**66.4**	**86.1**	**82.8**	**83.2**	**83.65**
Speedup		×2.65	×2.79	×2.55	×2.55	×4.98	×5.58	×4.66	×2.83	

Table 1: Comparison between our SP^3 and both the distillation methods and pruning methods. Note that, following CoFi, we do not count the number of parameters in the embedding layer.

	Params.	SST-2	QNLI	MNLI	QQP	RTE	STS-B	MRPC	SQuAD	Avg.
SP^3	25M	91.1	89.6	80.8	90.9	66.8	86.8	85.0	84.8	84.48
CoFi	25M	86.9	82.2	78.7	90.0	56.7	82.3	71.8	79.1	78.46

Table 2: Comparison of SP^3 and CoFi in Compressing the Hidden Dimension Only.

	SST-2	RTE	STS-B	MRPC
SP^3-10M	**91.3**	**66.8**	**87.8**	**85.5**
w/o PCA-Projection	91.2	52.7	85.0	80.6

Table 3: Effect of PCA Projection. SP^3-10M refers to $\text{BERT}_{\text{base}}$ compressed to 10M by omitting masks z_{head}, z_{MHA}, and z_{FFN}.

Number of tokens	SST-2	RTE	STS-B	MRPC
2,048	90.0	58.8	85.9	82.8
4,096	**91.4**	**64.2**	86.1	82.8
8,192	90.8	63.5	**86.4**	**83.1**

Table 4: Effect of sampled tokens on PCA projection.

pressing the hidden size d in our SP^3 . This entails retaining only the projection matrices $P^{\cdot}_{\text{in}}, P^{\cdot}_{\text{out}}$ and masks $z^{\cdot}_{\text{int}}, z^{\cdot}_{\text{out}}$ in SP^3 (refer to Eq. 9), while maintaining the masks $z_{\text{int}}, z_{\text{out}}$ in the CoFi approach (see Eq. 3). As indicated in Table 2, SP^3 demonstrates significant superiority over CoFi, which relies solely on masks for compressing the hidden dimensions. These experimental findings underscore the efficacy of PCA Projection in the compression of hidden dimensions, affirming its effectiveness in enhancing model performance.

4.3 Ablation Study on PCA Projection

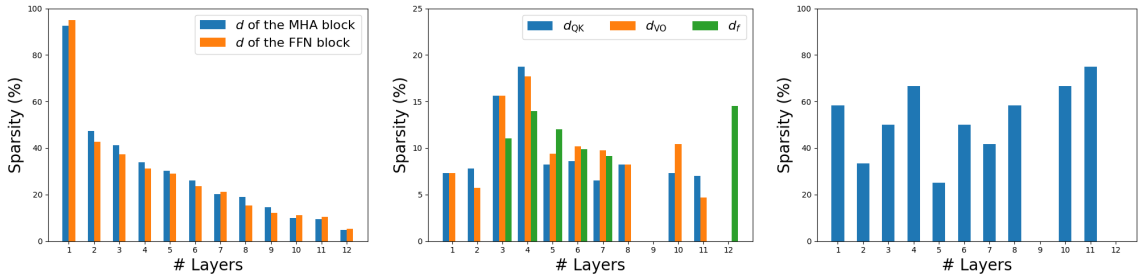
PCA Projection employs principal components obtained through SVD decomposition as the projection matrix. To assess the importance of these principal components, we conducted an experiment where we initialized the projection matrices as identity matrices. To prevent head-level and layer-level masks from interfering with the role of PCA Projection, we use only dimension-level masks in our experiments. The results, presented in Table 3, show that omitting PCA Projection leads to a notable drop in performance on the RTE and MRPC datasets, and also a slight decrease on the SST-2 and STS-B datasets. This indicates that PCA Projection plays a significant role in enhancing the performance of the compressed model.

Furthermore, PCA Projection relies on calibration data derived from the training dataset. We examine how the number of tokens in this data affects model performance. As shown in Table 4, the accuracy of the model varies with different sample token sizes. Considering that a higher number of samples adds complexity to the SVD computation, choosing 4,096 tokens strikes an optimal balance between accuracy and computational efficiency. Refer to Appendix F for more ablation experiments on head- and layer-level masks.

4.4 Structures of Pruned Model

We study the pruned structures produced by SP^3. Take the MRPC dataset as an example. Figure 4 shows the structural information of the pruned model. More results are shown in Appendix G.

From Fig. 4 (b) and (c), as well as figures for other datasets, it's evident that the model structure varies across different datasets. However, a consistent observation across these structures is that layers nearer the output are more compressed than those closer to the inputs. Additionally, the inter-



(a) Hidden dimensions (b) Intermediate dimensions (c) Attention head size

Figure 4: Structural information of the pruned model on the MRPC dataset, where sparsity denotes the ratio of the remaining dimension or size to the original dimension or size. (a) Output dimensions of each MHA and FFN block. (b) Intermediate dimensions of each MHA and FFN block. (c) The number of attention heads in each MHA block.

mediate dimensions of the FFN block are notably more compressed across all datasets compared to the intermediate dimensions of the MHA block. This distinction is highlighted when comparing the green bars to the blue and red bars in Fig. 4 (b). The compression patterns we observed are consistent with those reported in prior pruning studies (Xia et al., 2022). In addition, Fig. 4 (a) and similar figures for other datasets indicate a trend where the model's hidden dimension reduces as the number of layers increases. This trend suggests that the model incrementally compresses features into more compact dimensions during its inference process.

5 Related Work

Distillation. Knowledge distillation (Hinton et al., 2015; Hou et al., 2020; Jiao et al., 2020; Sanh et al., 2019) is a model compression approach that transfers knowledge from a larger teacher model to a smaller student model. In contrast to the distillation methods, our approach uses less computational cost while achieving the same performance.

Pruning. Existing pruning methods can be broadly divided into two categories: unstructured and structured. Unstructured pruning (Gale et al., 2019; Frankle and Carbin, 2018; Kurtic et al., 2022; Louizos et al., 2018; Sanh et al., 2020) aims to remove unimportant scalar values from the model's parameters. In contrast, structured pruning (Lin et al., 2020; Lagunas et al., 2021; Wang et al., 2020; Xia et al., 2022) are proposed to remove weight blocks in PLMs, including the entire layer (Fan et al., 2019; Sajjad et al., 2023, 2020), attention heads (Michel et al., 2019; Voita et al., 2019), and filters (McCarley et al., 2019; Prasanna et al., 2020). Unlike the previous structured pruning, we focus on exploring how to compress the hidden dimensions of the model efficiently.

Low-Rank Factorization. Low-rank factorization methods compress the PLMs by decomposing the weight matrices (Liu and Ng, 2022; Ma et al., 2019; Hua et al., 2022; Xiao et al., 2023; Yin et al., 2022; Zhou et al., 2019) based on low-rank property. In our approach, we propose PCA Projection based on the low-rank property of the features.

Re-parameterization (Re-p). Re-p methods have been proposed to improve existing model compression methods (Ding et al., 2021; Lin et al., 2021; Wu et al., 2023b). Its core idea is to represent the compressed model based on the original model. Therefore, this method can utilize the parameters of the original model more efficiently, which leads to better performance of the compressed model. SP^3 can also be regarded as a Re-p method, and the major difference between SP^3 and the existing methods is that we better initialize the parameters additionally inserted into the model by the Re-p method based on the PCA theory.

6 Conclusion

This study introduces SP^3, an enhanced structured pruning approach for compressing PLMs. SP^3 employs PCA Projection, facilitating easier optimization of the compression model, thereby boosting its performance. Concurrently, SP^3 uses non-shared masks in the hidden dimension. This allows the model to selectively determine its hidden dimensions size at different layers in line with the desired sparsity. When applied to $\text{BERT}_{\text{base}}$ and evaluated on the GLUE and SQuAD benchmarks, SP^3 notably achieves a 94% sparsity with only a minor 4% reduction in accuracy.

7 Limitations

Our proposed SP^3 introduces extra weight matrices in the residual parts when merging the inserted compactor matrices with the weight matrices. When the model is compressed to 5M, these extra parameters account for more than half of the model. This predominance hinders further compression. In future research, we aim to explore strategies to eliminate these extraneous parameters. Meanwhile, the efficient acceleration of the model mainly relies on coarse-grained pruning (head-level, layer-level), while our approach uses more fine-grained pruning, and how to balance the fine-grained and coarse-grained pruning to achieve higher acceleration also needs our further exploration.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Patrick Chen, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. 2021. Drone: Data-aware low-rank compression for large nlp models. *Advances in neural information processing systems*, 34:29321–29334.

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

Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. 2021. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4510–4520.

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. 2019. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635.

Hao Fu, Shaojun Zhou, Qihong Yang, Junjie Tang, Guiquan Liu, Kaikui Liu, and Xiaolong Li. 2021. Lrc-bert: latent-representation contrastive knowledge distillation for natural language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12830–12838.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

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*, 33:9782–9793.

Ting Hua, Yen-Chang Hsu, Felicity Wang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Numerical optimizations for weighted low-rank estimation on language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1404–1416, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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 the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.

Seonhoon Kim, Inho Kang, and Nojun Kwak. 2019. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6586–6593.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The optimal BERT surgeon: Scalable and accurate second-order pruning for large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4163–4181, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *arXiv preprint arXiv:2204.09656*.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jianquan Li, Xiaokang Liu, Honghong Zhao, Ruifeng Xu, Min Yang, and Yaohong Jin. 2020. Bert-emd: Many-to-many layer mapping for bert compression with earth mover's distance. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3009–3018.

Ye Lin, Yanyang Li, Ziyang Wang, Bei Li, Quan Du, Tong Xiao, and Jingbo Zhu. 2021. Weight distillation: Transferring the knowledge in neural network

parameters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2076–2088, Online. Association for Computational Linguistics.

Zi Lin, Jeremiah Liu, Zi Yang, Nan Hua, and Dan Roth. 2020. Pruning redundant mappings in transformer models via spectral-normalized identity prior. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 719–730, Online. Association for Computational Linguistics.

Ye Liu and Michael K Ng. 2022. Deep neural network compression by tucker decomposition with nonlinear response. *Knowledge-Based Systems*, page 108171.

Yuanxin Liu, Zheng Lin, and Fengcheng Yuan. 2021. Rosita: Refined bert compression with integrated techniques. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8715–8722.

Christos Louizos, Max Welling, and Diederik P Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. 2019. A tensorized transformer for language modeling. *Advances in neural information processing systems*, 32.

Andrzej Maćkiewicz and Waldemar Ratajczak. 1993. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342.

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? *Advances in neural information processing systems*, 32.

Azade Nova, Hanjun Dai, and Dale Schuurmans. 2023. Gradient-free structured pruning with unlabeled data. *arXiv preprint arXiv:2303.04185*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703.

Tianduo Wang Peiyuan Zhang, Guangtao Zeng and Wei Lu. 2023. Tinyllama.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When bert plays the lottery, all tickets are winning. *arXiv preprint arXiv:2005.00561*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Siyu Ren and Kenny Q Zhu. 2023. Low-rank prune-and-factorize for language model compression. *arXiv preprint arXiv:2306.14152*.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models. *arXiv preprint arXiv:2004.03844*, 2(2).

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Marzieh Tahaei, Ella Charlaix, Vahid Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. 2022. KroneckerBERT: Significant compression of pre-trained language models through kronecker decomposition and knowledge distillation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2116–2127, Seattle, United States. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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 *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Siyue Wu, Hongzhan Chen, Xiaojun Quan, Qifan Wang, and Rui Wang. 2023a. Ad-kd: Attribution-driven knowledge distillation for language model compression. *arXiv preprint arXiv:2305.10010*.

Taiqiang Wu, Cheng Hou, Zhe Zhao, Shanshan Lao, Jiayi Li, Ngai Wong, and Yujiu Yang. 2023b. Weight-inherited distillation for task-agnostic bert compression. *arXiv preprint arXiv:2305.09098*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528.

Jinqi Xiao, Miao Yin, Yu Gong, Xiao Zang, Jian Ren, and Bo Yuan. 2023. COMCAT: Towards efficient compression and customization of attention-based vision models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38125–38136. PMLR.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. Bert-of-theseus: Compressing bert by progressive module replacing. *arXiv preprint arXiv:2002.02925*.

Runxin Xu, Fuli Luo, Chengyu Wang, Baobao Chang, Jun Huang, Songfang Huang, and Fei Huang. 2022. From dense to sparse: Contrastive pruning for better pre-trained language model compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11547–11555.

Miao Yin, Huy Phan, Xiao Zang, Siyu Liao, and Bo Yuan. 2022. Batude: Budget-aware neural network compression based on tucker decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8, pages 8874–8882.

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International Conference on Machine Learning*, pages 26809–26823. PMLR.

Mingyi Zhou, Yipeng Liu, Zhen Long, Longxi Chen, and Ce Zhu. 2019. Tensor rank learning in cp decomposition via convolutional neural network. *Signal Processing: Image Communication*, 73:12–21.

A MHA Intermediate Dimension PCA Projection

We can verify the effectiveness of the Eq. 11 in the following way,

$$\|X^\top W_Q^{(i)\top} W_K^{(i)} X - X^\top W_Q^{(i)\top} U_{Q,:k}^{(i)\top} U_{K,:k}^{(i)} W_K^{(i)} X\|, \tag{19}$$

Eq. 19 measures the difference between the inner product computed using the first k principal components of $U_Q^{(i)}, U_K^{(i)}$ and the original result.

Next, we prove that $U_Q^{(i)}$ and $U_K^{(i)}$ satisfy the following equation,

$$\mathrm{M}_k = U_{Q,:k}^{(i)\top} U_{K,:k}^{(i)} \tag{20}$$

where,

$$\begin{aligned} \mathrm{M}_k = \arg\min_{\mathrm{M}} &\|X^\top W_Q^{(i)\top} W_K^{(i)} X - X^\top W_Q^{(i)\top} \mathrm{M} W_K^{(i)} X\|, \\ &\text{s.t. } rank(\mathrm{M}) = k \end{aligned} \tag{21}$$

Suppose

$$\begin{aligned} X_Q &= W_Q^{(i)} X, \\ X_K &= W_K^{(i)} X, \\ U_1, \Sigma_1, V_1^\top &= \mathrm{SVD}(X_Q), \\ U_2, \Sigma_2, V_2^\top &= \mathrm{SVD}(X_K). \end{aligned} \tag{22}$$

Then, we have

$$\begin{aligned} \mathrm{M}_k &= \arg\min_{\mathrm{M}} \|X_Q^\top X_K - X_Q^\top \mathrm{M} X_K\| \\ &= \arg\min_{\mathrm{M}} \|V_1^\top X_Q^\top X_K V_2 - V_1^\top X_Q^\top \mathrm{M} X_K V_2\| \\ &= \arg\min_{\mathrm{M}} \|\Sigma_1^\top U_1^\top U_2 \Sigma_2 - \Sigma_1^\top U_1^\top \mathrm{M} U_2 \Sigma_2\| \end{aligned} \tag{23}$$

Define

$$\begin{aligned} Z &= \Sigma_1^\top U_1^\top U_2 \Sigma_2, \\ U_Z, \Sigma_Z, V_Z^\top &= \mathrm{SVD}(Z), \end{aligned} \tag{24}$$

then, based on the PCA theory, we know that $Z_k = U_{Z,:k} \Sigma_{Z,:k} V_{Z,:k}^\top$ is the solution of

$$\arg\min_{Z^*} \|Z - Z^*\|, \quad \text{s.t. } \mathrm{rank}(Z^*) = k. \tag{25}$$

Therefore, we have

$$\begin{aligned} &\Sigma_Q^\top U_1^\top \mathrm{M_k} U_2 \Sigma_2 = U_{Z,:k} \Sigma_{Z,:k} V_{Z,:k}^\top \\ &\Rightarrow \mathrm{M}_k = U_1 \Sigma_1^{-1\top} U_{Z,:k} \Sigma_{Z,:k} V_{Z,:k}^\top \Sigma_2^{-1} U_2^\top. \end{aligned} \tag{26}$$

Refer to Eq. 11,

$$\begin{aligned} U_Q^{(i)} &= \Sigma_Z^{(i)\frac{1}{2}} U_Z^{(i)\top} \Sigma_1^{(i)-1} U_1^{(i)\top}, \\ U_K^{(i)} &= \Sigma_Z^{(i)\frac{1}{2}} V_Z^{(i)} \Sigma_2^{(i)-1} U_2^{(i)\top}, \end{aligned} \tag{27}$$

we also have

$$\begin{aligned} U_{Q,:k}^{(i)} &= \Sigma_{Z,:k}^{(i)\frac{1}{2}} U_{Z,:k}^{(i)\top} \Sigma_1^{(i)-1} U_1^{(i)\top}, \\ U_{K,:k}^{(i)} &= \Sigma_{Z,:k}^{(i)\frac{1}{2}} V_{Z,:k}^{(i)} \Sigma_2^{(i)-1} U_2^{(i)\top}. \end{aligned} \tag{28}$$

Therefore,

$$\mathrm{M}_k = U_{Q,:k}^{(i)\top} U_{K,:k}^{(i)} \tag{29}$$

B Pseudo Code of PCA Projection

The pseudo-code of PCA Projection is shown in Algorithm 1.

C Sparity

The expected sparsity $\hat{s}$ is computed as follow

$$\begin{aligned} \hat{s} = \frac{1}{M}(& \\ & \sum_i^L \sum_j^H \sum_k^d \sum_l^{d_h} z_{\mathrm{MHA}}^{(i)} \cdot z_{\mathrm{head}}^{(i,j)} \cdot z_{\mathrm{out,FFN,i-1}}^{(i,k)} \cdot z_Q^{(i,l)} + \\ & \sum_i^L \sum_j^H \sum_k^d \sum_l^{d_h} z_{\mathrm{MHA}}^{(i)} \cdot z_{\mathrm{head}}^{(i,j)} \cdot z_{\mathrm{out,FFN,i-1}}^{(i,k)} \cdot z_K^{(i,l)} + \\ & \sum_i^L \sum_j^H \sum_k^d \sum_l^{d_h} z_{\mathrm{MHA}}^{(i)} \cdot z_{\mathrm{head}}^{(i,j)} \cdot z_{\mathrm{out,FFN,i-1}}^{(i,k)} \cdot z_V^{(i,l)} + \\ & \sum_i^L \sum_j^H \sum_k^d \sum_l^{d_h} z_{\mathrm{MHA}}^{(i)} \cdot z_{\mathrm{head}}^{(i,j)} \cdot z_{\mathrm{in,MHA,i}}^{(i,k)} \cdot z_O^{(i,l)} + \\ & \sum_i^L \sum_k^d \sum_l^{d_f} z_{\mathrm{FFN}}^{(i)} \cdot z_{\mathrm{out,MHA,i}}^{(i,k)} \cdot z_f^{(i,l)} + \\ & \sum_i^L \sum_k^d \sum_l^{d_f} z_{\mathrm{FFN}}^{(i)} \cdot z_{\mathrm{in,FFN,i}}^{(i,k)} \cdot z_f^{(i,l)} + \\ & \sum_i^L \sum_k^d z_{\mathrm{out,FFN,i-1}}^{(i,k)} \cdot z_{\mathrm{in,MHA,i}}^{(i,k)} + \\ & \sum_i^L \sum_k^d z_{\mathrm{out,MHA,i}}^{(i,k)} \cdot z_{\mathrm{in,FFN,i}}^{(i,k)}), \end{aligned} \tag{30}$$

where M denotes the total number of parameters of Transformer.

D Experiment Details

D.1 Datasets

GLUE (Wang et al., 2018) benchmark consists of various tasks related to sentence similarity calculation, classification, textual entailment, and natural language inference. It includes 10 tasks, namely AX, COLA, QQP, MNLI, MRPC, QNLI, QQP, RTE, SST-2, STS-B, and WNLI. The number of training examples for each task is as follows: 1.1k, 10.7k, 432k, 5.8k, 105k, 364k, 3k, 70k, 67k, and 852, respectively. SQuAD 1.1 (Rajpurkar et al., 2016) dataset involves question-and-answer tasks, containing 88K training examples.

D.2 Experiment Setup

Our SP^3 is developed using PyTorch (Paszke et al., 2019) and executed on a server equipped with four NVIDIA 3090 GPU cards for all experiments. We source the BERT model from the HuggingFace

(Wolf et al., 2019) Transformers library. To prepare for compression, we first fine-tune BERT to create a task-specific model. This fine-tuning occurs on the training datasets from the GLUE benchmark and SQuAD, spanning 3 epochs, with a batch size of 32 and learning rates of 1e-5 and 2e-5. We retain the default parameters set by the HuggingFace framework for other settings.

The pruning of BERT follows the same configuration as the model fine-tuning process. Dimension-level pruning commences at the 3rd epoch, followed by the initiation of head-level and layer-level pruning at the 8th epoch.

E More Experiment Results

E.1 Compare SP^3 with More Baselines

The comparison results for SP^3 and more baselines are shown in Table 5.

E.2 Experiment Results on OPT-125m

The results of the experiment on opt-125m are shown in Table 6. For opt-125m, we use the same experimental setup as for BERT-base.

E.3 Experiment Results on TinyLLaMa

The results of the experiment on TinyLLaMa (1.1B) (Peiyuan Zhang and Lu, 2023) are shown in Table 7. we use the SFT dataset oasst_top1_2023-08-25 to prune the TinyLLaMa. We prune the model on the oasst_top1_2023-08-25 dataset for 2 epochs. Then, we tested the performance of the compressed model under a zero-shot setting.

F Impact of Multi-level Masks

SP^3 uses different levels of masks to compress the model. To explore the impact of different levels of masks on model performance, we conduct the following experiments: (1) Use all levels of masks. (2) Ignore head-level masks. (3) Ignore layer-level masks. (4) Ignore head-level and layer-level masks.

The findings from the experiment are presented in Table 8. Observations indicate superior model performance on the SST-2 and RTE datasets when all mask levels are utilized. For the STS-B datasets, the removal of the head-level mask results in the most precise models. Notably, the optimal performance on the MRPC dataset is achieved by a model that excludes both the head-level and layer-level masks. Given the data volume in each dataset, we hypothesize that minor alterations in the head-level and layer-level masks can significantly influence model outputs compared to the dimension-level mask. This implies that the head-level and layer-level masks might be more challenging to optimize. Consequently, removing either the head-level or layer-level mask in smaller datasets can stabilize the optimization process, leading to a more precise model. As dataset sizes increase, the need for model compression flexibility becomes evident, with multi-level masking yielding superior outcomes.

G Structures of Pruned Model

The structure of pruned models on RTE, SST-2, STS-B, MNLI, QNLI, QQP and SQuAD are shown in Fig. 5 and Fig. 6. We show the model structure in terms of dimension-level and head-level sparsity. Instead of directly showing the layer-level sparsity, we indirectly show the layer-level sparsity by dimension-level and head-level sparsity in the histograms, and if the value of a certain position is 0, it can be assumed that layer-level pruning has occurred at that position.

Algorithm 1: PCA Projection

Input: T: sampled tokens size, $\mathbb{D}$: a subset of training data, $L^{(1\sim N)}$: model
Output: Projection Matrix $P_{\text{in}}, P_{\text{out}}, P_Q, P_K, P_V, P_O$
$X_{\text{Q}}, X_{\text{K}}, X_{\text{V}}, X_{\text{M}}, X_{\text{F}} \leftarrow [], [], [], [], []$
for $j \leftarrow 1$ **to** $|\mathbb{D}|$ **do**
  $x^{(0)} \leftarrow \mathbb{D}[j]$
  for $i \leftarrow 1$ **to** N **do**
    $[W_Q^{(*)}, W_K^{(*)}, W_V^{(*)}, W_O^{(*)}] \leftarrow L^{(i)}$
    for $h \leftarrow 1$ **to** d_h **do**
      $X_Q^{(h),i} \leftarrow \text{Concate}(X_Q^{(h),i}, W_Q^{(h)} x^{(i-1)})$
      $X_K^{(h),i} \leftarrow \text{Concate}(X_K^{(h),i}, W_Q^{(h)} x^{(i-1)})$
      $X_V^{(h),i} \leftarrow \text{Concate}(X_V^{(h),i}, W_Q^{(h)} x^{(i-1)})$
    end
    $x_{\text{M}} = \text{MHA}^i(x^{(i-1)}) + x^{(i-1)}$
    $x_{\text{M}}^{\text{LN}} = LN(x_{\text{M}})$
    $x_{\text{F}} = \text{FFN}^i(x_{\text{M}}) + x_{\text{M}}$
    $x^{(i)} = \text{LN}(x_{\text{F}})$
    $X_{\text{M}}^i \leftarrow \text{Concate}(X_{\text{M}}^i, x_{\text{M}})$
    $X_{\text{F}}^i \leftarrow \text{Concate}(X_{\text{F}}^i, x_{\text{F}})$
  end
end
for $i \leftarrow 1$ **to** N **do**
  $[W_Q^{(*)}, W_K^{(*)}, W_V^{(*)}, W_O^{(*)}] \leftarrow L^{(i)}$
  for $h \leftarrow 1$ **to** d_h **do**
    $X_V^{(h),i} \leftarrow \text{Sample}(X_V^{(h),i}, T)$
    $X_Q^{(h),i} \leftarrow \text{Sample}(X_Q^{(h),i}, T)$
    $X_K^{(h),i} \leftarrow \text{Sample}(X_K^{(h),i}, T)$
    $U_V, \Sigma_V, V_V^\top \leftarrow \text{SVD}(X_V^{(h),i})$
    $U_1, \Sigma_1, V_1^\top \leftarrow \text{SVD}(X_Q^{(h),i})$
    $U_2, \Sigma_2, V_2^\top \leftarrow \text{SVD}(X_K^{(h),i})$
    $Z = \Sigma_1^\top U_1^\top U_2 \Sigma_2$
    $U_Z, \Sigma_Z, V_Z^\top = \text{SVD}(Z)$
    $U_Q = \Sigma_Z^{\frac{1}{2}} U_Z^\top \Sigma_1^{-1} U_1^\top$
    $U_K = \Sigma_Z^{\frac{1}{2}} V_Z \Sigma_2^{-1} U_2^\top$
    $P_V^{(h),i}, P_O^{(h),i} \leftarrow U_V^\top, U_V$
    $P_Q^{(h),i}, P_K^{(h),i} \leftarrow U_Q, U_K$
  end
  $[\gamma^M, \gamma^F] \leftarrow L^{(i)}$
  $R \leftarrow I - \frac{1}{d} 1 1^\top$
  $X_{\text{M}}^i \leftarrow \text{Sample}(X_{\text{M}}^i, T)$
  $X_{\text{F}}^i \leftarrow \text{Sample}(X_{\text{F}}^i, T)$
  $U_{\text{M}}, \Sigma_{\text{M}}, V_{\text{M}}^\top \leftarrow \text{SVD}(X_{\text{M}}^i)$
  $U_{\text{F}}, \Sigma_{\text{F}}, V_{\text{F}}^\top \leftarrow \text{SVD}(X_{\text{F}}^i)$
  $P_{\text{in}}^{\text{M},i}, P_{\text{out}}^{\text{M},i} \leftarrow U_{\text{M}}^\top R, \text{diag}(\gamma^{\text{M}}) U_{\text{M}}$
  $P_{\text{in}}^{\text{F},i}, P_{\text{out}}^{\text{F},i} \leftarrow U_{\text{F}}^\top R, \text{diag}(\gamma^{\text{F}}) U_{\text{F}}$
end

	Params.	SST-2	QNLI	MNLI	QQP	RTE	STS-B	MRPC	SQuAD
$\text{BERT}_{\text{base}}$	110M	93.1	91.5	84.8	91.2	70.4	89.1	85.6	88.4
SP^3	25M	91.4	87.6	81.6	90.1	66.4	86.1	82.8	83.2
DisitilBERT (Sanh et al., 2019)	13.6M	86.4	84.5	71.3	88.0	56.3	-	72.5	56.2
LPAF (Ren and Zhu, 2023)	13.6M	89.7	88.6	81.7	90.1	67.9	-	86.0	75.1
LRC-BERT (Fu et al., 2021)	14.5M	92.9	88.7	82.7	72.2	63.1	81.2	87.9	-
AD-KD (Wu et al., 2023a)	66M	91.8	90.0	82.6	88.9	65.8	83.4	87.1	-
DynaBERT (Hou et al., 2020)	10.6M	92.0	88.5	82.3	90.4	63.2	87.0	81.4	76.6
BERT-EMD (Li et al., 2020)	66M	93.3	90.7	84.7	72.0	71.7	86.8	89.8	-
KroneckerBERT (Tahaei et al., 2022)	5.2M	88.4	86.1	80.1	70.5	64.7	81.3	87.1	-
BERT-of-Theseus (Xu et al., 2020)	66M	91.5	89.5	82.3	89.6	68.2	88.7	89.0	-
Movement (Sanh et al., 2020)	3M	-	-	79.5	89.1	-	-	-	82.3
FLOP (Wang et al., 2020)	80M	92.1	89.1	-	-	-	88.2	88.6	-
ROSITA (Liu et al., 2021)	14.5M	87.6	83.8	77.7	88.3	-	-	-	-
PLATON (Zhang et al., 2022)	8.5M	90.5	88.9	82.2	90.2	65.3	87.1	84.3	79.0
CAP-f (Xu et al., 2022)	8.5M	89.7	-	81.2	90.2	-	-	-	70.2
Fast (Kwon et al., 2022)	66M	92.5	90.1	82.5	90.4	-	88.0	85.3	75.3
KCM (Nova et al., 2023)	50M	91.1	87.8	77.2	89.2	-	85.7	84.2	70.3

Table 5: Comparison between our SP^3 and other distillation and pruning methods.

	Params.	SST-2	QNLI	QQP	RTE	STS-B	MRPC
OPT_{125m}	85M	92.9	90.8	90.1	66.4	87.0	82.4
CoFi^*	7M	86.8	84.4	88.5	55.8	70.6	69.2
SP^3	7M	89.9	86.7	87.8	60.3	75.6	77.2

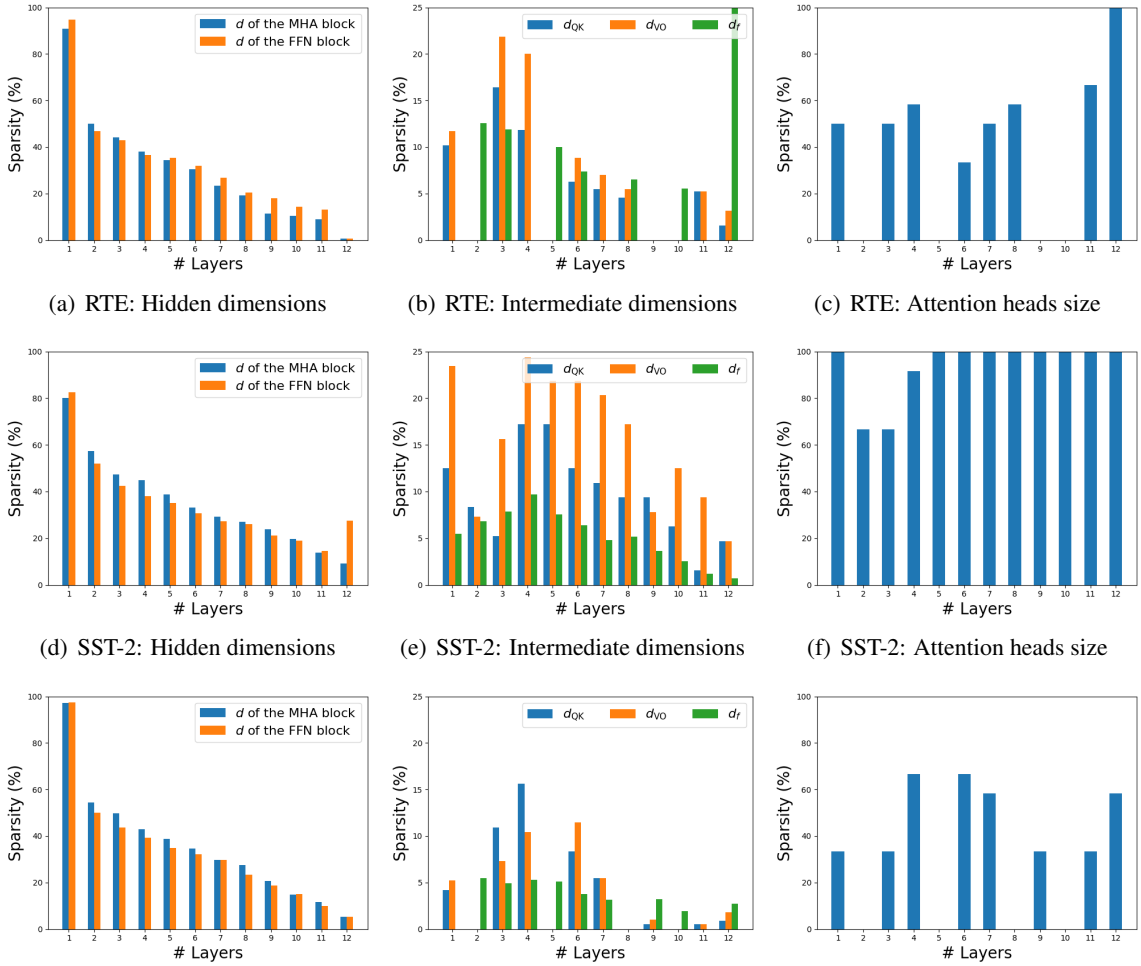
Table 6: Experiment results on OPT-125m, where CoFi^* means that the code of CoFi on opt-125m is implemented by ourselves.

	Pruning Ratio	ARC-e	ARC-c	BoolQ	HellaSwag	OBQA	PIQA	SciQ	WinoGrande
TinyLLaMa	0%	57.5	27.4	59.2	44.0	23.8	70.8	85.3	56.0
	20%	51.6	25.9	50.5	37.5	20.4	67.3	80.3	54.3
	40%	44.9	22.3	61.0	33.0	17.2	61.3	75.0	53.7

Table 7: Experiment results on TinyLLaMa.

	SST-2	RTE	STS-B	MRPC
SP^3-5M	**91.4**	**66.4**	86.1	82.8
w/o head	90.9	66.1	**86.6**	82.1
w/o layer	91.2	62.8	86.1	83.5
w/o head & layer	91.1	63.9	86.5	**83.8**

Table 8: Ablation studies on pruning units on SST-2, RTE, STS-B, and MRPC datasets.

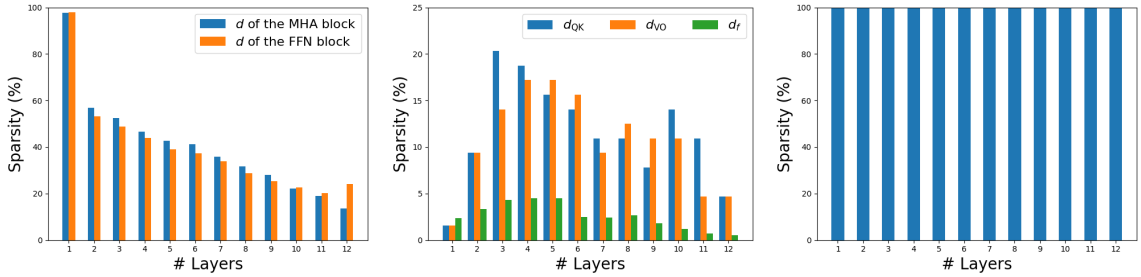


(a) RTE: Hidden dimensions (b) RTE: Intermediate dimensions (c) RTE: Attention heads size

(d) SST-2: Hidden dimensions (e) SST-2: Intermediate dimensions (f) SST-2: Attention heads size

(g) STS-B: Hidden dimensions (h) STS-B: Intermediate dimensions (i) STS-B: Attention heads size

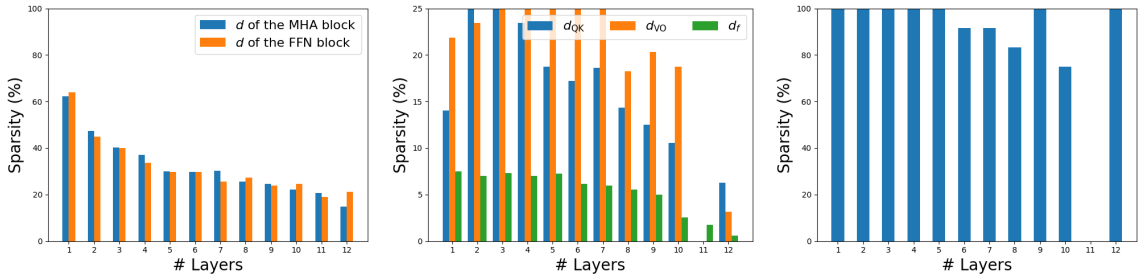
Figure 5: Pruned model structures on RTE, SST-2 and STS-B datasets



(a) MNLI: Hidden dimensions

(b) MNLI: Intermediate dimensions

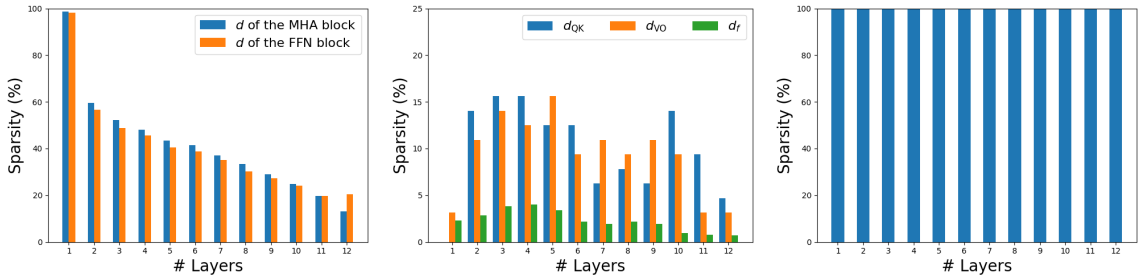
(c) MNLI: Attention heads size



(d) QNLI: Hidden dimensions

(e) QNLI: Intermediate dimensions

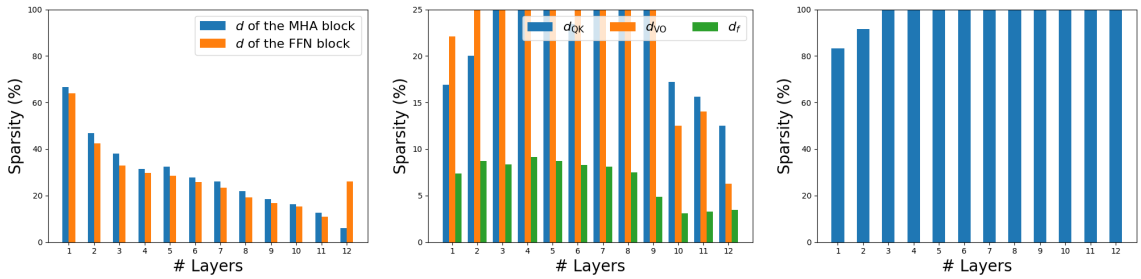
(f) QNLI: Attention heads size



(g) QQP: Hidden dimensions

(h) QQP: Intermediate dimensions

(i) QQP: Attention heads size



(j) SQuAD: Hidden dimensions

(k) SQuAD: Intermediate dimensions

(l) SQuAD: Attention heads size

Figure 6: Pruned model structures on MNLI, QNLI, QQP and SQuAD datasets