TRANSPA: TOWARDS EFFICIENT STRUCTURED SPARSE TRAINING FOR TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers have emerged as the backbone neural network architecture in today's AI applications. Due to their high complexity, sparsifying transformers, at both pre-training and fine-tuning stages, is very attractive for lower the training and inference costs. In this paper, we propose TranSpa, an efficient structured sparse training approach for language and vision transformers. Unlike prior works focusing on individual building blocks, TranSpa fully considers the correlation between the weight matrices and their component rows/columns, and performs the coupled estimation and coupled sparsification. To achieve that, TranSpa introduces the use of new granularity when calibrating the importance of structural components in the transformer and removing the insignificant parts. Evaluations across different models, in both pre-training and fine-tuning scenarios, demonstrate the effectiveness of the proposed approach. TranSpa can bring $1.6\times$ size reduction with 0.6 lower perplexity when training GPT-2 model from scratch. It also enables $1.6\times$ training speedup over the existing sparse pre-training method. For training sparse LLaMA-1B from scratch, our approach reduces GPU memory usage by 50%, decreases training time by 21%, and achieves a $1.6\times$ speedup in inference throughput while maintaining model performance. Experiments of applying TranSpa for fine-tuning tasks also show significant performance improvement with respect to model accuracy and pruning cost reduction.

1 INTRODUCTION

Thanks to their excellent performance for sequence modeling and scalability, transformers Vaswani et al. (2017) have served as the backbone neural network architecture across various important domains, such as natural language processing (NLP) Vaswani et al. (2017); Devlin et al. (2019); Wang et al. (2019a); OpenAI et al. (2023); Touvron et al. (2023), computer vision Dosovitskiy et al. (2020); Liu et al. (2021); Carion et al. (2020) and speech processing Dong et al. (2018); Gulati et al. (2020); Hsu et al. (2021). However, despite their unprecedented popularity, modern transformers suffer from high model complexity, causing expensive costs in both training and inference phases.

To alleviate these challenging issues, **sparse training**, a strategy that originated for optimizing convolutional neural networks (CNNs), has emerged as an attractive solution for efficient transformers. In general, given an input model θ, sparse training aims to output a sparse model θ_s with high task performance. As illustrated in Fig. 1, when θ is randomly initialized, sparse training serves as an efficient *pre-training* method that can reduce the computational and memory costs of the expensive training-from-scratch process. When θ is a pre-trained high-performance dense model, sparse training is essentially the well-known pruning technique, which imposes the sparsity on θ in the *fine-tuning* process and trims down the inference cost.

Although sparse training has been well studied for slimming CNN models, the fundamentally different mechanism of transformers, *e.g.*, multi-head self-attention, make the existing CNN-oriented solutions not suitable for transformers. To date, the efficient sparse transformer training is still under-explored. More specifically, 1) for applying sparse training at the pre-training stage, Frankle & Carbin (2018), Chen et al. (2021a;b) propose to train sparse vision and NLP transformers from scratch, with focus on exploring head-level sparsity. Also Dao et al. (2022) proposes to use specially structured matrix for sparse transformer training; 2) for applying sparse training at the fine-tuning stage, Ma et al. (2023); Chen et al. (2021b); Yu et al. (2022) explore the structured sparsity at head

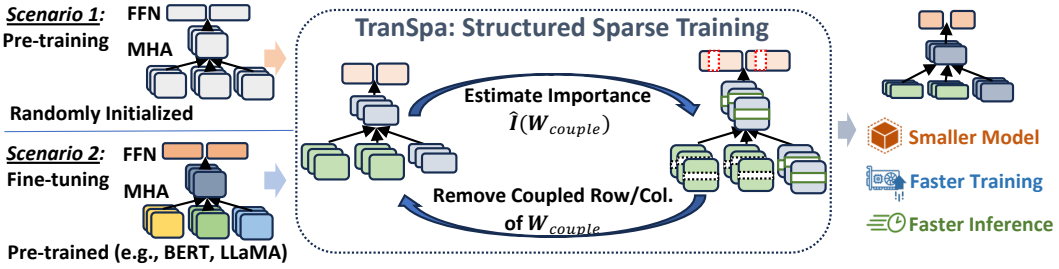

Figure 1: Our proposed TranSpa is a structured sparse training approach, applicable for both pre-training and fine-tuning stages.

levels. Observing the potential layer-wise redundancy, Chen et al. (2024); Men et al. (2024) propose to remove unimportant layers in the transformer architecture. In addition, Ashkboos et al. (2024); van der Ouderaa et al. (2023) use finer granularity at row and column level for structured pruning, achieving good compression performance.

Technical Contributions. In this paper, we propose to perform structured sparse transformer training [1] from a new perspective. Unlike prior works focusing on evaluating the importance of individual components of transformer architecture, *e.g.*, a weight matrix or its component row/column, we propose to fully consider the inter-matrix and inter-row/column correlation, leading to new mechanism for importance assessment and structural removal. More specifically, at the *coupled estimation* step, we introduce a new granularity, namely, coupled weight matrices, to calibrate the architectural importance of transformer model, providing more fine-grained measurement via exploring the inherent coupling effects of weight matrices. Then, at the *coupled sparsification* step, for the row and column within those unimportant coupled matrices, we propose to rank and remove them in a coupled way, maximally preserving the inherent inter-row/column correlation in the weight matrix.

We apply our approach, namely TranSpa, in both pre-training and fine-tuning scenarios. Evaluation across different model types (*e.g.*, vision transformers, and large language model (LLMs)) show that TranSpa brings significant performance improvement with respect to training speed, model accuracy, and cost reduction. For instance, when training the GPT-2 model from scratch, our approach can bring $1.6\times$ size reduction with 0.6 lower perplexity (PPL). It also brings $1.6\times$ training speedup compared to the existing sparse pre-training methods. For pre-training sparse LLaMA-1B from scratch, our approach cuts GPU memory usage by half, shortens training time by 21%, and boosts inference throughput by $1.6\times$. For training sparse DeiT from scratch, TranSpa brings 0.8% accuracy increase over the state-of-the-art solutions. Experiments on pruning LLMs also demonstrate the effectiveness of our approach.

2 Background

2.1 Preliminaries

Notation. We represent tensors using boldface calligraphic script, denoted as $\boldsymbol{\mathcal{X}}$. Matrices and vectors are indicated with boldface capital and lowercase letters, such as $\boldsymbol{X}$ and $\boldsymbol{x}$, respectively. Furthermore, non-boldface letters with indices, *e.g.*, $\mathcal{X}(i_1, \cdots, i_d)$, $X(i, j)$, and $x(i)$, denote the entries for a d-dimensional tensor $\boldsymbol{\mathcal{X}}$, a matrix $\boldsymbol{X}$, and a vector $\boldsymbol{x}$, respectively.

Transformer. For Transformer-based models, the key components include the Multi-Head Attention (MHA) and Feed-Forward Network (FFN). More specifcally, the MHA operation is defined as follows:

$$\mathrm{MHA}(\boldsymbol{X}_Q, \boldsymbol{X}_K, \boldsymbol{X}_V) = \mathrm{Concat}(head_1, \ldots, head_h)\boldsymbol{W}^O, \tag{1}$$

[1] In this paper we focus on structured sparsity, which can bring measured speedup on off-the-shelf hardware; while the unstructured sparsity typically cannot. Though some Nvidia GPUs now provide hardware support for semi-structured 2:4 sparsity, this feature is only available for the high-end GPUs such as A100 and up, limiting its practice in many budget-limited resource-limited applications.

where $\boldsymbol{X}_Q, \boldsymbol{X}_K, \boldsymbol{X}_V \in \mathbb{R}^{l \times e_m}$ are the length-l input sequences, e_m is the embedding dimension, and h is the number of attention heads. Here each attention head $head_i$ operates as below:

$$head_i = \text{Attention}(\boldsymbol{X}_Q \boldsymbol{W}_i^Q,\ \boldsymbol{X}_K \boldsymbol{W}_i^K,\ \boldsymbol{X}_V \boldsymbol{W}_i^V) = \text{softmax}\left(\frac{\boldsymbol{X}_Q \boldsymbol{W}_i^Q (\boldsymbol{X}_K \boldsymbol{W}_i^K)^\top}{\sqrt{e}}\right) \boldsymbol{X}_V \boldsymbol{W}_i^V, \tag{2}$$

where $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V \in \mathbb{R}^{e_m \times e}$, $\boldsymbol{W}^O \in \mathbb{R}^{e_m \times e_m}$, and $e_m = e \times h$. The FFN consists of two fully connected layers with a Gaussian Error Linear Unit (GELU) activation function applied in between. Let $\boldsymbol{X}$ represent the input embeddings, and $\boldsymbol{W}_{in} \in \mathbb{R}^{e_m \times d}, \boldsymbol{W}_{out} \in \mathbb{R}^{d \times e_m}, \boldsymbol{b}_{in}, \boldsymbol{b}_{out}$ denote the weight matrices and bias vectors, respectively. The operation of FFN can be then defined as follows:

$$\text{FFN}(\boldsymbol{X}) = \text{GELU}(\boldsymbol{X}\boldsymbol{W}_{in} + \boldsymbol{b}_{in})\boldsymbol{W}_{out} + \boldsymbol{b}_{out}. \tag{3}$$

2.2 Related Works

Sparse Training for Transformer Pre-training. Applying sparse training at the pre-training stage is very attractive for reducing the high complexity of the costly train-from-scratch process. However, unlike the well-studied sparse CNN pre-training, to date sparse transformer pre-training is still under-explored. Chen et al. (2021a) dynamically extracts and trains sparse sub-networks, either in unstructured or structured ways, thereby alleviating the training memory bottleneck for Vision Transformers. Inspired by the Lottery Ticket Hypothesis originated in CNN, Chen et al. (2021b) performs early detection of the structured winning lottery tickets for transformers, improving the pre-training efficiency. In Dao et al. (2022), a class of structured matrices, which can closely approximate the dense weight matrices, are proposed for hardware-efficient sparse pre-training, accelerating the overall training process. Notice that the methods developed in Chen et al. (2021b;a); Dao et al. (2022) can also be applied at the fine-tuning stage for transformer pruning.

Sparse Training for LLM Pruning. Due to the high costs of pre-trained LLM models, using sparse training to trim down LLM size is a promising solution for efficient deployment. In general, pruning LLM can be performed in either unstructured or structured way. *Unstructured pruning* Frantar & Alistarh (2023); Sun et al. (2023); Zhang et al. (2023); Sun et al. (2023); Xia et al. (2023); Malla et al. (2024) focus on removing unimportant individual weights, achieving high compression performance but limited computational efficiency due to the unstructured sparsity pattern. *Structured pruning* Ma et al. (2023); Xia et al. (2023); Ashkboos et al. (2024); An et al. (2024); Chen et al. (2023b;a); Zhao et al. (2024a); Yang et al. (2024) aims to sparsify some building components of transformer architecture, *e.g.*, head, row, layer, *etc.*, offering wall-clock time inference speedup. To guide the guide the selection of elements to be removed, typical pruning metrics include magnitude-based Sun et al. (2023); An et al. (2024) and loss-based Ma et al. (2023); van der Ouderaa et al. (2023). Magnitude-based methods use the absolute values of weights, whereas loss-based approaches assess the impact of pruning on model loss, often using the gradient information obtained from Taylor expansion.

Sparse Training for CNN Pre-training/Pruning. Sparse training for CNN, at pre-training and fine-tuning stages, has been well-studied in the literature. For efficient sparse CNN pre-training, Mocanu et al. (2018) explores to prune and grow the same amount of weights periodically and iteratively. Mostafa & Wang (2019) proposes to automatically adjust the sparsity levels, achieving good scalability and high computational efficiency. Evci et al. (2020) uses the gradient-based criteria to grow the weights with Erdos-Renyi Kernel initialization, and a memory-economic scheme is proposed in Yuan et al. (2021) for training on the edge devices. Recently, Chen et al. (2023c) proposes to automatically sparsify the CNN model from scratch, obtaining a compact model without iterative fine-tuning. On the other hand, CNN pruning can be roughly categorized to unstructured pruning (for weights) Han et al. (2015a;b) and structured pruning (for channels/filters) He et al. (2019); Wang et al. (2019b); Yvinec et al. (2021); Hou et al. (2022); Lin et al. (2020); Dubey et al. (2018); Sui et al. (2021); Tan & Motani (2020); Luo et al. (2017). Considering the importance of structured sparsity for inference speedup, most of the state-of-the-art CNN pruning works focus on structured pruning.

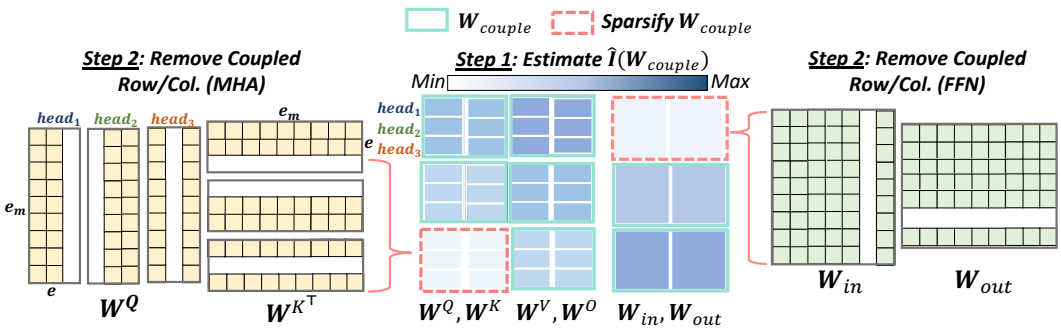

Figure 2: Key steps of TranSpa: 1) Estimate the importance of coupled weight matrix $\boldsymbol{W}_{couple}$; and 2) Remove the coupled row/column pair in unimportant $\boldsymbol{W}_{couple}$.

3 METHOD

Fig. 2 shows the overall procedure of TranSpa, which consists of two key steps. **(1) Coupled Estimation** (Section 3.1). This step assesses the importance of the building components of transformers, using our proposed new calibration granularity at the coupled weight matrix level. **(2) Coupled Sparsification** (Section 3.2). Once the unimportant coupled weight matrices are identified, TranSpa further gauges the fine-grained importance of the coupled row/column pairs in the component matrices and discards the insignificant ones.

3.1 COUPLED ESTIMATION: COUPLED WEIGHT MATRIX-WISE IMPORTANCE CALIBRATION

As extensively studied in the literature Frantar & Alistarh (2023); Sun et al. (2023); van der Ouderaa et al. (2023), estimating the unimportant components of neural networks plays a crucial role for model sparsification. In general, importance estimation can be performed at different granularity levels, such as weight, neuron, layer, and head. More specifically, when aiming for obtaining the structured sparse transformers, identifying the insignificant heads and layers is the most common practice Chen et al. (2021a;b); Men et al. (2024); Ma et al. (2023).

Unlike existing works, we propose to assess the importance of structural components within transformers using a new granularity, namely *coupled weight matrix*. This idea is motivated by the observation that the transformer has its unique computing pattern and model topology, and hence coupling the weight matrices can provide rich information for importance estimation. Next, we describe the details of our proposal.

Coupled Weight Matrix in MHA. Recall that Eq. 1 and Eq. 2 depict the computations of MHA, which consists of four types of weight matrices $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, $\boldsymbol{W}^V$ and $\boldsymbol{W}^O$. We propose to estimate the structural importance of MHA by analyzing the combined effect of these matrices. More specifically, consider the following mathematical reformulation of Eq. 2:

$$\text{MHA}(\boldsymbol{X}_Q, \boldsymbol{X}_K, \boldsymbol{X}_V) = \sum_{i=1}^{h} head_i \boldsymbol{W}_i^O = \sum_{i=1}^{h} \text{softmax}(\frac{\boldsymbol{X}_Q(\boldsymbol{W}_i{}^Q \boldsymbol{W}_i^{K\top})\boldsymbol{X}_K{}^\top}{\sqrt{e}} \boldsymbol{X}_V(\boldsymbol{W}_i{}^V \boldsymbol{W}_i{}^O), \tag{4}$$

where $\boldsymbol{W}_i^O \in \mathbb{R}^{e \times e_m}$ and $\boldsymbol{W}^O = \text{Concat}(\boldsymbol{W}_1^O, \ldots, \boldsymbol{W}_h^O)$. It is seen that $\boldsymbol{W}_i^{QK} = \boldsymbol{W}_i^Q \boldsymbol{W}_i^{K\top}$ and $\boldsymbol{W}_i^{VO} = \boldsymbol{W}_i^V \boldsymbol{W}_i^O$, as the combination of two weight matrices, can serve as the structural components in MHA. Following this perspective, we can then set the granularity of importance estimation at the level of those coupled weight matrices. We believe this strategy brings two benefits: *i*) it provides more fine-grained measurement than the commonly adopted head-level calibration; and *ii*) it meanwhile naturally explores the inter-matrix correlation within the attention heads, avoiding the limitations if only focusing on individual weight matrices.

Coupled Weight Matrix in FFN. Following the same philosophy, we also evaluate the importance of the building blocks in FFN from the lens of the coupled weight matrix. More specifically, consider

the FFN computation (Eq. 3) can be reformulated as follows:

$$\begin{aligned}\text{FFN}(\boldsymbol{X}) &= 0.5\boldsymbol{Y}_1 \odot (1 + \text{erf}(\boldsymbol{Y}_1/\sqrt{2})\boldsymbol{W}_{out} \\ &= 0.5\boldsymbol{X}\boldsymbol{W}_{in}\boldsymbol{W}_{out} + 0.5\text{erf}(\boldsymbol{Y}_1/\sqrt{2}) \odot \boldsymbol{X}\boldsymbol{W}_{in}\boldsymbol{W}_{out},\end{aligned} \tag{5}$$

where $\boldsymbol{Y}_1 = \boldsymbol{X}\boldsymbol{W}_{in}$, $\odot$ is the Hadamard product, and $\text{erf}(\cdot)$ denotes the error function as $\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt$ bounded by $[-1, 1]$. From Eq. 5 it is seen that the coupled weight matrix $\boldsymbol{W}_{12} = \boldsymbol{W}_{in}\boldsymbol{W}_{out}$ contributes a significant portion of the overall computation. Therefore, we propose to calibrate the importance of this combined matrix, as a good proxy, to identify the significance of the different structural components of FFN.

Coupled Weight Matrix-wise Importance. After setting the resolution of the importance estimator with the proposed granularity, we can gauge the structural criticality of MHA and FFN accordingly. To that end, we calibrate the empirical Fisher information Kunstner et al. (2019) in a coupled weight matrix-wise way as follows:

$$\hat{I}(\boldsymbol{W}_{couple}) = \frac{1}{2}\sum_{k=1}^{2}\frac{1}{|\boldsymbol{W}_k|}\frac{1}{|\boldsymbol{\mathcal{D}}|}\sum_{i,j}(\frac{\partial\mathcal{L}(\boldsymbol{W}_k;\boldsymbol{\mathcal{D}})}{\partial\boldsymbol{W}_k})^2_{i,j}, \text{where } \boldsymbol{W}_{couple} = \boldsymbol{W}_1\boldsymbol{W}_2. \tag{6}$$

Here for MHA, $\{\boldsymbol{W}_1, \boldsymbol{W}_2\}$ is $\{\boldsymbol{W}_i^Q, \boldsymbol{W}_i^{K^\top}\}$ and $\{\boldsymbol{W}_i^V, \boldsymbol{W}_i^O\}$; while for FFN, $\{\boldsymbol{W}_1, \boldsymbol{W}_2\}$ is $\{\boldsymbol{W}_{in}, \boldsymbol{W}_{out}\}$. Also, $\boldsymbol{\mathcal{D}}$ is the training data and $|\cdot|$ returns the size of the operand. Notice that different from prior works Hsu et al. (2022); Sung et al. (2021), the empirical Fisher information is calculated for the entire coupled weight matrix $\boldsymbol{W}_{couple}$ instead of its entries. Smaller $\hat{I}(\boldsymbol{W}_{couple})$ indicates the lower importance of $\boldsymbol{W}_{couple}$ when sparsifying the model.

3.2 Coupled Sparsification: Removing the Coupled Row/Column Pair

Upon obtaining the importance information of all the coupled weight matrices, the next step is to sparsify $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ belonging to those less significant $\boldsymbol{W}_{couple}$. To that end, we perform row/column-wise sparsification, a strategy with finer granularity than removing the entire $\boldsymbol{W}_i$, towards minimizing performance loss and preserving model structuredness. Notice that though removing the rows/columns in the weight matrices of transformers has been reported in Chen et al. (2021b;a); van der Ouderaa et al. (2023), we believe those individual weight matrix-oriented methods are not the best suited in the scenario involved with coupled weight matrices. In other words, the coupling effect between $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ should be fully considered and leveraged in the sparsification process. Aiming at that, we propose coupled sparsification, a solution that removes the coupled row/column pairs in $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, with details described as below.

Inspiration from tSVD. The key idea of coupled sparsification is inspired by the philosophy of truncated singular value decomposition (tSVD). More specifically, recall that a matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ can be exactly factorized to two matrices via SVD as:

$$\begin{aligned}\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top &= \begin{pmatrix}\boldsymbol{U}_{m,r} & \boldsymbol{U}_{m,(m-r)}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Sigma}_{r,r} & \\ & \boldsymbol{\Sigma}_{(m-r),(n-r)}\end{pmatrix}\begin{pmatrix}\boldsymbol{V}_{r,n}^\top \\ \boldsymbol{V}_{(n-r),n}^\top\end{pmatrix} \\ &= \begin{pmatrix}\boldsymbol{U}_{m,r}\boldsymbol{\Sigma}_{r,r}^{1/2} & \boldsymbol{U}_{m,(m-r)}\boldsymbol{\Sigma}_{(m-r),(n-r)}^{1/2}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Sigma}_{r,r}^{1/2}\boldsymbol{V}_{r,n}^\top \\ \boldsymbol{\Sigma}_{(m-r),(n-r)}^{1/2}\boldsymbol{V}_{(n-r),n}^\top\end{pmatrix} = \boldsymbol{M}_1\boldsymbol{M}_2,\end{aligned} \tag{7}$$

where $\boldsymbol{\Sigma}_{r,r} \in \mathbb{R}^{r\times r}$ is a diagonal matrix containing r largest singular values σ_i's and $\boldsymbol{\Sigma}_{(m-r),(n-r)} \in \mathbb{R}^{(m-r)\times(n-r)}$ is a diagonal rectangular matrix containing the smallest $(\min(m, n) - r)$ σ_i's. Then, consider a rank-r tSVD Wall et al. (2003) for approximating $\boldsymbol{M}$ is performed as:

$$\boldsymbol{M} \approx (\boldsymbol{U}_{m,r}\boldsymbol{\Sigma}_{r,r}^{1/2})(\boldsymbol{\Sigma}_{r,r}^{1/2}\boldsymbol{V}_{r,n}^\top) = (\boldsymbol{M}_1 \odot \boldsymbol{S}_{mask1})(\boldsymbol{M}_2 \odot \boldsymbol{S}_{mask2}). \tag{8}$$

Notice that here $\boldsymbol{S}_{maski}$ is the binary matrix that essentially removes a sets of rows/columns of $\boldsymbol{M}_i$. Meanwhile, it is theoretically proven that tSVD provides the *optimal* rank-r approximation for $\boldsymbol{M}$ Wall et al. (2003). Therefore, the row/column-wise sparsification policy for $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ in tSVD, by its nature, brings important insights for sparsifying two component matrices with closely approximating their combination. More specifically, comparing Eq. 7 and Eq. 8, we can see that the

removed rows/columns are $\boldsymbol{U}_{m,(m-r)}\boldsymbol{\Sigma}^{1/2}_{(m-r),(n-r)}$ and $\boldsymbol{\Sigma}^{1/2}_{(m-r),(n-r)}\boldsymbol{V}^{\top}_{(n-r),n}$, which exhibit the following characteristics:

Observation #1 (Small ℓ_1 Norm) *The removed rows/columns of $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ typically have smaller ℓ_1 norm, as $\boldsymbol{\Sigma}_{(m-r),(n-r)}$ only contains very least significant σ_i's.*

Observation #2 (Aligned Removal) *The removed rows in $\boldsymbol{M}_1$ and columns in $\boldsymbol{M}_2$ always have the same indices, exhibiting the positional alignment of the sparsification.*

Sparsifying Coupled Weight Matrices. Considering the mathematical format of Eq. 8 is highly similar to our desired task of sparsifying the coupled $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, a natural idea is directly performing tSVD (Eq. 8) on $\boldsymbol{W} = \boldsymbol{W}_1\boldsymbol{W}_2$ to obtain sparse $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. However, this strategy is mathematically infeasbile because Eq. 7 is generally not invertible, *e.g.*, we cannot use $\mathrm{SVD}(\boldsymbol{W}_1\boldsymbol{W}_2)$ to reconstruct $\boldsymbol{W}_1$ or $\boldsymbol{W}_2$. Fortunately, we can still leverage the observations extracted from tSVD process to design the sparsification policy for $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. More specifically, we propose to calculate the importance scores of the row/column vectors of $\boldsymbol{W}_i$:

$$\boldsymbol{\upsilon} = [\upsilon_1, \upsilon_2, \ldots, \upsilon_d] = \frac{\|\mathrm{vec}(\boldsymbol{W}_1)\|_{1,col} \odot \|\mathrm{vec}(\boldsymbol{W}_2)\|_{1,row}}{\|\|\mathrm{vec}(\boldsymbol{W}_1)\|_{1,col} \odot \|\mathrm{vec}(\boldsymbol{W}_2)\|_{1,row}\|_1}, \tag{9}$$

where $\|\mathrm{vec}(\cdot)\|_{1,col}$ and $\|\mathrm{vec}(\cdot)\|_{1,row}$ represent the column-wise and row-wise ℓ_1 norm of a matrix, respectively, *e.g.*, $\|\mathrm{vec}(\boldsymbol{W}_1)\|_{1,col} = \{\sum_{i=1}^{m} |\boldsymbol{W}_1(i,j)|, (j = 1, 2, \ldots, d)\}$. And $\|\mathrm{vec}(\cdot)\|_1$ calculates the ℓ_1 norm of a vector. From Eq. 9 it is seen that υ_i essentially calculates the normalized combined ℓ_1 for the i-th column of $\boldsymbol{W}_1$ and the i-th row of $\boldsymbol{W}_2$, reflecting the two key observations we obtain from tSVD. Hence we can rank υ_i's to determine the important coupled row/column in $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, and then remove those insignificant pairs.

Overall Sparse Training Procedure. Algorithm 1 describes the processing scheme of TranSpa. Here in each epoch, after identifying top-K least significant W_{couple}, the importance scores of the rows/columns in those component weight matrices are calculated. Once the cumulative important scores exceed the threshold θ, the pair of rows and columns corresponding to the smallest score are removed. This gradual sparsification process continues till the overall model reaches the target budget size.

Algorithm 1 Processing Scheme of TranSpa

Input: Random Initialized/Pre-trained model $\boldsymbol{\mathcal{W}}$, Dataset $\boldsymbol{\mathcal{D}}$,
top-K ratio, Target model size c, Cumulative threshold θ
Output: Sparse Model $\hat{\boldsymbol{\mathcal{W}}}$
for t *in* $[1, 2, \cdots, T]$ **do**
  Compute: Loss of $\boldsymbol{\mathcal{W}}$ on dataset $\boldsymbol{\mathcal{D}}$
  Update: $\boldsymbol{\mathcal{W}}_t = \boldsymbol{\mathcal{W}}_{t-1} + \delta\boldsymbol{\mathcal{W}}$
  if *Param*($\boldsymbol{\mathcal{W}}_t$) $> c$ **then**
    Compute: $\hat{I}(\boldsymbol{W}_{couple}), \forall \boldsymbol{W}_{couple} \in \boldsymbol{\mathcal{W}}$ ▷ Eq. 6
    Select: top-K $\boldsymbol{W}_{couple}$ with smallest $\hat{I} \rightarrow \{\boldsymbol{W}_{couple}\}^{\mathrm{top}-K}$
    for $\boldsymbol{W}_{couple}$ *in* $\{\boldsymbol{W}_{couple}\}^{\mathrm{top}-K}$ **do**
      Compute: $\boldsymbol{\upsilon}$ of $\boldsymbol{W}_{couple}$ ▷ Eq. 9
      while $\sum \upsilon_i > \theta$ **do**
        Remove smallest υ_i and i-th row/col. of $\boldsymbol{W}_{couple}$
        Remove i-th row/col. optimizer moments of $\boldsymbol{W}_{couple}$

4 Experiments

4.1 Experiments on Pre-training (Train from Scratch)

NLP Transformer (GPT-2): We pre-train sparse NLP transformers by following the training settings in Zhao et al. (2023). Specifically, we train sparse GPT-2 Radford et al. (2019) models from scratch on the WikiText-103 dataset Merity et al. (2016), using the AdamW optimizer with an initial learning rate of 0.001 and a batch size of 512.

Table 1: Evaluation of training sparse/low-rank GPT-2 models from scratch on WikiText-103. Validation perplexity is provided, along with memory estimates for total parameters and optimizer states in FP16 format.

Method	PPL	Training Time	Training Epochs	# Params. (M)	# Mem. (MB)	Inference Throughput (samples/s)
GPT2-Small (Baseline)	18.5	24.5h, 8×V100	100	124.4	711.8	42.64
In-Rank Zhao et al. (2023)	18.9	22.2h, 8×V100	100	91.2	521.9	-
Monarch Dao et al. (2022)	20.7	-	100	72	412.0	-
Galore Zhao et al. (2024b)	18.3	22.6h, 4×V100	50	124.4	504.9	42.64
TranSpa	**18.1**	**19.6h, 4×V100**	50	88.2	504.7	54.79
TranSpa	20.7	21.9h, 4×V100	60	**71.3**	**408.0**	**61.16**
GPT2-Medium (Baseline)	19.5	60.5h, 8×V100	100	354.8	2030.2	19.74
In-Rank Zhao et al. (2023)	19.9	48.6h, 8×V100	100	223.0	1276.0	-
Monarch Dao et al. (2022)	20.3	-	100	165	994.1	-
Galore Zhao et al. (2024b)	19.6	60.4h, 4×V100	50	354.8	1260.9	19.74
TranSpa	**19.0**	**52.1h, 4×V100**	50	219.9	1258.3	24.48
TranSpa	19.8	56.8h, 4×V100	60	**160.3**	**917.2**	**28.46**

Table 2: Evaluation of training LLaMA-1B models from scratch on C4 dataset for 100K steps. Validation perplexity is provided, along with memory estimates for total parameters and optimizer states in BF16 format. The results for LoRA and ReLoRA are sourced from Zhao et al. (2024b).

Method	PPL	Training Time 8×A100	# Params. (M)	# Mem. (GB)	Inference Throughput (tokens/s)
Baseline	15.56	51.1h	1339.08	7.80	21786.49
LoRA Hu et al.	19.21	-	1339.08	6.17	21786.49
ReLoRA Lialin et al. (2023)	18.33	-	1339.08	6.17	21786.49
Galore Zhao et al. (2024b)	15.64	60.6h	1339.08	4.38	21786.49
TranSpa	15.60	**40.3h**	**933.94**	**3.88**	**35211.27**

Large Language Model (LLaMA-1B): For large language models, we follow the training settings in Zhao et al. (2024b) to train sparse LLaMA-1B from scratch on the C4 dataset. The training employs the Galore optimizer with an initial learning rate of 0.01 and a batch size of 512.

Vision Transformer (DeiT): We apply our approach to train sparse DeiT models Touvron et al. (2021) from scratch on the ImageNet-1K dataset Deng et al. (2009). We use the same hyper-parameters as in Chen et al. (2021a), which include the AdamW optimizer with an initial learning rate of 0.0005 and a batch size of 512.

4.2 Comparison Results

Table 1 compares our approach with the existing sparse and low-rank pre-training works. It is seen that TranSpa achieves better performance than baseline and prior efforts using only half GPU resources and less training time. Specifically, our approach can train sparse GPT2-Small and GPT2-Medium models from scratch, with $1.4\times$ and $1.6\times$ model size reduction, respectively; and meanwhile, it brings 0.4 and 0.6 lower perplexity over the baseline models. Compared with Dao et al. (2022), TranSpa achieves the same or lower perplexity with $1.6\times$ training speedup (measured using the number of epochs). Compared to Galore Zhao et al. (2024b), a method designed for low-rank compression of optimizer memory, TranSpa delivers better results with faster training speeds under the same memory consumption and provides $1.2\times$ training speedup and $1.4\times$ inference acceleration.

Table 2 presents the pre-training results for LLaMA-1B. Our approach reduces GPU memory usage by 50%, decreases training time by 21%, and achieves a $1.6\times$ speedup in inference throughput without any performance loss. Compared to Galore, our approach reduces memory usage by 10%, increases training speed by $1.5\times$, and boosts inference speed by $1.6\times$. The results for LoRA Hu et al. and ReLoRA Lialin et al. (2023) are sourced from Zhao et al. (2024b). For low-rank methods, LoRA Hu et al. (2021) fine-tunes pre-trained models using low-rank adaptors: $W = W_0 + BA$, where W_0 is the fixed initial weights and BA is a learnable low-rank adaptor. For pre-training, W_0 is the full-rank

Table 3: Results of training sparse DeiT models from scratch on ImageNet-1k. Results for SSP-Tiny and SSP-Small are sourced from Chen et al. (2021a). Inference speedup is measured on Nvidia RTX 3090 GPU.

Method	# Params. (M)	# Mem. (MB)	FLOPs Saving (%)	Top-1 Acc. (%)	Inference Speedup
DeiT-Tiny (Baseline)	5.72	32.73	-	71.80	-
SSP-Tiny Bartoldson et al. (2020)	4.21	24.09	23.69	68.59	1.12×
S^2ViTE-Tiny Chen et al. (2021a)	4.21	24.09	23.69	70.12	1.12×
TranSpa	**3.95**	**22.60**	**32.01**	70.92	**1.20×**
DeiT-Small (Baseline)	22.10	126.46	-	79.78	-
SSP-Small Bartoldson et al. (2020)	14.60	83.54	33.13	77.74	1.29×
S^2ViTE-Small Chen et al. (2021a)	14.60	83.54	33.13	79.22	1.29×
TranSpa	**13.98**	**79.99**	**37.30**	79.57	**1.29×**

Table 4: Perplexity of compressed LLaMA2-7B on WikiText-2 with different target model sizes. SVD-LLM Wang et al. (2024) and SliceGPT Ashkboos et al. (2024) are low-rank based methods. LLM Surgeon van der Ouderaa et al. (2023) is a pruning method, K-OBD van der Ouderaa et al. (2023), as a baseline comparison method, uses Kronecker-factored curvature and only prunes without updating the remaining weights.

Method		K-OBD	SVD-LLM	LLM Surgeon	SliceGPT	**TranSpa**
Training Time		16h58m, H100	15m, A100	17h08m, H100	1h07m, H100	**1h41m, A100**
PPL @ Target Size	80%	9.14	7.94	6.18	6.64	6.36
	70%	15.43	9.56	7.83	8.12	**7.66**
	60%	28.03	13.11	10.39	-	**10.24**
	50%	46.64	23.97	15.38	-	**14.02**

initialization matrix. ReLoRA Lialin et al. (2023) is a variant of LoRA designed for pre-training. It periodically merges BA into W and reinitializes BA with a reset on optimizer states and learning rate. TranSpa surpasses these low-rank methods, reducing PPL by 3.6 and 2.7, respectively, while decreasing memory usage by 37% and offering $1.6\times$ acceleration in inference throughput.

Table 3 lists the performance results of various sparse vision transformer pre-training methods. TranSpa achieves a 0.8% increase in top-1 accuracy for DeiT-Tiny and a 0.35% increase for DeiT-Small over state-of-the-art solutions, along with greater model size reduction.

4.3 Experiments on Fine-tuning LLaMA2-7B (Structured Pruning)

Experimental Setting. We evaluate the pruning performance of TranSpa on the pre-trained LLaMA2 Touvron et al. (2023) models. We use WikiText-2 Merity et al. (2016) as the calibration dataset and evaluate the perplexity of the pruned model. We follow the same training settings adopted in van der Ouderaa et al. (2023), use 128 sequences with a sequence length of 2048 tokens from the training data set, and evaluate perplexity on the standard test split. Additionally, we also evaluate the performance of the pruned models on downstream zero-shot tasks.

Comparison Results. Table 4 presents a comparison of the perplexity performance between TranSpa and existing LLM pruning and low-rank factorization methods applied to LLaMA2-7B. Our approach consistently achieves lower perplexity across various target model size configurations compared to previous works. Additionally, Table 5 illustrates the zero-shot task performance of the pruned LLaMA2-7B model. In comparison to SliceGPT Ashkboos et al. (2024), our method demonstrates improved results across different target model sizes, indicating its effectiveness.

4.4 Discussion & Analysis

Selection of K and θ. As described in Algorithm 1, K determines the percentage of $\boldsymbol{W}_{couple}$'s that will be sparsified in each epoch, and threshold θ impacts the amount of to-be-removed coupled

Table 5: Comparison of downstream zero-shot task performance of LLaMA2-7B model when trained on WikiText2 dataset.

	Target Size	PIQA	WinoGrande	HellaSwag	ARC-e	ARC-c	Avg.
LLaMA2-7B	100%	79.11	69.06	75.99	74.58	46.25	69.00
TranSpa	80%	73.78	61.48	67.79	61.62	39.42	60.82
	75%	71.93	60.69	64.69	54.38	35.15	57.37
	70%	70.18	59.98	60.00	49.16	34.22	54.71
SliceGPT	80%	69.42	65.11	59.04	59.76	37.54	58.18
	75%	66.87	63.38	54.16	58.46	34.56	55.48
	70%	63.55	61.33	49.62	51.77	31.23	51.50

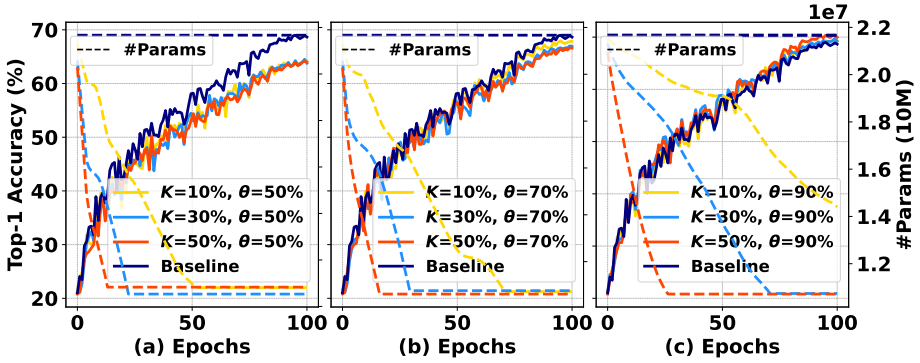

Figure 3: Pre-training sparse DeiT-Small model on CIFAR-10 dataset with various K and θ. The target model size is 11M.

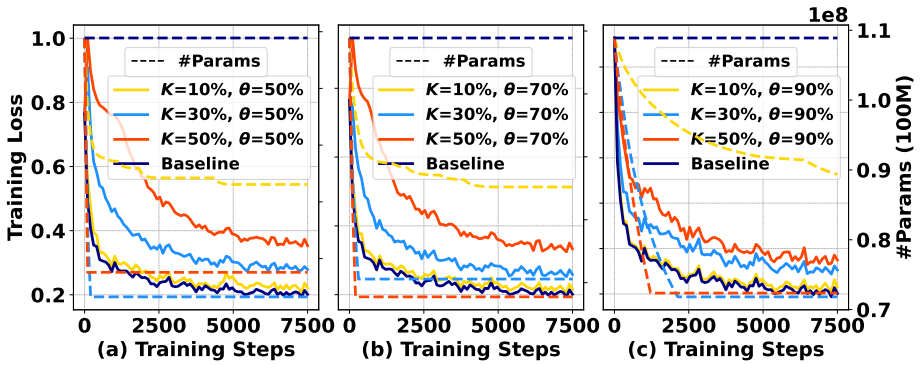

Figure 4: Fine-tuning Pre-trained sparse BERT-Base model on SQuAD benchmark with various K and θ. The target model size is 72.6M.

row/column pairs within those unimportant $\boldsymbol{W}_{couple}$'s. As shown in Fig. 3, when applying TranSpa at the pre-training stage, larger θ brings better training performance, while the model is less sensitive to the change of K. On the other hand, Fig. 4 shows that it is better to use smaller K at the fine-tuning stage, while the selection of θ is less significant in this scenario. We hypothesize that such difference might be due to the existence of a pre-trained model in the fine-tuning process since a pre-trained model typically has more diverse distribution of $\boldsymbol{W}_{couple}$ than the model being sparsely trained from random initialization, making the identification of unimportant $\boldsymbol{W}_{couple}$ more effective. Therefore, considering K cannot be too small (otherwise, it is challenging to meet the model size budget (see the change of parameters (dashed curve) in Fig. 4), we set $K = 30\%$ and $\theta = 90\%$ in our experiments.

Importance of Coupled Sparsification. Fig. 5 compares the training performance using different methods to compress $\boldsymbol{W}_{couple}$. It is seen that our proposed coupled row/column-wise sparsification scheme achieves the best performance. In particular, it outperforms the uncoupled row/column-based

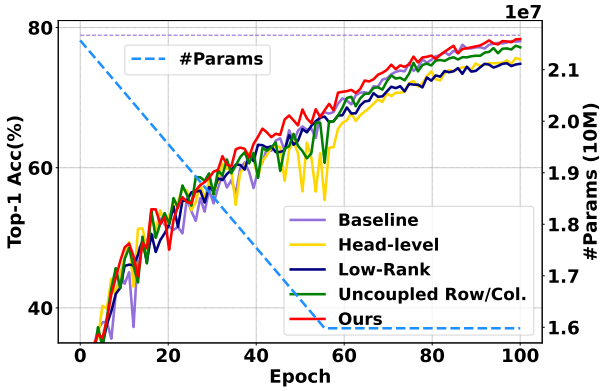

Figure 5: Pre-training DeiT-Small model on CIFAR-10 dataset using different $\boldsymbol{W}_{couple}$ compression methods. All the methods remove the same number of parameters within the same $\boldsymbol{W}_{couple}$'s in each epoch.

solution, demonstrating the importance of removing the row/column of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ in a coupled way. Notice that though our approach is inspired by tSVD, it achieves better performance than directly applying tSVD on $\boldsymbol{W}_{couple}$. This is because SVD not only changes the structure of the model but also alters the numerical distribution of the original model weights. Consequently, the optimizers that keep track of moment information, *e.g.*, Adam Kingma & Ba (2014), cannot work well since they will not be able to use previously accumulated information, thereby affecting the model performance.

5 Conclusion

In this paper, we propose TranSpa, an efficient structured sparse training approach for transformers. By estimating the coupled weight matrix-wise importance and removing the coupled row/column pair during training, TranSpa brings a significant reduction in training costs and model complexity with preserving high task performance. Experiments across various transformer models demonstrate the superior performance of TranSpa in both pre-training and fine-tuning scenarios.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10865–10873, 2024.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

Brian Bartoldson, Ari Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *Advances in Neural Information Processing Systems*, 33: 20852–20864, 2020.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pp. 213–229. Springer, 2020.

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

Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021a.

Tianyi Chen, Tianyu Ding, Badal Yadav, Ilya Zharkov, and Luming Liang. Lorashear: Efficient large language model structured pruning and knowledge recovery. *arXiv preprint arXiv:2310.18356*, 2023a.

Tianyi Chen, Luming Liang, Tianyu DING, Zhihui Zhu, and Ilya Zharkov. OTOv2: Automatic, generic, user-friendly. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=7ynoX1ojPMt.

Tianyi Chen, Luming Liang, Tianyu Ding, Zhihui Zhu, and Ilya Zharkov. Otov2: Automatic, generic, user-friendly. *arXiv preprint arXiv:2303.06862*, 2023c.

Xiaodong Chen, Yuxuan Hu, and Jing Zhang. Compressing large language models by streamlining the unimportant layer. *arXiv preprint arXiv:2403.19135*, 2024.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. Earlybert: Efficient bert training via early-bird lottery tickets. 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)*, pp. 2195–2207, 2021b.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. Quora question pairs, 2018.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pp. 4690–4721. PMLR, 2022.

Tristan Deleu and Yoshua Bengio. Structured sparsity inducing adaptive optimizers for deep learning. *arXiv preprint arXiv:2102.03869*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Bill Dolan, Chris Brockett, and Chris Quirk. Microsoft research paraphrase corpus. *Retrieved March*, 29(2008):63, 2005.

Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5884–5888. IEEE, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Abhimanyu Dubey, Moitreya Chatterjee, and Narendra Ahuja. Coreset-based neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 454–470, 2018.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015b.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4340–4349, 2019.

Zejiang Hou, Minghai Qin, Fei Sun, Xiaolong Ma, Kun Yuan, Yi Xu, Yen-Kuang Chen, Rong Jin, Yuan Xie, and Sun-Yuan Kung. Chex: Channel exploration for cnn model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12287–12298, 2022.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. *arXiv preprint arXiv:2207.00112*, 2022.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2023.

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1529–1538, 2020.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

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

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Srikanth Malla, Joon Hee Choi, and Chiho Choi. Copal: Continual pruning in large language generative models. *arXiv preprint arXiv:2405.02347*, 2024.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Neo Wei Ming, Zhehui Wang, Cheng Liu, Rick Siow Mong Goh, and Tao Luo. Ma-bert: Towards matrix arithmetic-only bert inference by eliminating complex non-linear functions. In *The Eleventh International Conference on Learning Representations*, 2022.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.

Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pp. 4646–4655. PMLR, 2019.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, and et al. Gpt-4 technical report, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

Yang Sui, Miao Yin, Yi Xie, Huy Phan, Saman Aliari Zonouz, and Bo Yuan. Chip: Channel independence-based pruning for compact neural networks. *Advances in Neural Information Processing Systems*, 34:24604–24616, 2021.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.

Chong Min John Tan and Mehul Motani. Dropnet: Reducing neural network complexity via iterative pruning. In *International Conference on Machine Learning*, pp. 9356–9366. PMLR, 2020.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, volume 139, pp. 10347–10357, July 2021.

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

Tycho FA van der Ouderaa, Markus Nagel, Mart Van Baalen, Yuki M Asano, and Tijmen Blankevoort. The llm surgeon. *arXiv preprint arXiv:2312.17244*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*, pp. 91–109. Springer, 2003.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1810–1822, 2019a.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. *arXiv preprint arXiv:2012.15828*, 2020.

Wenxiao Wang, Cong Fu, Jishun Guo, Deng Cai, and Xiaofei He. Cop: Customized deep model compression via regularized correlation-based filter-level pruning. *arXiv preprint arXiv:1906.10337*, 2019b.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*, 2024.

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

Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

Yifei Yang, Zouying Cao, and Hai Zhao. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*, 2024.

Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. *arXiv preprint arXiv:2203.08243*, 2022.

Geng Yuan, Xiaolong Ma, Wei Niu, Zhengang Li, Zhenglun Kong, Ning Liu, Yifan Gong, Zheng Zhan, Chaoyang He, Qing Jin, et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34:20838–20850, 2021.

Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. Red: Looking for redundancies for data-freestructured compression of deep neural networks. *Advances in Neural Information Processing Systems*, 34:20863–20873, 2021.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*, 2023.

Bowen Zhao, Hannaneh Hajishirzi, and Qingqing Cao. Apt: Adaptive pruning and tuning pretrained language models for efficient training and inference. *arXiv preprint arXiv:2401.12200*, 2024a.

Jiawei Zhao, Yifei Zhang, Beidi Chen, Florian Schäfer, and Anima Anandkumar. Inrank: Incremental low-rank learning. 2023.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024b.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

A APPENDIX

A.1 FINE-TUNING BERT MODEL ON GLUE AND SQUAD1.1 DATASET

Experimental Setting. We evaluate the pruning performance of TranSpa on the pre-trained BERT Devlin et al. (2018) models. For experiments on BERT, the model is pruned and evaluated on GLUE Wang et al. (2018) and SQuAD Rajpurkar et al. (2016) benchmarks. When performing fine-tuning on the GLUE benchmark, the learning rate and weight decay are set as 5×10^{-5} and 0, respectively, with 3 training epochs and batch size of 32. The maximum sequence length is 128. For fine-tuning on the SQuAD benchmark, the learning rate is set to 3×10^{-5} at the 3-epoch fine-tuning stage, and the batch size, maximum sequence length, and document stride are 12, 320, 128, respectively. Both training and testing are conducted on a single Nvidia GeForce RTX 3090. For single-sentence tasks, we include the CoLA Warstadt et al. (2019) task, evaluated based on Matthew's correlation, and the SST-2 task, assessed using classification accuracy. For sentence similarity tasks, we consider three distinct evaluations: MRPC Dolan et al. (2005) evaluated using F-1 score, STS-B Cer et al. (2017) assessed through Pearson-Spearman correlation, and QQP Chen et al. (2018) measured in F-1 score. The evaluation also covers three natural language inference tasks: MNLI Williams et al. (2017), measured in classification accuracy with the average of the matched and mismatched subsets, and QNLI Rajpurkar et al. (2016), assessed in terms of accuracy. For the reading comprehension task, SQuAD consists of questions generated by crowd workers based on a collection of Wikipedia articles. Each question requires extracting an answer, in the form of a segment of text or span, from the corresponding reading passage.

Table 6: Experimental results on GLUE. 'Avg." means the average of GLUE tasks. Average training and inference times are calculated by averaging the time consumption across all tasks, "G" denotes Generic Knowledge Distillation, where models are compressed, and then knowledge distillation is conducted on the datasets from English Wikipedia and Toronto Book Corpus Zhu et al. (2015) before fine-tuning on various task datasets. "G+TS" indicates that a task-specific teacher model is still utilized during fine-tuning for further knowledge distillation after generic knowledge distillation. Training times for DitilBERT and MA-DistilBERT are sourced from the Sanh et al. (2019); Ming et al. (2022), respectively. Average training and inference times are computed by averaging the time consumption across all tasks.

Method	MRPC F-1	QNLI Acc.	MNLI Acc.	STS-B Corr.	SST-2 Acc.	QQP F-1	CoLA Corr.	Avg.	# Params. (M)	Knowledge Distllation	Fine-tuning Epochs	Avg. Training Time	Avg. Inference Time (ms/batch)
BERT-Base (Baseline)	87.4	91.3	84.7	88.5	93	87.8	56.2	84.1	109.5	No	3	1402.7s, 1×RTX3090	118.1
TranSpa	88.2	90.1	83.2	86.5	91.6	87.2	52.9	82.8	75.1	No	**2**	**821.7s, 1×RTX3090**	84.3
TranSpa	89.5	89.4	82.7	86.7	92.7	87.3	51.6	**82.8**	**66.1**	No	3	1210.7s, 1×RTX3090	**70.3**
FWSVD Hsu et al. (2022)	88.0	89.5	83.0	87	91.2	87.6	49.4	82.2	66.5	No	4	1548.1s, 1×RTX3090	81.3
FISH Sung et al. (2021)	86.2	93.3	85.3	85.7	94.0	79.3	56.4	82.9	335.8	No	7	-	376.6
EarlyBERT Chen et al. (2021b)	-	89.2	81.8	-	90.7	-	-	-	77.0	No	2.2	-	-
DistilBERT Sanh et al. (2019)	88.7	89.3	82.2	86.1	90.4	86.7	49.8	81.9	67.0	Yes	3	3.75d, 8×V100	-
MiniLMv2 Wang et al. (2020)	89.1	90.6	84	88.1	91.4	86.7	43.3	81.9	67.0	Yes	3	-	-
MA-DistilBERT Ming et al. (2022)	86.7	88.4	82.3	87.5	90.8	-	51.3	-	67.2	Yes	5-10	3.5d, 1×RTX3090	-

Table 7: Experimental results on SQuAD1.1 for pruning BERT-Base model.

Method	# Params. (M)	Exact (%)	F1-score (%)	Training epochs	Training Time (s)	Inference Speedup
BERT-Base (Baseline)	108.9	81.00	88.30	3	3,834	-
ProxSSI Deleu & Bengio (2021)	90.8	72.30	82.00	3	-	1×
OTOv2 Chen et al. (2023b)	87.2	79.40	87.30	3	-	1.2×
OTOv2	59.9	74.60	83.80	3	-	1.4×
EarlyBERT Chen et al. (2021b)	77.0	-	86.13	2.2	-	-
TranSpa	72.3	78.59	86.53	2	2,138	1.5×
TranSpa	86.7	79.92	87.59	3	3,618	1.2×
TranSpa	58.6	75.97	84.64	3	2,939	1.7×

Comparison Results. Table 6 shows the performance results of compressing the BERT-Base model using different methods. It is seen that TranSpa consistently outperforms existing structured pruning and knowledge distillation solutions across a set of tasks. Meanwhile, because it does not require costly knowledge distillation at the fine-tuning stage, our approach enjoys a very fast fine-tuning speed. Table 7 shows the results on the SQuAD benchmark. It is seen that TranSpa consistently outperforms the existing structured pruning approaches with significantly higher exact and F-1 scores (at least 0.52% and 0.29% increase) and smaller model size.