Small Pre-trained Language Models Can be Fine-tuned as Large Models via Over-Parameterization

Ze-Feng Gao[1] , **Kun Zhou**[2,3] , **Peiyu Liu**[1,3] , **Wayne Xin Zhao**[1,3] , **Ji-Rong Wen**[1,2,3]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]School of Information, Renmin University of China
[3]Beijing Key Laboratory of Big Data Management and Analysis Methods
{zfgao,francis_kun_zhou,liupeiyustu,jrwen}@ruc.edu.cn, batmanfly@gmail.com

Abstract

With the help of more parameters, large pre-trained language models (PLMs) have shown remarkable performance in various natural language processing tasks, mostly outperforming small PLMs by a large margin. However, due to the high computational cost, the enormous number of parameters also restricts the applicability of large PLMs in the community. In this paper, we focus on just scaling up the parameters of PLMs during fine-tuning, to benefit from the over-parameterization but not increasing the inference latency. Given a relatively small PLM, we over-parameterize it by employing a matrix product operator, an efficient and almost lossless decomposition method to factorize its contained parameter matrices into a set of higher-dimensional tensors. Considering the efficiency, we further propose static and dynamic strategies to select the most important parameter matrices for over-parameterization. Extensive experiments have demonstrated that our approach can significantly boost the fine-tuning performance of small PLMs and even help small PLMs outperform $3\times$ parameterized larger ones. Our code is publicly available at https://github.com/zfgao66/OPF.

1 Introduction

Due to the state-of-the-art performance, fine-tuning large-scale pre-trained language models (PLMs) (Devlin et al., 2018; Liu et al., 2019; Raffel et al., 2020) has become the de facto method in the natural language process (NLP) area. With the help of large-scale pre-trained data and parameters, these large-scale PLMs are able to process a decent amount of world knowledge (Roberts et al., 2020; Jiang et al., 2020) and generalize well on a variety of tasks (Brown et al., 2020; Lester et al., 2021). Following this way, more data and more parameters have turned into an observed trend to improve the performance of PLMs in recent years (Chowdhery et al., 2022; Chen et al., 2022; Raffel et al., 2020), leading to the number expansion of PLM parameters from millions to billions.

Despite the remarkable performance, the large-scale parameters also limit the usage of large PLMs in the community. Specifically, the computation cost of pre-training and the efficiency of utilizing large PLMs are generally unaffordable for researchers and real-world applications. Therefore, as a compromise, a number of works (Gururangan et al., 2020; Chang et al., 2020; Zhang et al., 2020b) focus on pre-training relatively smaller language models (*e.g.,* BERT-base-uncased) on domain-specific or task-specific corpus. However, as small PLMs are not enough over-parameterized as large models, their generalization capability is generally weaker than them (Brown et al., 2020), leading to a sub-optimal fine-tuning performance on downstream tasks.

Considering narrowing the performance gap between small and large PLMs, this work seeks to over-parameterize small PLMs as large models during fine-tuning, in pursuit of improving their generalization capability. Generally, PLMs are based on the Transformer architecture (Vaswani et al., 2017), and most of their parameters are stored as matrices. According to the matrix decomposition techniques (Tucker, 1966; Henry and Hofrichter, 1992; Oseledets, 2011) (*e.g.,* Singular Value Decomposition), each matrix can be factorized as the multiplication of a set of matrices. In this way, the total number of parameters can be enlarged during fine-tuning, and after convergence, the factorized matrices can also be merged to re-organize the parameter matrix of the small PLMs. Such a paradigm just borrows the merit of over-parameterization during fine-tuning, while not increasing the inference latency of small PLMs.

Although it is promising to incorporate the matrix decomposition to over-parameterize small PLMs, there are two major concerns required to be investigated. First, the possible information

loss caused by the matrix decomposition strategy should be greatly reduced, since small computation errors may be exponentially accumulated and propagated in the stacked multiple Transformer layer of PLMs. Second, small PLMs also consists of numerous parameter matrices, and they do not always play key roles in fine-tuning different downstream tasks (Voita et al., 2019; Zhang et al., 2022). Thus, it is costly and unnecessary to over-parameterize all of them during fine-tuning. Therefore, there is a need to choose the proper matrix decomposition method and leverage it to over-parameterize carefully selected important parameter matrices.

To address the above concerns, we introduce the technique of matrix product operator (MPO) (Pirvu et al., 2010) as the matrix decomposition strategy. MPO has been widely used in the quantum many-body physics area, as it can efficiently factorize any matrix with arbitrary dimensions into a set of higher-dimensional tensors with arbitrary scales, and the factorized tensors can reconstruct the original matrix in almost lossless condition (Gao et al., 2020; Liu et al., 2021a). Such merits make MPO an ideal method for over-parameterizing small PLMs during fine-tuning. Based on MPO, we also devise static and dynamic strategies to adaptively select important parameter matrices for over-parameterizing. The static strategy estimates the importance of each parameter matrix based on the variation of the loss values after removing it from a fine-tuned model (Voita et al., 2019) and then over-parameterizes the top-N important ones. The dynamic strategy computes the variation of gradients within several fine-tuning steps, which is the approximation of the above loss variation (Hou et al., 2020) and can dynamically guide the matrix over-parameterization process during fine-tuning.

To this end, in this paper, we propose a general Over-Parameterization Framework, namely **OPF** to improve the fine-tuning performance of small PLMs. Given the pre-trained parameter matrices of a small PLM, we first utilize the static or dynamic strategies to select the most important ones and then over-parameterize them by the MPO decomposition. Such a framework only revises the fine-tuning process, hence it is general to various small PLMs and NLP tasks. We conduct extensive experiments on the GLUE benchmark (Wang et al., 2018), a widely-used natural language understanding benchmark. Experimental results show that our OPF can boost the performance of small PLMs on GLUE significantly, *e.g.,* improving BERT-base by +2.64 in average, improving T5-base by +2.41 in average. Besides, our approach also helps small PLMs outperform 3× parameterized ones, *e.g.,* BERT-base+Ours (83.68) *v.s.* BERT-large (83.60) in average metrics on GLUE.

2 Related Work

Pre-trained Language Models. Pre-trained language models (PLM) (Devlin et al., 2018; Liu et al., 2019) have yielded state-of-the-art performance on a wide range of natural language processing tasks. Based on the Transformer architecture (Vaswani et al., 2017), BERT (Devlin et al., 2018) incorporated the "pre-training + fine-tuning" paradigm and has significantly improved the performance on a variety of NLP benchmarks, *e.g.,* GLUE (Wang et al., 2018). Following this way, the T5 model (Raffel et al., 2020) and RoBERTa model (Liu et al., 2019) leveraged more data, more parameters and more pre-training steps, further improving the fine-tuning performance. Moreover, GPT-3 (Brown et al., 2020) showed that scaling up language models can greatly improve few-shot performance. In our approach, we improve the performance of PLMs by just scaling up the model during fine-tuning, which would not increase the inference latency.

Over-parameterization in Neural Network. Over-parameterization has shown the superiority on providing better model initialization (Arpit and Bengio, 2019), improving model convergence (Allen-Zhu et al., 2019b; Gao et al., 2021; Du et al., 2018) and generalization (Allen-Zhu et al., 2019a). After the lottery theory hypothesis (Frankle and Carbin, 2018) was introduced, a surge of works pointed out that over-parameterization might be helpful to enhance the training efficiency (Malach et al., 2020; Pensia et al., 2020) and improve the model performance (Chen et al., 2020; Brix et al., 2020; Prasanna et al., 2020). Among them, Liu et al. (2021b) employed in-time over-parameterization to narrow the performance gap between sparse and dense training. Our study aimed to use the over-parametrization strategy to better inspire the potentiality of PLMs, enhancing their fine-tuning performance.

Tensor Decomposition in Neural Network. Tensor decomposition methods have been widely applied in a neural network for efficient train-

ing and inference, *e.g.,* model compression (Gao et al., 2020) and lightweight fine-tuning (Liu et al., 2021a). There are a surge of typical applications using the tensor decomposition methods on the parameter matrices of deep models to compress the linear layers (Novikov et al., 2015) and convolutional kernels (Garipov et al., 2016). Besides, existing works also apply the MPO method for the lightweight fine-tuning of ALBERT (Liu et al., 2021a) and the efficient expansion for the MoE framework (Gao et al., 2022). Unlike existing methods, our approach focuses on the property that tensor decomposition can be used to map parameters from low-level spaces to high-dimensional spaces for over-parameterizing PLMs during fine-tuning, making PLMs benefit from more parameters.

3 Preliminary

Tensor. A tensor $\mathcal{T}_{i_1,i_2,\ldots,i_m}$ can be viewed as an array with m indices, where $\{i_1, i_2, \ldots, i_m\}$ denotes the dimensions of the m indices, respectively. In this way, a vector (*i.e.,* $\boldsymbol{v}$) and a matrix (*i.e.,* $\mathbf{W}$) can be regarded as a 1-order tensor and 2-order tensor, respectively.

Tensor Product. Suppose $\{\psi_1 \ldots \psi_p\}$ and $\{\phi_1 \ldots \phi_q\}$ are the orthonormal basis of tensors $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2)}$, respectively. The tensor product can be derived by contraction of $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2)}$, denoted as $\otimes$. Formally, the tensor contraction of $\mathcal{T}^{(1)} = \sum_{i=1}^{p} a_i \psi_{i_1}$ and $\mathcal{T}^{(2)} = \sum_{j=1}^{q} b_j \phi_{i_2}$ is defined as follow:

$$\begin{aligned}\mathcal{T}^{(1)} \otimes \mathcal{T}^{(2)} &= \left\{\sum_{i=1}^{p} a_i \psi_{i_1}\right\} \otimes \left\{\sum_{j=1}^{q} b_j \phi_{i_2}\right\} \\ &= \sum_{i=1}^{p}\sum_{j=1}^{q} a_i b_j \psi_{i_1} \otimes \phi_{i_2}. \qquad (1)\end{aligned}$$

Tensor Decomposition Tensor decomposition can be seen as the inverse operation of tensor product. A widely-used way is the singular value decomposition (SVD) algorithm. Given a Tensor $\mathcal{T} \in \mathbb{R}^{i_1 \times \cdots \times i_m}$, the m times SVD operation can decompose this tensor into m local tensors $\{\mathcal{T}^{(k)}\}_{k=1}^{m}$. Conversely, the decomposed tensors can also reconstruct the original tensor by sequentially performing the tensor product operator. The details of tensor decomposition are shown in Supplementary Materials A.1

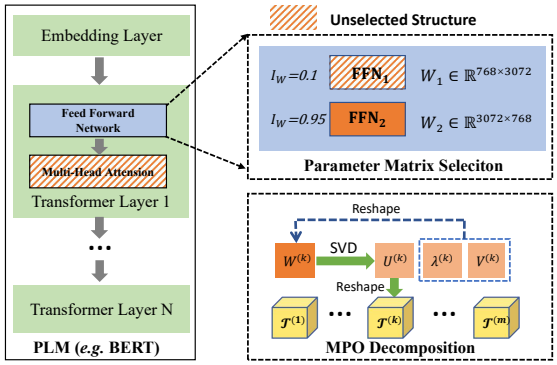


Figure 1: The overview of over-parameter framwork (OPF) in fine-tuning PLMs. $I_{\mathbf{W}}$ denotes the estimated important score of parameter matrices. We show a case that the parameter matrix $\mathbf{W}$ is selected for over-parameterization, and is decomposed into a set of high-order tensors $\{\mathcal{T}^{(k)}\}_{k=1}^{m}$.

4 Approach

In this part, we describe our proposed *OPF* based on parameter matrix decomposition for improving the fine-tuning performance of small PLMs. We first give an overview of our approach, then introduce the details of matrix decomposition and over-parameterized matrices selection strategies.

4.1 Overview

Existing works mostly require compressing a large PLM into a small one for benefiting from over-parameterization (Sun et al., 2019; Shen et al., 2020). Different from them, our approach can scale up the number of parameters of arbitrary small PLMs during fine-tuning without the usage of large ones. To achieve it, we leverage a matrix decomposition method, to factorize part of the important parameter matrices from the small PLM into a sequence of high-order tensors. These high-order tensors would greatly increase the trainable parameter number during fine-tuning and can be reconstructed into the original matrix. In this way, while in the inference phase, the number of parameters in the fine-tuned PLM will be also the same as the original one, without increasing the inference latency and parameter storage.

In our proposed OPF, we incorporate an MPO-based matrix decomposition strategy to scale up the parameter matrices in PLMs and devise static and dynamic selection strategies to determine important matrices for over-parameterization (Section 4.2). During fine-tuning, the static strategy first decides the important parameter matrices from

the PLM based on the variation of training loss after removing each matrix and then relies on MPO to over-parameterize the selected top-N ones. The dynamic strategy computes the variation of gradients to estimate the importance of each matrix once a few steps and dynamically selects important matrices for over-parameterization (Section 4.3). The overview of our approach is presented in Figure 1. We also present a thorough algorithm for our OPF in Algorithm 1.

4.2 Over-parameterizing PLMs via Matrix Product Operator

To make small PLMs benefit from the over-parameterization during fine-tuning, our approach relies on Matrix Product Operator (MPO), a matrix decomposition technique to expand the number of model parameters. In this part, we first introduce the details of the MPO method and then describe how to adapt it for over-parameterizing PLMs.

Matrix Product Operator. MPO is an efficient algorithm that can factorize a parameter matrix $\mathbf{W} \in \mathbb{R}^{I \times J}$ into a sequential product of multiple tensors (Gao et al., 2020), denoted as:

$$\mathrm{MPO}(\mathbf{W}) = \mathcal{T}^{(1)} \otimes \cdots \otimes \mathcal{T}^{(m)}, \qquad (2)$$

where $\{\mathcal{T}^{(k)}\}_{k=1}^{m}$ are the set of 4-order tensors with size $[d_{k-1}, i_k, j_k, d_k]$, in which $\prod_{k=1}^{m} i_k = I$, $\prod_{k=1}^{m} j_k = J$, and d_k is calculated by:

$$d_k = \min(\sum_{l=1}^{k} i_l \times j_l, \sum_{l=k}^{m} i_l \times j_l). \qquad (3)$$

Given the parameter matrix $\mathbf{W}$, the tensor sizes $\{d_k\}_{k=1}^{m}$, $\{i_k\}_{k=1}^{m}$ and $\{j_k\}_{k=1}^{m}$, MPO can be regarded as a determined mapping process from $\mathbf{W}$ to multiple high-order tensors $\{\mathcal{T}^{(k)}\}_{k=1}^{m}$. Concretely, the MPO process consists of m-turn iterative matrix reshaping and SVD decomposition operations (Henry and Hofrichter, 1992), where the parameter matrix will gradually shrink and the decomposed tensor will be generated one by one. In the k-th turn, given the output parameter matrix $\mathbf{W}_{k-1}$ from the last turn, we first reshape it into a new matrix $\mathbf{W}'_{k-1}$ whose first dimension is $d_{k-1} \times i_k \times j_k$. Then, we perform the SVD decomposition on it as:

$$\mathbf{U}\lambda\mathbf{V}^{\top} = \mathrm{SVD}\,(\mathbf{W}'_{k-1}) \qquad (4)$$

where $\mathbf{U}$ and $\mathbf{V}$ are complex unitary matrices, λ is a rectangular diagonal matrix with non-negative real numbers on the diagonal. Following truncated SVD methods (Henry and Hofrichter, 1992; Hansen et al., 1992), we extract the first d_k columns of $\mathbf{U}$ corresponding to the d_k largest singular values to compose the decomposed tensor $\mathcal{T}^{(k)}$, and reshape it to the size $[d_{k-1}, i_k, j_k, d_k]$. Besides, we adopt $\lambda\mathbf{V}^{\top}$ as the output parameter matrix $\mathbf{W}_k$ for the decomposition in the following turns. After m-turn iterations, we can obtain the decomposed multiple high-order tensors $\{\mathcal{T}^{(k)}\}_{k=1}^{m}$, and the contraction of these tensors in order would reconstruct the original parameter matrix $\mathbf{W}$ in almost lossless condition (Gao et al., 2020) (See Algorithm 2 in Appendix A.1).

Over-parameterizing PLMs. Based on the MPO method, we aim to expand the parameter scale of small PLMs during fine-tuning, for benefiting from over-parameterization. Generally, PLMs are based on the Transformer architecture (Vaswani et al., 2017), consisting of an embedding layer, stacked multi-head attention layers, and feed-forward networks. These modules contain necessary parameter matrices that have been pre-trained on large-scale corpus, *e.g.,* the query projection matrices in the multi-head attention layers. Therefore, we can utilize the MPO method to decompose part of the parameter matrices into multiple tensors as Eq. (2). After the MPO decomposition, the parameter number of the matrix $\mathbf{W}$ would be increased according to the values of $\{d_k\}_{k=1}^{m}$, $\{i_k\}_{k=1}^{m}$ and $\{j_k\}_{k=1}^{m}$. The detailed added parameter number N_{add} can be calculated as follows:

$$N_{add} = \sum_{k=1}^{m} i_k j_k d_{k-1} d_k - \prod_{k=1}^{m} i_k j_k. \qquad (5)$$

According to Eq. (8), $\{d_k\}_{k=1}^{m}$ are determined by $\{i_k; j_k\}_{k=1}^{m}$. Hence we can adjust the values of $\{i_k; j_k\}_{k=1}^{m}$ to control the number of added parameters by the MPO decomposition strategy. Therefore, during fine-tuning, we can adopt MPO on several selected parameter matrices from the PLM to generate their corresponding multiple tensors. In this way, we can scale up the total parameter the number of the PLM, and make it more over-parameterized. After fine-tuning the over-parameterized PLM to convergence, we will perform tensor contraction on these decomposed tensors, to reconstruct the parameter matrices of the PLM. This new PLM owns the same parameter number and inference latency as the original one and has benefited from over-parameterization during fine-tuning.

4.3 Over-parameterized Matrices Selection

Despite the efficiency and flexibility of the MPO method, it is still costly to utilize it for over-parameterizing all the parameter matrices in small PLMs. To concentrate the benefits of over-parameterization on the most important parameters, we only select the most important parameter matrices from PLMs for decomposition. In particular, we propose a static selection strategy as well as a dynamic selection strategy, which pre-selects the important parameter matrices or dynamically chooses the ones during fine-tuning, respectively.

Static Selection Strategy. The proposed static selection strategy requires to pre-compute the importance scores of all parameter matrices before fine-tuning and then leverages MPO to over-parameterize the top-N ones. After that, the architecture of the over-parameterized PLM would be static during fine-tuning. Inspired by network pruning methods (Molchanov et al., 2016; Voita et al., 2019), we utilize the change of the training loss $\mathcal{L}_{\mathbf{W}}$ after removing each parameter matrix $\mathbf{W}$, to measure the importance scores since important parameters would play a key role to predict the correct label (Voita et al., 2019). Therefore, the importance score $I_{\mathbf{W}}$ of a parameter matrix $\mathbf{W}$ can be computed as:

$$I_{\mathbf{W}} = |\mathcal{L}_{\mathbf{W}} - \mathcal{L}_{\mathbf{W}=0}|, \tag{6}$$

where $\mathcal{L}_{\mathbf{W}=0}$ denotes the value of loss after zeroing $\mathbf{W}$. To calculate the loss, we need to fine-tune a small PLM using the same pre-trained parameter as ours before. Generally, the parameter matrices from different modules of the PLM (*e.g.,* multi-head attention layer and feed-forward network) may have different sizes and functions, making it inappropriate to directly compare them. Thus, we first categorize all parameter matrices by module, where each group contains one module for L layers. Then we pick the top-N ones from each group for over-parameterization.

Dynamic Selection Strategy Our proposed dynamic selection strategy aims to dynamically calculate the importance scores and choose the immediate important parameter matrices for over-parameterization during fine-tuning. Such a way can dynamically capture the importance of change *w.r.t.* the optimization of the whole PLM. Following Hou et al. (2020), we perform the first-order Taylor expansion on Eq. (6) to obtain the approximation of the importance score as:

$$I_{\mathbf{W}} = |\mathcal{L}_{\mathbf{W}} - (\mathcal{L}_{\mathbf{W}} - \frac{\partial \mathcal{L}}{\partial \mathbf{W}}(\mathbf{W}-\mathbf{0}) + R_{\mathbf{W}=\mathbf{0}})| \approx |\frac{\partial \mathcal{L}}{\partial \mathbf{W}}\mathbf{W}|, \tag{7}$$

where once the remaining part $R_{\mathbf{W}=\mathbf{0}}$ is omitted, the important score can be estimated by the absolute values of the gradients of the parameter matrix. In practice, we accumulate the absolute values of the gradients for all the parameter matrices during fine-tuning. We dynamically calculate the importance score using Eq. (7) and over-parameterize the top-n parameter matrices from the categorized groups once t steps. The above process will be performed multiple times until N parameter matrices from each group have been selected.

Algorithm 1 Fine-tuning a PLM with our OPF.

Input: Parameters matrices set of a PLM $\{\mathbf{W}\}$.
Divide $\{\mathbf{W}\}$ into several groups by module.
if is Static Strategy **then**
  Fine-tuning the PLM until converged.
  Compute $I_{\mathbf{W}}$ for $\{\mathbf{W}\}$ using Eq. (6).
  Sort $\{\mathbf{W}\}$ in each group according to $I_{\mathbf{W}}$.
  Perform MPO on the top-N matrices.
  Train the other PLM until converged.
else
  Define $S = \{\}$
  while Len$(S) < N$ **do**
    Train the PLM for t steps.
    Compute $I_{\mathbf{W}}$ for $\{\mathbf{W}\}$ using Eq. (7).
    Sort $\{\mathbf{W}\}$ in each group according to $I_{\mathbf{W}}$.
    Add top-n matrices into S, and perform MPO.
  end while
  Continually train the PLM until converged.
end if

5 Experiments

In this section, we first set up the experiments, then report the results and give a detailed analysis.

5.1 Experimental Setup

Datasets. To verify the effectiveness of our approach, we conduct experiments on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), which consists of 8 datasets (MNLI, QQP, QNLI, RTE, MRPC, CoLA, SST-2, STS-B) to systematically evaluate the abilities of similarity matching, sentiment classification, linguistic acceptability estimation and natural language inference. Since the labels of their original test sets are not available, we randomly split their original validation sets in half, and use one half as the validation set and the other half as the test set. For the evaluation metrics, following existing

Datasets	MNLI Acc.	QNLI Acc.	SST-2 Acc.	RTE Acc.	QQP Acc.	CoLA Mcc.	STS-B Spear.	MRPC F1	Avg.	#To (M) Train	#To (M) Test
BERT-small											
+None	77.60	**86.40**	89.70	61.80	87.00	27.80	77.00	83.40	73.84	28	28
+OPF-SVD	77.73	86.06	89.04	62.31	88.10	27.90	79.31	83.25	74.21	34	28
+OPF-MPO$_S$	**77.76**	86.37	89.27	63.54	88.11	28.14	85.46	83.61	75.28	81	28
+OPF-MPO$_D$	77.75	86.10	**89.77**	**63.55**	**88.99**	**28.19**	**86.27**	**83.91**	**75.57**	81	28
BERT-medium											
+None	80.00	87.70	89.60	62.20	87.90	38.00	78.40	86.60	76.30	41	41
+OPF-SVD	**80.77**	87.50	89.68	62.45	89.35	39.16	79.61	87.35	76.98	46	41
+OPF-MPO$_S$	80.58	87.55	90.13	62.73	89.36	42.22	87.53	85.81	78.24	129	41
+OPF-MPO$_D$	80.61	**88.24**	**90.37**	**62.82**	**89.84**	**44.56**	**87.89**	**86.08**	**78.90**	129	41
BERT-base											
+None	83.60	90.50	92.50	66.40	89.30	52.10	85.80	88.10	81.04	109	109
+OPF-SVD	83.62	90.59	92.54	66.79	89.31	55.21	88.45	87.88	81.80	134	109
+OPF-MPO$_S$	83.78	90.87	**92.55**	68.87	89.30	56.12	88.53	88.40	82.30	341	109
+OPF-MPO$_D$	**84.08**	**91.54**	92.52	**72.32**	**89.40**	**60.62**	**89.03**	**89.95**	**83.68**	341	109
BERT-large											
+None	85.70	92.70	93.90	70.10	90.10	60.50	86.50	89.30	83.60	335	335
+OPF-SVD	85.33	91.78	93.22	71.48	90.12	56.82	88.04	88.74	83.19	410	335
+OPF-MPO$_S$	85.90	92.73	93.69	72.64	90.60	**63.56**	89.03	91.01	84.90	828	335
+OPF-MPO$_D$	**85.96**	**92.85**	**93.82**	**72.94**	**90.69**	62.63	**89.63**	**91.08**	**84.95**	828	335

Table 1: Performance comparison using BERT on GLUE benchmark (in percent). "# To (M)-Train" and "# To (M)-Test" denote the number (in millions) of total parameters during training and test, respectively. "+OPF-SVD" represents the use of SVD as the model over-parameterization method, whilst "+OPF-MPO$_S$ " and "+OPF-MPO$_D$ " signify the use of MPO decomposition as the over-parameterization method with static and dynamic matrix selection strategy, respectively. The best performance in each group is highlighted in bold. For all the results, we report the mean values of five runs using different random seeds.

works (Gao et al., 2022), we use Matthews correlation for CoLA, Spearman correlation for SST-B, F1 for MRPC, and accuracy for other tasks. We also compute the average score across all tasks.

Baseline Methods. We implement our approach on the following PLMs, **BERT** (Devlin et al., 2018), **T5** (Raffel et al., 2020) and **BART** (Lewis et al., 2019). **BERT** is a widely-used PLM based on the bidirectional Transformer architecture. We select the publicly released BERT-small, BERT-medium (Liu et al., 2021b), BERT-base and BERT-large (Devlin et al., 2018) for comparison. **T5** and **BART** adopt the sequence-to-sequence Transformer architecture, and we choose their base and large versions. Besides, we also compare our approach with **SVD** (Henry and Hofrichter, 1992), a classic matrix decomposition method that can also be used for over-parameterizing PLMs. Concretely, we leverage SVD to replace MPO in our framework and perform over-parameterization on all the parameter matrices of the PLM during fine-tuning.

5.2 Main Experimental Results

In this part, we report and analyze the experimental results on BERT, T5 and BART.

Evaluation on BERT. We present the results on BERT in Table 1. First, we can see that the BERT models with more parameters perform consistently better than smaller ones, *i.e.,* BERT-large > BERT-base > BERT-medium > BERT-small. It demonstrates that more parameters are helpful for PLMs to achieve better performance, showing the effectiveness of over-parameterization. Second, after combining PLMs with the over-parameterization methods, their performances are most improved. Although these methods just increase the model parameters during fine-tuning, they can also benefit from over-parameterization to improve the generalization capacity. Between the two matrix decomposition methods, we observe that SVD mostly underperforms MPO. As SVD just performs the matrix decomposition once in the 2D space based on the singular value, it is hard to greatly increase the number of the model parameters as our approach (*e.g.,* 34M *v.s.* 81M in BERT-small). As a comparison, MPO can factorize the matrix into arbitrary scales by increasing the order, making it more proper for over-parameterization.

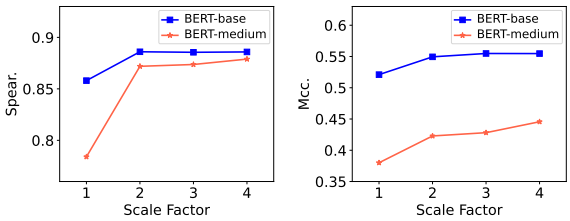
Finally, by comparing our approach with different matrix selection strategies, we can see that the dynamic strategy mostly outperforms the static

Datasets	MNLI Acc.	QNLI Acc.	SST-2 Acc.	RTE Acc.	QQP Acc.	CoLA Mcc.	STS-B Spear.	MRPC F1	Avg.	#To (M) Train	#To (M) Test
T5-Base											
+None	87.78	93.82	94.72	71.74	**91.11**	53.49	91.16	89.16	84.12	220	220
+OPF-MPO$_S$	87.95	93.27	92.88	74.64	89.89	62.72	**91.21**	90.76	85.42	663	220
+OPF-MPO$_D$	**88.78**	**93.91**	**95.14**	**77.42**	91.08	**63.51**	91.11	**91.30**	**86.53**	663	220
T5-large											
+None	**89.32**	94.03	96.20	83.94	91.54	55.10	91.90	90.15	86.51	770	770
+OPF-MPO$_S$	88.15	93.98	**96.21**	83.98	89.88	66.38	91.91	92.38	87.86	1426	770
+OPF-MPO$_D$	88.91	**94.11**	96.05	**84.12**	**91.67**	**66.51**	**91.85**	**92.41**	**88.20**	1426	770
BART-base											
+None	85.78	93.15	92.54	69.31	91.00	44.72	91.08	**90.58**	82.27	140	140
+OPF-MPO$_S$	85.84	93.62	93.58	67.57	**91.16**	45.78	91.07	90.32	82.36	418	140
+OPF-MPO$_D$	**85.89**	**93.94**	**93.81**	**71.56**	90.64	**46.75**	**91.11**	90.31	**83.07**	418	140
BART-large											
+None	88.60	93.98	**95.76**	79.92	91.08	59.56	**91.23**	90.14	86.28	407	407
+OPF-MPO$_S$	88.75	**94.21**	95.18	79.81	90.67	61.69	91.15	90.16	86.45	1198	407
+OPF-MPO$_D$	**89.09**	94.12	95.35	**82.31**	**91.16**	**62.55**	91.08	**91.31**	**87.12**	1198	407

Table 2: Performance comparison using T5 and BART on GLUE benchmark (in percent). "# To (M)-Train" and "# To (M)-Test" denote the number (in millions) of total parameters during training and test, respectively. The best performance in each group is highlighted in bold. For all the results, we report the mean values of five runs using different random seeds.

one, under the setting of the same parameter scale. The reason may be that the dynamic strategy can estimate the importance of immediate parameter matrices *w.r.t.* the training steps. Such a way is able to adapt to the change of parameter importance during fine-tuning, and better guides the over-parameterization. Surprisingly, by using our framework with the dynamic strategy, the BERT-base model can be fine-tuned to achieve comparable performance as the BERT-large model, where the number of its parameters is just increased into a similar scale during fine-tuning.

Evaluation on T5 and BART. We show the results on T5 and BART in Table 2. Similar to BERT, we can also see that the large models consistently outperform base models, and our proposed over-parameterization method narrows this performance gap. It indicates that our approach is general to different model architectures and pre-training tasks and can benefit from over-parameterization to improve the fine-tuning performance of different PLMs. Besides, the performance of T5 is improved more than BART under a similar parameter-increasing rate, and the over-parameterized T5-base model also achieves comparable performance with T5-large. A possible reason is that T5 has been pre-trained using a much large corpus C4 (Raffel et al., 2020), and over-parameterization can better inspire its potentiality during fine-tuning.



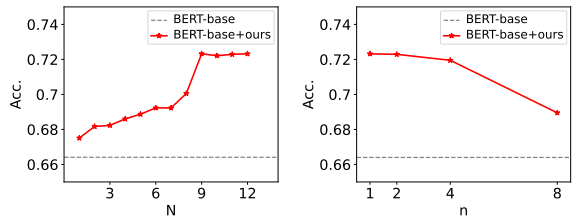
(a) Performance on STS-B. (b) Performance on CoLA.

Figure 2: Comparison of different scale factors of parameter number after over-parameterizing BERT-medium and BERT-base in STS-B and CoLA tasks.

5.3 Further Analysis

Next, we continue to investigate our proposed approach in a more detailed analysis.

Performance Comparison *w.r.t.* Parameter Increasing Rate. During fine-tuning, our approach can increase the number of model parameters for improving the over-parameterization of PLMs. As our approach is a general and flexible way to increase the model parameters into arbitrary scales, here we investigate how the performance changes *w.r.t.* a different number of increased model parameters. Based on BERT-base and BERT-medium, we expand their parameter scales after over-parameterizing from $1\times$ to $4\times$, reporting the performance on STS-B and CoLA tasks. As shown in Figure 2, we can see that the model performance



(a) Performance *w.r.t* N. (b) Performance *w.r.t* n.

Figure 3: Comparison of the different total parameter matrices selection number N and the selection number n at one time in each parameter matrix. We conduct experiments on RTE using BERT-base.

is consistently improving *w.r.t.* the increasing of parameter scales. Comparing the improved performance between BERT-base and BERT-medium, BERT-medium has gained more boost. It indicates that a small PLM is much more hungry for more parameters. Besides, after reaching the $4\times$ parameter scale, the improvement becomes relatively smaller. It shows that the $4\times$ parameter scale seems to be the limit that can significantly improve the model performance via over-parameterization.

Hyper-parameters Tuning. For our OPF using the dynamic strategy, the numbers of total selected parameter matrices N and the selection number at one time n in each parameter matrix group are important hyper-parameters that require tuning. Larger N means that more parameter matrices are selected and over-parameterized and larger n denotes that more matrices are over-parameterized at one time. To investigate the effect of their values on the model performance, we conduct experiments on the CoLA task using BERT-base as the backbone. As shown in Figure 3, we can see that the performance steadily improves as N increases and eventually reaches a plateau as a result. The reason may be that over-parameterizing too few matrices is not able to sufficiently over-parameterize the PLM. Besides, we can see that too large n would degrade the performance. A possible reason is that too large n will over-parameterize too many parameter matrices at one time, causing the dynamic strategy to degrade into the static one. Whereas, we can see that our approach consistently outperforms the baseline method. It shows that our approach is not very sensitive to the above hyper-parameters.

Sensitivity Analysis. As our approach is based on the matrix decomposition method to over-parameterize the PLM, once a small error arises during performing decomposition, it would accumulate into an extremely large value that may ruin the PLM. To avoid it, our approach incorporates the MPO method, which can factorize the parameter matrix in almost lossless conditions. Such a way could stabilize the performance of our approach and make it less sensitive to perturbation on hyper-parameters. To validate it, we select a commonly-used hyper-parameter, the learning rate to evaluate the sensitivity of our approach on RTE, CoLA and STS-B tasks using BERT-base, and report the performance change *w.r.t.* tuning it in the set {5e-6, 1e-5, 3e-5, 5e-5, 1e-4} in Table 3. We can observe that the performance of our approach consistently stables around certain values, *i.e.,* 72.0 for RTE, 60.0 for CoLA, and 88.5 for STS-B. It indicates that our approach is not sensitive to the learning rate during fine-tuning. Besides, setting the learning rate to a commonly-used value 3e-5 is enough for our approach to achieving good performance, no longer requiring time-consuming parameter tuning.

Learning Rate	5e-6	1e-5	3e-5	5e-5	1e-4
RTE	71.08	72.24	72.12	72.31	70.25
CoLA	59.86	60.44	60.54	60.61	59.31
STS-B	88.32	88.89	89.01	88.95	88.14

Table 3: Comparison of different learning rates on RTE, CoLA and STS-B tasks using our approach on BERT-base (in percent).

6 Conclusion

In this paper, we proposed OPF, a novel over-parameterization framework to scale up the number of parameters for PLMs just during fine-tuning, for benefiting from more parameters. In our OPF, we incorporated the matrix product operator method, which decomposes the parameter matrices in PLMs into high-order tensors for increasing the parameter number, and also devised the static and dynamic strategies to select the most important parameter matrices for over-parameterization. Extensive experiments have demonstrated that our OPF can boost the performance of small PLMs significantly, and even help small PLMs outperform big ones.

In future work, we will investigate more efficient and effective tensor decomposition methods for PLM over-parameterization. In addition, we will also apply OPF to other important backbone models in computer vision and multimodal domains.

Limitations

Further research is needed to understand the robustness of our over-parameterization framework properly. The results given in this study are constrained by the natural language processing tasks and datasets used for evaluation. Even though we employ standard classifications from the literature, the choice of downstream tasks and datasets is still subjective. Furthermore, due to computing limitations, we could not investigate the scaling behavior of the Large PLMs. Additional study is needed in this area. In addition, as our approach is based on PLMs that may learn biased information from pre-trained corpus, a potential risk is that our approach may also be affected by it and generates improper texts.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China under Grants No. 62206299 and 62222215, Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098 and CCF-Zhipu AI Large Model Fund. Xin Zhao is the corresponding author.

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. 2019a. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019b. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR.

Devansh Arpit and Yoshua Bengio. 2019. The benefits of over-parameterization at initialization in deep relu networks. *arXiv preprint arXiv:1901.03611*.

Christopher Brix, Parnia Bahar, and Hermann Ney. 2020. Successfully applying the stabilized lottery ticket hypothesis to the transformer architecture. *arXiv preprint arXiv:2005.03454*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932*.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. 2022. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. 2009. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons.

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*.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. 2018. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*.

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

Tianxiang Gao, Hailiang Liu, Jia Liu, Hridesh Rajan, and Hongyang Gao. 2021. A global convergence theory for deep relu implicit networks via over-parameterization. *arXiv preprint arXiv:2110.05645*.

Ze-Feng Gao, Song Cheng, Rong-Qiang He, Zhi-Yuan Xie, Hui-Hai Zhao, Zhong-Yi Lu, and Tao Xiang. 2020. Compressing deep neural networks by matrix product operators. *Physical Review Research*, 2(2):023300.

Ze-Feng Gao, Peiyu Liu, Wayne Xin Zhao, Zhong-Yi Lu, and Ji-Rong Wen. 2022. Parameter-efficient mixture-of-experts architecture for pre-trained language models. *arXiv preprint arXiv:2203.01104*.

Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry Vetrov. 2016. Ultimate tensorization: compressing convolutional and fc layers alike. *arXiv preprint arXiv:1611.03214*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

Per Christian Hansen, Takashi Sekii, and Hiromoto Shibahashi. 1992. The modified truncated svd method for regularization in general form. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1142–1150.

ER Henry and J Hofrichter. 1992. [8] singular value decomposition: Application to analysis of experimental data. *Methods in enzymology*, 210:129–192.

Frank L Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189.

Lu Hou, Lifeng Shang, Xin Jiang, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *arXiv preprint arXiv:2004.04037*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Peiyu Liu, Ze-Feng Gao, Wayne Xin Zhao, Zhi-Yuan Xie, Zhong-Yi Lu, and Ji-Rong Wen. 2021a. Enabling lightweight fine-tuning for pre-trained language model compression based on matrix product operators. 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 5388–5398.

Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. 2021b. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, pages 6989–7000. PMLR.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. 2020. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2016. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov. 2015. Tensorizing neural networks. *arXiv preprint arXiv:1509.06569*.

Ivan V Oseledets. 2011. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.

Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. 2020. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. *Advances in Neural Information Processing Systems*, 33:2599–2610.

Bogdan Pirvu, Valentin Murg, J Ignacio Cirac, and Frank Verstraete. 2010. Matrix product operator representations. *New Journal of Physics*, 12(2):025012.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332.

Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

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

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.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020a. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020b. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278.

Appendices

A More Details about Tensors

A.1 Tensor and Matrix Product Operator

As introduced in (Cichocki et al., 2009), a tensor can be defined as follows.

Tensor. Let $D_1, D_2..., D_M \in M$ denote the index upper bounds. A tensor $\mathcal{T} \in \mathbb{R}^{D_1,...,D_M}$ of order M is an M-way array where elements $\mathcal{T}[d_1, d_2, ..., d_M]$ are indexed by $d_m \in \{1, 2, ..., D_M\}$ for $1 \leq m \leq M$.

Matrix Product Operator. The k-th order (Kolda and Bader, 2009) and $k \in \{1, \ldots, D\}$. The bond dimension d_k is defined by:

$$d_k = \min\left(\prod_{m=1}^{k} i_m \times j_m, \prod_{m=k+1}^{n} i_m \times j_m\right). \quad (8)$$

From Eq. (8), we can see that is going to be large in the middle and small on both sides. Algorithm 2 presents a thorough algorithm for MPO decomposition.

Algorithm 2 MPO decomposition procedure.

Input: matrix $\mathbf{W} \in \mathbb{R}^{I \times J}$, the number of local tensor m
Output : local tensor set $\{\mathcal{T}^{(k)}\}_{k=1}^{m}$

for $k = 1, \ldots, m-1$ **do**
  $\mathbf{W}[d_{k-1} \times i_k \times j_k, -1] \leftarrow \text{Reshape}(\mathbf{W}[\text{I, J}])$
  $\mathbf{U}\lambda\mathbf{V}^\top \leftarrow \text{SVD}\ (\mathbf{W})$
  $\mathcal{T}^{(k)}[d_{k-1}, i_k, j_k, d_k] \leftarrow \text{Reshape}(\mathbf{U})$
  Calculate $\mathbf{W} = \lambda\mathbf{V}^\top$
end for
Let $\mathcal{T}^{(m)} \leftarrow \mathbf{W}$
Normalization
return local tensor set $\{\mathcal{T}^{(k)}\}_{k=1}^{m}$

The MPO representation of $\mathbf{W}$ is obtained by factorizing it into a sequential product of local tensors. The algorithm has been depicted in Section 4.2 of the main text. With the MPO decomposition technique, we can get local tensor as follows:

$$\mathbf{W}_{i_1 \cdots i_n, j_1 \cdots j_n} = \mathcal{T}^{(1)}[i_1, j_1] \cdots \mathcal{T}^{(m)}[i_m, j_m] \quad (9)$$

where $\mathcal{T}^{(k)}[j_k, i_k]$ is a $D_{k-1} \times D_k$ matrix with D_k the virtual basis dimension on the bond linking $\mathcal{T}^{(k)}$ and $\mathcal{T}^{(k+1)}$ with $D_0 = D_m = 1$. With Eq. (9) we can decompose an original matrix $\mathbf{W}$ to a sequential product of the derived local tensors.

A.2 Theorem

Theorem 1. *Suppose that the tensor $\boldsymbol{W}^{(k)}$ of matrix W that is satisfy*

$$\mathbf{W} = \mathbf{W}^{(k)} + \mathbf{E}^{(k)}, D(\boldsymbol{W}^{(k)}) = d_k,$$
$$where \quad ||\mathbf{E}^{(k)}||_F^2 = \epsilon_k^2, k = 1, ..., d-1. \quad (10)$$

Then MPO$(\mathbf{W})$ with the k-th bond dimension d_k upper bound of truncation error satisfy:

$$||\mathbf{W} - \text{MPO}\ (\mathbf{W})||_F \leq \sqrt{\sum_{k=1}^{d-1} \epsilon_k^2} \quad (11)$$

Proof. The proof is by induction. For $n = 2$ the statement follows from the properties of the SVD. Consider an arbitrary $n > 2$. Then the first unfolding $\mathbf{W}^{(1)}$ is decomposed as

$$\mathbf{W}^{(1)} = \mathbf{U}_1\lambda_1\mathbf{V}_1 + \mathbf{E}^{(1)} = \mathbf{U}_1\mathbf{B}^{(1)} + \mathbf{E}^{(1)}, \quad (12)$$

where $\mathbf{U}_1$ is of size $r_1 \times i_1 \times j_1$ and $||\mathbf{E}^{(1)}||_F^2 = \epsilon_1^2$. The matrix $\mathbf{B}_1$ is naturally associated with a $(n-1)$-dimensional tensor $\mathcal{B}^{(1)}$ with elements $\mathcal{B}^{(1)}(\alpha, i_2, j_2, ..., i_n, j_n)$, which will be decomposed further. This means that $\mathbf{B}_1$ will be approximated by some other matrix $\hat{\mathbf{B}}_1$. From the properties of the SVD it follows that $\mathbf{U}_1^T\mathbf{E}^{(1)} = 0$, and thus

$$\begin{aligned} &||\mathbf{W} - \mathcal{B}^{(1)}||_F^2 \\ &= ||\mathbf{W}_1 - \mathbf{U}_1\hat{\mathbf{B}}_1||_F^2 \\ &= ||\mathbf{W}_1 - \mathbf{U}_1(\hat{\mathbf{B}}_1 + \mathbf{B}_1 - \mathbf{B}_1)||_F^2 \\ &= ||\mathbf{W}_1 - \mathbf{U}_1\mathbf{B}_1||_F^2 + ||\mathbf{U}_1(\hat{\mathbf{B}}_1 - \mathbf{B}_1)||_F^2 \quad (13) \end{aligned}$$

and since $\mathbf{U}_1$ has orthonormal columns,

$$||\mathbf{W} - \mathcal{B}^{(1)}||_F^2 \leq \epsilon_1^2 + ||\mathbf{B}_1 - \hat{\mathbf{B}}_1||_F^2. \quad (14)$$

and thus it is not difficult to see from the orthonormality of columns of $\mathbf{U}_1$ that the distance of the k-th unfolding $(k = 2, ..., d_k - 1)$ of the $(d-1)$-dimensional tensor $\mathcal{B}^{(1)}$ to the d_k-th rank matrix cannot be larger than ϵ_k. Proceeding by induction, we have

$$||\mathbf{B}_1 - \hat{\mathbf{B}}_1||_F^2 \leq \sum_{k=2}^{d-1} \epsilon_k^2, \quad (15)$$

combine with Eq. (14), this complets the proof.

Experiments	N	n	Feed-forward Network	Multi-head Attention	LR
BERT-small					
OPF+SVD	-	-	$\mathcal{T}_{32,16}^{32,32}(D)$	$\mathcal{T}_{32,16}^{16,32}(D)$	3e-5
OPF+MPO$_S$	4	4	$\mathcal{T}_{32,1,1,1,1,1,1,1,1,1,1,32}^{32,1,1,1,1,1,1,1,1,1,1,32}(D)$	$\mathcal{T}_{32,1,1,1,1,1,1,1,1,1,1,16}^{16,1,1,1,1,1,1,1,1,1,1,32}(D)$	3e-5
OPF+MPO$_D$	4	2	$\mathcal{T}_{32,1,1,1,1,1,1,1,1,1,1,16}^{32,1,1,1,1,1,1,1,1,1,1,32}(D)$	$\mathcal{T}_{32,1,1,1,1,1,1,1,1,1,1,16}^{16,1,1,1,1,1,1,1,1,1,1,32}(D)$	3e-5
BERT-medium					
OPF+SVD	-	-	$\mathcal{T}_{32,16}^{32,32}(D)$	$\mathcal{T}_{32,16}^{16,32}(D)$	3e-5
OPF+MPO$_S$	12	12	$\mathcal{T}_{32,1,1,1,1,1,1,1,1,1,1,32}^{32,1,1,1,1,1,1,1,1,1,1,32}(D)$	$\mathcal{T}_{32,1,1,1,1,1,1,1,1,1,1,16}^{16,1,1,1,1,1,1,1,1,1,1,32}(D)$	3e-5
OPF+MPO$_D$	12	2	$\mathcal{T}_{32,1,1,1,1,1,1,1,1,1,1,16}^{32,1,1,1,1,1,1,1,1,1,1,32}(D)$	$\mathcal{T}_{32,1,1,1,1,1,1,1,1,1,1,16}^{16,1,1,1,1,1,1,1,1,1,1,32}(D)$	3e-5
BERT-base					
OPF+SVD	-	-	$\mathcal{T}_{64,48}^{32,24}(D)$	$\mathcal{T}_{32,24}^{24,32}(D)$	3e-5
OPF+MPO$_S$	8	8	$\mathcal{T}_{64,1,1,1,1,1,1,1,1,1,1,24}^{32,1,1,1,1,1,1,1,1,1,1,24}(D)$	$\mathcal{T}_{64,1,1,1,1,1,1,1,1,1,1,48}^{32,1,1,1,1,1,1,1,1,1,1,24}(D)$	3e-5
OPF+MPO$_D$	8	2	$\mathcal{T}_{64,1,1,1,1,1,1,1,1,1,1,24}^{32,1,1,1,1,1,1,1,1,1,1,24}(D)$	$\mathcal{T}_{64,1,1,1,1,1,1,1,1,1,1,48}^{32,1,1,1,1,1,1,1,1,1,1,24}(D)$	3e-5
BERT-large					
OPF+SVD	-	-	$\mathcal{T}_{64,64}^{32,32}(D)$	$\mathcal{T}_{32,32}^{32,32}(D)$	3e-5
OPF+MPO$_S$	24	24	$\mathcal{T}_{64,1,1,1,1,1,1,1,1,1,1,64}^{32,1,1,1,1,1,1,1,1,1,1,32}(D)$	$\mathcal{T}_{64,1,1,1,1,1,1,1,1,1,1,64}^{32,1,1,1,1,1,1,1,1,1,1,32}(D)$	3e-5
OPF+MPO$_D$	24	4	$\mathcal{T}_{64,1,1,1,1,1,1,1,1,1,1,64}^{32,1,1,1,1,1,1,1,1,1,1,32}(D)$	$\mathcal{T}_{64,1,1,1,1,1,1,1,1,1,1,64}^{32,1,1,1,1,1,1,1,1,1,1,32}(D)$	3e-5

Table 4: The experiments setting in various BERT models."LR" denote the learning rate.

B Additional Experimental Details

Experimental Details in Pre-trained Language Modeling In this paper, the MPO decomposition is proposed for enlarging model parameters. In order to show the process of incorporating several MPO structures into BERT, T5, and BART respectively. Moreover, we denote an MPO, defined by Eq. (2), as:

$$\mathcal{T}_{i_1,i_2,\ldots,i_m}^{j_1,j_2,\ldots,j_m}(D) \tag{16}$$

BERT-small and BERT-medium models were released by Google Research in (Turc et al., 2019), which have 4 and 8 Transformer layers respectively, and each of the layers has 512 hidden nodes. We perform the pre-compute stage, which would compute the importance scores of all parameter matrices before fine-tuning and then leverages MPO to over-parameterize the top-N ones. The important score can be computed by Eq. (6) and Eq. (7). In our main experiments, the detail hyper-parameter setting was shown in Table 4.

We have conducted sufficient trials for both the T5 and the BART models using the following particular experimental parameter configurations in Table 5.

Hardware We trained our model on one machine with 8 NVIDIA V100 GPUs. For our base models, we adopt all these models released by Huggingface [1]. The hyperparameters are described in Table 6.

Optimizer We used the Adam optimizer and vary the learning rate over the course of training. The vary formula (Vaswani et al., 2017) follows in our work. We also used the $warmup\_steps = 1000$.

Details of Fine-tuning Datasets GLUE benchmark covers multiple datasets (MNLI, QNLI, SST-2, RTE, QQP, CoLA, STS-B, MRPC) [2].

Details of Evaluation Metrics Following Gao et al. (2022), the metrics that we use for the GLUE benchmark are Matthew's correlation for CoLA (Mcc.), Spearman for STS-B (Spear.), F1 for MRPC, and accuracy (Acc.) for the remaining tasks. We compute and report the average scores for all of the aforementioned metrics based on all of the test samples taken into consideration. Since the original test sets are not accessible, we divide the original validation set in half and use one half for validation and the other for the test for datasets with fewer than 10,000 samples (RTE, MRPC, STS-B, CoLA) (Zhang et al., 2020a).

[1] https://huggingface.co/

[2] In line with Raffel et al. (2020), we do not test WNLI due to its adversarial character with respect to the training set.

Experiments	N	n	Feed-forward Network	Multi-head Attention	LR
T5-base					
OPF+MPO$_S$	8	8	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,24}_{64,1,1,1,1,1,1,1,1,1,48}(D)$	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,24}_{64,1,1,1,1,1,1,1,1,1,48}(D)$	3e-5
OPF+MPO$_D$	8	2	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,24}_{64,1,1,1,1,1,1,1,1,1,48}(D)$	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,24}_{64,1,1,1,1,1,1,1,1,1,48}(D)$	3e-5
T5-large					
OPF+MPO$_S$	16	16	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,32}_{64,1,1,1,1,1,1,1,1,1,64}(D)$	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,32}_{64,1,1,1,1,1,1,1,1,1,64}(D)$	3e-5
OPF+MPO$_D$	16	4	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,32}_{64,1,1,1,1,1,1,1,1,1,64}(D)$	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,32}_{64,1,1,1,1,1,1,1,1,1,64}(D)$	3e-5
BART-base					
OPF+MPO$_S$	8	8	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,24}_{64,1,1,1,1,1,1,1,1,1,48}(D)$	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,24}_{64,1,1,1,1,1,1,1,1,1,48}(D)$	3e-5
OPF+MPO$_D$	8	2	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,24}_{64,1,1,1,1,1,1,1,1,1,48}(D)$	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,24}_{64,1,1,1,1,1,1,1,1,1,48}(D)$	3e-5
BART-large					
OPF+MPO$_S$	16	16	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,32}_{64,1,1,1,1,1,1,1,1,1,64}(D)$	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,32}_{64,1,1,1,1,1,1,1,1,1,64}(D)$	3e-5
OPF+MPO$_D$	16	4	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,32}_{64,1,1,1,1,1,1,1,1,1,64}(D)$	$\mathcal{T}^{32,1,1,1,1,1,1,1,1,1,32}_{64,1,1,1,1,1,1,1,1,1,64}(D)$	3e-5

Table 5: The experiments setting in T5 and BART models."LR" denote the learning rate.

Model	d_{head}	d_{ff}	L	N$_{sl}$
BERT-small	512	2048	4	128
BERT-medium	512	2048	8	128
BERT-base	768	3072	12	128
BERT-large	1024	4096	24	128
T5-base	768	3072	12	128
T5-large	1024	4096	24	128
BART-base	768	3072	12	128
BART-large	1024	4096	24	128

Table 6: The hyperparameter in experiments of the main text mentioned models. "L" denotes the number of Transformer layers. "N$_{sl}$" denotes the sequence length.

Category	Method	Inference Time
MPO	MPO$_{(m>2)}$	$\mathcal{O}(mID^3)$
	MPO$_{(m=2)}$(SVD)	$\mathcal{O}(2ID^3)$
Tucker	Tucker$_{(D>1)}$	$\mathcal{O}(mID + D^m)$
	Tucker$_{(D=1)}$(CP)	$\mathcal{O}(mID^2)$

Table 7: The amount of time and complexity that various low-rank approximation algorithms need for inference. Here, m denotes the number of the tensors, I denotes $\max(\{i_k\}_{k=1}^m)$ means the largest i_k in input list, and D denotes $\max(\{D'_k\}_{k=0}^m)$ means the largest dimension D'_k in the truncated dimension list.

C Additional Discussion

Different Tensor Decompostion In the field of mathematics, the MPO-based approximation may be seen as an alternative form of the low-rank approximation approach. Now we will evaluate it in light of many other low-rank approximation techniques, such as SVD (Henry and Hofrichter, 1992), CPD (Hitchcock, 1927), and Tucker decomposition (Tucker, 1966).

We present the categorization of these methods in Table 7. Because the work of low-rank decomposition only needs to be done once, and it does not take a long time, thus we mainly focus on the forward propagation time in practical use. In point of fact, each of the techniques may either be based on a tensor-based decomposition (that is, a list of tensors for factorization) or a matrix-based decomposition, and we quantify the amount of time each approach requires using standard parameters. Indeed, MPO and Tucker are examples of two different classes of low-rank approximation algorithms. In most cases, the capacity of the algorithm will rise in proportion to the value of m (more tensors). When m is more than three, the temporal complexity of MPO is lower than that of Tucker decomposition. It is clear that SVD may be thought of as a special example of MPO when the dimension of the tensor is equal to two, and that CPD is a particular case of Tucker when the super-diagonal matrix is the core tensor. Both of these relationships can be observed here.