Parameter-Efficient Mixture-of-Experts Architecture for Pre-trained Language Models

Ze-Feng Gao[1,4,5*] , **Peiyu Liu**[1,4*] , **Wayne Xin Zhao**[1,4†] , **Zhong-Yi Lu**[2] and **Ji-Rong Wen**[1,3,4]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Department of Physics, Renmin University of China
[3] School of Information, Renmin University of China
[4]Beijing Key Laboratory of Big Data Management and Analysis Methods
[5]Beijing Academy of Artificial Intelligence, Beijing, 100084, China
{zfgao,liupeiyustu,zlu,jrwen}@ruc.edu.cn, batmanfly@gmail.com

Abstract

Recently, Mixture-of-Experts (short as MoE) architecture has achieved remarkable success in increasing the model capacity of large-scale language models. However, MoE requires incorporating significantly more parameters than the base model being extended. In this paper, we propose building a parameter-efficient MoE architecture by sharing information among experts. We adopt matrix product operator (MPO, a tensor decomposition from quantum many-body physics) to reconstruct the parameter matrix in the expert layer and increase model capacity for pre-trained language models by sharing parameters of the central tensor (containing the core information) among different experts while enabling the specificity through the auxiliary tensors (complementing the central tensor) of different experts. To address the unbalanced optimization issue, we further design the gradient mask strategy for the MPO-based MoE architecture. Extensive experiments based on T5 and GPT-2 show improved performance and efficiency of the pre-trained language model (27.2x reduction in total parameters for the superior model performance, compared with the Switch Transformers). Our code is publicly available at `https://github.com/RUCAIBox/MPOE`.

1 Introduction

Large-scale pre-trained language models (PLMs), such as BERT (Devlin et al., 2018) and T5 (Raffel et al., 2020), have become the de facto standard in natural language processing (NLP). By involving a huge number of parameters pre-trained on the general-purpose corpus, PLMs can achieve excellent performance in many NLP tasks. In order to increase the model capacity, a promising direction is to explore the scaling properties with the mixture-of-experts (MoE) paradigm (Jacobs et al., 1991; Shazeer et al., 2017) for developing more powerful PLMs. By incorporating multiple expert networks, MoE schedules the learning of data samples through a routing component that is usually implemented by some gating function, which increases model capacity without a proportional increase in computation costs. Despite the effectiveness, it has been shown that the MoE architecture is parameter inefficient (Zuo et al., 2021), considering the yielded improvement *w.r.t.* the involved costs. Most of the existing studies (Yang et al., 2021; Roller et al., 2021; Lewis et al., 2021) attribute this issue to the unbalanced load of experts, focusing on improving the routing strategies.

However, an important question about the MoE architecture has been neglected in previous studies: whether the increased parameters from the experts are all necessary to increase the model capacity. As different experts from an MoE network are often trained with correlated data samples (*e.g.,* sample correlation from training data), it is likely to lead to parameter redundancy across experts. Indeed, *expert redundancy* has been identified in existing studies, where Fedus et al. (2021) distills sparse MoE models into dense models and Kim et al. (2021) prunes experts to compress MoE models. This finding motivates us to develop a parameter-efficient MoE architecture by reducing its parameter redundancy. Intuitively, a straightforward approach is to share a certain proportion of parameters among experts. However, it is difficult to identify and optimize the key parameters that encode the shared information across experts, since expert networks typically consist of dense matrices.

To address this issue, we propose a novel parameter sharing approach inspired by the matrix product operators (MPO) decomposition from quantum many-body physics (Gao et al., 2020), which decomposes a matrix into a sequential product of local tensors (either *central* or *auxiliary* tensors). Unlike other matrix decomposition methods, MPO

*Authors contributed equally.
†Corresponding author.

can effectively reorganize and aggregate important information of the original matrix into the *central tensor*. The auxiliary tensors, on the other hand, serve to complement the central tensor for recovering the original matrix (Liu et al., 2021). In the setting of MoE, considering the small parameter variations among experts, we speculate that the central tensors of different experts (with MPO decomposition for each expert) are likely to be very similar. If the central tensors could be shared for all expert networks, we would significantly reduce the parameters of the MoE architecture.

To this end, we propose a novel MPO-based parameter-efficient MoE architecture, called **MPOE**. Based on classic MoE architecture (Shazeer et al., 2017), our approach introduces a major extension allowing experts to share a global central tensor while keeping expert-specific auxiliary tensors. In our setting, the parameter matrix of in a single expert is formed by the product of the globally shared central tensor and the corresponding auxiliary tensors. Since the central tensor contains most of the parameters from an MPO decomposition, our MPOE approach can significantly reduce the parameters of the original MoE architecture. Another major merit of MPO is that auxiliary tensors are closely entangled with the central tensor (Pirvu et al., 2010), and it is theoretically guaranteed that any change from auxiliary tensors can be propagated to the central tensor. That is to say, though a large proportion of parameters are shared, local auxiliary tensors still enable the experts to capture specific variations or differences according to routing data samples. However, directly optimizing the MPOE architecture is likely to lead to an *unbalanced optimization* issue, where the central tensors are updated more frequently than auxiliary tensors during fine-tuning. Therefore, we further propose a gradient mask strategy that masks the central tensor gradient to effectively alleviate the unbalanced optimization issue.

To the best of our knowledge, this is the first attempt to reduce the parameter redundancy of the MoE architecture with structural matrix decomposition. We conduct extensive experiments to evaluate the effectiveness of the MPOE architecture on two representatives PLMs, T5 and GPT. Experiments have demonstrated the effectiveness of our approach in increasing model capacity (27.2x fewer parameters for the superior model performance, compared with several competitive MoE-enhanced PLMs.

2 Preliminary

2.1 Mixture-of-Experts (MoE)

We first describe the mixture-of-experts architecture (MoE) (Shazeer et al., 2017), which has been used to enhance the model capacity of Transformer based models. Let $G(x)$ and $E_i(x)$ denote the output vectors of the gating network and the output of the i-th expert network for a given input x, respectively. The output of MoE architecture y can be formally computed as:

$$y = \sum_{i=1}^{n} G(x) \cdot E_i(x). \tag{1}$$

The $\mathrm{softmax}$ function is widely adopted as the gating function $G(x)$. The sparsely-gated MoE architecture, which uses a noisy top-k gating mechanism to reduce the computational cost, has been proposed in Shazeer et al. (2017). It adds tunable Gaussian noise with $H(\cdot)$, and then keeps only the top-k values with $\mathrm{KeepTopK}(\cdot)$ and sets the rest $-\infty$. This keeps only the top k experts to be evaluated with:

$$G(x) = \text{softmax}(\text{KeepTopK}(H(x), k)). \tag{2}$$

Furthermore, Switch Transformer designs a switch routing strategy to simplify this gating function by routing to a *single* expert (Fedus et al., 2021).

2.2 Tensor and Matrix Product Operators

We refer to one-dimensional arrays as *vectors* (denoted by bold lowercase letters, *e.g.,* $\boldsymbol{v}$), two-dimensional arrays as *matrices* (denoted by bold uppercase letters, *e.g.,* $\mathbf{W}$), and arrays of higher dimensions as *tensors* (denoted by calligraphic bold uppercase letters, *e.g.,* $\mathcal{T}$).

MPO decomposition (Oseledets, 2011) (*a.k.a.* tensor-train decomposition) has been a widely used matrix decomposition technique from quantum many-body physics, which decomposes a matrix (2-order tensor) into m local tensors (Pirvu et al., 2010). Given a matrix $\mathbf{W}_{I\times J} \in \mathbb{R}^{I\times J}$, the MPO decomposition is given in the following format:

$$\text{MPO}\ (\mathbf{W}) = \prod_{k=1}^{m} \mathcal{T}_{(k)}[d_{k-1}, i_k, j_k, d_k], \tag{3}$$

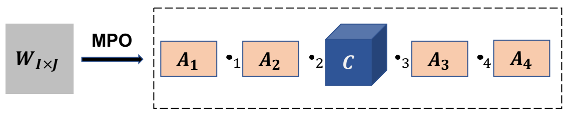


Figure 1: MPO decomposition for matrix $\mathbf{W}_{I\times J}$ with five local tensors. Auxiliary tensors ($\{\mathcal{A}_i\}_{i=1}^4$) and central tensor ($\mathcal{C}$) are marked in orange and blue, respectively.

where $I = \prod_{k=1}^{n} i_k$ and $J = \prod_{k=1}^{n} j_k$, $\mathcal{T}_{(k)}$ is a 4-order tensor with size $d_{k-1} \times i_k \times j_k \times d_k$. The d_k is dimension of bond linking $\mathcal{T}_{(k)}$ and $\mathcal{T}_{(k+1)}$.

According to Gao et al. (2020), the original matrix $\mathbf{W}$ can be exactly reconstructed by tensor contraction of MPO($\mathbf{W}$) without truncation of the connection bond $\{d_k\}_{k=1}^{m}$. Figure 1 presents the illustration of the MPO decomposition procedure for a matrix ($m = 5$). More detailed analysis on different factorization ways (*i.e.,* $m = 3, 5, 7, 9$) will be given in Section 4.5. After MPO decomposition, the central tensor (the tensor right in the middle) with most of the parameters can encode the core information of the original matrix, while the auxiliary tensors (the rest of these tensors) with only a small proportion of parameters play the role of complementing the central tensor.

3 Approach

To reduce the information redundancy across different experts, we design an MPO-based MoE architecture for increasing the model capacity in a parameter-efficient way. We firstly describe the MPO-based MoE architecture and then introduce an improved optimization algorithm for learning the parameters in this architecture.

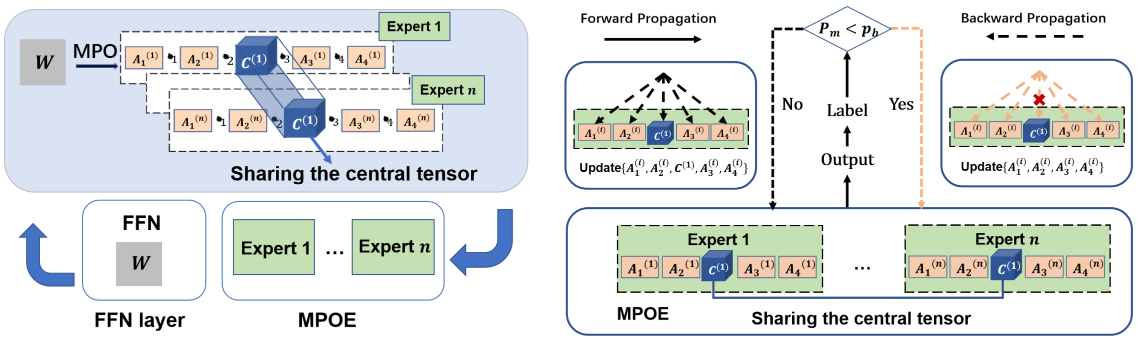
3.1 MPO-based Mixture-of-Experts

Previous MoE architecture (Jacobs et al., 1991; Shazeer et al., 2017) usually treats different experts as individual components, requiring a compete copy of network parameters for each expert. Although it has been found (Fedus et al., 2021; Kim et al., 2021) that there exists redundant information among different experts in the MoE architecture, it is not easy to identify the shareable parameters from the highly coupling network.

Considering this issue, our solution is inspired by an important merit of MPO decomposition: it can reorganize and aggregate the core information in central tensors (Gao et al., 2020) as aforementioned. Based on this property, the core idea of our approach is to share the central tensors for all the expert layers and enable specificity via expert-specific auxiliary tensors.

Parameter-Efficient MoE Architecture. The Transformer network consists of two major neural components, namely FFN and multi-head attention. Following previous work on MoE-based PLMs (Shazeer et al., 2017; Fedus et al., 2021), we consider FFN layers as experts to be extended, while our approach is generally applicable to various matrix-based model components. A straightforward method to reducing information redundancy is to share a proportion of parameters across experts. However, in Transformer-based networks, the experts (*i.e.,* FFN here) are mainly composed of large dense matrices, which are difficult for sharing partial parameters from these matrices. As our solution, we consider parameter sharing through the MPO decomposition, so that the derived central tensors can be flexibly shared across matrices.

Lightweight MoE Design. Specifically, we simplify the discussion by assuming that an expert corresponds to one parameter matrix at each layer, and it is similar for the multi-matrix cases. We consider a MoE architecture of n experts each with L layers, so that there are $L \times n$ matrices in total, denoted by $\{\mathbf{W}^{(l,i)}\}_{l=1,i=1}^{L,n}$. As discussed in Section 2.2, a matrix can be decomposed into m tensors, consisting of one central tensor and $m-1$ auxiliary tensors. In this work, we consider five decomposed tensors, *i.e.,* $m = 5$. At the l-th layer, the decomposition results can be denoted by $\{\mathcal{C}^{(l,i)}, \mathcal{A}_1^{(l,i)}, \mathcal{A}_2^{(l,i)}, \mathcal{A}_3^{(l,i)}, \mathcal{A}_4^{(l,i)}\}_{i=1}^{n}$, where $\mathcal{C}^{(l,i)}$ and $\mathcal{A}_{\cdot}^{(l,i)}$ are the central and auxiliary tensors of the i-th parameter matrix, respectively, at the l-th layer. To develop the MPO-based MoE architecture, the core idea is to share the central tensors as global parameters and keep expert-specific auxiliary tensors as local parameters, *i.e.,* $\mathcal{C}^{(l,1)} = \mathcal{C}^{(l,2)} \cdots = \mathcal{C}^{(l,n)}$ ($\forall\ l = 1 \cdots L$), and we denote the global central tensor at the l-th layer by $\mathcal{C}^{(l)}$. In this way, we can only keep L central censors for a L-layer MoE architecture. For MPO, the decomposition process is transparent to external modules, so that we can reuse the previous routing mechanism (Section 2.1) by distributing data samples to different experts. A slight difference is that we only need to consider the routing to local tensors for each matrix since the global tensor is shared across experts. We call such an MPO-based MoE architecture as **MPOE**.


W
MPO
Expert 1
Expert n
Sharing the central tensor
FFN
FFN layer
Expert 1
Expert n
MPOE
Forward Propagation
Backward Propagation
$P_m < p_b$
No
Yes
Label
Output
Update$\{A_1^{(i)}, A_2^{(i)}, C^{(1)}, A_3^{(i)}, A_4^{(i)}\}$
Update$\{A_1^{(i)}, A_2^{(i)}, A_3^{(i)}, A_4^{(i)}\}$
MPOE
Sharing the central tensor

(a) MPO-based mixture-of-experts architecture.

(b) Gradient mask strategy.

Figure 2: Illustration the proposed MPOE architecture and gradient mask strategy. We decompose the weight matrix of each expert in the MoE architecture into five local tensors using MPO, containing four auxiliary tensors and one central tensor, which are marked in orange and blue, respectively. In our approach, the central tensor of the n experts is shared in the MPOE architecture. During optimization, each backward propagation process updates a set of auxiliary tensors while updating the central tensor with a probability of p_b (the mask probability of the central tensor), which can effectively avoid the unbalanced optimization of the central tensor.

Discussion. Since the central tensor contains most of the information from original parameter matrices (Gao et al., 2020), a key question is whether the current architecture enables sufficient flexibility and specificity for each expert. To answer this question, we refer to an important property of MPO decomposition from quantum many-body physics (Pirvu et al., 2010): it is guaranteed in principle, that any change on one tensor will be propagated to the entire local tensor set. In other words, only tuning the auxiliary tensors (keeping the central tensor fixed) can lead to the same effect as tuning the whole matrix. Since the parameters of the central tensor are shared, our approach can significantly reduce the number of actual parameters given the MoE architecture with the same number of experts. Assuming the original model consisting of n experts with T parameters each, we have a total number of $n \cdot T$ parameters. Specifically, let γ denote the parameter ratio of the auxiliary tensor to the central tensor for expert networks. Given the total number T for an expert network, the central and auxiliary tensors correspond to the parameters numbers of $\frac{\gamma}{\gamma+1}T$ and $\frac{1}{\gamma+1}T$, respectively. Since our MPOE approach shares the central tensor, the final number of parameters will be $\frac{\gamma}{\gamma+1}T + \frac{n}{\gamma+1}T$. Thus, our MPOE approaches corresponds to a ratio of $\frac{n+\gamma}{n(\gamma+1)}$ of the original parameter scale. In our experiments, the ratio γ is about 12, and $\frac{n+\gamma}{n(\gamma+1)}$ approximately equals to 0.19 when $n = 8$. Such a ratio will be further decreased when we have more experts. It can be seen that our MPOE approach is able to effectively reduce the parameter scale.

3.2 Alleviate Unbalanced Optimization

As the experts share the central tensor in the MPOE approach, the corresponding parameters of the central tensor will be updated more frequently than those in the auxiliary tensors during fine-tuning. It tends to lead to the unbalanced optimization issue as reported by Xu et al. (2021), due to deviation from the pre-trained weights. As a result, it is crucial to develop a more stable optimization technique that is suited to the MPOE architecture.

Inspired by the solution of gradient dropout strategy (Tseng et al., 2020; Xu et al., 2021), we propose to mask the gradients for the central tensor to improve model optimization for the MPO-based MoE architecture. At each iteration, we take a certain probability p_b to discard the update in the central tensor. This can effectively alleviate the unbalanced optimization which is caused by the frequent updates of the central tensor. Specifically, we generate a binary mask b drawn from Bernoulli distribution with a mask probability p_b, which can be calculated by $b \sim \text{Bernoulli}(p_b)$. We denote the $\Delta\mathcal{C}$ as the update of the central tensor at each iteration:

$$\Delta\mathcal{C} = \eta \frac{\partial \mathcal{L}(\mathcal{C})}{\partial \mathcal{C}} \odot (1 - b). \tag{4}$$

The larger p_b is, the less frequently the central tensor is updated. In particular, when p_b is equal to 1, it means that the parameters of the central tensor are frozen for each input of the data. The computational cost of central tensor update can be also reduced with this trick.

Note that the gradient mask trick is only applied to central tensors. For auxiliary tensors, we perform the standard gradient update for learning the parameters. Compared with two alternative ways to implement the gradient mask technique, *i.e.,* mask pre-activation or post-activation in FFN layers, we find that such a sampling-based masking strategy can effectively improve the model performance in our experiments.

3.3 The Overall Algorithm Procedure

Our approach can be generally applied to various MoE-based models for increasing the model capacity. In this work, we adopt the MoE-extended PLMs (Radford et al., 2019) for study.

Algorithm 1 presents a complete procedure for the proposed update procedure, which can be briefly summarized as follows. First, we obtain the PLM and perform MPO decomposition for each weight matrix of the FFN layers in the Transformer. For each weight matrix, we decompose it into one central tensor $\mathcal{C}$ and a list of auxiliary tensors $\mathcal{A}$. In the original MoE architecture, we will have n sets of such decomposed parameters. Next, the key point lies in that we share the central tensor $\mathcal{C}$ in the decomposition process but keep expert-specific auxiliary tensors. In this way, each expert is composed of a set of auxiliary tensors and a shared central tensor. To recover the original FFN matrix in some specific expert, we can simply multiply the shared central tensor by expert-specific auxiliary tensors. Then, we apply the gradient mask strategy to update the parameters in these experts, *i.e.,* masking the gradient of the central tensor.

Since the parameters of the central tensor are two orders of magnitude larger than the parameters of the auxiliary tensors (Liu et al., 2021), the cost of MoE-based networks will be largely reduced by sharing the central tensor.

Algorithm 1 The proposed updating procedure.

```
Require: $\{\{\mathcal{A}_j\}_{j=1}^4, \mathcal{C}\}$: Initialize experts
Require: $\alpha$: learning rate
Require: $p_b$: mask probability
Require: time step $t \leftarrow 0$ (Initialize timestep)
1: while not converged do
2:     $t \leftarrow t + 1$
3:     $g_{\mathcal{C}}^t \leftarrow \frac{\partial \mathcal{L}(\mathcal{C}^t)}{\partial(\mathcal{C}^t)}$,  $g_{\mathcal{A}}^t \leftarrow \frac{\partial \mathcal{L}(\mathcal{A}^t)}{\partial(\mathcal{A}^t)}$
       (Get gradients at timestep $t$)
4:     $b \leftarrow \text{GenerateMask}(p_b)$
       (Compute gradient mask)
5:     $\mathcal{C}^t \leftarrow \mathcal{C}^{t-1} - \alpha \cdot g_{\mathcal{C}}^t \odot (1 - b)$
       (Update central tensors)
6:     $\mathcal{A}^t \leftarrow \mathcal{A}^{t-1} - \alpha \cdot g_{\mathcal{A}}^t$
       (Update the routed auxiliary tensors)
7: end while
8: return $\{\{\mathcal{A}_j^t\}_{j=1}^4, \mathcal{C}^t\}$ (Resulting parameters)
```

3.4 Discussion

For the parameter inefficiency issue of MoE-based networks, existing studies mainly focus on alleviating the unbalanced load of experts, which have proposed different routing methods to balance the routing probabilities of different experts, such as BASE-Layer (Lewis et al., 2021), HASHLayer (Roller et al., 2021), GShard (Lepikhin et al., 2021) and Switch Transformers (Fedus et al., 2021). As a comparison, we aim to reduce information redundancy by sharing common parameters among experts. Actually, the MPOE approach can be further enhanced with existing improved routing methods.

Specifically, Deepspeed-MoE proposed to use pyramid residual MoE architecture to reduce the parameters of the MoE architecture (Rajbhandari et al., 2022), while our work takes a different perspective to improve the original MoE architecture by sharing parameters among different experts.

4 Experiments

In this section, we first set up the experiments and then report the results and analysis. Then, we conduct a detailed analysis under different experimental settings. Here, we use T5 (Raffel et al., 2020) and GPT-2 (Radford et al., 2019) models as the base model in our experiments.

4.1 Experimental Setup

Datasets. To evaluate the effectiveness of the proposed MPOE as an efficient strategy to improve the model capacity of PLMs, we follow the setting of T5 and GPT-2 to perform experiments on Natural Language Understanding (NLU) and Natural Language Generation (NLG) tasks. Specifically, we evaluate the NLG tasks in GLUE benchmark (Wang et al., 2018), the language modeling task with WikiText-2 (Merity et al., 2017), the text generation task with IMDB (Maas et al., 2011) and EMNLP2017 WMT News (Guo et al., 2018). Furthermore, we follow the setup of Raffel et al. (2020) on the GLUE benchmark for a direct comparison with the T5 model.

GLUE benchmark covers multiple datasets

NLU with T5

Experiments	MNLI	QNLI	SST-2	RTE	QQP	CoLA	MRPC	STS-B	Avg.	#To (M)
T5-Large	**89.23**	94.03	96.20	83.94	**91.54**	55.10	90.15	91.90	86.51	737
+MoEfication♦	87.50	93.20	95.40	86.40	90.20	55.50	87.50	90.60	85.79	737
+MoEfication$_{++}$ ♦	88.70	93.60	96.20	87.50	91.30	59.40	89.30	91.00	87.13	737
+Switch♣	/	/	/	/	/	/	/	/	88.50	26000
+MPOE	87.16	**94.12**	**96.80**	**88.60**	90.63	**67.63**	**93.65**	**91.97**	**88.82**	956
T5-Base	87.78	93.82	94.72	71.74	91.11	53.49	89.16	91.16	84.12	223
+Switch♣	/	/	/	/	/	/	/	/	86.70	3800
+Switch♠	87.73	93.85	**94.87**	**77.53**	91.59	59.90	91.64	91.16	86.03	1015
+MoE★	86.98	92.82	94.60	69.56	90.02	64.56	87.68	90.89	84.64	1015
+MPOE	87.60	93.30	94.81	77.13	90.81	65.53	**93.14**	91.17	86.69	294
+MPOE$_{++}$	**87.78**	**93.93**	94.83	77.42	**91.61**	**65.90**	91.14	**91.65**	**86.78**	365

NLG with GPT-2

Experiments	WikiText-2			EMNLP News		IMDB		#To (M)
	PPL (↓)	BLEU-2	BLEU-4	BLEU-2	Self-BLEU-2	BLEU-2	Self-BLEU-2	
GPT-2	21.27	28.69	9.46	62.61	74.67	73.12	83.85	124
+MoE★	21.86	28.27	9.14	65.27	79.79	74.46	90.01	578
+Switch♠	21.25	28.71	9.44	64.62	81.11	75.35	91.82	578
+MPOE	**20.72**	28.78	9.51	66.99	83.10	76.30	92.72	157
+MPOE$_{++}$	20.73	**28.82**	**9.57**	**68.49**	**83.11**	**76.82**	**93.08**	171

Table 1: Performance comparison of different models on NLU and NLG tasks (in percent). "#To (M)" denote the number (in millions) of total parameters. We set the number of experts $n = 8$ in these models, MPOE. Furthermore, we use $n = 16$ for a more powerful version of our approach, denoted by MPOE$_{++}$. We report the average test performance of three runs, and the best results are highlighted in bold. ♦: Experimental results by Zhang et al. (2021b) ♣: Experimental results by Fedus et al. (2021) ♠: Our re-implementation by Fedus et al. (2021). ★:Apply method by Shazeer et al. (2017).

(MRPC, QQP, SST-2, MNLI, RTE, QNLI, CoLA)[1]. The original test sets are not publicly available, and following Zhang et al. (2021a), for datasets fewer than $10K$ samples (RTE, MRPC, STS-B, CoLA), we divide the original validation set in half, using one half for validation and the others for the test.

Evaluation Metrics. We use perplexity (PPL) (Brown et al., 1992) to measure how well the probability model predicts a sample compared with the ground-truth. To evaluate the ratios of the overlapping n-grams between generated and real samples, we use BLEU-n score (Papineni et al., 2002). We also take into account the Self-BLEU-n score (Zhu et al., 2018) to evaluate the diversity of generated samples especially. For metrics used in the GLUE benchmark, we follow Mahabadi et al. (2021) and use Matthew's correlation for CoLA, Pearson for STS-B, and accuracy for the other tasks.

Comparison methods. We adopt the T5 and GPT-2 as the base architectures for both MoE and MPOE. Following Shazeer et al. (2017), we extend the FFN components with the MoE architecture containing n experts in each Transformer block of the T5 and GPT-2 model. We refer to this method as "+MoE". The Switch Transformers (Fedus et al., 2021) use a simplified strategy that routes to only a single expert instead of top-2 routing in MoE. We refer to this method as "+Switch". To ensure a fair comparison, we maintain the same number ($n = 8$) of experts for baselines and MPOE. We also implement an enhanced version of MPOE with $n = 16$ experts, which is referred to as "+MPOE$_{++}$". Based on the released *gpt2* model[2], *t5-base* model[3] and *t5-large* model[4] provided by Huggingface, we first initialize the experts, then fine-tune the models on the downstream tasks. For the T5 model, we follow the setting in Mahabadi et al. (2021) and fine-tune all parameters of the model on all tasks. For different downstream tasks, we run a hyperparameter sweep and select the best configuration according to the accuracy results on the validation set. The hyperparameters that we tune include the epochs, batch size and learning rates.

[1]Following Raffel et al. (2020), as a common practice, due to the adversarial nature of WNLI with respect to the training set, we do not experiment with WNLI

[2]https://huggingface.co/gpt2

[3]https://huggingface.co/t5-base

[4]https://huggingface.co/t5-large

4.2 Mains Results

In our main experiments, we adopt T5 (Raffel et al., 2020), GPT-2 (Radford et al., 2019), Switch Transformers (Fedus et al., 2021) and MoEfication (Zhang et al., 2021b) as baselines, and report the comparison results of both NLU and NLG tasks in Table 1.

Overall, compared to these MoE variants, our proposed MPOE approach achieves performance improvement while being more parameter-efficient. For the NLU task, our proposed approach ("+MPOE") outperforms the best baseline method, *i.e.,* "+Switch" (88.82 vs. 88.50 for T5-Large) with up to 27.2x reduction in total parameters in the GLUE benchmark. By zooming into low-resource datasets such as CoLA and MRPC, our approach yields more significant improvements. This suggests that sharing parameters across experts reinforces the positive transfer effects[5] of information from other datasets toward the learning of low-resource datasets. For the NLG task, GPT-2+MPOE achieves gains in BLEU-2 score (1.72 for GPT-2+MoE and 2.37 for GPT-2+Switch) with 3.7x reduction in total parameters on the EMNLP News dataset. This indicates that GPT-2 also benefits from sharing central tensors.

Moreover, T5+MPOE$_{++}$ and GPT-2+MPOE$_{++}$ perform better when we add more auxiliary tensors as additional experts. This demonstrates the necessity of improving model capacity (Shazeer et al., 2017), as more parameters of experts tend to result in an improved model capacity.

4.3 Evaluation on Multi-task Learning

To demonstrate the efficiency of MPOE in multi-task learning, we adopt the T5-Base model for analysis to be comparable with Hyperformer (Mahabadi et al., 2021). We conduct experiments on the multi-task GLUE benchmark. The detailed metrics can be found in Section 4.1. Note that compared to Hyperformer, MPOE approach does not incorporate additional neural network components, thus it is more flexible to be used with the PLMs.

Table 2 shows the results on GLUE benchmark for T5-base (Raffel et al., 2020), Hyperformer (Mahabadi et al., 2021) and MPOE. As we can see, the performance of the MPOE approach is consistently better than the Hypernetwork in all cases, while

[5]Here, the positive transfer effects can be referred to Mahabadi et al. (2021), which means that the transferred knowledge can lead to improved performance for unseen in-domain tasks.

Datasets	T5-Base	Hyper♣	+MPOE
MNLI (acc)	87.73	85.74	**87.83**
QNLI (acc)	93.51	93.02	**93.89**
SST-2 (acc)	92.50	94.03	**94.73**
RTE (acc)	75.41	75.36	**75.51**
QQP (acc)	91.12	90.28	**91.17**
CoLA (mcc)	54.93	63.73	**65.85**
MRPC (acc)	89.21	89.66	**90.10**
STS-B (pearson)	90.75	90.00	**90.92**
Avg.	84.39	85.23	**86.25**
#To (M)	223	343	258

Table 2: Performance of multi-task learning on GLUE benchmark obtained by fine-tuning T5-Base (in percent). ♣: Experimental results from Hyperformer (Mahabadi et al., 2021).

Variants	WikiText-2			#To (M)
	PPL (↓)	B2	B4	
+MoE★	21.86	28.27	9.14	578
w/o PS	21.28	28.67	9.44	153
w/o GM	21.17	28.71	9.47	157
+MPOE	**20.72**	**28.78**	**9.51**	157

Table 3: Ablation study on the WikiText-2 dataset about the NLG tasks (in percent). "B2" and "B4" are short for BLEU-2 and BLEU-4, respectively. ★: The method from Shazeer et al. (2017)

the MPOE is more parameter-efficient (258M vs. 343M in total parameters). It further demonstrates the potential benefits of the MPOE approach in a multi-task learning setting, where the central tensor learns common information across tasks and the auxiliary tensor learns task-specific information.

4.4 Ablation Results

Our approach has incorporated two novel improvements: (1) MoE architecture with parameters sharing (PS) among experts based on MPO decomposition and (2) gradient mask (GM) to alleviate unbalanced optimization.

To verify the effectiveness of each component, we conduct the ablation study on the WikiText-2 dataset to analyze the contribution of each part. We adopt PPL, BLEU-2 and BLEU-4 as the evaluation metrics, and consider removing the parameters sharing and gradient mask strategy respectively. The ablation results of our MPOE approach are shown in Table 3. We can see that removing any component would lead to a decrease in the model performance. It shows the effectiveness of all these components in our approach. Besides, parameter sharing seems more important than the gradient

Variants	WikiText-2 PPL (↓)	B2	B4	#To (M)
GPT-2	21.27	28.69	9.46	124.4
MPOE (m=3)	24.01	27.86	8.93	130.3
MPOE (m=5)	20.72	28.77	9.48	157.4
MPOE (m=7)	20.73	28.76	9.47	198.7
MPOE (m=9)	20.78	28.45	9.38	214.6

Table 4: Evaluation with different factorization manner on the WikiText-2 dataset about the NLG tasks (in percent). "B2" and "B4" are short for BLEU-2 and BLEU-4, respectively.

mask strategy, which yields a larger performance drop after being removed.

4.5 Detailed Analysis

MPO decomposition has different factorization manners. However, the MPOE approach requires a defined MPO decomposition form to be given before it can be used. Therefore, different factorization manners may affect the efficiency of the MPOE approach. To vertify this, we perform a detailed analysis on different factorization manners of MPO decomposition. We present three variants of MPOE with different lengths of local tensors produced by MPO decomposition empirically. Tabel 4 shows the evaluation results on the WikiText-2 dataset about NLG tasks. As we can see, the variants of $m > 3$ are all superior to the GPT-2 model. Additionally, we can observe that more local tensors performs similarly but leads to higher memory cost. Thus we finally choose to set $m = 5$ for MPOE architecture considering the trade-off between the cost and quality.

5 Related Work

We will review the related works in four aspects.

PLMs with MoE. It has been reported that models with more parameters are usually considered to have a larger model capacity (Fedus et al., 2021; Zuo et al., 2021). In order to increase the model capacity, a promising direction is to explore the scaling properties with MoE architecture which was introduced by Jacobs et al. (1991). Thus, Shazeer et al. (2017) first applied the MoE architecture to large-scale language models. Then, Switch Transformers (Fedus et al., 2021), GShard (Lepikhin et al., 2021), BASELayer (Lewis et al., 2021) and HashLayer (Roller et al., 2021) studied how to build large-scale Transformer-based model with MoE as well as improving routing strategy, which can better utilize the model capacity. In addition, Zhang et al. (2021b) proposed a strategy for sparse activation of MoE architecture. He et al. (2021) suggested a distributed training system for fast training of MoE. Zoph et al. (2022) proposed a sparse expert model with more stable training. Yu et al. (2022) proposed a sparse expert model based on all-MLP architecture. In contrast, our approach aims to reduce information redundancy by sharing parameters among experts.

Matrix Product Operators Decomposition. Matrix product operators (MPO) (Pirvu et al., 2010) decomposition was proposed in quantum many-body physics, *a.k.a.* tensor-train (TT) decomposition (Oseledets, 2011). A major category of MPO studies relies on model compression (Gao et al., 2020). They focus on compressing weight matrix and convolutional layers (Novikov et al., 2015; Garipov et al., 2016; Sun et al., 2020). Furthermore, the MPO decomposition was used to compress the PLMs as well as enable lightweight fine-tuning in downstream tasks (Liu et al., 2021). In this work, we utilize such a decomposition mechanism for parameter sharing to construct a parameter-efficient MoE architecture.

Improved Variants of MoE. Despite the achieved performance performance, MoE architecture has been hindered by the model complexity and high memory costs (Shazeer et al., 2017; Fedus et al., 2021). This problem can be alleviated by using distillation (Fedus et al., 2021) and expert pruning (Kim et al., 2021). Then, Kudugunta et al. (2021) and Zuo et al. (2021) indicated that sub-networks can be employed when using the model. Indeed, our approach can be further enhanced by these existing methods for improving inference time.

Multi-task Learning. The exploitation of MoE architectures for multi-task learning is a very promising direction in recent years (Ma et al., 2018). Houlsby et al. (2019) suggested training adapters for each task separately while keeping the model fixed. Further research suggested that model parameters could be shared across tasks, and task-specific adapter parameters were introduced (Stickland and Murray, 2019). Based on this idea, Mahabadi et al. (2021) and Pilault et al. (2020) proposed that parameter-efficient multi-task fine-tuning for transformer-based models via shared hypernetworks. Our approach differs from these works in that the MPOE approach allows us to reduce model size while keeping the same number of experts, and meanwhile achieve performance improvement for multi-task learning.

6 Conclusion

In this paper, we proposed a parameter-efficient MoE architecture for increasing model capacity based on the MPO decomposition. First, we shared the central tensors among different experts based on MPO decomposition, which largely reduced the model parameters of MoE architecture. Then, we designed the gradient mask strategy to alleviate the unbalanced optimization issues and ensured that different tensors capture different types of information efficiently. Extensive experiments have shown that our approach outperforms several competitive PLM scaling strategies, especially in terms of improving the parameter efficiency of the MoE architecture.

In the future, we will enhance the proposed MPOE approach with recently proposed routing methods, such as BASELayer (Lewis et al., 2021), HASHLayer (Roller et al., 2021) and GShard (Lepikhin et al., 2021). We will also consider exploring additional decomposition methods for developing parameter-efficient MoE architecture.

Acknowledgments

This work was partially supported by Beijing Natural Science Foundation under Grant No. 4222027, National Natural Science Foundation of China under Grants No. 62206299 and 11934020, Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098 and Beijing Academy of Artificial Intelligence (BAAI). Xin Zhao is the corresponding author.

References

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. 1992. An estimate of an upper bound for the entropy of english. *Comput. Linguistics*, 18(1):31–40.

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*.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

Ze-Feng Gao, Song Cheng, Rong-Qiang He, ZY 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.

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

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5141–5148. AAAI Press.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. 2021. Fastmoe: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Young Jin Kim, Ammar Ahmad Awan, Alexandre Muzio, Andres Felipe Cruz Salinas, Liyang Lu, Amr Hendy, Samyam Rajbhandari, Yuxiong He, and Hany Hassan Awadalla. 2021. Scalable and efficient moe training for multitask multilingual models. *arXiv preprint arXiv:2109.10465*.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021. Beyond distillation: Task-level mixture-of-experts for efficient inference. *arXiv preprint arXiv:2110.03742*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. *arXiv preprint arXiv:2103.16716*.

Peiyu Liu, Ze-Feng Gao, Wayne Xin Zhao, Zhi-Yuan Xie, Zhong-Yi Lu, and Ji-Rong Wen. 2021. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5388–5398. Association for Computational Linguistics.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1930–1939. ACM.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 565–576. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Jonathan Pilault, Amine Elhattami, and Christopher Pal. 2020. Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data. *arXiv preprint arXiv:2009.09139*.

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

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

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

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. *arXiv preprint arXiv:2201.05596*.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. 2021. Hash layers for large sparse models. *arXiv preprint arXiv:2106.04426*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Xingwei Sun, Ze-Feng Gao, Zhong-Yi Lu, Junfeng Li, and Yonghong Yan. 2020. A model compression method with matrix product operators for speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2837–2847.

Hung-Yu Tseng, Yi-Wen Chen, Yi-Hsuan Tsai, Sifei Liu, Yen-Yu Lin, and Ming-Hsuan Yang. 2020. Regularizing meta-learning via gradient dropout. In *Proceedings of the Asian Conference on Computer Vision*.

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. *EMNLP 2018*, page 353.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 9514–9528. Association for Computational Linguistics.

An Yang, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, Di Zhang, Wei Lin, Lin Qu, Jingren Zhou, and Hongxia Yang. 2021. Exploring sparse expert models and beyond. *CoRR*, abs/2105.15082.

Ping Yu, Mikel Artetxe, Myle Ott, Sam Shleifer, Hongyu Gong, Ves Stoyanov, and Xian Li. 2022. Efficient language modeling with sparse all-mlp. *arXiv preprint arXiv:2203.06850*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2021a. Revisiting few-sample BERT fine-tuning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2021b. Moefication: Conditional computation of transformer models for efficient inference. *arXiv preprint arXiv:2110.01786*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100. ACM.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. 2021. Taming sparsely activated transformer with stochastic experts. *CoRR*, abs/2110.04260.