ScaleOT: Privacy-utility-scalable Offsite-tuning with Dynamic LayerReplace and Selective Rank Compression

Kai Yao[1,2], Zhaorui Tan[3], Tiandi Ye[4], Lichun Li[2], Yuan Zhao[2], Wenyan Liu[2], Wei Wang[2*], Jianke Zhu[1*]

[1]Zhejiang University, Hangzhou, China
[2]Ant Group
[3]University of Liverpool, Liverpool, the United Kingdom
[4]East China Normal University, Shanghai China
jiumo.yk@antgroup.com, jkzhu@zju.edu.cn

Abstract

Offsite-tuning is a privacy-preserving method for tuning large language models (LLMs) by sharing a lossy compressed emulator from the LLM owners with data owners for downstream task tuning. This approach protects the privacy of both the model and data owners. However, current offsite tuning methods often suffer from adaptation degradation, high computational costs, and limited protection strength due to uniformly dropping LLM layers or relying on expensive knowledge distillation. To address these issues, we propose ScaleOT, a novel privacy-utility-scalable offsite-tuning framework that effectively balances privacy and utility. ScaleOT introduces a novel layerwise lossy compression algorithm that uses reinforcement learning to obtain the importance of each layer. It employs lightweight networks, termed harmonizers, to replace the raw LLM layers. By combining important original LLM layers and harmonizers in different ratios, ScaleOT generates emulators tailored for optimal performance with various model scales for enhanced privacy protection. Additionally, we present a rank reduction method to further compress the original LLM layers, significantly enhancing privacy with negligible impact on utility. Comprehensive experiments show that ScaleOT can achieve nearly lossless offsite tuning performance compared with full fine-tuning while obtaining better model privacy.

Introduction

Recent years have witnessed the bloom of large language models (LLMs) (Devlin et al. 2019; Radford et al. 2019; Touvron et al. 2023; Du et al. 2022). Pre-trained on massive datasets, these models acquire robust general capabilities to tackle a comprehensive range of tasks by zero-shot (ZS) predictions or in-context learning (Dong et al. 2024; Wies, Levine, and Shashua 2024) and able to be fine-tuned for complex downstream tasks (Wortsman et al. 2022; Zhou et al. 2022; Wei et al. 2022; Ouyang et al. 2022; Tan et al. 2024; Liu et al. 2024a; Dai et al. 2024). However, conventional centralized fine-tuning processes require the model and the data to be co-located - either the data owner uploads the data, which may threaten the data owner's data privacy, or the model owner shares the model weights - posing a risk of leaking the expensively trained models (Chua et al. 2023; Xiao, Lin, and Han 2023). Moreover, the possible exposure of parameters for the latter may increase the susceptibility of their fine-tuned models to be attacked (Li et al. 2020; Zou et al. 2023; Ye et al. 2024; Liu et al. 2024b). All the above issues may hinder the long-run sustenance of LLMs.

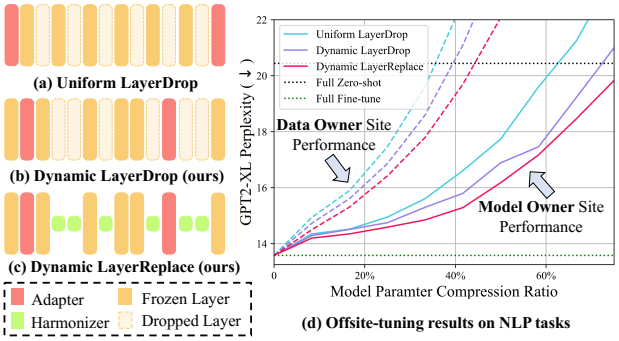

Figure 1: Comparison of layerwise compression strategies. (a) Uniform LayerDrop. (b) Our Dynamic LayerDrop drops layers with the estimated importance scores. (c) Our Dynamic LayerReplace with harmonizers. (d) Results of using different compression ratios. Our approach achieves better performance at the owner site while maintaining the performance discrepancy.

To protect both model ownership and data privacy effectively, the concept of offsite-tuning (Xiao, Lin, and Han 2023; Chua et al. 2023; Zhang et al. 2023a) has been proposed. As shown in Fig. 2(b), instead of training with the full model, it enables the data owner to conduct fine-tuning using a lossy compressed emulator provided by the model owner, whose paradigm would result in a performance-degraded emulator for data owners. The trained adapter is then returned to the model owner and plugged into the full model to create an adapted model with high performance. Particularly, as the key factor for model privacy, the performance discrepancy between the data owner's and owner's sites encourages downstream users to use the adapted full model. Thus, the primary challenge of offsite-tuning lies in efficiently compressing LLM to protect model privacy by maintaining the performance discrepancy but improving the

*Corresponding authors.

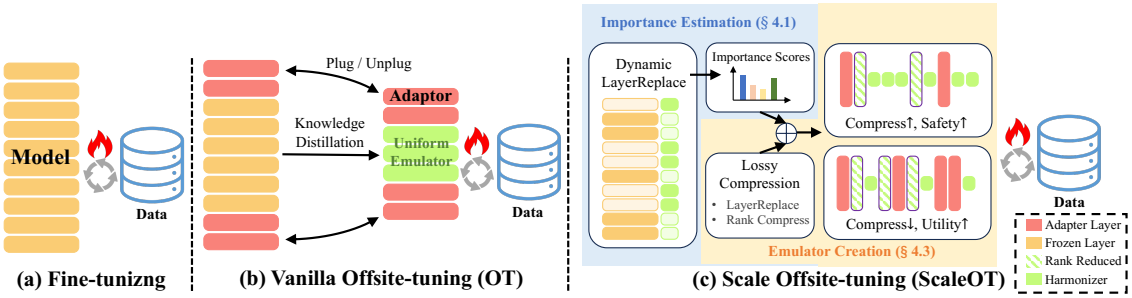

Figure 2: Comparison in various tuning methods. (a) Fine-tuning requires access to full model parameters and necessitates the co-location of data and model. (b) Vanilla OT (Xiao, Lin, and Han 2023) allows downstream users to fine-tune adapters on a lossy compressed emulator and then return the adapter. However, knowledge distillation (Sanh et al. 2019; Hinton, Vinyals, and Dean 2015) is very expensive, limiting its application. (c) The proposed ScaleOT introduces a layerwise importance-aware compression method Dynamic LayerReplace, providing privacy-utility-scalable emulators for downstream task tuning.

adapted full model.

Following the offsite tuning strategy, vanilla method OT (Xiao, Lin, and Han 2023) employs an *Uniform LayerDrop* (Sajjad et al. 2023) that uniformly drops a subset of layers from the full model, as shown in Fig. 1(a). However, despite many parameters in large models being redundant (Michel, Levy, and Neubig 2019), the importance of each layer varies significantly (Yao et al. 2024), and such uniform drop may lead to performance degradation of the adapted full model. Moreover, the direct LayerDrop causes a misalignment between the input and output hidden spaces of the dropped layers, which also leads to a decline in the performance of the owner site. Although knowledge distillation (Sanh et al. 2019; Hinton, Vinyals, and Dean 2015) can alleviate this issue, the massive cost of training a required emulator at least half the size of the LLM means that the massive cost of training raises a significant drawback for providing emulators with varying compression ratios.

To address these issues, this paper introduces a novel offsite-tuning framework, named ScaleOT, that provides lossy compressed emulators with various scales for model privacy and facilitates lossless tuning compared with full fine-tuning. As shown in Fig. 2(c), our framework consists of two stages: importance estimation and emulator generation. For the first stage, we propose an importance-aware layer-replace-based algorithm *Dynamic LayerReplace* (see Fig. 1(c)) that involves a reinforcement learning method to determine the importance of each layer within LLM. Simultaneously, for the less important layers, a set of trainable harmonizers, which are lightweight networks for better alignment between the remaining layers, are dynamically selected and trained as substitutions. In the second stage, according to the learned importance scores, the original model layers and their corresponding harmonizers can be combined variously to generate emulators while maintaining satisfactory performance at the model owner's site, as shown in Fig. 1(d). Empirically, we unveiled that using a rank-decomposition to further compress the remaining model layers can better certify privacy protection with mere modal performance degradation, from which we propose the Selective Rank Compression (SRC) method.

Extensive experiments on multiple models and datasets show that our approach outperforms the previous approaches while adjusting the compressed emulator model size and the rank reduction ratio in SRC, validating its effectiveness and feasibility.

In a nutshell, our contributions are threefold:

- **A flexible approach to produces various-sized compressed models for offsite-tuning:** We introduce an importance-aware lossy compression algorithm, Dynamic LayerReplace, oriented for offsite-tuning with LLMs that can scale up emulators by reinforcement learning and the harmonizers. All these components enable the flexibility of producing compressed models in various sizes.
- **Further compression for better privacy with mere fine-tuning degradation:** We propose the Selective Rank Compression strategy that can further enhance the model privacy with minimal performance loss.
- Comprehensive experiments show that our proposed ScaleOT outperforms the state-of-the-art methods.

Related Work

Large Language Models. The rapid development of hardware and the abundance of data available in the information age has made it possible to train large foundational models for various tasks, such as GPT-3 (Brown et al. 2020), CLIP (Radford et al. 2021), SAM (Kirillov et al. 2023), etc. These models are designed to have powerful general capabilities, capable of broadly solving problems through zero-shot learning or in-context learning (Dong et al. 2024; Wies, Levine, and Shashua 2024). However, when facing downstream tasks, transfer learning with a small set of data is the mainstream choice because it not only offers better performance but also avoids the extensive resources required for training models from scratch. Although the open-sourcing of many models has significantly advanced academic progress, the parameter exposure increases the risk of adversarial attacks (Li et al. 2020; Zou et al. 2023). On the other hand, closed-source models (Brown et al. 2020) present contradictions between model ownership and data privacy due to the co-location of data and model in conventional fine-tuning.

In summary, addressing these pain points, offsite tuning can provide a privacy-preserving, secure, and sustainable solution.

Offsite-tuning. Previous works mainly focus on protecting user data privacy, such as federated learning (Nguyen et al. 2021). However, the widespread usage of LLMs raises concerns about the ownership of the model, considering the significant cost and time required to train large models. Therefore, how to protect model privacy has emerged as a research hotspot. OT (Xiao, Lin, and Han 2023) was proposed as an effective bidirectional privacy-preserving solution for both model and data owners. Owing to its efficiency, it has been integrated into federated learning frameworks to facilitate privacy-preserving federated fine-tuning (Chua et al. 2023; Fan et al. 2023; Kuang et al. 2024). CRaSh (Zhang et al. 2023a) improved OT by generating emulators by clustering, dropping and sharing layers. Additionally, some studies (Hong et al. 2024) have explored the utilization of smaller language models to generate discrete prompts for offsite prompt tuning of large language models, though ensuring performance remains challenging.

Model Compression. The discovery of sparse sub-networks in convolutional, fully connected and Transformer models (Lv et al. 2023; Murty et al. 2023) has been validated by recent studies (Hoefler et al. 2021; Frankle and Carbin 2018). Previous works have found that models can be greatly compressed (often removing over 90% of parameters) with very little drop in accuracy. However, conventional compression methods like pruning (Han, Mao, and Dally 2016) and quantization (Jacob et al. 2018; Radford et al. 2023), while effective in compressing models, have been found to be difficult to use in offsite tuning (Xiao, Lin, and Han 2023), mainly due to consistent performance loss in both model and data sites. LayerDrop (Sajjad et al. 2023), another explored method, although useful, can degrade the final performance of the adapted model, unless applying knowledge distillation. In this study, we propose a layer-replace-based compression method oriented for offsite tuning.

Preliminary

In our study, we consider the privacy concerns preventing the sharing and co-location of data and LLMs between their respective owners. Our objective is to tune the model using the data owner's data without accessing the model owner's model weights. Starting with a pretrained LLM $\mathcal{M}$, parameterized by weights Θ, and a downstream dataset $\mathcal{D}$, we fine-tune this model on the downstream data to achieve $\mathcal{M}_\Theta \rightarrow \mathcal{M}_{\Theta+\Delta}, \Delta = \arg\min_\delta \mathcal{L}(\Theta + \delta, \mathcal{D})$. Our objective is to facilitate private transfer learning by finding an alternative, smaller, and weaker model $\mathcal{M}^*_{\Theta^*}$ (referred to as an Emulator) than $\mathcal{M}_\Theta$. This approach ensures that sharing $\mathcal{M}^*$ with downstream users does not threaten the ownership of the LLMs. Then, data owners fine-tune the substitute model with their datasets, resulting in $\mathcal{M}^*_{\Theta^*+\Delta^*}$. We hope that by reintegrating the trained weights Δ^* into the original model (represented as $\mathcal{M}_{\Theta+\Delta^*}$), we can nearly mirror the performance observed when directly optimizing $\mathcal{M}$ on the dataset (denoted as $\mathcal{M}_{\Theta+\Delta}$), thereby eliminating the need to access to $\mathcal{M}$ itself.

For convenience, we make definations as follows: zero-shot (ZS) performance with $\mathcal{M}_\Theta$, fine-tuning (FT) performance with $\mathcal{M}_{\Theta+\Delta}$, emulator ZS and FT performance with $\mathcal{M}^*_{\Theta^*}$ and $\mathcal{M}^*_{\Theta^*+\Delta^*}$, and plug-in performance with $\mathcal{M}_{\Theta+\Delta^*}$. An effective offsite-tuning should satisfy: 1) ZS $<$ Plug-in so that necessitates the fine-tune process. 2) Emulator FT $<$ Plug-in to discourage the downstream users to use the fine-tuned emulators. 3) Plug-in $\approx$ FT to encourage the downstream users to use the plugged model.

Main Methodology

Problem setting. In this paper, we concentrate on the design of offsite-tuning for the Transformer architecture (Vaswani et al. 2017), which is extensively used in LLMs (Radford et al. 2019; Brown et al. 2020; Touvron et al. 2023). We consider each transformer layer as the basic unit, and a LLM can be characterized as $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$, where n is the total number of the layers. Our approach involves dividing $\mathcal{M}$ into two components: a compact, trainable adapter $\mathcal{A} = \{m_i\}_{i\in\phi_\mathcal{A}}$ and the remaining part of the model $\mathcal{E} = \{m_i\}_{i\in\phi_\mathcal{E}}$, making $\mathcal{M} = [\mathcal{A}, \mathcal{E}]$. The sets of layer indices are defined such that $\phi_\mathcal{A} \cap \phi_\mathcal{E} = \emptyset$ and $\phi_\mathcal{A} \cup \phi_\mathcal{E} = \{1, 2, \ldots, n\}$. To safeguard the privacy of the model, a lossy compression is conducted on the frozen component $\mathcal{E}$, producing an emulator $\mathcal{E}^*$, thereby facilitating model tuning through updates to $\mathcal{A}$. After training at the data owner site, the updated adapter $\mathcal{A}'$ is returned to model owner side and replace the original $\mathcal{A}$ in the $\mathcal{M}$. Hence, the final updated LLM is denoted as $\mathcal{M}' = [\mathcal{A}', \mathcal{E}]$. Notably, the lossy compression inherently limits the model performance of $[\mathcal{A}', \mathcal{E}^*]$ for the downstream users, enabling the protection of model ownership.

This paper tackles the two crucial keys of the problem: obtaining suitable division of $\mathcal{A}$ and $\mathcal{E}$ and achieving better compression from $\mathcal{E}$ to $\mathcal{E}^*$ for the effective fine-tuning and maintaining the privacy protection. For the former, we introduce the importance scores on the model layers which could be used to guide the selection of $\mathcal{A}$ and $\mathcal{E}$. In particular, the importance scores are estimated through reinforcement learning during the dynamically replacing the original layers with lightweight networks. Those lightweight networks, termed as harmonizers, can be further adopted as the substitutions for the layers in the $\mathcal{E}$, which promotes the performance of full adapted model. Moreover, for the remained layers in $\mathcal{E}$ that replaced by the harmonizers, we additionally propose the selective rank compression method, which certify better privacy while maintaining the full adapted model's performance. The following parts of the section introduce details of each proposed components.

Importance-aware Dynamic LayerReplace

We propose a novel layer-replace-based compression algorithm called Dynamic LayerReplace. Our goal is to estimate the importance of each layer within LLM, and replace the less important layers with lightweight networks, named harmonizers, to preserve semantic consistency between layers. To accomplish this, we utilize a dual-process approach comprising reinforcement learning (RL) to assess the importance

of each LLM layer and deep learning (DL) for training the harmonizers via gradient descent. These processes alternate iteratively during the training phase to maintain stability.

Formally, we begin with a LLM denoted as $\mathcal{M}$. We then initialize the importance scores $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, and the harmonizers $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$. Two subsets of the dataset intended for pre-training are taken as the training sets $\mathcal{D}^T$ and validation sets $\mathcal{D}^V$, which are unrelated to the downstream tasks. During the training process, we update $\mathcal{S}$ utilizing RL and train $\mathcal{H}$ via DL, while keeping $\mathcal{M}$ unchanged. In the following, we will introduce LayerReplace sampling, which is the basic action of RL, and describe how to attain the importance score.

LayerReplace Sampling. We start by defining the state space for the RL process as the configuration of layers within the network, which includes both the presence of original layers and harmonizers. The decision to replace a specific layer with the corresponding harmonizer acts as the action, influenced by an action policy π_i based on the importance score of each layer:

$$\pi_i \triangleq U(0, \texttt{Sigmoid}(s_i)), \tag{1}$$

where $U(a, b)$ denotes the uniform distribution between a and b. For each time, we randomly sample a probability $p_i \sim \pi_i$, forming the probability set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ for all layers. Empirically, half of the layers in LLMs are sampled following $\mathcal{P}$ and then replaced with harmonizers. However, since the LLMs are typically deep, and the action policy in the early stages of training is inaccurate, selecting half of all layers directly risks selecting a significant number of adjacent layers, potentially leading to training collapse. To counter this and ensure training stability, we regroup the network's layers into N_g groups of index of adjacent layers and replace half of the layers in each group. The set of the group is denoted as $\{g_j\}_{j=1}^{N_g}$. The index set for the kept layers is constructed as follows:

$$\phi = \{i \text{ where } p_i > p^{g_j}, i \in g_j\}, \tag{2}$$

where p^{g_j} is the median probability in the j-th group, N_g is empirically set to 4 by default. According to ϕ, the sampled LayerReplace candidate network can be formed as:

$$F = f_1 \circ f_2 \circ \cdots \circ f_n, \quad f_i = \begin{cases} m_i & \text{if } i \in \phi \\ h_i & \text{otherwise} \end{cases}, \tag{3}$$

where $f_i \circ f_{i+1}$ represents compositional function.

Importance and Harmonizer Updating. We propose jointly updating importance scores with harmonizer to improve efficiency, involving training for DL and RL. For the training of DL, we perform LayerReplace Sampling for once and update the parameters of harmonizers within the sampled candidate network through task loss using training dataset $\mathcal{D}^T$, i.e., $\theta_{\mathcal{H}} \leftarrow \nabla_\theta L(\theta)$. Here, we uses negative log-likelihood loss, a widely accepted standard for next-token prediction. Subsequently, the gradients necessary for updating the harmonizer are derived through backpropagation.

For the training of RL, we samples N_c LayerReplace candidate networks based on the sampled probability sets $\{\mathcal{P}_j\}_{j=1}^{N_c}$. Then, we generate the index sets $\{\phi_j\}_{j=1}^{N_c}$, and their corresponding losses $\{\mathcal{L}_j\}_{j=1}^{N_c}$ with the held-out validation dataset $\mathcal{D}^V$. Last, the reward for j-th action policy can be written as:

$$r_j = e^{-\mathcal{L}_j} - \frac{1}{N_c} \sum\nolimits_{t=1}^{N_c} e^{-\mathcal{L}_t}. \tag{4}$$

If the reward is greater than 0, it indicates that this policy is more optimal due to experiencing a smaller loss with the sampled candidate network, compared to other policies. Accordingly, the layers contained within the policy are more important. Therefore, the importance score is updated by:

$$s_i = \begin{cases} s_i + r_j * \sigma(s_i) * (1 - \sigma(s_i)) & \text{if } i \in \phi_j \\ s_i & \text{otherwise} \end{cases}, \tag{5}$$

where σ represent sigmoid function.

After training with Dynamic LayerReplace, we can obtain the importance scores for each layer of the LLM and the harmonizers to replace these layers, which will be used in the subsequent process of emulator creation.

Selective Rank Compression

In many studies, LLMs have been found to hold an inherent over-parameterization nature (Hoefler et al. 2021; Hu et al. 2022). Consequently, the model weights can be low-rank approximated without substantially impacting the model's performance. Intuitively, we propose to compress the emulator's weights through low-rank approximation to enhance model privacy. As the higher-order components of weights are reduced, the emulator's expressive capability is further diminished, enabling a larger performance gap. Meanwhile, the remaining lower-order components of weights still provide approximate gradient directions for updating adapters during tuning. In the following, we will briefly introduce the rank-r approximation, and describe how we reduce the rank of specific layers for offsite tuning tasks.

SVD-based Rank-r Approximation. Given a matrix $W \in \mathbb{R}^{m \times n}$, and $r \in \mathbb{N}$, a rank-r approximation problem seek to find a matrix $\hat{W}$ that minimizes $\|W - \hat{W}\|_2$ while ensuring that the rank of $\hat{W}$ is at most r. Singular Value Decomposition (SVD) is proven optimal to this problem according to Eckart-Young-Mirsky theorem (Eckart and Young 1936). To achieve the rank-r approximation using SVD, the matrix W is decomposed as follows:

$$W = U \Sigma V^T, \tag{6}$$

where U and V are orthogonal matrices containing the left and right singular vectors of W, respectively, and Σ is a diagonal matrix with the singular values of W in descending order.

To form the rank-r approximation $\hat{W}$, only the first r singular values and corresponding singular vectors are used:

$$\hat{W} = U_r \Sigma_r V_r^T. \tag{7}$$

Here, U_r and V_r consist of the first r columns of U and V, and Σ_r includes only the top r singular values.

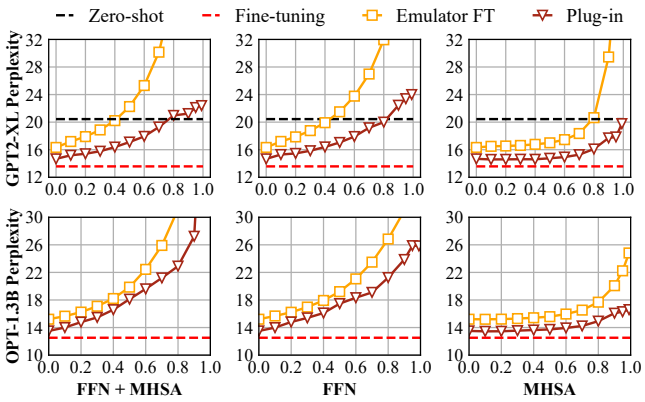

Figure 3: Rank Compression Study with varies β on multi-head self-attention (MHSA) layer and a Feedforward Network (FFN) layer in Transformer Block on WikiText dataset.

Rank Compression on Specific Modules. A typical Transformer (Vaswani et al. 2017) layer is composed of a multi-head self-attention (MHSA) layer followed by a Feedforward Network (FFN) layer. The MHSA layers facilitate interactions among tokens, while the FFN layer further processes information transformation within tokens. For expressive capability, the FFN's hidden dimension is empirically set to be very high, significantly exceeding its input and output dimensions (around $2.5\times$ to $4\times$). Considering that FFNs could be inherently high-rank, we leverage this characteristic and propose selectively applying rank compression to the weights in the MHSA layers to enhance the model privacy of the emulator. Empirical experiments, as shown in Fig. 3, indicate that applying rank compression to the weights in all layers (MHSA + FFN) and to the FFN alone both exponentially degrade the performance of both the model and data sites. In contrast, rank compression of the weights in MHSA leads to a rapid decline in the performance of emulator FT but a slow decrease in the performance of plug-in, especially when the rank compression ratio is larger than 0.6. Therefore, we choose to rank compress MHSA layers in the emulator to enhance model privacy, thus form Selective Rank Compression strategy.

Privacy-utility-scalable Emulator Creation

The creation of a privacy-utility-scalable emulator is defined by three quantities (N_a, α, β), consisting of the number of the adapted layers N_a, the proportion of layers replaced with harmonizers within LLM α, and the rank reduction ratio of SRC β. Taken together, these values describe how the emulator $\mathcal{E}^*$ is created with LLMs $\mathcal{M}$, the important scores $\mathcal{S}$ and the harmonizers $\mathcal{H}$.

Formally, given $\{g_j\}_{j=1}^{N_g}$, we identify two sets of indices:

$$\phi_{\mathcal{A}} = \{i \text{ where } s_i \geq s^{g_j,k}, i \in g_j\}, \tag{8}$$

$$\phi_{\mathcal{E}} = \{i \text{ where } s_i < s^{g_j,k}, i \in g_j\}, \tag{9}$$

where $k = \frac{N_a}{N_g}$, $s^{g_j,k}$ is the k-th largest importance score in j-th group. Our goal is to tune the most important layers while keeping the less ones within the LLM unchanged, thus forming $\mathcal{A} = \{m_i\}_{i\in\phi_{\mathcal{A}}}$ and $\mathcal{E} = \{m_i\}_{i\in\phi_{\mathcal{E}}}$. Subsequently, the indices set of harmonizers is denoted as follows:

$$\phi_{\mathcal{H}} = \{i \text{ where } s_i < s^{g_j,\kappa}, i \in g_j\}, \tag{10}$$

where $\kappa = \frac{n\times\alpha}{N_g}$, $s^{g_j,\kappa}$ is the κ-th largest importance score in j-th group, $\phi_H \in \phi_E$. The emulator $\mathcal{E}^*$ can be formed as:

$$\mathcal{E}^* = \{h_i\}_{i\in\phi_{\mathcal{H}}} \cup \{\text{SRC}(m_i, \beta)\}_{i\in\phi_{\mathcal{E}}\wedge i\notin\phi_{\mathcal{H}}}. \tag{11}$$

By modulating α and β, we effectively manage the model's privacy. Increasing values of α and β enhances privacy via higher compression rates, while decreasing these values enhances the adapted model's performance. After tuning $\mathcal{A}$ with $\mathcal{E}^*$, the downstream users return $\mathcal{A}'$ to model owner, to form the well-tuned LLMs $\mathcal{M}' = [\mathcal{A}', \mathcal{E}]$.

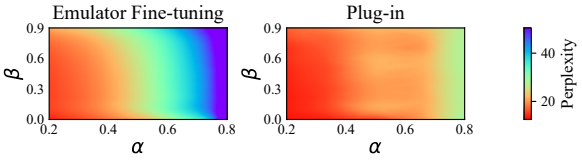

Figure 4: Joint effect of α and β in emulator generation.

Experiments

Experimental Setup

Models and Datasets. We evaluate our method on large language models, including GPT-2-XL (Radford et al. 2019), OPT-1.3B (Zhang et al. 2023b), OPT-6.7B (Zhang et al. 2023b) and LLaMA (Touvron et al. 2023). We validate our method across one generation task WikiText (Merity et al. 2017), and eight question answering benchmarks: OBQA (Mihaylov et al. 2018), PIQA (Bisk et al. 2020), ARC (Clark et al. 2018), HellaSwag (Zellers et al. 2019), SciQ (Welbl, Liu, and Gardner 2017), WebQuestions (Berant et al. 2013) and RACE (Lai et al. 2017).

Evaluation Protocols. For a fair comparison, we adopt the same evaluation metric used in previous studies (Xiao, Lin, and Han 2023). Specifically, we measure perplexity on the WikiText dataset and report accuracy on the other tasks. We use `lm-eval-harness` [1] for language model evaluation.

Implementation Details. In the training of the Dynamic LayerReplace, we utilize the Pile corpus (Gao et al. 2020) datasets for language. For Dynamic LayerReplace, $N_c = 3$ and $N_g = 4$ are set empirically. For harmonizer, we utilize a simple low-rank FFN with ReLU activation and rank of 64 and 256 for medium and large size LLM respectively. For the construction of emulators, we set $\alpha = 0.25$ and $\beta = 0.8$ by default to balance the privacy-utility trade-off, unless otherwise specified. For fair comparison, N_a is set to be consistent with OT (Xiao, Lin, and Han 2023), meaning that only about 10% of the parameters are tuned, as opposed to full fine-tuning. For the offsite tuning phase, we employ the AdamW Optimizer, experimenting with a range of learning rates: [2e-5, 5e-5, 1e-4, 2e-4, 3e-4]. All experiments are conducted on a workstation with 8 V100 GPUs.

[1] https://github.com/EleutherAI/lm-evaluation-harness

Method	Setting	OBQA	PIQA	ARC-E	ARC-C	HellaSwag	SciQ	WebQs	RACE	Avg.	Δ ↑
GPT2-XL											
Full Model	Zero-shot (ZS)	23.0	70.9	58.2	25.1	40.0	83.2	1.5	33.0	41.9	-
	Fine-tuning (FT)	30.0	73.2	62.9	30.0	40.7	92.5	26.4	43.2	49.9	
OT	Emulator ZS ↓	18.8	67.7	53.2	20.8	33.5	77.0	0.2	30.0	37.7	
	Emulator FT ↓	24.0	70.3	58.2	23.9	35.8	92.7	18.9	39.4	45.4	2.9
	Plug-in ↑	28.2	73.6	61.4	28.5	41.6	93.2	19.9	39.9	48.3	
ScaleOT w/o SRC	Emulator ZS ↓	19.4	69.5	52.9	23.9	36.6	81.3	1.2	33.3	39.8	
	Emulator FT ↓	29.6	72.7	58.6	26.5	39.6	93.2	24.4	43.4	48.5	1.6
	Plug-in ↑	31.2	73.6	63.5	29.4	41.8	93.9	24.4	42.6	**50.1**	
ScaleOT	Emulator ZS ↓	15.4	61.3	37.9	19.5	28.6	53.8	0.0	24.2	**30.1**	
	Emulator FT ↓	29.8	70.9	54.4	23.0	35.7	90.7	16.3	38.9	**45.0**	**4.3**
	Plug-in ↑	31.6	73.4	63.6	29.1	41.3	94.2	21.2	40.3	49.3	
OPT-1.3B											
Full Model	Zero-shot (ZS)	23.4	71.6	56.9	23.5	41.5	84.4	4.6	34.2	42.5	-
	Fine-tuning (FT)	31.4	75.2	61.3	27.7	42.7	92.5	31.2	37.0	49.9	
OT	Emulator ZS ↓	19.4	68.7	53.9	21.5	35.1	80.9	1.3	33.0	39.2	
	Emulator FT ↓	24.8	71.6	58.1	26.1	37.0	92.2	24.3	38.6	46.6	2.4
	Plug-in ↑	29.0	74.5	59.4	27.8	43.3	92.9	26.2	38.9	49.0	
ScaleOT w/o SRC	Emulator ZS ↓	20.2	69.8	54.1	24.6	38.2	85.2	1.4	36.3	41.2	
	Emulator FT ↓	28.4	71.0	52.9	27.5	38.7	90.9	27.3	39.9	47.1	3.0
	Plug-in ↑	29.6	74.5	61.6	27.9	43.7	93.3	29.8	41.9	**50.3**	
ScaleOT	Emulator ZS ↓	17.2	63.1	41.8	19.5	32.1	59.9	0.1	27.2	**32.6**	
	Emulator FT ↓	27.2	70.9	52.5	26.5	37.8	90.0	25.3	39.1	**46.2**	**3.7**
	Plug-in ↑	28.2	75.2	61.9	28.3	42.9	94.0	28.2	40.8	49.9	

Table 1: Comparative results of offsite-tuning with medium-size language model on eight question answering benchmarks. Δ denote the performance difference between Emulator fine-tuning and Plug-in. Best in **bold** and second best in underline.

Scalability in Emulator Generation

To showcase the scalability of privacy and utility using our proposed ScaleOT, we conducted experiments using OPT-1.3B with WikiText by tuning α and β. As demonstrated in Fig. 4, gradually increasing α causes a simultaneous decline in both emulator fine-tuning and plug-in performance. Meanwhile, increasing β results in a larger performance discrepancy without harming plugin performance. We observed a broad parameter space to create a reasonable emulator, particularly when α is less than 0.5 and β is greater than 0.6. This flexibility allows us to adjust our strategy to create either high-compressed, high-privacy emulators or low-compressed, high-performance emulators.

Main Results

We first evaluate our proposed ScaleOT on medium-sized models in Tab. 1, including GPT2-XL and OPT-1.3B with around one billion parameters. All methods meet the conditions of offsite tuning, i.e., the performance of the plug-in surpasses both the full model zero-shot and the performance of emulator fine-tuning. Moreover, ScaleOT without SRC achieves nearly lossless performance compared to full fine-tuning. This highlights the efficacy of Dynamic Layer-Replace against the Uniform LayerDrop used in the baseline OT. Notably, due to the selection of important layers for updating, the performance of the plug-in can exceed that of directly fine-tuning on LLMs, thanks to the better convergence brought by sparse training (Pan et al. 2024; Yao et al. 2024). Finally, the incorporation of SRC significantly reduces the performance of emulator zero-shot and fine-tuning by average 9.2% and 2.2%, with minimal drop in the performance of the plugin. Overall, ScaleOT not only achieves better performance but also ensures good model privacy.

Subsequently, we validated the effectiveness of larger LLMs, including OPT-6.7B and LLaMA-7B, each with approximately 7 billion parameters. As shown in Tab 2, OT did not achieve satisfactory performance due to its inability to perform knowledge distillation on limited hardware. CRaSh improved performance through LayerSharing, but its lack of healing performance after compression resulted in suboptimal outcomes. In contrast, ScaleOT made the compression of large models feasible by requiring only about 1-2% of the parameters to be trained during the compression phase. Notably, our method achieve strong plug-in performance on the WebQs task, where the zero-shot accuracy is zero, highlighting its potential for new downstream applications. Furthermore, ScaleOT achieved commendable results, demonstrating that its effectiveness is not restricted to specific model sizes. This makes ScaleOT a valuable strategy for enhancing offsite-tuning outcomes across models of varying scales.

Analytical Study

Effect of Selective Rank Compression (SRC). To evaluate the effectiveness of SRC in improving model privacy, we conducted experiments on the WikiText dataset with GPT2-XL and OPT-1.3B. As depicted in Fig. 3, we linearly increase the compression ratio β from 0 to 1, resulting in lower rank in the network. A consistent decreases in both emulator fine-tuning and plug-in performance were observed as β increased, particularly in configurations that included the feed-

Setting	OBQA	PIQA	ARC-E	ARC-C	HellaSwag	SciQ	WebQs	RACE	Avg.
OPT-6.7B									
Zero-shot	27.6	76.2	65.5	30.6	50.5	90.1	8.8	38.2	48.4
Emulator ZS	23.9	59.5	58.1	26.4	30.9	86.6	2.0	33.4	40.0
Emulator FT	30.2	73.0	67.8	28.0	45.1	92.4	21.9	44.0	49.5
OT Plug-in	33.8	77.7	66.8	33.9	52.1	91.9	23.9	44.1	53.0
CRaSh Plug-in	38.8	78.0	70.7	36.3	53.4	95.3	26.1	45.2	55.3
ScaleOT w/o SRC Plug-in	41.2	76.6	70.2	35.9	53.5	94.5	35.2	46.9	**56.8**
ScaleOT Plug-in	33.2	78.1	70.1	35.3	52.2	95.7	33.9	45.3	55.5
LLaMA-7B									
Full Zero-shot	28.2	78.3	67.3	38.2	56.4	89.7	0.0	40.0	49.8
Emulator ZS	23.6	72.4	57.8	29.9	46.1	81.5	0.0	35.6	43.4
Emulator FT	31.0	74.7	62.3	28.8	44.2	92.3	21.8	42.9	49.8
OT Plug-in	33.0	78.8	69.6	39.0	57.4	83.5	27.3	44.0	54.1
CRaSh Plug-in	34.6	80.0	71.3	41.8	58.4	95.1	29.8	45.6	57.1
ScaleOT w/o SRC Plug-in	40.8	79.6	75.3	47.7	59.5	96.2	43.2	48.5	**61.2**
ScaleOT Plug-in	37.4	79.7	73.2	42.3	58.1	95.7	33.7	45.6	58.2

Table 2: Comparative results of offsite-tuning with large-size language model on eight question answering benchmarks.

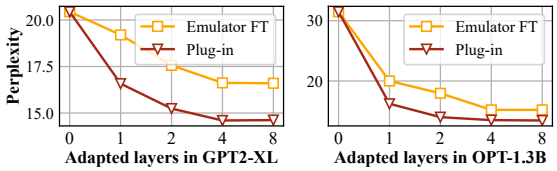

Figure 5: Number of adapted layers in ScaleOT.

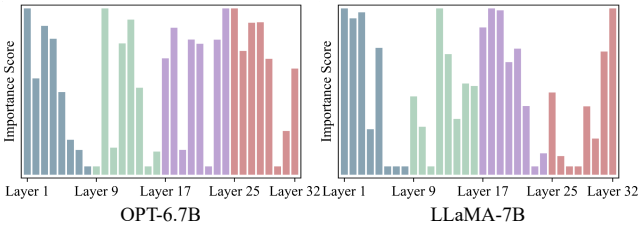

Figure 6: Visualization of the estimated importance score.

forward network (FFN), where a linear relationship was evident. In contrast, for MHSA configurations, within a range of 0.6 to 1, emulator FT performance showed an exponential decline, whereas plug-in performance exhibited a linear reduction. This indicates that SRC has potential to enhance model privacy without degrading overall performance.

Number of Adapted Layers. We analyzed the impact of the number of adapted layers N_a on model performance. As illustrated in the Fig. 5, With the increase in the number of adapted layers, the performance grows linearly and saturates at $N_a = 4$. Doubling N_a to 8 does not yield any significant improvement. Consequently, $N_a = 4$ is selected for optimal balance between performance and computational efficiency.

Importance Score. We visualize the estimated importance scores for OPT-6.7B and LLaMA-7B. As illustrated in Fig. 6, it is evident that there is considerable variation in importance distribution across networks. However, a consistent pattern emerges: the first layer holds significant importance. This finding echoes the observations of OT (Xiao, Lin, and Han 2023), albeit lacking an explicit explanation.

Orthogonality with Parameter-efficient Fine-tuning. ScaleOT is designed to integrate seamlessly with parameter-efficient fine-tuning (PEFT) methods, facilitating a combined approach that significantly decreases the trainable parameters and enhances efficiency. This can be accomplished by employing PEFT methods in adapter layers, including strategies such as Adapter-tuning (Houlsby et al. 2019) and LoRA (Hu et al. 2022). As shown in Tab. 3, we observed that Adapter-tuning and LoRA substantially reduced the trainable parameters while maintaining plug-in performance.

Methods	#Total Params	#Trainable Params	Emu. FT ↑	Plug-in ↓
Zero Shot	1.64B	0	-	20.44
Full Finetune	1.64B	1.48B	-	13.58
ScaleOT w/o SRC	1.27B	127M	16.62	14.60
+Adapter (d=64)		1.65M	17.29	14.92
+LoRA (r=4)		410K	16.75	**14.53**
ScaleOT	1.05B	127M	20.65	15.95
+Adapter (d=64)		1.65M	21.69	16.03
+LoRA (r=4)		410K	21.18	**15.85**

Table 3: ScaleOT is orthogonal to most existing parameter-efficient fine-tuning techniques. Scores are validated with GPT2-XL on WikiText-2 in terms of perplexity.

Conclusion

In this paper, we propose a novel privacy-utility-scalable offsite-tuning framework, namely ScaleOT. We introduce a layerwise importance-aware lossy compression algorithm, Dynamic LayerReplace, to scale up the emulator and adapter by reinforcement learning and the harmonizers. Then, we propose Selective Rank Compression (SRC) that can enhance model privacy with minimal impact on performance. Comprehensive experiments across various LLMs and tasks showcase the superiority of our proposed ScaleOT.

Acknowledgments

This work was supported by Ant Group Postdoctoral Programme.

References

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *Conference on Empirical Methods in Natural Language Processing*, 1533–1544.

Bisk, Y.; Zellers, R.; Gao, J.; Choi, Y.; et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *AAAI*, volume 34, 7432–7439.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901.

Chua, T. J.; Yu, W.; Zhao, J.; and Lam, K.-Y. 2023. FedPEAT: Convergence of Federated Learning, Parameter-Efficient Fine Tuning, and Emulator Assisted Tuning for Artificial Intelligence Foundation Models with Mobile Edge Computing. *arXiv preprint arXiv:2310.17491*.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Dai, W.; Li, J.; Li, D.; Tiong, A. M. H.; Zhao, J.; Wang, W.; Li, B.; Fung, P. N.; and Hoi, S. 2024. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Ma, J.; Li, R.; Xia, H.; Xu, J.; Wu, Z.; Chang, B.; Sun, X.; and Sui, Z. 2024. A Survey on In-context Learning. In *Conference on Empirical Methods in Natural Language Processing*, 1107–1128.

Du, Z.; Qian, Y.; Liu, X.; Ding, M.; Qiu, J.; Yang, Z.; and Tang, J. 2022. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. In *Association for Computational Linguistics*, 320–335.

Eckart, C.; and Young, G. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3): 211–218.

Fan, T.; Kang, Y.; Ma, G.; Chen, W.; Wei, W.; Fan, L.; and Yang, Q. 2023. Fate-llm: A industrial grade federated learning framework for large language models. *arXiv preprint arXiv:2310.10049*.

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

Gao, L.; Biderman, S.; Black, S.; Golding, L.; Hoppe, T.; Foster, C.; Phang, J.; He, H.; Thite, A.; Nabeshima, N.; et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hoefler, T.; Alistarh, D.; Ben-Nun, T.; Dryden, N.; and Peste, A. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241): 1–124.

Hong, J.; Wang, J. T.; Zhang, C.; LI, Z.; Li, B.; and Wang, Z. 2024. DP-OPT: Make Large Language Model Your Privacy-Preserving Prompt Engineer. In *International Conference on Learning Representations*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2790–2799.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4015–4026.

Kuang, W.; Qian, B.; Li, Z.; Chen, D.; Gao, D.; Pan, X.; Xie, Y.; Li, Y.; Ding, B.; and Zhou, J. 2024. FederatedScope-LLM: A Comprehensive Package for Fine-tuning Large Language Models in Federated Learning. In *Conference on Knowledge Discovery and Data Mining*, 5260–5271.

Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; and Hovy, E. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Conference on Empirical Methods in Natural Language Processing*, 785–794.

Li, L.; Ma, R.; Guo, Q.; Xue, X.; and Qiu, X. 2020. BERT-ATTACK: Adversarial Attack Against BERT Using BERT. In *Conference on Empirical Methods in Natural Language Processing*, 6193–6202.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2024a. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Liu, H.; Liu, Z.; Tang, R.; Yuan, J.; Zhong, S.; Chuang, Y.-N.; Li, L.; Chen, R.; and Hu, X. 2024b. LoRA-as-an-Attack! Piercing LLM Safety Under The Share-and-Play Scenario. *arXiv preprint arXiv:2403.00108*.

Lv, X.; Zhang, P.; Li, S.; Gan, G.; and Sun, Y. 2023. Lightformer: Light-weight transformer using svd-based weight transfer and parameter sharing. In *Findings of the Association for Computational Linguistics*, 10323–10335.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2017. Pointer Sentinel Mixture Models. In *International Conference on Learning Representations*.

Michel, P.; Levy, O.; and Neubig, G. 2019. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, 32: 14014–14024.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Murty, S.; Sharma, P.; Andreas, J.; and Manning, C. D. 2023. Characterizing intrinsic compositionality in transformers with Tree Projections. In *International Conference on Learning Representations*.

Nguyen, D. C.; Ding, M.; Pathirana, P. N.; Seneviratne, A.; Li, J.; and Poor, H. V. 2021. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3): 1622–1658.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744.

Pan, R.; Liu, X.; Diao, S.; Pi, R.; Zhang, J.; Han, C.; and Zhang, T. 2024. LISA: Layerwise Importance Sampling for Memory-Efficient Large Language Model Fine-Tuning. *arXiv preprint arXiv:2403.17919*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 8748–8763.

Radford, A.; Kim, J. W.; Xu, T.; Brockman, G.; McLeavey, C.; and Sutskever, I. 2023. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, 28492–28518. PMLR.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Sajjad, H.; Dalvi, F.; Durrani, N.; and Nakov, P. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77: 101429.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Tan, Z.; Yang, X.; Wang, Q.; Nguyen, A.; and Huang, K. 2024. Interpret Your Decision: Logical Reasoning Regularization for Generalization in Visual Classification. In *Advances in Neural Information Processing Systems*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30.

Wei, J.; Bosma, M.; Zhao, V. Y.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

Welbl, J.; Liu, N. F.; and Gardner, M. 2017. Crowdsourcing Multiple Choice Science Questions. In *EMNLP Workshop*, 94–106. Association for Computational Linguistics.

Wies, N.; Levine, Y.; and Shashua, A. 2024. The learnability of in-context learning. *Advances in Neural Information Processing Systems*, 36.

Wortsman, M.; Ilharco, G.; Kim, J. W.; Li, M.; Kornblith, S.; Roelofs, R.; Lopes, R. G.; Hajishirzi, H.; Farhadi, A.; and Namkoong, H. 2022. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7959–7971.

Xiao, G.; Lin, J.; and Han, S. 2023. Offsite-tuning: Transfer learning without full model. *arXiv preprint arXiv:2302.04870*.

Yao, K.; Gao, P.; Li, L.; Zhao, Y.; Wang, X.; Wang, W.; and Zhu, J. 2024. Layer-wise Importance Matters: Less Memory for Better Performance in Parameter-efficient Fine-tuning of Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 1977–1992.

Ye, T.; Chen, C.; Wang, Y.; Li, X.; and Gao, M. 2024. BapFL: You can Backdoor Personalized Federated Learning. *ACM Trans. Knowl. Discov. Data*, 18(7): 166.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Association for Computational Linguistics*, 4791–4800.

Zhang, K.; Ding, N.; Qi, B.; Zhu, X.; Long, X.; and Zhou, B. 2023a. CRaSh: Clustering, Removing, and Sharing Enhance Fine-tuning without Full Large Language Model. In *Conference on Empirical Methods in Natural Language Processing*, 9612–9637.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2023b. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhou, K.; Yang, J.; Loy, C. C.; and Liu, Z. 2022. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16816–16825.

Zou, A.; Wang, Z.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.