

# SP<sup>3</sup>: Enhancing Structured Pruning via PCA Projection

Yuxuan Hu<sup>1,2</sup>, Jing Zhang<sup>1,3\*</sup>, Zhe Zhao<sup>5</sup>, Chen Zhao<sup>1,2</sup>,  
Xiaodong Chen<sup>4</sup>, Cuiping Li<sup>1,3</sup>, Hong Chen<sup>1,3†</sup>

<sup>1</sup>School of Information, Renmin University of China, Beijing, China

<sup>2</sup>Key Laboratory of Data Engineering and Knowledge Engineering, MOE, China

<sup>3</sup>Engineering Research Center of Database and Business Intelligence, MOE, China

<sup>4</sup>Tencent AI Lab, Tencent, Beijing, China

<sup>5</sup>School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China

## 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<sup>3</sup>), 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<sup>3</sup> can reduce  $d$  by 70%, compress 94% of the BERT<sub>base</sub> model, and maintain over 96% accuracy and outperform other methods that compress  $d$  by 6% in accuracy at the same compression ratio. SP<sup>3</sup> has also proven effective with other models, including OPT and Llama. Our data and code are available at [ours repo](#).

## 1 Introduction

Pre-trained language models (PLMs) like BERT (Devlin et al., 2018), GPT-2 (Radford et al.), ChatGPT (Ouyang et al., 2022), and Llama (Touvron et al., 2023), leveraging transformer architectures, have gained prominence in Natural Language Processing due to their exceptional performance. However, their substantial storage requirements and long inference times present practical challenges, prompting the need for 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 under-explored, 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 for two main reasons. First, conventional mask-based pruning methods directly reduce dimensions in the original feature space, which doesn't effectively eliminate unimportant dimensions. We interpret the prun-

\*Corresponding author.

†{huyuxuan1999,zhang-jing}@ruc.edu.cn,  
nlpzhezhaoo@tencent.com, zhaochen100@ruc.edu.cn, chenxiaodong6677@163.com, {licuiping,chong}@ruc.edu.cn

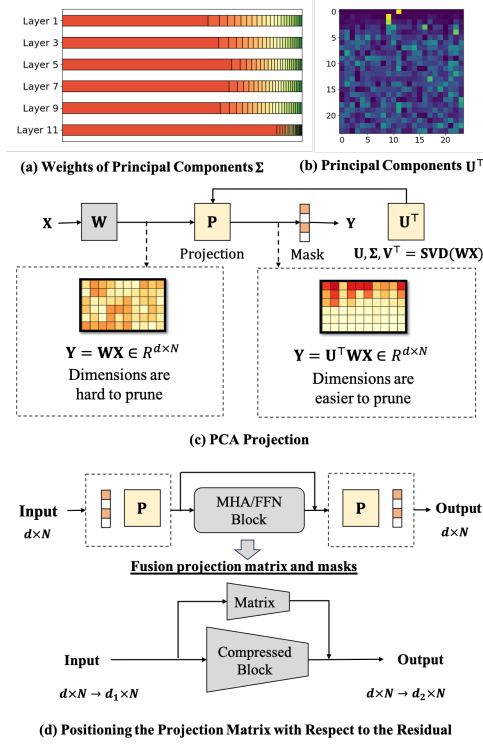


Figure 1: (a) Depiction of principal component weights within  $\Sigma$ . The length of each rectangle responds to the percentage of that principal component’s weight among all principal components; (b) Illustration of the principal component matrix. Each row of the matrix represents a principal component. Brighter colors indicate a higher correlation between the feature dimension and the principal component under investigation. (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.

ing of  $d$  from the perspective of features’ principal components. Drawing inspiration from 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 that represents 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 BERT<sub>base</sub> 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 an MHA block and an FFN block. We take BERT (Devlin et al., 2018) as an example to describe this architecture.

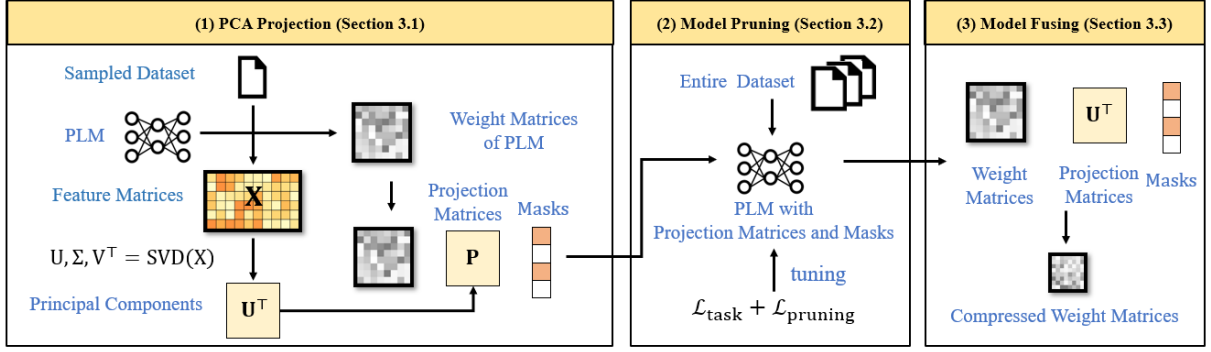


Figure 2: Illustration of the workflow of SP<sup>3</sup>.

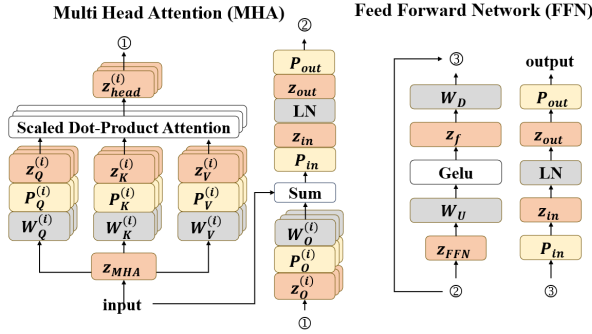


Figure 3: Illustration of the SP<sup>3</sup> architecture, in which the gray rectangles represent the weight matrices, the yellow rectangles signify the projection matrices, and the red rectangles indicate the masks.

**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 LayerNorm (Ba et al., 2016) step.

$$x_M = \text{LN}(x + \text{MHA}(x)),$$

$$\text{MHA}(x) = \sum_{i=1}^H \text{Att}^{(i)}(x),$$

$$\text{Att}^{(i)}(x) = W_O^{(i)\top} W_V^{(i)} x \cdot \text{Softmax}\left(\frac{(W_K^{(i)} x)^\top (W_Q^{(i)} x)}{\sqrt{d_h}}\right), \quad (1)$$

where  $W_Q^{(i)}, W_K^{(i)} \in \mathbb{R}^{d_{QK} \times d}$ ,  $W_V^{(i)}, W_O^{(i)} \in \mathbb{R}^{d_{VO} \times d}$  denote the query, key, value, and output matrices, respectively,  $d$  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.

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

$$\begin{aligned} x_F &= \text{LN}(x_M + \text{FFN}(x_M)), \\ \text{FFN}(x_M) &= W_D^\top \text{gelu}(W_U x_M), \end{aligned} \quad (2)$$

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

## 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:

$$\begin{aligned} x_M &= \text{diag}(z_{\text{out}}) \text{LN}(\text{diag}(z_{\text{in}})(x + z_{\text{MHA}} \text{MHA}(x))), \\ \text{MHA}(x) &= \sum_{i=1}^H z_{\text{head}}^{(i)} \text{Att}^{(i)}(x), \\ x_F &= \text{diag}(z_{\text{out}}) \text{LN}(\text{diag}(z_{\text{in}})(x_M + z_{\text{FFN}} \text{FFN}(x_M))), \\ \text{FFN}(x_M) &= W_D^\top \text{diag}(z_f) \text{gelu}(W_U x_M), \end{aligned} \quad (3)$$

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

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

where  $\theta$  denotes the parameters of the model, and  $t$  denotes the pre-defined target sparsity.

## 3 SP<sup>3</sup>

**Workflow of SP<sup>3</sup>.** The SP<sup>3</sup> 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 MHA block or the 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 ex-

ceeds the number of feature dimension. This feature matrix  $X$  acts as the input for the 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  $U, \Sigma, V^\top = \text{SVD}(X - \bar{X})$ , where  $\bar{X}$  represents the matrix obtained by broadcasting the average values of each column.

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

$$\begin{aligned} \text{LN}(X) &= \text{diag}(\gamma) \text{norm}(X - \bar{X}) \\ &= \text{diag}(\gamma) \text{norm}(UU^\top(X - \bar{X})), \end{aligned} \quad (5)$$

where LN signifies LayerNorm module (the bias term is omitted for simplicity),  $\text{norm}$  represents the normalization function defined as  $\text{norm}(x) = x/\|x\| * d$ , and  $UU^\top = I$  because columns of  $U$  are orthogonal. Given the following formulas,

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

$$\begin{aligned} X - \bar{X} &= (I - \frac{1}{d} \mathbf{1}\mathbf{1}^\top)X \\ &= RX, \end{aligned} \quad (7)$$

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

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

Based on this, we assign projection matrices  $P_{\text{in}} = U^\top R$  and  $P_{\text{out}} = \text{diag}(\gamma)U$ , respectively.

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

$$\begin{aligned} x_M &= P_{\text{out}}^M \text{diag}(z_{\text{out}}^M) \text{LN}(\text{diag}(z_{\text{in}}^M) P_{\text{in}}^M (x + z_{\text{MHA}} \text{MHA}(x))), \\ x_F &= P_{\text{out}}^F \text{diag}(z_{\text{out}}^F) \text{LN}(\text{diag}(z_{\text{in}}^F) P_{\text{in}}^F (x_M + z_{\text{FFN}} \text{FFN}(x_M))), \end{aligned} \quad (9)$$

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:

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

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  $U_Q^{(i)}, U_K^{(i)}$ :

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

where

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

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,

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

**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 dimension  $d$ , 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:

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

where  $\hat{s}$  is the expected sparsity,  $t$  is the target sparsity, and  $\lambda_1, \lambda_2$  are two Lagrange multipliers. Please refer to Appendix C for more detail of  $\hat{s}$ . Additionally, we incorporate the same task-specific loss,  $\mathcal{L}_{\text{task}}$ , 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<sup>3</sup>, we implement masks at three hierarchical levels: dimension-level ( $z_{\text{in}}, z_{\text{out}}, z_f, z_Q, z_K, z_V, z_O$ ), head-level ( $z_{\text{head}}$ ), and layer-level ( $z_{\text{MHA}}, z_{\text{FFN}}$ ). 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 spans 5 epochs, followed by the second stage, which lasts 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  $\text{Pr}$  denotes pruning all zero rows and columns of the matrix.

**MHA Layer Fusing.** For the MHA layer, we have

$$\begin{aligned}\hat{W}_Q^{(i)} &= \text{Pr}(\text{diag}(z_Q)P_QW_Q^{(i)}P_{\text{out}}^M\text{diag}(z_{\text{out}}^M)), \\ \hat{W}_K^{(i)} &= \text{Pr}(\text{diag}(z_K)P_KW_K^{(i)}P_{\text{out}}^M\text{diag}(z_{\text{out}}^M)), \\ \hat{W}_V^{(i)} &= \text{Pr}(\text{diag}(z_V)P_VW_V^{(i)}P_{\text{out}}^M\text{diag}(z_{\text{out}}^M)), \\ \hat{W}_O^{(i)\top} &= \text{Pr}(\text{diag}(z_{\text{in}}^F)P_{\text{in}}^F W_O^{(i)\top} P_O\text{diag}(z_O)),\end{aligned}\quad (15)$$

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

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

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

$$x^{(i)} = \text{LN}(W_R^{(i)}x + \text{Block}^{(i)}(x^{(i-1)})), \quad (17)$$

where  $W_R$  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.,

$$W_R^{(i)} = \text{Pr}(\text{diag}(z_{\text{in}}^{(i)})P_{\text{in}}^{(i)}P_{\text{out}}^{(i-1)}\text{diag}(z_{\text{out}}^{(i-1)})). \quad (18)$$

## 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. 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<sup>3</sup> against established distillation methods like TinyBERT (Jiao et al., 2020), WID (Wu et al., 2023b), and the pruning approach CoFi (Xia et al., 2022).

**Settings.** Our primary experiments focus on compressing the BERT<sub>base</sub> model. SP<sup>3</sup> 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<sup>3</sup>, we randomly choose 512 samples from each dataset and concatenate them, and then from them, we 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 BERT<sub>base</sub> model and compare the performance of SP<sup>3</sup> with other methods under similar sparsity. First, compared to the original model, SP<sup>3</sup> retains 96% of the model performance while removing 94% of the model parameters. Second, compared to other baselines, SP<sup>3</sup> has extra degrees of freedom. Compared to the pruning method CoFi, SP<sup>3</sup> can additionally compress the hidden dimensions and intermediate dimensions of the MHA block. Unlike distillation methods, SP<sup>3</sup> can use different hidden dimensions at different layers with additional head-level and layer-level pruning. Overall, SP<sup>3</sup> obtains the best performance on all datasets, indicating that utilizing these extra degrees of freedom can benefit the performance during compression. The comparisons with additional baselines on BERT<sub>base</sub> and experiments on TinyBERT with augmentation and OPT-125m are presented in Appendix E.

We delve into the impact of specifically compressing the hidden size  $d$  in our SP<sup>3</sup>. This entails retaining only the projection matrices  $P_{\text{in}}$ ,  $P_{\text{out}}$  and masks  $z_{\text{in}}^*$ ,  $z_{\text{out}}^*$  in SP<sup>3</sup> (refer to Eq. 9), while main-

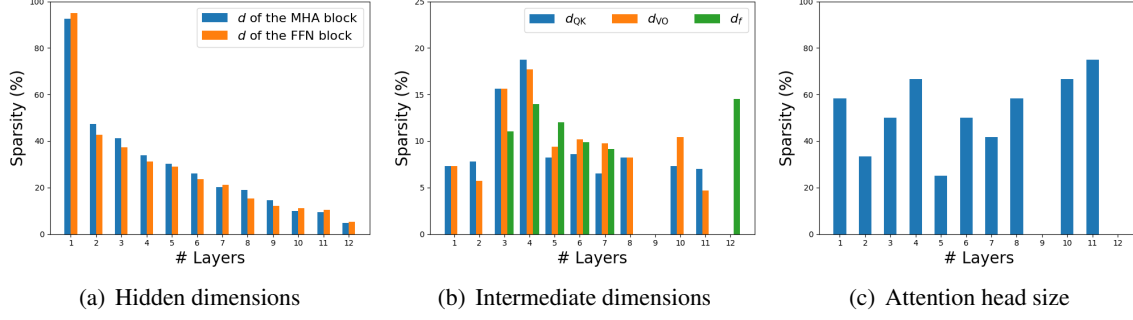


Figure 4: Structural information of the pruned model on the MRPC dataset, where sparsity denotes the ratio of the remaining dimension or size to the original dimension or size. (a) Output dimensions of each MHA and FFN block. (b) Intermediate dimensions of each MHA and FFN block. (c) The number of attention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

	Params.	SST-2	QNLI	MNLI	QQP	RTE	STS-B	MRPC	SQuAD	Avg.
BERT <sub>base</sub>	85M	93.1	91.5	84.8	91.2	70.4	89.1	85.6	88.4	86.76
TinyBERT <sub>4</sub>	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	<b>90.1</b>	64.7	83.1	82.6	82.6	82.55
SP <sup>3</sup>	~5.0M	<b>91.4</b>	<b>87.6</b>	<b>81.6</b>	<b>90.1</b>	<b>66.4</b>	<b>86.1</b>	<b>82.8</b>	<b>83.2</b>	<b>83.65</b>
Speedup		×2.65	×2.79	×2.55	×2.55	×4.98	×5.58	×4.66	×2.83	

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

taining the masks  $z_{\text{int}}, z_{\text{out}}$  in the CoFi approach (see Eq. 3). As indicated in Table 2, SP<sup>3</sup> 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.

We also apply our method to Large Language Models (LLMs), with experimental settings and results detailed in Appendix F. Table 8 illustrates the effective compression achieved by our method on TinyLlaMa. Additionally, thanks to the pre-RMSNorm architecture of TinyLlaMa, we propose Group PCA Projection. This approach unifies the computation of main components of features across several consecutive blocks, effectively reducing parameters introduced at the residuals and improving performance compared to distinct main components in each block, as demonstrated in Table 9.

### 4.3 Ablation Study on PCA Projection

PCA Projection employs principal components as the projection matrix. To assess the importance of these principal components, we conduct an experiment where we initialize the projection matrices

as identity matrices. To prevent other masks from interfering with the role of PCA Projection, we use only dimension-level masks in our experiments. The results, as depicted in Table 3, indicate that excluding PCA Projection results in a significant performance decline on the RTE and MRPC datasets, with a slight decrease observed on the SST-2 and STS-B datasets. This is because, for larger datasets, the model can directly optimize suitable compactor parameters from the data. The results underscore the significant role of PCA Projection 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. We also conduct experiments using different random seeds to sample calibration data and show its robustness. Appendix G presents the details as well as the studies about different masks.

	Params.	SST-2	QNLI	MNLI	QQP	RTE	STS-B	MRPC	SQuAD	Avg.
SP <sup>3</sup>	25M	<b>91.1</b>	<b>89.6</b>	<b>80.8</b>	<b>90.9</b>	<b>66.8</b>	<b>86.8</b>	<b>85.0</b>	<b>84.8</b>	<b>84.48</b>
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<sup>3</sup> and CoFi in compressing the hidden dimension of BERT<sub>base</sub>.

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

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

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

Table 4: Effect of sampled tokens on PCA projection.

#### 4.4 Structures of Pruned Model

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

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 intermediate 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 observe 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 trans-

fers 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 (LRF).** LRF methods compress the PLMs by decomposing the weight matrices (Hua et al., 2022; Xiao et al., 2023) based on low-rank property. In contrast, we propose PCA Projection based on the low-rank property of the features. For more discussions refer to Appendix I.

**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, these methods can utilize the parameters of the model more efficiently, which leads to better performance. SP<sup>3</sup> can also be regarded as a Re-p method, and the major difference between SP<sup>3</sup> and the existing methods is that we better initialize the parameters based on the PCA theory.

## 6 Conclusion

This study introduces SP<sup>3</sup>, an enhanced structured pruning approach for compressing PLMs. SP<sup>3</sup> employs PCA Projection, facilitating easier optimization of the compression model, thereby boosting its



performance. Concurrently, SP<sup>3</sup> 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 BERT<sub>base</sub> and evaluated on the GLUE and SQuAD benchmarks, SP<sup>3</sup> notably achieves a 94% sparsity with only a minor 4% reduction in accuracy.

## Limitations

Our proposed SP<sup>3</sup> 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 further exploration.

When applying to large language models, integrating compactors during training in our methodology may lead to increased memory consumption, necessitating efforts to reduce memory overhead. Additionally, experimental results indicate that our tuning-based pruning technique may face challenges with overfitting, particularly when tuning using a limited SFT dataset. Therefore, investigating a tuning-free pruning technique is warranted.

## Ethical Considerations

**Intellectual Property.** The datasets we use include oasst\_top1\_2023-08-25, MNLI, QNLI, QQP, SST-2, MRPC, STS-B, RTE, SQuAD, ARC-e, ARC-c, BoolQ, HellaSwag, openbookQA, PIQA, and WinoGrande. The models we use include BERT<sub>base</sub>, OPT<sub>125m</sub>, and TinyLlama. These are publicly accessible and well-established resources aimed at facilitating diverse AI and NLP research endeavors.

**Data annotation.** We utilize the annotations provided by existing datasets, thereby eliminating the need for manual annotation in our study.

**Intended Use.** SP<sup>3</sup> is a model pruning technique

designed to compress the parameters of a transformer model while preserving performance, effectively reducing the computational resources needed for deploying the model.

**Misuse risks.** SP<sup>3</sup> is a pruning method for transformer models and incorrect use of SP<sup>3</sup> might degrade the performance of some applications.

**Misuse Control.** We intend to make our approach available to the open-source community, enabling users to gain a deeper understanding of our methodology and mitigate the risk of misuse.

## Acknowledgments

This work is supported by the National Key Research & Develop Plan (2023YFF0725100) and the National Natural Science Foundation of China (62322214, U23A20299, 62076245, 62072460, 62172424, 62276270). This work is supported by Public Computing Cloud, Renmin University of China.

## References

- Saleh Ashkboos, Maximilian L Croci, Marcelo Genari do Nascimento, Torsten Hoeffler, and James Hensman. 2024. SliceGPT: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. [Layer normalization](#).
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.
- Patrick Chen, Hsiang-Fu Yu, Inderjit Dhillon, and Chojui Hsieh. 2021. Drone: Data-aware low-rank compression for large nlp models. *Advances in neural information processing systems*, 34:29321–29334.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

- 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*.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. [A framework for few-shot language model evaluation](#).
- Isha Garg, Priyadarshini Panda, and Kaushik Roy. 2019. A low effort approach to structured cnn design using pca. *IEEE Access*, 8:1347–1360.
- 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.
- Zixuan Jiang, Jiaqi Gu, Hanqing Zhu, and David Z Pan. 2023. Pre-rmsnorm and pre-crmsnorm transformers: Equivalent and efficient pre-In transformers. *arXiv preprint arXiv:2305.14858*.
- 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<sub>0</sub> 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.
- Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*.
- 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.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.
- 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*.
- Marc Riera, Jose Maria Arnau, and Antonio González. 2022. Dnn pruning with principal component analysis and connection importance estimation. *Journal of Systems Architecture*, 122:102336.
- 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.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.
- 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.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.
- 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 Senrich, 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, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.
- 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.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.
- 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.
- Wei Zhang and Zhiming Wang. 2022. Pca-pruner: Filter pruning by principal component analysis. *Journal of Intelligent & Fuzzy Systems*, 43(4):4803–4813.
- 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,

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

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,



$$M_k = U_{Q,:k}^{(i)\top} U_{K,:k}^{(i)} \quad (20)$$

where,

$$M_k = \arg \min_M \|X^\top W_Q^{(i)\top} W_K^{(i)} X - X^\top W_Q^{(i)\top} M W_K^{(i)} X\|, \quad \text{s.t. } \text{rank}(M) = k \quad (21)$$

Suppose

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

Then, we have

$$\begin{aligned} M_k &= \arg \min_M \|X_Q^\top X_K - X_Q^\top M X_K\| \\ &= \arg \min_M \|V_1^\top X_Q^\top X_K V_2 - V_1^\top X_Q^\top M X_K V_2\| \\ &= \arg \min_M \|\Sigma_1^\top U_1^\top U_2 \Sigma_2 - \Sigma_1^\top U_1^\top M U_2 \Sigma_2\| \end{aligned} \quad (23)$$

Define

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

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

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

Therefore, we have

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

Refer to Eq. 11,

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

we also have

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

Therefore,

$$M_k = U_{Q,:k}^{(i)\top} U_{K,:k}^{(i)} \quad (29)$$

## B Pseudo Code of PCA Projection

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

## C Sparsity

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

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

where  $M$  denotes the total number of parameters of the model.

## D Experiment Setting 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 answering tasks, containing 88K training examples.

### D.2 Experiment Setup

Our SP<sup>3</sup> 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

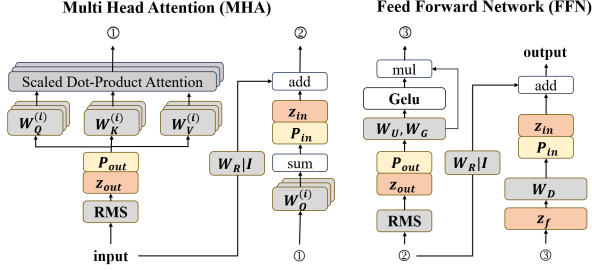


Figure 5: Illustration of the SP<sup>3</sup> architecture for LLM, in which the gray rectangles represent the weight matrices, the yellow rectangles signify the projection matrices, and the red rectangles indicate the masks.

SQuAD, spanning 3 epochs, with a batch size of 32 and learning rates of 1e-5 and 2e-5, respectively. We maintain the default hyperparameters provided by HuggingFace.

The pruning of BERT follows the same configuration as the model fine-tuning process. We first fine-tune SP<sup>3</sup> with the task-specific objective for 2 epochs, then dimension-level pruning begins at the 3rd epoch, followed by the head-level and layer-level pruning at the 8th epoch.

## E More Experimental Results

### E.1 Comparing SP<sup>3</sup> with More Baselines

The comparison results between SP<sup>3</sup> and more baselines are shown in Table 5.

### E.2 Comparing SP<sup>3</sup> with TinyBERT

We skip the data augmentation step in TinyBERT, following the approach of CoFi (Xia et al., 2022), as this data augmentation significantly enlarges the training dataset, leading to a substantial increase in training time. Here, we just present the results of TinyBERT with data augmentation (DA) for reference. The comparison results are shown in Table 6. Even with data augmentation, our proposed SP<sup>3</sup> demonstrates comparable performance.

### E.3 Experiment Results on OPT<sub>125m</sub>

The results of the experiment on OPT<sub>125m</sub> are shown in Table 7. For OPT<sub>125m</sub>, we use the same experimental setup as for BERT<sub>base</sub>.

## F SP<sup>3</sup> for Large Language Model

SP<sup>3</sup> can also be used to prune large language model (LLM), e.g., Llama (Touvron et al., 2023)). To mitigate computational overhead of LLM, our approach restricts using SFT data exclusively for model pruning. Concurrently, to reduce pruning complexities,

we only prune the hidden dimensions and the filters within the FFN block, while ignoring the pruning of attention heads and layers, because coarse-grained pruning requires more training data to recover the performance of the model. We also ignore the pruning of the intermediate dimensions of the MHA block, as it may conflict with the RoPE positional encoding (Su et al., 2024) which is commonly used for LLM. The structure used to prune the LLM is shown in Fig. 5.

### F.1 Group PCA Projection

As shown in section 3.3, to compress the hidden dimensions of the model, we need to add additional matrices to each residual. These additional added matrices increase the total number of parameters in the model by 15%. Fortunately, we find that if we use the same principal components to initialize the compactors of successive blocks, we can avoid introducing additional matrices at the residuals. Taking Llama as an example, for the  $i$ -th block of the Llama model, we have

$$x^{(i)} = \text{Block}^{(i)}(\text{RMSNorm}(x^{(i-1)})) + x^{(i-1)}. \quad (31)$$

After computing the principal component matrix  $U$ , we have

$$\begin{aligned} x^{(i)} &= \text{Block}^{(i)}(\text{RMSNorm}(UU^\top x^{(i-1)})) + x^{(i-1)}, \\ &= \text{Block}^{(i)}(U \cdot \text{RMSNorm}(U^\top x^{(i-1)})) + x^{(i-1)}. \end{aligned} \quad (32)$$

Suppose  $\hat{x}^{(i-1)} = U^\top x^{(i-1)}$ ,  $\hat{x}^{(i)} = U'^\top x^{(i)}$ . Then, we have

$$\hat{x}^{(i)} = U'^\top \text{Block}^{(i)}(U \cdot \text{RMSNorm}(\hat{x}^{(i-1)})) + U'^\top U \hat{x}^{(i-1)}. \quad (34)$$

It can be seen that if we make  $U'$  equal to  $U$ , we don't need to add extra matrices at the residuals, because at this point we have  $U'^\top U = I$ . We refer to this unified computation of the main components of features in several consecutive blocks as Group PCA Projection. For LLM, the matrix introduced when the group size is 8 only increases the number of model parameters by 1%.

It is also worth noting that Group PCA Projection requires models to use RMSNorm with a pre-norm architecture, but fortunately, based on existing work (Jiang et al., 2023), all models that use LayerNorm can be directly converted to use RMSNorm. Moreover, nearly all LLMs use the pre-norm architecture.

## F.2 Experiments

**Setup.** Following existing works on pruning LLM (Xia et al., 2023; Ma et al., 2023), we consider the commonsense reasoning and question-answering tasks including Hellaswag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), ARC-Easy and ARC-Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), and PIQA (Bisk et al., 2020). We evaluate our models using the Language Model Evaluation Harness (Gao et al., 2023) framework. We use the SFT dataset oasst\_top1\_2023-08-25 to prune the TinyLlama-1.1B-Chat-v0.3 (Peiyuan Zhang and Lu, 2023). Without additional declarations, for TinyLlama we use Group PCA Projection with group size 8.

**Results.** The experimental results on TinyLlama are shown in Table 8. We finetune the model on the dataset for 2 epochs with a learning rate of  $1e-5$  and a batch size of 4 to obtain the pruned model. Then, we test the performance of the pruned model under a zero-shot setting. Our method updates both the model parameters and the mask. Since for limited SFT data, training too many epochs may lead to model overfitting, while too few updates can lead to non-convergence of the mask. Therefore we use smaller batch sizes so that convergence of the mask is guaranteed in case the model does not overfit the data.

**Impact of Group Size.** We also explore the effect of using different group sizes in Group PCA Projection on model performance. The experimental results are shown in Table 9. To prevent the interference of additional factors, we only consider the compression of the hidden dimension in this experiment. From the experimental results, it can be seen that using a smaller group size does not improve the performance of the model. On the contrary, a group size of 8 can be utilized without introducing too many additional parameters and performs better than a smaller group size. We speculate that larger group sizes with fewer additional parameters at residuals act as a form of regularization, helping to prevent overfitting of the calibration data by the results of principal component analysis.

**Combining with Existing Pruning Methods.** Our method can also be combined with other pruning methods, e.g., LLM-Pruner (Ma et al., 2023). For these methods, we first initialize the compactors

using PCA projection, after which the compactors are directly fused to the model. Additionally, to maintain compatibility with existing pruning methods and avoid altering the model’s structure, we utilize the same principal component matrix across all layers. In this way,  $SP^3$  can be viewed as a re-representation of the model making it easier to be pruned. In this experiment, we combine  $SP^3$  with LLM-Pruner. The experimental results are shown in Table 10. It can be seen that combining our method can improve the performance of the LLM-pruner.

## G More Ablation Experiments

### G.1 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 experimental results are presented in Table 11. 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.2 Impact of Different Random Seeds

$SP^3$  needs to use the calibration data to initialize all the compactors. To verify the robustness of  $SP^3$  under different calibration data, we conduct experiments using different random seeds to sample the calibration data. We use  $SP^3$  to prune  $BERT_{base}$  and  $OPT_{125m}$  on datasets QNLI, SST-2, MRPC, STS-B, and RTE using a range of random seeds

(0, 1, 2, 3, 4, 42), where  $\text{BERT}_{\text{base}}$  are pruned to 5M and  $\text{OPT}_{125\text{m}}$  are pruned to 7M. The results of the experiment are shown in Table 12. Employing these six random seeds has produced various calibration sets, and our experimental findings indicate that these different calibration sets have a minimal impact on the model’s performance after compression due to their small standard deviation.

## H Structures of Pruned Model

The structure of pruned models on RTE, SST-2, STS-B, MNLI, QNLI, QQP and SQuAD are shown in Fig. 6 and Fig. 7. We show the model structure in terms of dimension-level and head-level sparsity. Rather than directly displaying layer-level sparsity, we indirectly illustrate it through histograms depicting dimension-level and head-level sparsity. If a value at a certain position is 0, it suggests layer-level pruning has taken place at that position.

## I Difference Between $\text{SP}^3$ and Low-rank Factorization Methods

Existing low-rank factorization methods (Garg et al., 2019; Liu and Ng, 2022; Ma et al., 2019; Riera et al., 2022; Yin et al., 2022; Zhou et al., 2019; Zhang and Wang, 2022) primarily focus on decomposing the weight matrix using PCA. In contrast, our approach applies PCA to the features generated during the model inference process and re-represents these features based on the principal components. This feature representation enhances compression efficacy and offers a distinct perspective from traditional weight matrix decomposition using PCA.

SliceGPT (Ashkboos et al., 2024) stands out as the most pertinent work to our research, as it focuses on model compression via principal component analysis of the feature matrix. However, notable distinctions exist between our approach and SliceGPT. Firstly, SliceGPT overlooks the variability in principal component distributions across layers, opting for a uniform compression rate of hidden dimension  $d$  across all layers, while our approach adaptively determines the compression rate for different layers through learnable masks. Secondly, it neglects strategies for eliminating parameters introduced by residuals, which affects the compression rate of the compressed model. To address this problem, we instead propose the Group PCA Projection approach to reduce the number of parameters added at the residuals. Lastly, SliceGPT

exclusively addresses decoder-only model architectures, omitting consideration for other model configurations. In contrast, we consider models with a decoder architecture, such as Llama, and OPT, as well as models with an encoder architecture, such as BERT.



---

**Algorithm 1: PCA Projection**


---

**Input:**  $T$ : sampled tokens size,  $\mathbb{D}$ : calibration data,  $L^{(1 \sim N)}$ : model

**Output:** Projection Matrix  $P_{\text{in}}, P_{\text{out}}, P_Q, P_K, P_V, P_O$

$X_Q, X_K, X_V, X_M, X_F \leftarrow \square, \square, \square, \square, \square$

**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**

$X_Q^{(h,i)} \leftarrow \text{Concate}(X_Q^{(h,i)}, W_Q^{(h)} x^{(i-1)})$

$X_K^{(h,i)} \leftarrow \text{Concate}(X_K^{(h,i)}, W_K^{(h)} x^{(i-1)})$

$X_V^{(h,i)} \leftarrow \text{Concate}(X_V^{(h,i)}, W_V^{(h)} x^{(i-1)})$

**end**

$x_M = \text{MHA}^{(i)}(x^{(i-1)}) + x^{(i-1)}$

$x_M^{\text{LN}} = \text{LN}(x_M)$

$x_F = \text{FFN}^{(i)}(x_M) + x_M$

$x_F^{\text{LN}} = \text{LN}(x_F)$

$x^{(i)} = x_F^{\text{LN}}$

$X_M^{(i)} \leftarrow \text{Concate}(X_M^{(i)}, x_M)$

$X_F^{(i)} \leftarrow \text{Concate}(X_F^{(i)}, x_F)$

**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**

$X_V^{(h,i)} \leftarrow \text{Sample}(X_V^{(h,i)}, T)$

$X_Q^{(h,i)} \leftarrow \text{Sample}(X_Q^{(h,i)}, T)$

$X_K^{(h,i)} \leftarrow \text{Sample}(X_K^{(h,i)}, T)$

$U_V, \Sigma_V, V_V^\top \leftarrow \text{SVD}(X_V^{(h,i)})$

$U_1, \Sigma_1, V_1^\top \leftarrow \text{SVD}(X_Q^{(h,i)})$

$U_2, \Sigma_2, V_2^\top \leftarrow \text{SVD}(X_K^{(h,i)})$

$Z = \Sigma_1^\top U_1^\top U_2 \Sigma_2$

$U_Z, \Sigma_Z, V_Z^\top \leftarrow \text{SVD}(Z)$

$U_Q = \Sigma_Z^{\frac{1}{2}} U_Z^\top \Sigma_1^{-1} U_1^\top$

$U_K = \Sigma_Z^{\frac{1}{2}} V_Z \Sigma_2^{-1} U_2^\top$

$P_V^{(h,i)}, P_O^{(h,i)} \leftarrow U_V^\top, U_V$

$P_Q^{(h,i)}, P_K^{(h,i)} \leftarrow U_Q, U_K$

**end**

$[\gamma^M, \gamma^F] \leftarrow L^{(i)}$

$R \leftarrow I - \frac{1}{d} 11^\top$

$X_M^{(i)} \leftarrow \text{Sample}(X_M^{(i)}, T)$

$X_F^{(i)} \leftarrow \text{Sample}(X_F^{(i)}, T)$

$U_M, \Sigma_M, V_M^\top \leftarrow \text{SVD}(X_M^{(i)})$

$U_F, \Sigma_F, V_F^\top \leftarrow \text{SVD}(X_F^{(i)})$

$P_{\text{in}}^{(M,i)}, P_{\text{out}}^{(M,i)} \leftarrow U_M^\top R, \text{diag}(\gamma^M) U_M$

$P_{\text{in}}^{(F,i)}, P_{\text{out}}^{(F,i)} \leftarrow U_F^\top R, \text{diag}(\gamma^F) U_F$

**end**

---

	Params.	SST-2	QNLI	MNLI	QQP	RTE	STS-B	MRPC	SQuAD
BERT <sub>base</sub>	110M	93.1	91.5	84.8	91.2	70.4	89.1	85.6	88.4
SP <sup>3</sup>	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<sup>3</sup> and other distillation and pruning methods on BERT<sub>base</sub>.

	SST-2	QNLI	MRPC	RTE
TinyBERT w/ DA	<b>91.6</b>	<b>87.6</b>	<b>83.6</b>	62.5
TinyBERT w/o DA	89.7	86.7	81.4	63.2
SP <sup>3</sup>	91.4	<b>87.6</b>	82.8	<b>66.4</b>

Table 6: Comparison between our SP<sup>3</sup> on BERT<sub>base</sub> and TinyBERT, where DA means data augmentation.

	Params.	SST-2	QNLI	QQP	RTE	STS-B	MRPC
OPT <sub>125m</sub>	85M	92.9	90.8	90.1	66.4	87.0	82.4
CoFi*	7M	86.8	84.4	<b>88.5</b>	55.8	70.6	69.2
SP <sup>3</sup>	7M	<b>89.9</b>	<b>86.7</b>	87.8	<b>60.3</b>	<b>75.6</b>	<b>77.2</b>

Table 7: Pruning performance of SP<sup>3</sup> on OPT<sub>125m</sub>, where CoFi\* means that CoFi on OPT<sub>125m</sub> is implemented by ourselves.

	Params.	ARC-e	ARC-c	BoolQ	HellaSwag	OBQA	PIQA	WinoGrande
TinyLlaMa	1.1B	57.5	27.4	59.2	44.0	23.8	70.8	56.0
SP <sup>3</sup>	0.85B	51.6	25.9	50.5	37.5	20.4	67.3	54.3
	0.65B	44.9	22.3	61.0	33.0	17.2	61.3	53.7

Table 8: Pruning performance of SP<sup>3</sup> on TinyLlaMa.

	Params.	ARC-e	ARC-c	BoolQ	HellaSwag	OBQA	PIQA	WinoGrande
SP <sup>3</sup> -gs1	0.85B + 0.10B	38.5	<b>23.6</b>	51.0	30.0	14.8	57.9	50.8
SP <sup>3</sup> -gs2	0.85B + 0.05B	42.0	21.7	52.1	31.7	17.0	60.2	<b>55.6</b>
SP <sup>3</sup> -gs4	0.85B + 0.03B	44.2	23.4	58.3	33.0	<b>18.4</b>	61.5	52.6
SP <sup>3</sup> -gs8	0.85B + 0.01B	<b>46.0</b>	<b>23.6</b>	<b>59.0</b>	<b>33.6</b>	17.6	<b>62.0</b>	54.0

Table 9: Pruning performance of SP<sup>3</sup> on TinyLlaMa under different group sizes (gs).

	Params.	ARC-e	ARC-c	BoolQ	HellaSwag	OBQA	PIQA	WinoGrande
LLM-Pruner	0.85B	46.2	23.1	45.2	35.4	16.8	65.6	<b>53.7</b>
LLM-Pruner + SP <sup>3</sup>	0.85B	<b>47.3</b>	<b>23.3</b>	<b>48.4</b>	<b>35.9</b>	<b>17.0</b>	<b>68.0</b>	52.0

Table 10: Pruning performance of combining SP<sup>3</sup> and LLM-Pruner on TinyLlama.

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

Table 11: Ablation studies of different levels of masks on datasets SST-2, RTE, STS-B, and MRPC using BERT<sub>base</sub>.

	Random seed	0	1	2	3	4	42	Mean	Std
BERT <sub>base</sub>	MPRC	82.8	81.3	81.8	83.8	83.1	82.8	82.6	0.82
	STS-B	86.0	86.6	85.7	85.9	86.1	86.1	86.1	0.29
	RTE	64.2	62.5	62.8	64.2	63.2	66.1	63.8	1.2
	QNLI	86.9	86.9	87.3	87.1	86.9	87.6	87.1	0.26
	SST-2	91.1	91.3	91.3	90.8	90.7	91.4	91.1	0.26
OPT <sub>125m</sub>	MPRC	78.9	78.4	78.1	76.9	76.7	77.2	77.7	0.81
	STS-B	75.9	75.4	75.6	75.6	75.6	75.6	75.6	0.15
	RTE	57.7	58.6	57.8	59.6	57.5	60.3	58.6	1.0

Table 12: Pruning performance on datasets QNLI, SST-2, MRPC, STS-B, and RTE via the calibration data sampled by various random seeds.

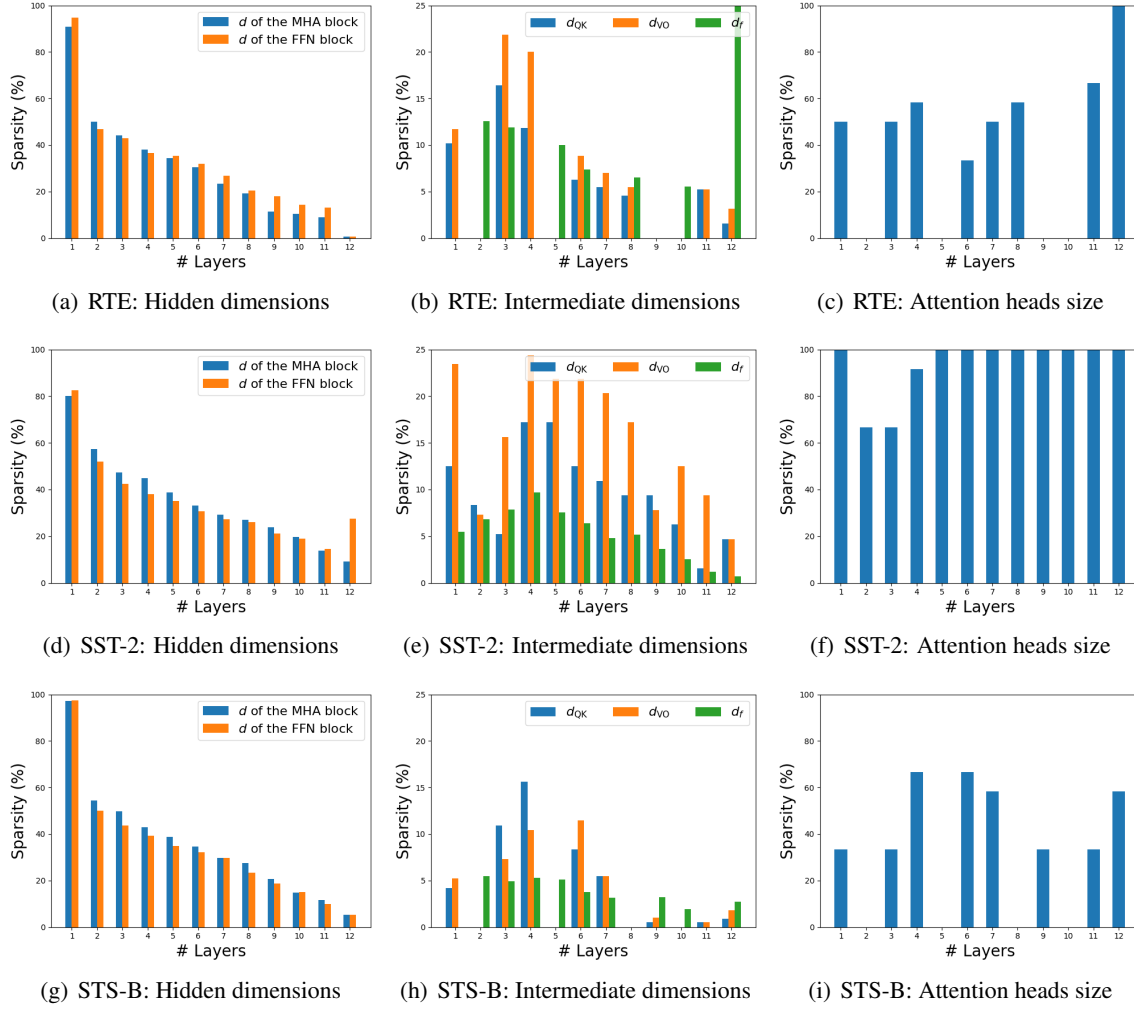


Figure 6: Pruned model structures on RTE, SST-2 and STS-B datasets



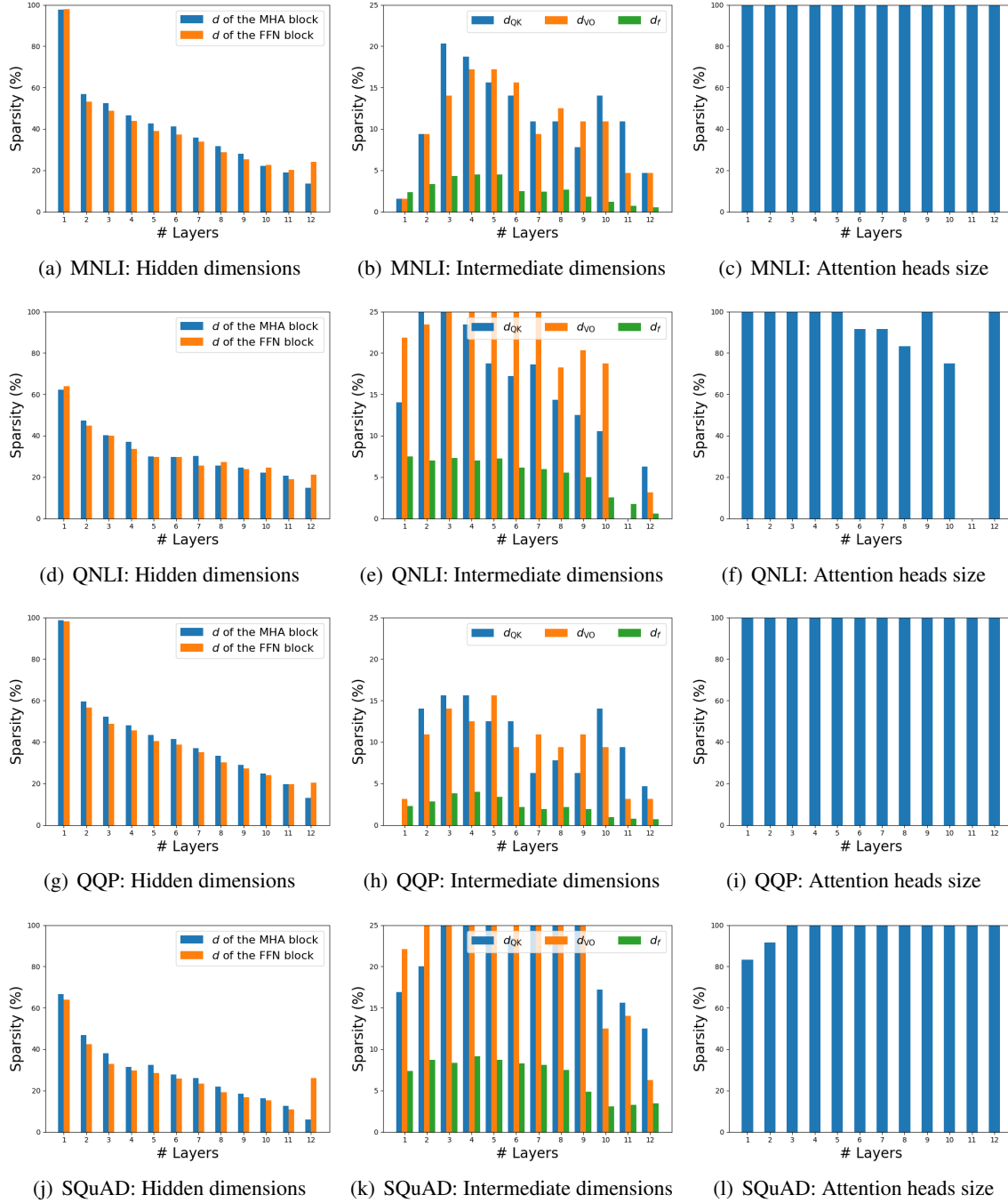


Figure 7: Pruned model structures on MNLI, QNLI, QQP and SQuAD datasets