CPET: Effective Parameter Efficient Tuning for Compressed Large Models

Anonymous ACL submission

Abstract

In recent times, parameter-efficient tuning (PET) has been widely explored, as it tunes significantly fewer parameters than full-parameter fine-tuning (FT) while still stimulating sufficient knowledge from large language models (LLMs) for downstream tasks. Moreover, when adopting PET to serve multiple tasks, various tiny task-specific PET modules can be built on a frozen backbone LLM, avoiding redundantly deploying LLMs. Although PET methods have significantly reduced the cost of tuning and deploying LLMs, the inference still suffers from the computation bottleneck of the LLM. To address this issue, we build an effective PET framework based on compressed backbone LLMs, named "CPET". In CPET, we systematically evaluate the impact of mainstream compression techniques on the performance of PET modules, and then introduce knowledge inheritance and knowledge recovery to restore the knowledge loss caused by compressing the backbone LLM. Our experimental results demonstrate that, owing to the restoring strategies of CPET, collaborating task-specific PET modules with a compressed LLM can achieve comparable performance to collaborating with its non-compressed version, and significantly outperform directly applying FT or PET to the compressed LLM.

1 Introduction

In recent years, the rise in data scale and computing power has boosted the parameter size of pre-trained language models (PLMs). While some small and medium language models with millions of parameters have shown proficiency in capturing linguistic (Jawahar et al., 2019), semantic (Yenicelik et al., 2020), syntactic (Hewitt and Manning, 2019), and world knowledge (Petroni et al., 2019), large language models (LLMs) with billions of parameters (Brown et al., 2020; Black et al., 2022; Chowdhery et al., 2022) exhibit more powerful and comprehensive abilities, especially in terms of cognition and embodiment (Lewkowycz et al., 2022; Nakano et al., 2021; Driess et al., 2023).

Despite the success of LLMs, one significant challenge in adapting LLMs to specific real-world tasks is the cost-effectiveness of tuning and deploying LLMs. In traditional full-parameter fine-tuning (FT), a single LLM is tuned to multiple task-specific versions to serve different downstream tasks, leading to the high resource consumption of tuning and deployment. To address this challenge, parameter-efficient tuning (PET) (Houlsby et al., 2019; Hu et al., 2021; Li and Liang, 2021; Ben Zaken et al., 2022) has been proposed, which freezes a LLM as the backbone and adopts tiny tunable PET modules to stimulate the knowledge of the LLM for specific tasks. Compared with FT, PET tunes much fewer parameters while achieving comparable performance (Ding et al., 2023), and thus has lower computation and storage overhead in multi-task serving (Zhou et al., 2022).

Although PET has shown potential in reducing the cost of tuning and deploying LLMs, the computation of the whole backbone LLM is inevitable, i.e., the inference process is still resource-intensive, which prevents companies from applying it to real-world model services. To this end, we employ task-agnostic compression techniques (Hinton et al., 2015; Bai et al., 2021; Liang et al., 2021), which can compress an LLM into a smaller version while retaining most of its capabilities (Zhang et al., 2022a), and build an effective PET framework based on the compressed backbone LLM. We name this framework "CPET". Since the computation of a compressed LLM relies on much lower resources than its non-compressed version, CPET can achieve better inference efficiency than existing PET works after deployment. Considering the compression process of the backbone LLM may cause some knowledge loss, as shown in Figure 1, CPET introduce the following two mechanisms to

restore the loss.

(1) **PET Knowledge Inheritance**. A stronger backbone model can make learning PET modules easier, and meanwhile, the PET modules based on the stronger backbone can also better grasp how to stimulate task-specific knowledge distributed in the backbone model. Therefore, we propose to adopt the PET modules learned on the non-compressed backbone LLM as the initialization to learn the PET modules for the compressed backbone LLM. In this way, the task-related knowledge of PET modules learned with the help of the non-compressed backbone can be inherited to obtain more effective PET modules for the compressed backbone.

(2) **Model Knowledge Recovery**. In addition to the knowledge of PET modules, the knowledge of the backbone LLM is also important to perform well on downstream tasks. Since task-agnostic compression techniques may result in losing some task-related knowledge within the backbone LLM, we add extra knowledge recovery modules into the compressed model to bridge the knowledge gap that arises from compressing the LLM. We point out that compression techniques may weaken multiple capabilities of the backbone LLM while restoring only one of the lost capabilities requires only a small number of parameters. Through the supervision of task-specific data, we can recover most of the lost task-related knowledge through some tiny recovery modules.

In experiments, we first conduct a comprehensive evaluation of the performance impact brought by various compression methods. The results show that compression results in a significant model performance drop without using any knowledge recovery mechanisms. Based on the above observation, we apply CPET for performance recovery, and the experimental results indicate that CPET can restore the model performance to the level before model compression. Furthermore, computing the compressed backbone LLM requires much lower resources than computing the non-compressed backbone, making CPET finally an effective and efficient PET framework.

2 Methodology

In this section, we will introduce how to build the effective PET framework CPET for compressed LLMs. Before introducing CPET, we first explain some essential preliminaries.

2.1 Preliminary

First, we briefly describe the core architecture of transformer (Vaswani et al., 2017). A transformer consists of multiple transformer blocks, and each block includes two components: a multi-head attention and a feed-forward network. Multi-head attention can be formalized as

$$\begin{aligned} \texttt{MH-ATT}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= [\mathbf{H}_1,\cdots,\mathbf{H}_n]\mathbf{W}^O, \\ \mathbf{H}_i &= \texttt{ATT}(\mathbf{Q}\mathbf{W}_i^Q,\mathbf{K}\mathbf{W}_i^K,\mathbf{V}\mathbf{W}_i^V), \\ \texttt{ATT}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \texttt{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{d})\mathbf{V}, \end{aligned} \tag{1}$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$ are the matrices of the linear transformations in the i-th attention head, d is the head dimension, $\mathbf{H}_i$ is the result of the i-th head, $[\cdot,\ldots,\cdot]$ is the concatenation of vectors. Feed-forward network can be formalized as

$$\texttt{FFN}(\mathbf{X}) = \sigma(\mathbf{X}\mathbf{W}_1+\mathbf{b}_1)\mathbf{W}_2+\mathbf{b}_2, \tag{2}$$

Where $\mathbf{W}_1$ and $\mathbf{W}_2$ are the matrices of the linear transformations in the feed-forward network, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the bias vectors of the linear transformations, $\sigma(\cdot)$ is the activation function.

For simplicity, we denote a LLM $\mathcal{M}$ as $\mathbf{Y} = f(\mathbf{X};\theta_{\mathcal{M}})$, where $f(\cdot)$ is the function of the whole transformer architecture, $\theta_{\mathcal{M}}$ is the parameters of the LLM, $\mathbf{X}$ is the input and $\mathbf{Y}$ is the output. In the FT setting, all parameters of $\mathcal{M}$ (i.e., $\theta_{\mathcal{M}}$) are tuned as follows

$$\theta_{\mathcal{M}}^t = \arg\min_{\theta_{\mathcal{M}}} \mathcal{L}(f(\mathbf{X}^t;\theta_{\mathcal{M}}),\mathbf{Y}^t), \tag{3}$$

where $\mathbf{X}^t,\mathbf{Y}^t$ is the data of the downstream task t, $\mathcal{L}$ is the loss function of the task t. $\theta_{\mathcal{M}}^t$ is the final task-specific model parameters of the LLM $\mathcal{M}$.

In the PET setting, $\mathcal{M}$ is frozen, and additional PET modules $\mathcal{P}$ are tuned on task-specific data. We denote the parameters of PET modules injected to LLM $\mathcal{M}$ as $\theta_{\mathcal{P}(\mathcal{M})}$. As shown in Figure 1, the computation of the transformer architecture is slightly changed due to the injected PET modules and becomes $\mathbf{Y} = f_{\text{PET}}(\mathbf{X};\theta_{\mathcal{M}},\theta_{\mathcal{P}(\mathcal{M})})$. The tuning process is formalized as

$$\theta_{\mathcal{P}(\mathcal{M})}^t = \arg\min_{\theta_{\mathcal{P}(\mathcal{M})}} \mathcal{L}(f_{\text{PET}}(\mathbf{X}^t;\theta_{\mathcal{M}},\theta_{\mathcal{P}(\mathcal{M})}),\mathbf{Y}^t), \tag{4}$$

where $\theta_{\mathcal{P}(\mathcal{M})}^t$ is the final task-specific PET modules collaborating with the LLM $\mathcal{M}$.

This paper aims to build PET modules based on a compressed LLM. To this end, after applying compression algorithms to compress the LLM $\mathcal{M}$,

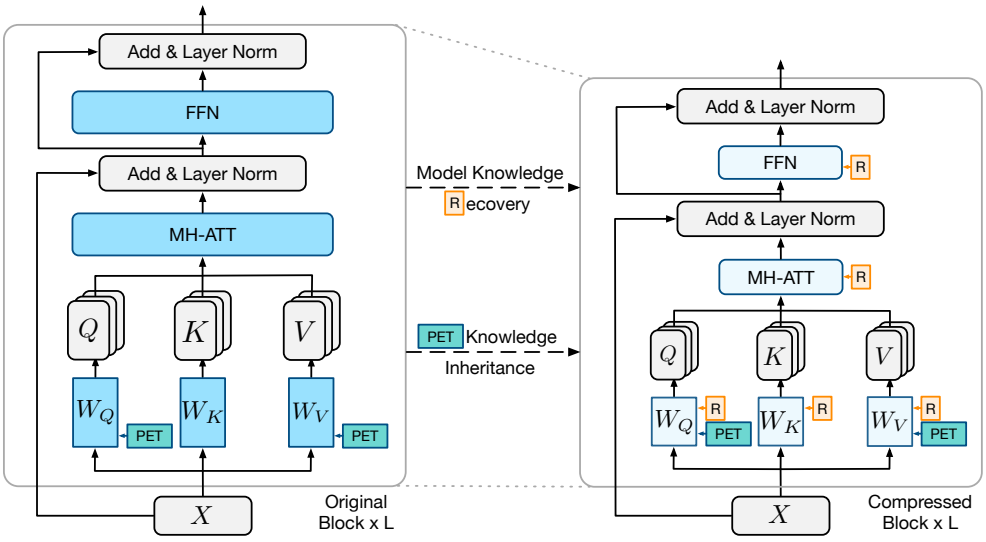

Figure 1: The overall design of our CPET. We use LoRA (Hu et al., 2021) as an example of PET.

making $\mathcal{M}$ have fewer parameters or lower-bit representations, we denote the compressed LLM and its parameters $\mathcal{C}$ and $\theta_{\mathcal{C}}$ respectively. Then the computation of the compressed model can be described as $\mathbf{Y} = f(\mathbf{X}; \theta_{\mathcal{C}})$.

2.2 Framework

PET methods do not change the LLM backbone , thus adopting PET methods is orthogonal to compressing the backbone LLM [1]. Therefore, we propose a more efficient PET framework CPET, by first compressing the backbone LLM using task-agnostic model compression methods and then applying PET methods to the compressed backbone. Formally, CPET can be formalized as

$$\theta^t_{\mathcal{P}(\mathcal{C})} = \arg\min_{\theta_{\mathcal{P}(\mathcal{C})}} \mathcal{L}(f_{\text{PET}}(\mathbf{X}^t; \theta_{\mathcal{C}}, \theta_{\mathcal{P}(\mathcal{C})}), \mathbf{Y}^t), \quad (5)$$

where $\theta_{\mathcal{P}(\mathcal{C})}$ are the parameters of PET modules injected to the compressed LLM $\mathcal{C}$.

By combining model compression with PET, on the one hand, we can take the advantage of PET to deploy a unified backbone LLM to serve multiple downstream tasks, while only needing to maintain tiny task-specific PET modules for each downstream task. On the other hand, by adopting a compressed LLM instead of a non-compressed LLM, the inference time and resource requirements of the backbone model can be significantly reduced. It is worth noting that this acceleration is not free. It is not difficult to imagine that adopting task-agnostic compression methods may weaken the backbone model, which will inevitably affect the search for the optimal parameters $\theta^t_{\mathcal{P}(\mathcal{C})}$ and the effect of the final model $f_{\text{PET}}(\mathbf{X}; \theta_{\mathcal{C}}, \theta^t_{\mathcal{P}(\mathcal{C})})$.

As shown in Figure 1, to better learn PET modules for the compressed backbone LLM, we adopt the mechanism to inheriting the PET knowledge from those modules based on the non-compressed backbone. To restore the knowledge loss caused by compressing the backbone LLM, in addition to the PET modules $\mathcal{P}$, we add some knowledge recovery modules $\mathcal{R}$, and Eq. (5) is modified to

$$\begin{aligned} &\theta^t_{\mathcal{P}(\mathcal{C})}, \theta^t_{\mathcal{R}} = \\ &\arg\min_{\theta_{\mathcal{P}(\mathcal{C})}, \theta_{\mathcal{R}}} \big[\mathcal{L}(f_{\text{PET}}(\mathbf{X}^t; \theta_{\mathcal{C}}, \theta_{\mathcal{P}(\mathcal{C})}, \theta_{\mathcal{R}}), \mathbf{Y}^t) \\ &\qquad + \alpha \mathcal{L}_{\text{DIST}}(\mathbf{X}^t; \theta_{\mathcal{M}}, \theta_{\mathcal{C}}, \theta_{\mathcal{P}(\mathcal{C})}, \theta_{\mathcal{R}})\big], \end{aligned} \quad (6)$$

where $\theta^t_{\mathcal{R}}$ is the parameters of the task-specific recovery modules for the task t, and $f_{\text{PET}}(\mathbf{X}; \theta_{\mathcal{C}}, \theta^t_{\mathcal{P}(\mathcal{C})}, \theta^t_{\mathcal{R}})$ is finally used to serve the task t. $\mathcal{L}_{\text{DIST}}(\mathbf{X}^t; \theta_{\mathcal{M}}, \theta_{\mathcal{C}}, \theta_{\mathcal{P}(\mathcal{C})}, \theta_{\mathcal{R}})$ is the loss function for model knowledge recovery, which will be introduced later. In subsequent sections, we will elaborate on how to conduct PET knowledge inheritance and model knowledge recovery.

2.3 PET Knowledge Inheritance

Instead of training PET modules based on the compressed LLM from scratch, we propose training PET based on the original non-compressed LLM first, then adapting the learned PET modules to the compressed LLM. The adaption from the non-compressed LLM to the compressed LLM can save convergence time and improve performance for

[1] Compression methods involved in this article do not change the model structure, which ensure that the insertion position and quantity of PET modules remain unchanged.

learning PET modules on the compressed LLM. Intuitively, it is more effective for a teacher to teach students the fundamentals of a discipline and then let students adapt their comprehension based on their circumstances rather than directly letting students learn from scratch. Formally, we first use Eq. (4) to obtain the parameters of task-specific PET modules $\theta^t_{\mathcal{P}(\mathcal{M})}$ based on the non-compressed LLM $\mathcal{M}$, and then use $\theta^t_{\mathcal{P}(\mathcal{M})}$ as the initialization to obtain the parameters of task-specific PET modules for the compressed LLM $\mathcal{C}$, i.e., obtain $\theta^t_{\mathcal{P}(\mathcal{C})}$ in Eq. (5) and Eq. (6).

2.4 Model Knowledge Recovery

Since the reduction of parameters in the compressed LLM $\mathcal{C}$ may cause performance degradation, we thus propose to inject knowledge recovery modules $\mathcal{R}$ into $\mathcal{C}$ to recover the lost knowledge. As shown in Figure 1, we add a bypass next to each linear layer to add a small amount of change to the output states of these linear layers. To avoid introducing too many parameters, the recovery modules $\mathcal{R}$ adopt the typical bottleneck MLP structure. Formally, we denote an arbitrary matrix in the compressed LLM as $\mathbf{W}$ and the linear transformation as $\mathbf{XW}$, the modified transformation becomes $\mathbf{XW} + \sigma(\mathbf{XD})\mathbf{U}$, where $\mathbf{D}$ is the down-sampling matrix, $\sigma(\cdot)$ is the activation function, and $\mathbf{U}$ is the up-sampling matrix. $\mathbf{D}$ and $\mathbf{U}$ together form $\theta_{\mathcal{R}}$.

To help obtain $\theta^t_{\mathcal{R}}$ for the task t, we design a distillation objective. Specifically, we first select the PET modules trained with Eq. (4) as the teacher, and then select the PET modules and recovery modules in Eq. (6) as the student, the whole distillation loss is given as

$$\mathcal{L}_{\text{DIST}}(\mathbf{X}^t; \theta_{\mathcal{M}}, \theta_{\mathcal{C}}, \theta_{\mathcal{P}(\mathcal{C})}, \theta_{\mathcal{R}}) = \frac{1}{|\mathbf{X}^t|} \left\| f_{\text{PET}}(\mathbf{X}^t; \theta_{\mathcal{M}}, \theta^t_{\mathcal{P}(\mathcal{M})}) - f_{\text{PET}}(\mathbf{X}^t; \theta_{\mathcal{C}}, \theta_{\mathcal{P}(\mathcal{C})}, \theta_{\mathcal{R}}) \right\|_2^2, \tag{7}$$

where $\mathbf{X}^t$ is the input data of the task t. As shown in Eq. (6), instead of first learning the recovery modules and then adding the inherited PET modules for further adaptation, we simultaneously conduct knowledge recovery and tune PET modules.

3 Experiment

3.1 Datasets

Since PET methods are usually evaluated on language understanding tasks, we thus evaluate CPET on 5 datasets from SuperGLUE (Wang et al., 2019), a public leaderboard commonly used to measure the language understanding performance of models, including BoolQ (Clark et al., 2019), CB (De Marneffe et al., 2019), RTE (Bentivogli et al., 2009), COPA (Roemmele et al., 2011), and WiC (Pilehvar and Camacho-Collados, 2018). More details are shown in Table 1. For all datasets, we use accuracy (%) as the metric for evaluation. Considering our goal is to evaluate the impact of a compressed backbone LLM on PET modules and to demonstrate whether the mechanisms in CPET can restore the lost knowledge caused by compression methods, for simplicity, we thus directly use the validation sets in SuperGLUE for test.

Dataset	#training	#validation
BoolQ	9427	3270
CB	250	56
RTE	2490	277
COPA	400	100
WiC	5428	638

Table 1: The statistics of the datasets used for experiments, which are from SuperGLUE (Wang et al., 2019).

3.2 Baseline and Implementation Details

As both PET and CPET need a backbone LLM, we thus use T5-3B (Raffel et al., 2020) (with 3 billion parameters) as the backbone model in our experiments. To compress the backbone, we directly use the open-source toolkit BMCook (Zhang et al., 2022a) to compress T5-3B into various compressed versions, including quantization, structured pruning, unstructured pruning, and MoEfication. A subset of The Pile (Gao et al., 2020) data is used for compression.

To evaluate the effectiveness of CPET, we compare our framework with 4 baseline paradigms that adapt LLMs to specific tasks: (1) **FT (LLM)**: no extra parameters are involved, and all parameters of the original T5-3B are tuned to handle specific tasks. (2) **PET (LLM)**: PET modules are attached to the original T5-3B, and only the parameters of PET modules are tunable while the parameters of the backbone LLM are frozen. (3) **FT (CLM)**: various compressed versions of T5-3B are used as the starting point, and then all parameters of these compressed LLMs are tuned on task-specific data. (4) **PET (CLM)**: PET modules are attached to various compressed versions of T5-3B, and then these compressed backbone LLMs are frozen and PET modules are tuned on task-specific data.

Datasets	Paradigm	Original LLM	Quantization	Unstructured Pruning	Structured Pruning	MoEfication
BoolQ	FT	88.0	87.0	87.4	83.0	85.7
	PET	88.4	87.3 (0.3 ↑)	86.8 (0.6 ↓)	79.6 (3.4 ↓)	87.8 (2.1 ↑)
	CPET	-	89.2 (2.2 ↑)	87.6 (0.2 ↑)	85.7 (2.7 ↑)	88.4 (2.7 ↑)
CB	FT	100.0	98.2	94.6	91.1	96.4
	PET	98.2	94.6 (3.6 ↓)	98.2 (3.6 ↑)	75.0 (16.1 ↓)	92.9 (3.5 ↓)
	CPET	-	98.2 (0.0 ↑)	98.2 (3.6 ↑)	94.6 (3.5 ↑)	98.2 (1.8 ↑)
RTE	FT	89.3	88.2	89.3	79.6	86.8
	PET	89.6	84.3 (3.9 ↓)	79.6 (9.7 ↓)	65.4 (14.2 ↓)	86.1 (0.7 ↓)
	CPET	-	91.4 (3.2 ↑)	85.7 (3.6 ↓)	76.8 (2.8 ↓)	89.3 (2.5 ↑)
COPA	FT	88.0	90.0	85.0	73.0	84.0
	PET	87.0	85.0 (5.0 ↓)	84.0 (1.0 ↓)	73.0 (0.0 ↑)	83.0 (1.0 ↓)
	CPET	-	88.0 (2.0 ↓)	85.0 (0.0 ↑)	76.0 (3.0 ↑)	87.0 (3.0 ↑)
WiC	FT	74.9	73.5	75.1	70.1	73.5
	PET	75.4	73.4 (0.1 ↓)	71.3 (3.8 ↓)	67.5 (2.6 ↓)	73.8 (0.3 ↑)
	CPET	-	74.7 (1.2 ↑)	74.4 (0.7 ↓)	70.6 (0.5 ↑)	74.2 (0.7 ↑)

Table 2: Comparisons between CPET and baselines (%). We present the highest scores of the model on the development set. In parentheses "()", we compare both PET (CLM) method and CPET method with FT (CLM), and then use ↑ and ↓ to indicate relative performance improvement or decrease.

All the above paradigms and our CPET are implemented with the open-source toolkit OpenDelta (Ding et al., 2023). For a fair comparison, we use LoRA method (Hu et al., 2021) as a representative PET method for both baseline paradigms and CPET. We inject LoRA modules into those weight matrices $\mathbf{Q}$ and $\mathbf{K}$ in the attention layer (Eq. (1)). We set the bottleneck dimension of the LoRA modules to 32 in all settings. We also set the bottleneck dimension of our recovery modules to 32. For FT (LLM) and FT (CLM), the learning rate is among $\{1e-5, 3e-5, 5e-5\}$. For PET (LLM), PET (CLM), and our CPET, the learning rate is among $\{1e-3, 5e-4, 1e-4\}$, considering tuning PET modules usually requires a higher learning rate than fine-tuning LLMs. The batch size is among $\{16, 32, 64\}$. The weight decay is set to $1e-2$. For all datasets, we tune 20 epochs and use the first 10% steps for warmup. The best model checkpoint is selected by considering the performance on validation sets. The coefficient in Eq. (6) is default to $\alpha = 0.05$.

3.3 Overall Results

Table 2 shows the performance of CPET and baselines. From the table, we can find that:

(1) Experiments comparing FT and PET show that using PET can consistently achieve comparable results to using FT, whether on the original LLM or on a compressed version. This indicates that utilizing PET can significantly decrease the number of parameters needed without compromising the task performance when serving multiple downstream tasks.

(2) Comparing the original LLM and its compressed versions, the results of the fine-tuning process show that the compressed LLMs perform not as well as the original LLM. It suggests that task-agnostic compression methods lead to losing some knowledge related to downstream tasks. That is to say, in order to improve the inference speed, the performance of the compressed model may decrease due to the acceleration process. At this time, if there is a mechanism to make up the performance gap without affecting the inference speed, applying compression methods will be more reasonable.

(3) Within the compressed model, CPET consistently outperforms vanilla PET methods. Such results indicate that task capabilities are effectively migrated through knowledge inheritance and knowledge recovery mechanisms. Among all methods, CPET can match the best performance of those fine-tuning baselines, showing the migrated knowledge can effectively improve the model performance on downstream tasks. In other words, our method is more effective than directly building PET modules based on a compressed LLM and can maintain good results in low-resource applications.

LLM	Q	UP	SP	M	PET	CPET
2.8	0.7	1.4	1.4	1.3	0.02	0.12

Table 3: Tunable parameters (billion) of different methods. "Q", "UP", "SP", and "M" represent quantization, unstructured pruning, structured pruning, and MoEfication, respectively. In "PET" and "CPET", the number of parameters in the backbone has been excluded.

I	R	D	Q	UP	SP	M
			87.3	86.8	79.6	87.8
✓			88.7	87.5	81.0	87.7
✓	✓		88.3	88.1	83.6	88.5
✓	✓	✓	89.2	87.6	85.7	88.4

Table 4: The ablation study on BoolQ (%). "I" means the tunable parameters are inherited from the PET modules trained on the non-compressed LLM. "R" means the recovery modules are injected into the compressed model, but the distillation objective is not applied. "D" means that the distillation objective is used.

On the other hand, as shown in Table 3, the size of tunable parameters for CPET is relatively close to that of vanilla PET methods, which is much smaller than the size of the original and compressed LLMs.

(4) Through cross-comparisons between different compression models, we find that quantization and MoEfication have relatively little loss on the model performance, and can completely restore the performance of the original LLM using our method. However, the pruning methods cause more performance loss. Although our method can significantly recover the performance loss caused by the pruning methods, it still cannot fully recover to the level before compression.

3.4 Ablation Studies

To further analyze how different mechanisms in CPET help restore the loss caused by model compression methods, we use the BoolQ dataset for further ablation studies. The results are shown in Table 4. From the results, we can find that:

(1) PET knowledge inheritance is effective. By initializing the tunable parameters with the PET modules trained on the non-compressed LLM, the task performance of the combination of the final PET modules and the compressed LLM has been greatly improved. This indicates that in the optimization space of the compressed LLM, it is difficult to optimize the optimal task-specific parameters of PET modules based on random initialization, but the optimal PET modules can be achieved much easier using PET knowledge inheritance.

(2) Model knowledge recovery is effective. Table 2 shows that CPET improves the task performance by adding our knowledge recovery mechanisms. We need to further prove that such performance improvement is not only due to the direct increase in the parameter size brought by these additional recovery modules. From Table 4, we find that only adding the extra recovery modules brings less performance improvement than our complete framework. This suggests combining recovery modules with knowledge distillation to enhance PET modules is necessary.

3.5 Convergence of CPET

Although we are primarily concerned with the final speed and performance at the end of the training process, the method usability may be compromised if the training process spends too much time. Therefore, based on four different compression LLMs, we compare the convergence speed of tuning PET modules to handle the BoolQ dataset.

From Figure 2, we find that due to our PET knowledge inheritance mechanism, our method is superior to the vanilla PET methods in terms of convergence speed and final results. Furthermore, when quantization, unstructured pruning or MoEfication is used, the inheritance mechanism gives our method a better starting point for tuning PET modules. While in the case of structured pruning, even though the initial point of our method does not work well on tasks, it is closer to the optimal point in the optimization space and converges faster.

Moreover, due to the existence of numerous downstream tasks, the PET modules based on a unified backbone LLM may be trained by community users on their own data and then uploaded to the Internet. When adapting these PET modules to a compressed backbone LLM, there may not be any task-specific data available for the adaptation process. From Figure 2, we can find that when quantization or MoEfication is used, we can achieve ideal results under the zero-shot adaptation condition by directly using the PET inheritance mechanism we proposed.

3.6 Performance on the Mixture of Compression Methods

To evaluate CPET on a higher compression ratio, we further adopt the mixture of compression meth-

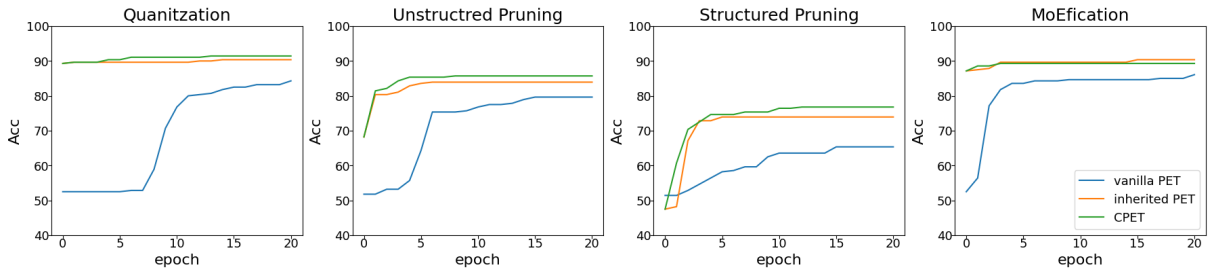

Figure 2: The convergence of vanilla PET, inherited PET, and CPET.

Method	BoolQ	CB	RTE	COPA	WiC
FT	88.0	100	89.3	88.0	74.9
CPET	86.7	100	86.1	85.0	75.3

Table 5: The results of applying CPET on the mixture of compression methods (%).

ods, including quantization, unstructured pruning, and MoEfication, to compress the backbone LLM. The compressed model is around $16\times$ smaller than T5-3B. We compare the performance of CPET, based on this $16\times$ smaller compressed model, to the performance of fine-tuning T5-3B. Table 5 shows the experimental results. From the table, we conclude that our method is compatible and can be easily applied to highly compressed models that use multiple compression methods.

4 Related Work

This work is related to PLMs, LLMs, PET, and model compression. We mainly introduce typical PET and model compression methods. More details on PLMs and LLMs can refer to the survey (Qiu et al., 2020; Bommasani et al., 2021).

4.1 Parameter-Efficient Tuning

In recent years, transformer-based (Vaswani et al., 2017) PLMs have been widely explored, such as GPT (Brown et al., 2020) and BERT (Devlin et al., 2018). With the increase in both the amount of pre-training data and the size of PLMs' parameters, large-scale PLMs (Kaplan et al., 2020), i.e., LLMs (Brown et al., 2020; Black et al., 2022; Chowdhery et al., 2022) also emerge and show excellent capabilities (Wei et al., 2022), especially in some cognitive and embodied scenarios (Lewkowycz et al., 2022; Nakano et al., 2021; Driess et al., 2023). Although an LLM can acquire rich knowledge from massive pre-training data to handle complex tasks in a zero-shot or few-shot manner (Brown et al., 2020; Black et al., 2022), to better stimulate the knowledge stored in the LLM to serve downstream tasks, there is still a need for adapting the LLM to various task-specific scenarios. For traditional PLMs, fine-tuning all parameters of PLMs is the mainstream way to adapt them (Church et al., 2021), yet its parameter inefficiency makes this way costly to adapt LLMs (Ding et al., 2023). Moreover, maintaining a task-specific version of LLMs for each downstream task is unacceptably resource-intensive (Zhou et al., 2022).

To adapt LLMs to task-specific scenarios in a more efficient manner, various PET methods (Lester et al., 2021; Houlsby et al., 2019; Hu et al., 2021; Li and Liang, 2021; Ben Zaken et al., 2022) have been proposed, where LLMs are frozen and some model-independent tunable modules are injected into the transformer architecture of LLMs to help the adaptation process. PET modules are usually tiny, which can significantly reduce the cost of adapting LLMs. PET modules can be inserted into different locations within the transformer architecture. For instance, prompt tuning (Lester et al., 2021) and prefix tuning (Li and Liang, 2021) are two methods that prepend tunable embeddings to the input and hidden states, respectively. Adapter tuning (Houlsby et al., 2019) applies tunable transformation between adjacent modules. BitFit (Ben Zaken et al., 2022) and LoRA (Hu et al., 2021) make minor internal modifications to the modules of the transformer architecture.

As mentioned before, LLMs have acquired rich capabilities and just need an efficient way to stimulate these capabilities. The role of tunable PET modules is to learn task features and serve as triggers to stimulate task-specific capabilities in LLMs (Ding et al., 2023). Sufficient experiments show that collaborating task-specific PET modules and a frozen LLM can reach comparable performance to fine-tuning all parameters of the LLM. Furthermore, since different task-specific

PET modules can share a unified frozen LLM as their backbone, this also leads to lower computation and storage overhead in multi-task serving and switching (Zhou et al., 2022). In general, the emergence of PET methods significantly reduces the cost of tuning and deploying LLMs.

4.2 Model Compression

Although PET methods can reduce the cost of tuning and deploying LLMs, the computation bottleneck of the LLM itself still exists. Therefore, to further improve efficiency for model serving, it is crucial to speed up the computation of LLMs, and model compression is a commonly used solution. Considering that the PET modules of different tasks usually work together on a unified backbone LLM, here we mainly introduce task-agnostic model compression (Sanh et al., 2019) rather than task-specific compression (Sun et al., 2019) for LLMs, including quantization, pruning, and MoEfication.

In traditional PLMs, 32-bit floating-point numbers are mainly used to represent models. As the model size gradually increases, representing LLMs in a 32-bit format consumes too much GPU memory and computing time. To address this issue, mixed-precision training (Micikevicius et al., 2017) is adopted to represent LLMs with 16-bit floating-point numbers. To further reduce the memory overhead and improve the model speed, quantization methods are applied to represent models with fixed-point numbers, from 8-bit (Zafrir et al., 2019), 4-bit (Frantar et al., 2023) to 1-bit (Bai et al., 2021). To avoid the performance degradation caused by quantization, quantization-aware training (QAT) (Stock et al., 2021) has also been proposed to use a small amount of data to adjust the distribution of model parameters for quantization.

Different from quantization methods that compress the representation of each parameter, pruning methods directly discard some parameters. Commonly used pruning methods include structured pruning (Fan et al., 2020; Wang et al., 2020; Zhang et al., 2021; Xia et al., 2022) and unstructured pruning (Han et al., 2015; Chen et al., 2020; Xu et al., 2021). Structured pruning aims to find useless modules and remove them completely, such as erasing all parameters in a linear layer. Unstructured pruning only removes individual parameters, such as deleting some parameters to form a sparse matrix.

MoEfication (Zhang et al., 2022b), inspired by the mixture-of-experts (MoE) transformer (Lepikhin et al., 2021), aims to divide the parameters of LLMs into multiple partitions, and each time only a few partitions are used to compute the final results. Although most of the currently popular LLMs are dense models, studies have shown that dense LLMs are activated sparsely, and different parameter areas are activated by different data to form some skill partitions (Wang et al., 2022; Dai et al., 2021; Suau et al., 2020; Panigrahi et al., 2023). Specifically, by analyzing the sparse pattern of activation states in LLMs, the linear layers of LLMs are sliced to MoE, and an expert router is trained to select experts. During the computation process, a certain proportion of relevant experts is dynamically activated according to the input data.

Typically, to make a compressed LLM behave the same as its original version, distillation objectives are often used to align the pre-compression and post-compression models, including aligning both output and intermediate states (Hinton et al., 2015; Sun et al., 2019; Jiao et al., 2020; Liu et al., 2022; Park et al., 2021). Due to space limitations, more compression details can refer to the survey (Liang et al., 2021; Xu and McAuley, 2022).

5 Conclusion

In this paper, we propose an effective PET framework based on the compressed LLM (named CPET) to further reduce the resource requirements and inference speed when deploying LLM and PET modules to serve downstream tasks. Considering task-agnostic compression methods may cause losing some task-specific knowledge, we introduce PET knowledge inheritance and model knowledge recovery to restore the lost knowledge. By inheriting the prior task knowledge of the PET modules learned on the non-compressed LLM, searching for the optimal PET modules for the compressed LLM becomes easier. Moreover, by introducing knowledge recovery modules to recover task-specific capabilities lost in the compression phase, collaborating PET modules with the compressed LLM can achieve comparable performance to those PET modules based on the non-compressed LLM. The experimental results show that CPET can outperform baselines based on the compressed LLM, and meanwhile, CPET maintains the advantages of PET methods for multi-task serving. This paper mainly accelerates the inference of PET methods and LLMs. We leave the computation bottleneck of LLMs in the tuning process as future work.

6 Limitations

In this paper, we only use T5-3B as the backbone LLM. For the selection of PET methods, we only choose LoRA as a representative. In fact, our framework can be applied to any LLM and PET method. Therefore, we will conduct experiments on more combinations of LLMs and PET modules to demonstrate the generalization of our framework. The compression method adopted in this paper does not change the the number of layers of the backbone LLM. However, for those compression methods that change the hidden dimensions of the model, how to transfer the knowledge of PET modules on the non-compressed LLM remains an open problem for our future work.

References

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2021. Binarybert: Pushing the limit of bert quantization. In *Proceedings of ACL/IJCNLP*, pages 4334–4348.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of ACL*, pages 1–9.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of TAC*.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

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. In *Proceedings of NeurIPS*, volume 33, pages 1877–1901.

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. In *Proceedings of NeurIPS*, volume 33, pages 15834–15846.

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*.

Kenneth Ward Church, Zeyu Chen, and Yanjun Ma. 2021. Emerging trends: A gentle introduction to fine-tuning. *Natural Language Engineering*, 27(6):763–778.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *Proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

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*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, pages 1–16.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *Proceedings of ICLR*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate quantization for generative pre-trained transformers. In *Proceedings of ICLR*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural network. In *Proceedings of NeurIPS*, pages 1135–1143.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of NAACL*, pages 4129–4138.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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 ICML*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *Proceedings of ACL*, pages 3651–3657.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of EMNLP*, pages 4163–4174.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam M. Shazeer, and Z. Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *Proceedings of ICLR*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of ACL-IJCNLP*, pages 4582–4597.

Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. 2021. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403.

Chang Liu, Chongyang Tao, Jiazhan Feng, and Dongyan Zhao. 2022. Multi-granularity structural knowledge distillation for language model compression. In *Proceedings of ACL*, pages 1001–1011.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Frederick Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. 2023. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*.

Geondo Park, Gyeongman Kim, and Eunho Yang. 2021. Distilling linguistic context for language model compression. In *Proceedings of EMNLP*, pages 364–378.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of EMNLP*, pages 2463–2473.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

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. *JMLR*, 21:1–67.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Proceedings of AAAI*, pages 90–95.

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

Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. 2021. Training with quantization noise for extreme model compression. In *Proceedings of ICLR*.

Xavier Suau, Luca Zappella, and Nicholas Apostoloff. 2020. Finding experts in transformer models. *arXiv preprint arXiv:2005.07647*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings EMNLP-IJCNLP*, pages 4323–4332.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*, volume 30.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of NeurIPS*, volume 32.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. 2022. Finding skill neurons in pre-trained transformer-based language models. In *Proceedings of EMNLP*, pages 11132–11152.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of EMNLP*, pages 6151–6162.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of ACL*, pages 1513–1528.

Canwen Xu and Julian McAuley. 2022. A survey on model compression for natural language processing. *arXiv preprint arXiv:2202.07105*.

Dongkuan Xu, Ian En-Hsu Yen, Jinxi Zhao, and Zhibin Xiao. 2021. Rethinking network pruning – under the pre-train and fine-tune paradigm. In *Proceedings of NAACL*, pages 2376–2382.

David Yenicelik, Florian Schmidt, and Yannic Kilcher. 2020. How does bert capture semantics? a closer look at polysemous words. In *Proceedings of BlackboxNLP*, pages 156–162.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. In *Proceedings of EMC2-NIPS*, pages 36–39. IEEE.

Zhengyan Zhang, Baitao Gong, Yingfa Chen, Xu Han, Guoyang Zeng, Weilin Zhao, Yanxu Chen, Zhiyuan Liu, and Maosong Sun. 2022a. Bmcook: A task-agnostic compression toolkit for big models. In *Proceedings of EMNLP Demonstration*, pages 396–405.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022b. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of ACL*, pages 877–890.

Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Qun Liu, and Maosong Sun. 2021. Know what you don't need: Single-shot meta-pruning for attention heads. *AI Open*, 2:36–42.

Zhe Zhou, Xuechao Wei, Jiejing Zhang, and Guangyu Sun. 2022. PetS: A unified framework for Parameter-Efficient transformers serving. In *Proceedings of ATC*, pages 489–504.