AegisGuard: RL-Guided Adapter Tuning for TEE-Based Efficient & Secure On-Device Inference

Che Wang[1,2]*, Ziqi Zhang[3], Yinggui Wang[5], Tiantong Wang[2], Yurong Hao[2],
Jianbo Gao[4], Tao Wei[5], Yang Cao[6], Zhong Chen[1], Wei Yang Bryan Lim[2]
School of Computer Science, Peking University[1],
Nanyang Technological University[2], University of Illinois Urbana-Champaign[3],
Beijing Jiaotong University[4], Ant Group[5], Institute of Science Tokyo[6]
chewang@stu.pku.edu.cn

Abstract

On-device large models (LMs) reduce cloud dependency but expose proprietary model weights to the end-user, making them vulnerable to white-box model stealing (MS) attacks. A common defense is TEE-Shielded DNN Partition (TSDP), which places all trainable LoRA adapters (fine tuned on private data) inside a trusted execution environment (TEE). However, this design suffers from excessive host-to-TEE communication latency. We propose `AegisGuard`, a fine tuning and deployment framework that selectively shields the MS sensitive adapters while offloading the rest to the GPU, balancing security and efficiency. `AegisGuard` integrates two key components: i) RL-based Sensitivity Measurement (RSM), which injects Gaussian noise during training and applies a lightweight reinforcement learning to rank adapters based on their impact on model stealing; and (ii) Shielded-Adapter Compression (SAC), which structurally prunes the selected adapters to reduce both parameter size and intermediate feature maps, further lowering TEE computation and data transfer costs. Extensive experiments demonstrate that `AegisGuard` achieves black-box level MS resilience (surrogate accuracy around 39%, matching fully shielded baselines), while reducing end-to-end inference latency by 2–3× and cutting TEE memory usage by 4× compared to state-of-the-art TSDP methods.

1 Introduction

The rapid development of large models (LMs) [6, 10, 28] has enabled efficient on-device LM inference, attracting attention from both academia and industry [2, 22, 39]. Particularly with parameter-efficient fine-tuning (PEFT; e.g., Low-Rank Adaptation [15]), LMs can quickly adapt to various downstream tasks. A popular solution to provide LM-based services is to deploy proprietary LMs fine-tuned on private data to the users' devices. However, such deployments introduce serious security risks, as attackers with visibility of model parameters can exploit exposed components (white-box accessible to the user) to effectively perform Model Stealing attacks [26], replicating and redistributing proprietary model capabilities by a surrogate model.

Existing works proposed TEE-Shielded DNN Partition (TSDP) [45, 4, 47, 36] to enhance the security of on-device inference. TSDP partitions the model into two parts and deploy them across TEE and untrusted GPU for accelerating the efficiency of trusted inference. The goal of TSDP is to reduce the workload in TEE by outsourcing the computation-intensive layers to untrusted GPUs, premised on model security. Compared to cryptographic solutions like multi-party computation (MPC)[42]

*Work done at NTU

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

and homomorphic encryption (HE)[50], TEEs introduce less overhead and thus are more practical in real-world LMs deployment.

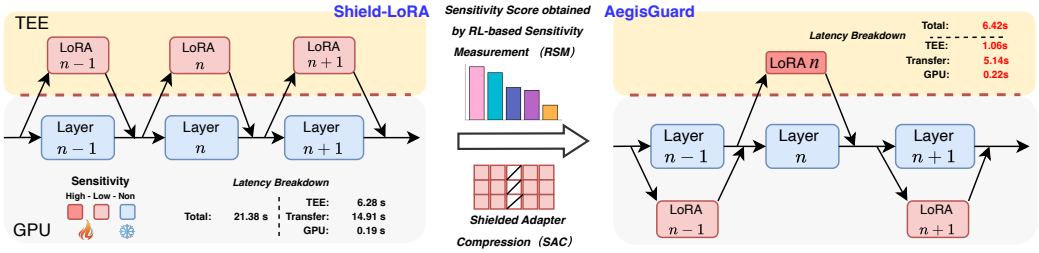

Figure 1: Shield-LoRA (Left), shields all adapters in TEEs. `AegisGuard` (Right), selectively shields LoRA n, the higher sensitive adapter in TEE to reduce data transfer and computation costs.

A straightforward solution that uses TSDP to protect LM is to shield LoRA adapters (trained by private data) in TEE and leave the base model layers (publicly available) offloaded on GPU (i.e., Shield-LoRA), as shown in the left part of Fig. 1. However, this solutions face a new challenge: frequent data transfer between adapters (in TEE) and base models (on GPU). According to our preliminary evaluation (16 batch size) on LLaMA-7B inference, the data transfer (14.91s) takes over 60% of the total inference latency (21.38s). Existing TSDP work mainly focuses on how to select critical model weights to shield [49, 47] or how to obfuscate offloaded weights [45, 34]. Thus, how to reduce the communication cost is still an open problem for TSDP in LM scenario.

To address this challenge, we propose `AegisGuard`, a novel fine-tuning and deployment framework that minimizes communication overhead while maintaining strong model protection against MS attacks. The key insight behind `AegisGuard` is that not all LoRA adapters contribute equally to privacy leakage. The sensitivity of an adapter (defined as how much its exposure helps the performance of model stealing attack) varies across layers. This motivates a selective shielding strategy, where only the most sensitive adapters are executed inside the TEE, while less sensitive ones are offloaded to the untrusted GPU.

One challenge is how to evaluate the adapter sensitivity. Directly evaluating the sensitivity via MS attacks is prohibitively costly. We instead propose a lightweight approach, i) Reinforcement Learning-based Sensitivity Measurement (RSM), which estimates each adapter's privacy risk by combining reinforcement learning with Gaussian noise perturbations. More sensitive adapters are then prioritized for fine-tuning and selected for shielding; ii) Shielded Adapter Compression (SAC), which structurally compresses both parameter size and intermediate feature map dimensions of adapters identified by RSM, improving execution efficiency within the TEE.

Extensive experiments demonstrate `AegisGuard` outperforms existing approaches, achieving 2-3$\times$ inference latency speedup towards Shield-LoRA. `AegisGuard` provides an effective defense against MS attack. The attack accuracy is averagely 39.1%, similar to the black box protection (accuracy of 38.9%). `AegisGuard` introduces a negligible accuracy drop by 0.12%.

The contributions of this paper are as follows:

- We propose to reduce the communication workload between TEE and GPU for TSDP solutions by an RL-based sensitivity measurement approach for on-device LM inference.
- We design `AegisGuard`, a fine-tuning framework that measures the adapter's sensitivity and compresses intermediate feature maps to reduce the communication cost after deployment.
- We conduct comprehensive experiments for deployed LMs, demonstrating superior inference efficiency, robust security against model stealing attack, and minimal accuracy loss of `AegisGuard`.

2 Related works

2.1 TEE

Trusted Excution Environments (TEEs) is a secure space that is isolated from other parts of the host machine [9, 1]. We follow prior work to regard TEE as a secure and computation-limited area

(without GPU) on a malicious adversary host machine [47, 49]. TEEs, such as ARM TrustZone [1], Intel SGX [9], provide hardware-enforced isolated enclaves designed to secure sensitive data and computations against adversarial access. Recent works have explored utilizing TEEs to protect model security on users' devices against possibly malicious device owners [4, 49, 17, 47, 34, 23]. The goal of TEE-based defenses is to downgrade the accurate and efficient white-box model stealing attacks to inaccurate and slow black-box attacks.

2.2 TEE-Shielded DNN Partition (TSDP)

Due to the limited computational capability of TEE, it is challenging to protect the entire model using TEE [36, 47]. To address this issue, researchers propose TEE-Shielded DNN Partition (TSDP) solutions that utilize co-located untrusted GPUs to accelerate the computation phase [49, 47, 14, 34]. TSDP solutions partition the model into two parts. One is a critical and computationally lightweight part, and the other is a computationally intensive part. The TEE shields the first part, and the second part is offloaded to GPUs.

Limitation of existing TSDP. Although TSDP can mitigate the computation overhead by utilizing GPU accelerators, existing solutions face another bottleneck: the cost of data transfer between TEE and GPU. According to the measurement of the latest TSDP work, the time of data transfer can take as much as 35% of the total inference latency [47]. It is because the DNN layers are heterogeneously deployed across TEE and GPU, and TSDP solutions have to transfer the internal feature map between TEE and GPU. The core goal of this paper is to reduce the communication workload by minimizing the number of shielded adapters in TEEs while not compromising on security.

Gray-box Model Stealing (MS) Attack. The goal of a MS attack is to steal the functionality of a victim model by acquiring a surrogate model with a similar performance to the victim model [27]. The attack against the TSDP models is a gray-box model stealing because TSDP offloads some computation to the untrusted GPU. The adversary can utilize computation operators (e.g., model weights) to obtain more information than is available through black-box protection (shielding the whole model in TEE). The goal of TSDP is to provide a black-box-level protection against MS attack in this gray-box setting [34, 33, 45].

2.3 Large Model Compression

Recent studies have validated the existence of sparse sub-networks in Transformer models, achieving significant compression (e.g., over 50% parameter reduction) with minimal accuracy degradation [20, 43, 19, 13]. While some compression methods like Singular Value Decomposition (SVD) [37] can effectively reduce model size, they do not compress the intermediate feature dimensions. Because TSDP deploys an LM across TEE and GPU, it needs to transfer intermediate features between TEE and GPU. The feature dimension determines the amount of data communication and affects the efficiency of TSDP. Thus, structural pruning[43] applied during fine-tuning provides more hardware-friendly compression: it directly removes redundant parameters (e.g., attention heads or ffn layers) as long as output dimensions, thereby generating sparse matrices that align with the computation and memory optimizations in TEEs.

3 Design of `AegisGuard`

3.1 Threats model

We adopt a gray-box security model consistent with prior TSDP literature [47, 34, 33, 45]. The model owner is the defender, who deploys a fine-tuned model to a user device for on-device inference. The deployed model has two components: i) an open-source pretrained backbone such as LLaMA, and ii) LoRA adapters fine-tuned using private data. The device owner is an honest-but-curious attacker that follows the correct inference protocol, but attempt to perform a MS attack, in which a surrogate model is trained to approximate the victim model's behavior using only the exposed components. The defender uses a TEE (e.g., Intel SGX) to shield selected LoRA adapters while offloading the rest of the model to a co-located GPU. We assume the TEE is secure and cannot be compromised. However, the attacker can observe: i) all model weights and parameters that are offloaded to the GPU, and ii) inference outputs from the full model. Our defense objective is to minimize the performance of the surrogate model while maintaining low inference latency and accuracy for the legitimate user.

Our threat model focuses on protecting the model's proprietary fine-tuning knowledge against parameter-extraction and functional imitation attacks. We do not address input reconstruction or training data inversion attacks based on feature leakage from unshielded components. These are orthogonal threats, and we note that complementary techniques such as differentially private training, input anonymization, or secure enclaved preprocessing may be integrated with `AegisGuard` to provide stronger end-to-end protection.

3.2 Problem formulation

`AegisGuard` aims to achieve a three-fold goal during the inference phase: security, efficiency, and accuracy. Security means that `AegisGuard` should defend against gray-box MS attacks [47] and minimize the surrogate model's performance. Efficiency means `AegisGuard` should reduce the overall inference latency by minimizing the amount of communication and computation in the slow TEE. Accuracy means `AegisGuard` should maintain a comparable accuracy with the model trained by standard fine-tuning. To better understand the `AegisGuard` approach, we define some symbols.

Let $\mathcal{M}$ denote the base model, with $\mathcal{D}_{train}$ representing training datasets. Similar to existing PEFT work [15], `AegisGuard` trains a set of adapters parameterized as $\Delta W = BA$. We set a decoder layer as the unit. Let $\mathcal{S} = [s_1, s_2, ..., s_n]$ denote the sensitivity scores for n layers, and $\mathcal{A} = [a_1, a_2, ..., a_t]$ represent policy actions of RSM, denotes selecting sensitive layers as trainable at each step, where t is step number. The objective of our approach can be formulated as:

$$F = \arg\min_{\Delta W} \mathcal{L}_{\text{FT}}(\mathcal{M}(\mathcal{D}_{train}); \Delta W) + \arg\min_{|\Delta W|} \mathcal{T}_{\text{SAC}}(\Delta W; \mathcal{S}) + \arg\max_{\mathcal{S}} \mathcal{J}_{\text{RSM}}(\mathcal{S}|\Delta W; \mathcal{D}_{train}; \mathcal{A}) \quad (1)$$

The first term $\mathcal{L}_{\text{FT}}$ minimizes task-specific loss through gradient-based fine-tuning. The second term $\mathcal{T}_{\text{SAC}}$ jointly optimizes higher sensitive adapter sparsity by pruning parameters based on sensitivity score $\mathcal{S}$, achieving dynamic model compression. The third term $\mathcal{J}_{\text{RSM}}$ maximizes privacy protection via RL. It updates $\mathcal{S}$ by evaluating sensitivity via noise perturbations to adapters and uses policy actions $\mathcal{A}$ to sample and train these sensitive adapters more progressively.

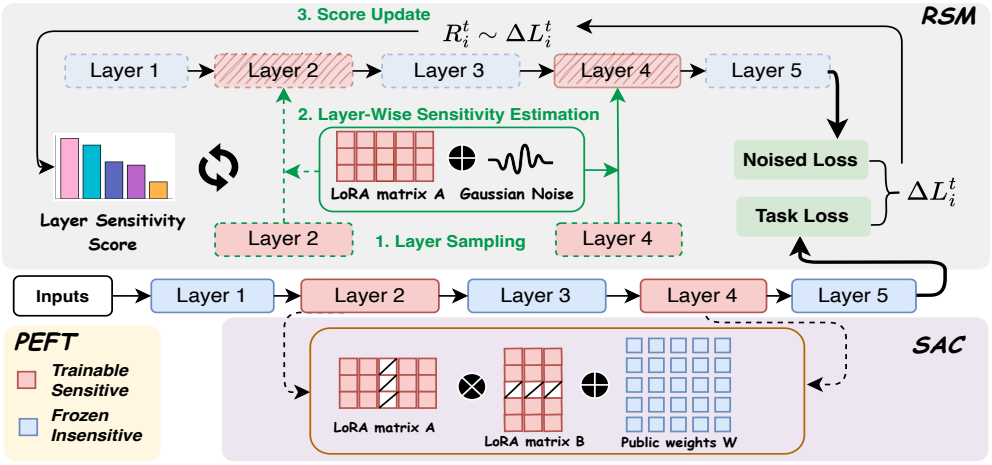

Figure 2: Fine-tuning of `AegisGuard`, integrates two main modules: RL-based Sensitivity Measurement (upper) and Shielded Adapter Compression (lower).

3.3 Overview

Fig. 2 illustrates the design of `AegisGuard`, which comprises two modules: RL-based Sensitivity Measurement (**RSM**) and Shielded Adapter Compression (**SAC**). The RSM outputs $\mathcal{S}$, which represents the privacy sensitivity of each layer, while the SAC outputs the compressed sensitive adapters. The two modules are integrated to the fine-tuning phase to dynamically estimate layer-wise sensitivity and compress the sensitive layers. Specifically, during fine-tuning phase, `AegisGuard` conducts **RSM** and **SAC** at regular intervals to monitor the change in layer-wise sensitivity and adjust

it accordingly. RSM consists of three stages, which follows the RL pipeline: policy action, reward feedback, and state update [11, 30]. The first stage (1 in Fig.2) samples sensitive layers based on updated sensitivity scores. The second stage (2 in Fig.2) independently evaluates the sensitivity of layers through gaussian noise. The third stage (3 in Fig.2) is to compute the rewards for each evaluated layer, and then update corresponding sensitivity scores. SAC only prunes the adapter weights of active layer (sampled by RSM), with the pruning ratio dynamically adjusted according to its updated sensitivity score. The pseudocode algorithm of **RSM** and **SAC** is presented in Appendix. A.

3.4 RL-based Sensitivity Measurement

To better search sensitive layers, we model it as a contextual multi-armed bandit problem [3, 40], which learns to select candidate higher sensitive layers for TEE protection based on continuously updated sensitivity scores $\mathcal{S}$. The measurement consists of three stages: (1) sensitive layer selection (policy actions), (2) layer sensitivity estimation (reward feedback), (3) sensitivity score update (state update). An episode of the measurement is conducted at every t_{rsm} fine-tuning steps (e.g., 10 steps). We provide convergence analysis of layer selection based fine tuning strategy in Appendix.B.

① **Sensitive Layer Selection.** Given a model $\mathcal{M}$ with adapters, `AegisGuard` regards the sensitivity scores $\mathcal{S}$ as environmental states. The scores are initialized to $\mathcal{S}_0 = [0, 0, \ldots, 0]$. The policy action space $\mathcal{A}$ constitutes a sequence of decisions a_t that select a subset of sensitive layers in each step for training. We utilize a uniform distribution combined with scores $\mathcal{S}$ to sample candidate sensitive layers. The policy action π_i is formally defined as follows:

$$\pi_i \sim U\big(0, sigmoid(s_i)\big) \tag{2}$$

where U represents a weighted random sampling in the range of $[0, \sigma(s_i)]$. For sampling, the layer selection probabilities $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ are generated from π_i. We select N_t highest scoring layers as trainable for the next fine-tuning steps. After one episode (t_{rsm} steps of fine tuning), `AegisGuard` injects noise to the weights of the selected N_t layers to quantify their sensitivity. However, due to the depth of LMs and the inaccuracy of initial layer selection, a group of adjacent layers may be identified as sensitive. Empirically, consolidating sensitive layers into contiguous sequences tends to cause unstable training. Therefore, we partition the total layers into $\{g_i\}_{i=1}^{G}$ groups and select $\frac{N_t}{G}$ layers from each group. The candidate model for next fine tuning step is:

$$l_i = \begin{cases} 1, & \text{if } i \in \theta \\ 0, & \text{otherwise} \end{cases}, \quad \theta = \{\, i \mid p_i > p_{g_\tau} \,\}, \tag{3}$$

where θ is the selection strategy that i-th layer l_i would be chosen as a trainable layer if p_i larger than threshold p_{g_τ} in corresponding group, then evaluate its sensitivity in current episode.

② **Layer Sensitivity Estimation.** The layer sensitivity is estimated through noise perturbation for each layer's adapters. In RSM episode, we inject Gaussian noise ϵ into the low-rank matrix $A \in \mathbb{R}^{d_{\text{rank}} \times d_{\text{out}}}$ of each layer independently. Then, we conduct forward computations on the noised model to evaluate sensitivity by measuring the change of output losses. This design is motivated by differential privacy (DP), indicating that sensitive parameters correlate with private data patterns [48]. Perturbing sensitive parameters leads to a larger loss turbulence. Fig. 3 shows the loss change after perturbing different layers on ViT-Base. The y-axis is the layer index and x-axis shows loss change. We can observe that the layer's influence on the model loss varies by a large margin. `AegisGuard` aims to utilize the correlation to select the sensitive layers during RSM episode.

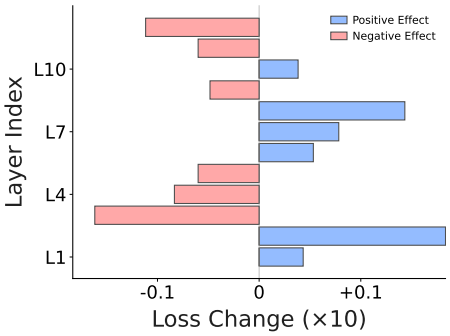

Figure 3: Loss change when perturbing different layers.

Then, the loss changes are used to guide the next action. The goal is to gather as much private information as possible in the current sensitive layers. Formally, the noise perturbation operation for the adapter can be expressed as:

$$f = xW + xB \cdot A \longrightarrow f = xW + x(A + \mathcal{N}(0, \sigma^2)) \cdot B, \;\; \epsilon \sim \mathcal{N}(0, \sigma_i^2) \tag{4}$$

$\mathcal{N}$ denotes gaussian noise, with a mean of 0 and variance σ^2. The variance is based on the standard deviation of the adapters to ensure that the noise is only functional for the adapter's weights but not impact the base model's weights, which is demonstrated by the weight distribution of base model and adapters presented in Appendix D.3. The i-th layers' change of loss is defined as:

$$\Delta\mathcal{L}_i^t = \frac{1}{B}\sum_{b=1}^{B}\left[\mathcal{L}(x_b; l_i') - \mathcal{L}(x_b; l_i)\right], \tag{5}$$

where B is the batch size of train data $\mathcal{D}_{train}$. l_i' denotes noise perturbation of i-th layer's adapter. The layer-wise reward is computed according to the loss change of injected noise in Eq. 5 as:

$$R_i^t = \text{sgn}(\Delta\mathcal{L}_i^t)\left(e^{\Delta\mathcal{L}_i^t} - \frac{1}{N}\sum_{j=1}^{N} e^{\Delta\mathcal{L}_j^t}\right). \tag{6}$$

R_i^t denotes the reward for i-th layer at step t. $sgn(\cdot)$ is the indicator function and $sgn(\Delta\mathcal{L}_i^t)$ is the sensitivity signal. Since the loss change would be small, we use exponential to amplify highly sensitive signals while subtracting mean to stabilize reward across layers. Then, if $\Delta\mathcal{L}_i^t$ is greater than zero, the layer is considered to be more sensitive to noise. Otherwise, the layer is considered to be less sensitive. The sensitive layers will be rewarded, whereas the less sensitive layers will be penalized.

③ **Sensitivity Score update.** The update of sensitivity score is based on the reward feedback of policy action a_i as shown in Eq.6. We only update layers that are trainable in current episode and do not update frozen layers. Otherwise, frozen layers may consistently increase or decrease the sensitivity scores without any training. Specifically, the update rule for the sensitivity score is formulated as follows:

$$s_i = s_i + r_i^t \cdot \mu \cdot sigmoid(s_i) \cdot (1 - sigmoid(s_i)), \quad \text{if } l_i = 1 \tag{7}$$

where μ is the learning rate of reinforcement learning, controlling the convergence of sensitivity scores. After identifying sensitive layers, they will be selectively deployed in TEEs for model protection.

3.5 Shielded Adapter Compression

Besides the frequent data transfer between TEEs and GPUs, reducing the amount of transmitted data and computation costs of TEEs can also mitigate the inference latency. Therefore, we utilize structural pruning to compress shielded adapters' feature dimensions. In the following, we briefly introduce gradient-guided weight pruning mechanism and describe how we utilize it to dynamically compress the adapter of sensitive layers combined with RSM module.

LoRA Importance Estimation. According to prior work, the importance of the LoRA matrix element $BA_{i,j}$ can be quantified by measuring the loss impact after setting $BA_{i,j} = -W_{i,j}$ [43, 24, 20]. For an input x and the ground-truth prediction y, the importance of $BA_{i,j}$ can be given as:

$$I_{i,j} = \left[\mathcal{L}(x, y, W + B \cdot A) - \mathcal{L}(x, y, W + B \cdot A | BA_{i,j} = -W_{i,j})\right]^2, \tag{8}$$

where $\mathcal{L}$ denotes the loss of model. We utilize the first-order Taylor expansion to approximate the importance $\hat{I}_{i,j}$ in gradient-based manner, in order to minimize the expensive computations of weight matrix. The importance of parameters in LoRA matrices is formulated as:

$$\hat{I}_{i,j} = \left[\left(\frac{\partial\mathcal{L}}{\partial B_{i,:}}A_{:,j} + B_{i,:}\frac{\partial\mathcal{L}}{A_{:,j}} - \frac{\partial\mathcal{L}}{\partial B_{i,:}}\frac{\partial\mathcal{L}}{\partial A_{:,j}}\right)(W_{i,j} + (BA)_{i,j})\right]^2 \tag{9}$$

Shielded Adapter pruning. Since only selected layers are learnable in an RSM episode, gradients are unavailable for other frozen adapter parameters, making it challenging to directly apply Eq. 9 in dynamic model fine-tuning. Furthermore, pruning during the initial training phase may lead to bias and performance degradation, as all layers exhibit relatively low and random sensitivity to private data. Estimating weight importance and performing pruning with limited training data also increase the inaccuracies. Therefore, we propose a dynamic pruning strategy that is compatible with our

sensitivity measurement module, ensuring both structural pruning compatibility and training stability. Formally, we design a dynamic pruning ratio for adapters in i-th layer as:

$$\text{ratio}_i(t) = \frac{R_{\text{total}} - R_{\text{t}}}{(T - t + \delta) \cdot sigmoid(s_i)} \cdot \left(1 - \alpha + \alpha \cdot \frac{t}{T}\right), \tag{10}$$

where T is the total number of fine-tuning steps, t is the current fine tuning step, R_{total} is overall prune ratio, and R_{t} is cumulative prune ratio. α is a hyper-parameter to control the pruning rate. δ used to prevent division by zero. In this way, the layer-wise pruning ratio will progressively increase in a linear manner. This design aims to prune relatively more weights at the later training phase.

In addition, the prune is executed at head-level. The head importance H_i is computed by averaging importance of weight elements in this head via moving average $H_i^t = \beta H_i^t + (1-\beta)H_i^{t-1}$ to mitigate bias caused by a single batch of data. Afterwards, weight mask is computed by ranking the head importance that partial lowest importance head would be set to 0:

$$\text{mask} = \begin{cases} 0, & \text{if } H_i^t \in \text{rank}\left(H^t; \lfloor \text{ratio}_i(t) \cdot N_{head} \rfloor\right) \\ 1, & \text{otherwise.} \end{cases} \tag{11}$$

where N_{head} is the total number of heads. Afterwards, in the following forward computation, the masked adapter would be calculated as :

$$f = xW + xBA \odot \text{mask}, \tag{12}$$

where $\odot$ denotes the Hardamard product. The module reduces intermediate feature maps throughput for data transfer and mitigates computation costs by compressing the parameters in TEEs.

4 Experiments

To demonstrate the effectiveness of `AegisGuard`, in this section, we study four research questions:

- RQ1: How is the on-device efficiency of `AegisGuard` compared with prior work?
- RQ2: How is the defense effectiveness against model stealing attacks?
- RQ3: Does `AegisGuard` introduce much accuracy degradation?
- RQ4: How does the sensitivity vary across different layers?

4.1 Setup

Models and Datasets. We select large models from different domains and sizes, including generative models (OPT-2.7B [44], LLaMA-7B [35]) and vision transformer (ViT-Base, ViT-Large-14[42]). We employ LoRA [15] for parameter-efficient fine-tuning. For the dataset, we use CommonSenseQA[16] to fine-tune generative models. We evaluate the model performance using six popular question-answering benchmarks: ARC-Challenge and ARC-Easy[7], HellaSwag [41], OBQA [21], PIQA [5], and WinoGrande [29]. For ViT models, we use six diverse datasets: CIFAR10, CIFAR100, [18], UTKFace [46], MNIST [8], GTSRB [32], and SUN397 [38]. Following prior works [49, 4], we use CIFAR and UTKFace to evaluate the defense effectiveness against MS attack for fair comparison. Due to space constraints, we provide more details (such as attack pipeline, evaluation metrics, and hyperparameters) in Appendix.C.

Implementation. We implement all code in PyTorch 2.5.1. The fine-tuning is conducted on a server with one NVIDIA A6000 GPUs. For inference, we follow existing work [47] to build a prototype framework on a PC with an Intel SGX enclave (SDK 2.6, GCC 7.5), and an NVIDIA RTX4090D 24GB GPU to evaluate the on-device LM performance.

Baselines. We select four baselines. Two are state-of-the-art TSDP solutions, two are bound baselines. The SOTA TSDP solutions are TEESlice [47] and Phantom [4], which are published at top-tier conferences. The two bound baselines are No-Shield and Shield-LoRA, following [47]. No-Shield deploys the entire model to the untrusted GPU, which has the highest efficiency (all on GPU) but the lowest security (no TEE). Shield-LoRA refers to deploying all adapters in the TEE,

rather than the GPU. This baseline offers the highest security (all in TEE) but the lowest efficiency (no GPU). Besides, No-Shield and Shield-LoRA also have the highest accuracy because they directly apply LoRA fine-tuning to the base model and do not compress the model weights.

4.2 Inference Efficiency

Table 1: Inference overhead on a real device. We report inference latency of AegisGuard compared with two baselines and Shield-LoRA setting (slowest). indicate lowest inference overhead

	Inference Costs (ViT-Base)						Inference Costs (LLaMA-7B)				
Time (ms)	Shield-LoRA	Phantom	TEESlice	AegisGuard	No-Shield	**Time (s)**	Shield-LoRA	Phantom	TEESlice	AegisGuard	No-Shield
GPU	6.88	6.03	5.73	5.94	6.45	GPU	0.19	0.32	0.21	0.22	0.96
TEE	12.24	4.25	8.93	4.31	–	TEE	6.28	1.15	2.54	1.06	–
Transfer	13.65	13.67	11.02	4.89	–	Trans	14.91	11.96	9.03	5.14	–
End-to-End	32.77 (1.00×)	23.95	25.68	15.14 (2.16×)	6.45	End-to-End	21.38 (1.00×)	13.43	11.78	6.42 (3.33×)	0.96

AegisGuard shows the best efficiency in inference latency. It presents the real-device efficiency of ViT-Base and LLaMA-7B. Table 1 shows the latency of GPU computation, TEE computation, data transfer, and the end-to-end latency. The end-to-end latencies of AegisGuard are 15.14ms and 6.42s for ViT-Base and LLaMA-7B, respectively. AegisGuard outperforms TEESlice and Phantom by an average of 1.60×. Compared to Shield-LoRA, AegisGuard accelerates the latency by reducing the number of adapters shielded in TEEs for minimizing communication and computations costs. Besides, the advantage of AegisGuard is larger when the model size increases. Compared to Shield-LoRA, AegisGuard accelerates by 3.33× on LLaMA-7B, higher than ViT-Base (2.16×). We also measure the memory cost of AegisGuard. AegisGuard has average 4.00× less parameters (54.96MB for LLaMA 7B, 0.88MB for ViT-Base) in TEEs than baselines. The memory results are in Appendix D.2.

The improvement of AegisGuard comes from TEE computation and data transfer. All methods have similar computation costs in GPUs, about 6 milliseconds for ViT-Base and 0.2 to 0.3 seconds for LLaMA-7B. For ViT-Base, AegisGuard selectively deploys only four sensitive adapters in TEE, whereas TEESlice places nine adapters. Thus, TEESlice exhibits more computation and data transfer costs. Phantom needs to insert three redundant adapters into models to obfuscate the real execution path. The MUX operation of Phantom is in TEEs to choose the correct path in each layer based on the user's key. Thus, Phantom has a larger data transfer costs than AegisGuard.

4.3 Defense Effectiveness

Table 2: Performance of MS attack accuracy. A lower attack accuracy (surrogate model) represents a higher defense effectiveness. indicates the best defensive capability measured by the surrogate model's accuracy.

	Model Stealing Attack (ViT-Base)						Model Stealing Attack (LLaMA-7B)				
Dataset	No-Shield	Phantom	TEESlice	AegisGuard	Shield-LoRA	**Dataset**	No-Shield	Phantom	TEESlice	AegisGuard	Shield-LoRA
C10	97.97	56.09	41.89	40.70	42.73	ARC-E	65.06	57.43	39.87	40.12	39.60
C100	86.07	30.79	23.18	22.67	19.25	PIQA	77.85	69.37	60.98	60.45	61.31
UTKFace	75.01	27.57	18.86	19.10	20.80	WinoGrande	68.75	60.48	50.73	51.48	50.27
Average	86.35	38.15	27.97	27.49	27.59	Average	73.21	62.42	50.52	50.68	50.39

AegisGuard shows the comparable defensive effectiveness against Black-box setting after deployment. Table 2 shows the performance of the MS attack on ViT-Base and LLaMA-7B. The accuracy of the stolen surrogate model evaluates the defense performance. A lower accuracy represents a better MS attack defense. The last row shows the average accuracy and the value in parentheses is the performance gain over the black-box baseline (lowest attack accuracy). AegisGuard can effectively reduce the MS attack accuracy. In all cases, the attack accuracy of AegisGuard are similar to the black-box baseline (1.00× higher than black-box). AegisGuard has a better protection than Phantom (1.38× lower) and a similar performance to TEESlice, which also provides black-box-level protection. The comparatively high MS attack accuracy of Phantom is because its adapter parameters are exposed in an untrusted GPU and difficult to find the optimal obfuscation forward path for LMs. Thus, the adversary can recover a certain amount of model functionality. TEESlice offers black-box-level protection because it shields all weights that are updated by private data in TEE. However, as shown in Table 1, TEESlice has a (1.5-2.0)× higher latency than AegisGuard because AegisGuard only shields partial highly sensitive layers in TEEs. We also present randomly selecting shielded layers experiments, which requires approximate 2× layers than AegisGuard in Appendix.D.4.

4.4 Accuracy

Table 3: Accuracy of AegisGuard on nlp and cv tasks. indicates AegisGuard has a similar downstream task accuracy (1.00×) with Shield-LoRA fine-tuning. RL_{sens} denotes RL-based sensitivity measurement module. SAC means Shielded Adapter Compression.

Model	Variant	CIFAR-10	CIFAR-100	UTKFace	MNIST	GTSRB	Sun397	Avg.
ViT-Base	Shield-LoRA/Phantom	97.47	85.96	75.03	98.90	93.35	53.37	84.01
	TEESlice	95.73	82.21	73.87	96.28	90.36	54.28	82.12
	Shield-LoRA+RL$_{sens}$	97.35	85.80	74.92	98.72	93.21	53.20	83.87
	Shield-LoRA+SAC(30%)	96.12	84.77	74.61	97.41	92.89	52.48	83.05
	AegisGuard	97.97	86.07	75.01	98.25	93.20	53.76	84.04
Model	**Variant**	ARC-C	ARC-E	HellaSwag	OBQA	PIQA	WinoGrande	Avg.
LLaMA-7B	Shield-LoRA/Phantom	41.43	65.06	72.39	45.67	77.92	68.61	61.85
	TEESlice	39.96	63.98	71.82	43.18	75.27	66.28	60.08
	Shield-LoRA+RL$_{sens}$	41.32	65.18	72.25	45.79	78.05	68.48	61.85
	Shield-LoRA+SAC(30%)	39.55	62.95	69.46	42.56	72.80	65.72	58.84
	AegisGuard	41.89	65.40	73.04	45.40	77.85	68.75	62.06

AegisGuard causes negligible accuracy fluctuation in downstream tasks. One concern of AegisGuard is that it may introduce too much accuracy drop than the standard fine-tuning with additional modules. Table 3 shows the accuracy of AegisGuard compared to standard fine-tuning over six nlp and cv tasks, as well as ablation studies. For all cases, AegisGuard is comparable to standard fine-tuning that the average accuracy fluctuation of LLaMA-7B and ViT-Base is approximately 0.12%.

According to the ablation study, **each module in AegisGuard has no obvious negative impact on performance.** It is because the RL-based sensitivity measurement and fine-tuning steps are performed iteratively. And this module only designed to affect sampling behavior. The adapter compression has a slight impact on performance, when prune 30% weights in Shield-LoRA. But there has less impact on AegisGuard, cause only partial shielded adapters are pruned and its slight negative impact can also be elimiated by next fine tuning epochs. We also conduct experiments in other large models (e.g., OPT-2.7B, ViT-Large), and the results are provided in Appendix D.1.

4.5 Sensitivity Visualization of Layers

Fig. 4 shows the sensitivity comparison across different layers of two models: ViT-Base on CIFAR100 (upper figure) and LLaMA-7B on CommenSense (lower figure). ViT-Base shows higher sensitivity in deeper layers and middle layers. For LLaMA-7B, the sensitive layers are distributed across the model from shallow layers to deep layers. For example, layer 2, 4, and 28 have the highest sensitivity score. Therefore, a fixed scheme that only shields shallow or deep layers is insufficient for comprehensive model protection. This observation demonstrate the necessary of a dynamic layer selection scheme to identify and protect the sensitive layers.

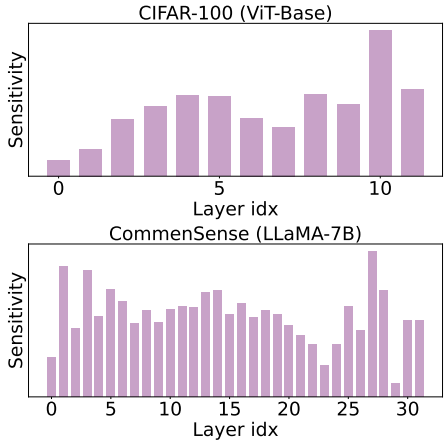

Figure 4: Layer-wise Sensitivity Scores

5 Conclusion

In this paper, we propose a noval fine tuning framework, namely AegisGuard. It constructs a hybrid LoRA-LM architecture for efficient and security on-device LM inference. We introduce a RL-based Sensitivity Measurement, to proactively search sensitive layers and fine tune more steps on them. We propose Shielded Adapter Compression that reduce the intermediate feature maps and enhance the efficiency of computation in TEEs. Comprehensive experiments across various large models and tasks showcases high-efficiency, high-security protection of on-device LM inference by AegisGuard.

6 Acknowledge.

Ziqi Zhang and Jianbo Gao are corresponding authors. This research is supported by Primary Research & Development Plan of Jiangsu Province (BE2023025, BE2023025-5), National Natural Science Foundation of China (62202011, 62172010). This work is also supported by Ant Group. The Singapore authors are supported by the NTU startup grant and the RIE2025 Industry Alignment Fund-Industry Collaboration Projects (IAF-ICP) (Award I2301E0026) as well as its Japan-Singapore Joint Call: JST-A*STAR 2024 (Project ID: R24I6IR139), administered by A*STAR. The Japan author is supported by JSPS KAKENHI JP23K24851, JST PRESTO JPMJPR23P5, JST CREST JPMJCR21M2, JST NEXUS JPMJNX25C4. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of Sponsors.

References

[1] Tiago Alves. Trustzone: Integrated hardware and software security. *Information Quarterly*, 3:18–24, 2004.

[2] Apple. New MacBook Pro features M4 family of chips and Apple Intelligence. `https://www.apple.com/sg/newsroom/2024/10/new-macbook-pro-features-m4-family-of-chips-and-apple-intelligence/`, 2024. Accessed: 2025-04-01.

[3] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[4] Juyang Bai, Md Hafizul Islam Chowdhuryy, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. Phantom: Privacy-preserving deep neural network model obfuscation in heterogeneous tee and gpu system. In *Proceedings of the USENIX Security Symposium '25*. Zenodo, 2025.

[5] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[6] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[7] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[8] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[9] Victor Costan and Srinivas Devadas. Intel sgx explained. *Cryptology ePrint Archive*, 2016.

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[11] Minghao Guo, Zhao Zhong, Wei Wu, Dahua Lin, and Junjie Yan. Irlas: Inverse reinforcement learning for architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9021–9029, 2019.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[14] Jiahui Hou, Huiqi Liu, Yunxin Liu, Yu Wang, Peng-Jun Wan, and Xiang-Yang Li. Model protection: Real-time privacy-preserving inference service for model privacy at the edge. *IEEE Transactions on Dependable and Secure Computing*, 19(6):4270–4284, 2021.

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[16] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276, 2023.

[17] Md Shihabul Islam, Mahmoud Zamani, Chung Hwan Kim, Latifur Khan, and Kevin W Hamlen. Confidential execution of deep learning inference at the untrusted edge with arm trustzone. In *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy*, pages 153–164, 2023.

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[19] Xiuqing Lv, Peng Zhang, Sunzhu Li, Guobing Gan, and Yueheng Sun. Lightformer: Lightweight transformer using svd-based weight transfer and parameter sharing. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10323–10335, 2023.

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

[21] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

[22] MLC LLM. Mlc llm: Universal llm deployment engine with ml compilation. `https://llm.mlc.ai/#android`, 2024. Accessed: 2025-04-01.

[23] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. Darknetz: towards model privacy at the edge using trusted execution environments. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 161–174, 2020.

[24] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2019.

[25] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[26] Daryna Oliynyk, Rudolf Mayer, and Andreas Rauber. I know what you trained last summer: A survey on stealing machine learning models and defences. *ACM Computing Surveys*, 55(14s):1–41, 2023.

[27] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4954–4963, 2019.

[28] Konstantinos I Roumeliotis and Nikolaos D Tselikas. Chatgpt and open-ai models: A preliminary review. *Future Internet*, 15(6):192, 2023.

[29] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[30] Ashish Kumar Shakya, Gopinatha Pillai, and Sohom Chakrabarty. Reinforcement learning algorithms: A brief survey. *Expert Systems with Applications*, 231:120495, 2023.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

[33] Yu Sun, Gaojian Xiong, Jianhua Liu, Zheng Liu, and Jian Cui. Tsqp: Safeguarding real-time inference for quantization neural networks on edge devices. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1–1. IEEE Computer Society, 2024.

[34] Zhichuang Sun, Ruimin Sun, Changming Liu, Amrita Roy Chowdhury, Long Lu, and Somesh Jha. Shadownet: A secure and efficient on-device model inference system for convolutional neural networks. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1596–1612. IEEE, 2023.

[35] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[36] Florian Tramer and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *International Conference on Learning Representations*.

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

[38] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[39] Xiaomi. Mace: Mobile ai compute engine. `https://github.com/XiaoMi/mace`, 2024. Accessed: 2025-04-01.

[40] Kai Yao, Penglei Gao, Lichun Li, Yuan Zhao, Xiaofeng Wang, Wei Wang, and Jianke Zhu. 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*, pages 1977–1992, 2024.

[41] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[42] Wenxuan Zeng, Meng Li, Wenjie Xiong, Tong Tong, Wen-jie Lu, Jin Tan, Runsheng Wang, and Ru Huang. Mpcvit: Searching for accurate and efficient mpc-friendly vision transformer with heterogeneous attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5052–5063, 2023.

[43] Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*, 2023.

[44] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[45] Zheng Zhang, Na Wang, Ziqi Zhang, Yao Zhang, Tianyi Zhang, Jianwei Liu, and Ye Wu. Groupcover: A secure, efficient and scalable inference framework for on-device model protection based on tees. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[46] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818, 2017.

[47] Ziqi Zhang, Chen Gong, Yifeng Cai, Yuanyuan Yuan, Bingyan Liu, Ding Li, Yao Guo, and Xiangqun Chen. No privacy left outside: On the (in-) security of tee-shielded dnn partition for on-device ml. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3327–3345. IEEE, 2024.

[48] Maxim Zhelnin, Viktor Moskvoretskii, Egor Shvetsov, Egor Venediktov, Mariya Krylova, Aleksandr Zuev, and Evgeny Burnaev. Gift-sw: Gaussian noise injected fine-tuning of salient weights for llms. *arXiv preprint arXiv:2408.15300*, 2024.

[49] Tong Zhou, Yukui Luo, Shaolei Ren, and Xiaolin Xu. Nnsplitter: an active defense solution for dnn model via automated weight obfuscation. In *International Conference on Machine Learning*, pages 42614–42624. PMLR, 2023.

[50] Itamar Zimerman, Moran Baruch, Nir Drucker, Gilad Ezov, Omri Soceanu, and Lior Wolf. Converting transformers to polynomial form for secure inference over homomorphic encryption. In *Forty-first International Conference on Machine Learning*, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes] .

   Justification: Yes, the main claims made in the abstract and introduction include our primary contributions and the assumptions underlying our scenarios.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes] .

   Justification: Yes, we discuss the scenarios in which our approach is applicable and the limitations we assumed in Appendix.F.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes] .

Justification: Yes, we provide the convergence analysis of `AegisGuard` for its randomized selection fine tuning strategy in Section. 1 with detailed analyses provided in Appendix B.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes] .

Justification: Yes, we describe our method concept in detail, providing overview diagrams (Figure 2), pseudocode in Algorithm A, specific implementations in experiments (Appendix C).

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes] .

   Justification: Yes, we provide the implementation details in appendix, and artifacts are released in code address presented in abstract.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] .

   Justification: Yes, we provide all the training and test details in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes] .

   Justification: Yes, we provide the standard error of mean for LLMs and ViT models. All results are presented in Appendix.D.1

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes] .

   Justification: Yes, we provide the computer resources in Section 4.1. We also provide communication and memory costs in Appendix.D.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes] .

   Justification: Yes, our research conforms to the NeurIPS Code of Ethics in every respect. We have thoroughly reviewed the guidelines and ensured that our work adheres to the ethical standards set forth.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes] .

    Justification: Yes, our work secures LMs inference by using TEEs to protect core adapter layers while reducing TEE-GPU communication. This enables privacy-preserving, tamper-resistant AI services on-device, benefiting sensitive domains like healthcare and finance.

However, improved model confidentiality may hinder transparency, auditing, or accountability, and could be misused to conceal harmful models. TEE encapsulation also raises trust concerns in critical applications. Future integration with auditing and attestation mechanisms is important. Overall, our method enhances security and efficiency, we provide the attack results in Appendix.D.4.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA] .

Justification: The research described in our paper does not involve data or models that have a high risk for misuse, and thus does not require specific safeguards.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes] .

Justification: Yes, all assets used in our paper are properly credited, and the license and terms of use are explicitly mentioned. Detailed citations, including version numbers and URLs, are provided in the main content Section 4.1.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA] .

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA] .

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA] .

Justification: This work does not involve human subjects or crowdsourced data collection. It focuses on the design and evaluation of a system for secure and efficient LLM inference using Trusted Execution Environments.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA] .

Justification: LLMs were only used for writing and language editing purposes, which do not affect the core methodology, scientific rigor, or originality of the research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Pseudocode algorithm of AegisGuard

Algorithm 1 AegisGuard: RL-based Sensitivity Measurement & Shielded Adapter Compression

Initialize W with LoRA adapters $\Delta W = B \cdot A$
Set $\mathcal{S}_0 = \mathbf{0}$ for n layers
Set T as total steps, t_{rsm} as episode steps, t_{prn} as prune interval steps
Initialize π, $\mathcal{A} = [a_0]$
for $t = 1 \rightarrow T$ **do**
    Update $\Delta B_t \cdot A_t \leftarrow \Delta B_{t-1} \cdot A_{t-1}$
    if t mod t_{rsm}== 0 **then**
        for i-th layer in all layers **do**
            Noise injection: $f_i = xW_i + x(A_i + \epsilon) \cdot B_i$
            Layer Evaluation: ΔL_i^t // loss change after noise perturbation
            Compute Rewards: R_i^t
            Update Sensitivity Scores: $s_i = s_i + R_i \cdot \mu \cdot \text{sigmoid}(s_i) \cdot (1 - \text{sigmoid}(s_i))$
        end for
        Sampling a_t: Select a subset of sensitive layers l_i via π
        Update: $\mathcal{A} \leftarrow a_t$
        Partial selected sensitive layers l_i fine tuning.
    end if
    if t mod t_{prn} == 0 **then**
        for layer l_i **do**
            Calculate prune ratio: $\text{ratio}_i(t)$
            Apply in matrix B, A: mask == $\vec{0} \leftarrow \text{Rank}(\vec{H}, \lfloor \text{ratio}_i(t) \cdot N_{head} \rfloor)$
        end for
    end if
end for
Return W with compressed ΔW and $\mathcal{S}(T)$

Here we present the pseudocode of AegisGuard, for better understanding of entire framework of Section 3 and Fig. 2.

B Convergence Analysis

In order to theoretical prove the effectiveness of AegisGuard (especially the RSM module), we provide the convergence analysis of a randomized layer gradient descent algorithm, with reference to [25], extending it to a setting where at iteration k, the total sampling mass $C_k = \sum_{i=1}^{L} p_i^k$ may vary, but is lower-bounded by $C_k \geq C > 0$. Under block-wise Lipschitz smoothness and a fixed step size $\eta \leq \frac{1}{\max_i L_i}$, we prove almost-sure convergence to stationary points with an $O(1/k)$ rate, and linear convergence when the objective function f satisfies the Polyak–Łojasiewicz (PL) condition.

Let $\theta \in \mathbb{R}^n$ be partitioned into L disjoint blocks (i.e., layers), such that $\theta = (\theta^{(1)}, \ldots, \theta^{(L)})$. At iteration k, each block i is selected with probability $p_i^k \geq 0$, such that $\sum_{i=1}^{L} p_i^k = C_k$. Let S_k denote the random set of selected indices. For each $i \in S_k$, the update rule is given by:

$$\theta_{k+1}^{(i)} = \theta_k^{(i)} - \eta \nabla_i f(\theta_k),$$

while the other blocks remain unchanged.

Assumption 1: *Block-wise smoothness.* There exist constants $L_i > 0$ such that

$$\|\nabla_i f(\theta + U_i h) - \nabla_i f(\theta)\| \leq L_i \|h\| \quad \text{for all} \quad \theta, h.$$

Assumption 2: *Step size.* A fixed step size η satisfies $0 < \eta \leq \frac{1}{\max_i L_i}$.

Assumption 3: *Uniform mass bound.* There exists a constant $C > 0$ such that $C_k \geq C$ for all k.

Assumption 4: *Polyak–Łojasiewicz condition.* The objective function f satisfies the Polyak–Łojasiewicz condition:

$$\frac{1}{2}\|\nabla f(\theta)\|^2 \geq \mu(f(\theta) - f^*),$$

where f^* is the minimum of f.

Gradient Update and Function Value Bound The update of block i by $\Delta^{(i)} = -\eta \nabla_i f(\theta)$ results in the following bound for the objective function value:

$$f(\theta + U_i \Delta^{(i)}) \leq f(\theta) - \eta \|\nabla_i f(\theta)\|^2 + \frac{1}{2} L_i \eta^2 \|\nabla_i f(\theta)\|^2 \leq f(\theta) - \frac{\eta}{2} \|\nabla_i f(\theta)\|^2.$$

Expected Function Value Change Conditioned on θ_k, the expectation of the function value at the next step is:

$$\begin{aligned}
\mathbb{E}[f(\theta_{k+1}) \mid \theta_k] &= (1 - C_k) f(\theta_k) + \sum_{i=1}^{L} p_i^k f(\theta_k + U_i \Delta^{(i)}) \\
&\leq f(\theta_k) - \frac{\eta}{2} \sum_{i=1}^{L} p_i^k \|\nabla_i f(\theta_k)\|^2 \\
&\leq f(\theta_k) - \frac{\eta C}{2} \|\nabla f(\theta_k)\|^2.
\end{aligned}$$

Rearranging and taking the full expectation:

$$\frac{\eta C}{2} \mathbb{E}[\|\nabla f(\theta_k)\|^2] \leq \mathbb{E}[f(\theta_k)] - \mathbb{E}[f(\theta_{k+1})].$$

Summation Over Iterations Summing from $k = 0$ to $N - 1$ gives:

$$\frac{\eta C}{2} \sum_{k=0}^{N-1} \mathbb{E}[\|\nabla f(\theta_k)\|^2] \leq f(\theta_0) - f(\theta_N) \leq f(\theta_0) - f^*,$$

so:

$$\sum_{k=0}^{\infty} \mathbb{E}[\|\nabla f(\theta_k)\|^2] < \infty,$$

which implies that $\mathbb{E}[\|\nabla f(\theta_k)\|^2] \to 0$ as $k \to \infty$. A standard supermartingale argument then implies that $\|\nabla f(\theta_k)\| \to 0$ almost surely.

Convergence Rate Define:

$$\delta_N = \min_{0 \leq k < N} \mathbb{E}[\|\nabla f(\theta_k)\|^2].$$

From the finite sum above, we have:

$$N \delta_N \leq \sum_{k=0}^{N-1} \mathbb{E}[\|\nabla f(\theta_k)\|^2] \leq \frac{2(f(\theta_0) - f^*)}{\eta C},$$

which implies that $\delta_N = O(1/N)$.

Linear Convergence Under the Polyak–Łojasiewicz Condition Under the Polyak–Łojasiewicz condition, we have:

$$\begin{aligned}\mathbb{E}[f(\theta_{k+1}) - f^*] &\leq \mathbb{E}[f(\theta_k) - f^*] - \frac{\eta C}{2}\mathbb{E}[\|\nabla f(\theta_k)\|^2] \\ &\leq (1 - \mu\eta C)\mathbb{E}[f(\theta_k) - f^*],\end{aligned}$$

which implies global linear convergence to the minimum f^*.

In conclusion, the randomized layer selection method for gradient descent converges almost surely to stationary points with an $O(1/k)$ rate, and linearly under the Polyak–Łojasiewicz condition.

C Experiment Configuration

In this part, we provide the hyperparameter used in our experiments, including Reinforcement Learing in RSM, and fine tuning. We provide the model stealing attack pipeline and setup. All artifacts will be released through a public GitHub repository upon paper acceptance, ensuring full transparency and replicability of our research.

C.1 Parameter Efficient Fine Tuning

During adapter fine-tuning for general tasks, we primarily apply LoRA adapters to the multi-head attention and feedforward layers, specifically to the query, key, value, and dense components. The exact placement of LoRA modules depends on the model's performance on task-specific datasets. Our goal is to minimize the number of trainable parameters while maintaining competitive accuracy. For instance, in relatively simple tasks such as CIFAR-10, LoRA is only applied to the query and value projections. In contrast, for more challenging NLP tasks, we insert LoRA adapters into the query, key, value, and dense components of each layer. The rank hyperparameter of LoRA is chosen from 16, 32, 64, depending on the size of the model and the dataset.

For dynamic pruning, we set the total pruning ratio with 20% and 50% to balance performance and efficiency. The pruning frequency is set to 20 steps, and the warm-up ratio is set to 0.1.

We use a micro-batch size is 8,32 and gradient-accumulation-steps as 8,1 for LLaMA and ViT, respectively. The training epochs of nlp tasks are setting in range of [2,5]. For vision tasks, the epoch is setting from 6 to 20 depends on the task complexity. We employ the AdamW as optimizer, experimenting with a range of learning rates: [2e-5,4e-5,2e-4,3e-4].

C.2 Reinforcement Learning in RSM

Several hyperparameters are involved in this phase. The first is gaussian noise, which is set within the range of 0.01 to 0.1 for vision models. For LLaMA and OPT models, however, a smaller noise value is required, as a larger gaussian noise may lead to loss vanishing or explosion due to the depth of layers. The sampling group size is fixed at 3. The reinforcement learning (RL) steps are scheduled every 20 fine-tuning steps, meaning that the RL procedure is executed once after every 20 steps of standard fine-tuning. The parameter N_t is set to 40% of the total number of layers. The learning rate for the reinforcement learning process is selected in the range of 3 to 8, depending on the convergence behavior of the Sensitivity score distributions.

C.3 Model Stealing Attack

Model Stealing Attack aims to create a "knockoff" or surrogate model that replicates the functionality of a victim model by adversaries, using only black-box access to query the victim model. The attack process typically involves initializing a surrogate model using a publicly available pre-trained model, sampling images/text from an out-of-distribution dataset to query the victim model and collect predictions, and then retraining the surrogate model on these collected prediction pairs to mimic the victim models' behavior. The effectiveness of model stealing is evaluated by measuring the surrogate models' accuracy on the victim model's private test dataset, indicating how well the surrogate model performs after being trained on the adversary's constructed dataset.

To perform fair comparison, we employ standard query-based stealing techniques where the attacker trains a model from a set of collected data labeled by the partially-shielded victim model. Query-based MS attack has been widely adopted in literature[47, 4, 27]. For surrogate model initialization, we assume that adversaries have knowledge of the LM architectures, since many public LMs are open-sourced. Therefore, adversaries can compare differences between their local models and publicly available models to identify the trained parameters.

Model Stealing Attack Setup. Model stealing attacks aim to extract a surrogate model that replicates the functionality of the victim model. To simulate this, we adopt the same architecture as the victim model and fine-tune it using a limited query-based training dataset, which consists of approximately 1% of the full dataset. This setting has been shown to be realistic and practical in real-world scenarios [47]. Moreover, `AegisGuard` follow the same fine-tuning configurations as baselines for a fair comparison. During this process, we shield approximately 40% of the total adapters based on sensitivity scores, and reinitialize these adapters with random weights before retraining. The learning rate and number of training epochs are set to be consistent with those used in the initial fine-tuning phase.

D Supplementary Experimental Results

D.1 Ablation Study

Table 4: **NLP Model Performance Across Tasks (Accuracy %)**

Model	Variant	ARC-C	ARC-E	HellaSwag	OBQA	PIQA	WinoGrande	Average
OPT-2.7B	Shield-LoRA/Phantom	37.99	51.94	51.95	32.80	72.03	61.16	51.65
	TEESlice	36.12	49.57	50.31	30.85	71.49	60.06	49.90
	Shield-LoRA+RL$_{sens}$	37.95	52.41	51.75	33.92	71.65	59.70	51.56
	Shield-LoRA+SAC(30%)	34.15	48.19	48.50	31.72	67.90	57.95	48.07
	`AegisGuard`	38.07	52.27	51.62	33.80	71.82	59.82	52.40
Llama-7B	Shield-LoRA/Phantom	41.43	65.06	72.39	45.67	77.92	68.61	61.85
	TEESlice	39.96	63.98	71.82	43.18	75.27	66.28	60.08
	Shield-LoRA+RL$_{sens}$	41.32	65.18	72.25	45.79	78.05	68.48	61.85
	Shield-LoRA+SAC(30%)	39.55	62.95	69.46	42.56	72.80	65.72	58.84
	`AegisGuard`	41.89	65.40	73.04	45.40	77.85	68.75	62.06

Table 5: **Vision Model Performance Across Tasks (Accuracy %)**

Model	Variant	CIFAR-10	CIFAR-100	UTKFace	MNIST	GTSRB	Sun397	Average
ViT-Base	Shield-LoRA/Phantom	97.47	85.96	75.03	98.90	93.35	53.37	84.01
	TEESlice	95.73	82.21	73.87	96.28	90.36	54.28	82.12
	Shield-LoRA+RL$_{sens}$	97.35	85.80	74.92	98.72	93.21	53.20	83.87
	Shield-LoRA+SAC(30%)	96.12	84.77	74.61	97.41	92.89	52.48	83.05
	`AegisGuard`	97.97	86.07	75.01	98.25	93.20	53.76	84.04
ViT-Large	Shield-LoRA/Phantom	98.58	91.97	88.38	98.46	94.79	63.81	89.33
	TEESlice	95.76	89.37	85.46	95.58	91.84	62.05	86.68
	Shield-LoRA+RL$_{sens}$	98.41	91.78	88.25	98.19	94.62	63.55	89.13
	Shield-LoRA+SAC(30%)	97.11	90.62	86.53	97.12	92.95	62.54	87.81
	`AegisGuard`	99.01	91.05	88.25	98.78	95.35	63.97	89.40

In this section, we compare the performance of `AegisGuard`, Phantom, TEESlice, and Shield-LoRA methods. The results are presented in Table 4 and Table 5. Phantom employs a post-selection strategy; thus, its model performance are identical to the Shield-LoRA method, which uses standard fine-tuned LoRA-LMs without additional modifications. TEESlice exhibits slightly inferior performance compared to other methods due to the removal of several adapters aimed at minimizing the number of adapters shielded within TEEs. Although TEESlice can be theoretically applied to LM scenarios, its practicality is limited, as the adapter removal is passive and only a small number of layers can

be removed while satisfying accuracy loss constraints. Ablation studies have demonstrate that each module has no obvious negative impact on overall model performance, except for the pruning ratio. Besides, due to the size of Table, we shows the standard error of NLP models in Table.6.

Table 6: Model Performance Across Tasks (Accuracy %) with Standard Error of the Mean

Model		ARC-C	ARC-E	HellaSwag	OBQA	PIQA	WinoGrande	Average
OPT-2.7B	AegisGuard	38.07 ± 1.35	52.27 ± 1.02	51.62 ± 0.48	33.80 ± 2.13	71.82 ± 1.01	59.82 ± 1.37	52.40 ± 1.22
Llama-7B	AegisGuard	41.89 ± 1.41	65.40 ± 1.01	73.04 ± 0.45	45.40 ± 2.00	77.85 ± 0.94	68.75 ± 1.29	62.07 ± 1.18
Model		CIFAR-10	CIFAR-100	UTKFace	MNIST	GTSRB	Sun397	Average
ViT-Base	AegisGuard	97.97 ± 0.20	86.07 ± 0.55	75.01 ± 1.03	98.25 ± 0.52	93.20 ± 0.45	53.76 ± 1.21	84.04 ± 1.01
ViT-Large	AegisGuard	99.01 ± 0.20	91.05 ± 0.56	88.25 ± 1.02	98.78 ± 0.41	95.35 ± 0.34	63.97 ± 1.36	89.40 ± 1.02

D.2 Utility Analysis

Table 7: Memory Consumption Comparison Across Models and Methods. **Bold** denotes the minimal memory consumption

Models	**Methods**	#Public Params	#Trainable Params	#Params in TEE	TEE Memory
LLaMA 7B (r=32) (ratio=30%)	Shield-LoRA	6.79B	56.10M	56.10M	224.4 MB
	Phantom	6.79B	72.93M	16.83M	67.32 MB
	TEESlice	6.79B	44.88M	44.88M	179.52 MB
	AegisGuard	6.79B	56.10M	**13.74M**	**54.96 MB (4.08×)**
OPT 2.7B (r=32) (ratio=30%)	Shield-LoRA	2.70B	47.19M	47.19M	188.76 MB
	Phantom	2.70B	58.99M	11.80M	47.20 MB
	TEESlice	2.70B	37.75M	37.75M	151.00 MB
	AegisGuard	2.70B	47.19M	**11.56M**	**46.24 MB (4.08×)**
ViT-Base (r=16) (ratio=30%)	Shield-LoRA	86.71M	0.88M	0.88M	3.52 MB
	Phantom	86.71M	1.32M	0.44M	1.76 MB
	TEESlice	86.71M	0.62M	0.62M	2.48 MB
	AegisGuard	86.71M	0.88M	**0.22M**	**0.88 MB (4.00×)**
ViT-Large (r=16) (ratio=30%)	Shield-LoRA	305.74M	2.36M	2.36M	9.44 MB
	Phantom	305.74M	3.54M	1.18M	4.72 MB
	TEESlice	305.74M	1.65M	1.65M	6.60 MB
	AegisGuard	305.74M	2.36M	**0.58M**	**2.32 MB (4.06×)**

In this section, we present the memory consumption of AegisGuard compared with baseline methods. Obviously, AegisGuard achieves minimal memory occupation in TEEs due to adapter compression and only partial layers shielded in TEEs. AegisGuard achieves averagely 4.00× less memory consumption than baselines.

Table 8: Inference and Computation Overhead Evaluation of ViT-Large and OPT-2.7B. indicate lowest inference overhead

	Inference Costs (ViT-Large)						Inference Costs (OPT-2.7B)				
Time (ms)	Shield-LoRA	Phantom	TEESlice	AegisGuard	No-Shield	**Time (s)**	Shield-LoRA	Phantom	TEESlice	AegisGuard	No-Shield
GPU	24.33	23.56	28.90	23.24	32.21	GPU	0.15	0.16	0.18	0.16	0.17
TEE	58.51	25.67	50.55	23.27	–	TEE	1.86	0.08	1.35	0.08	–
Trans	685.51	629.24	537.40	333.60	–	Trans	4.03	3.98	3.15	1.25	–
Total	768.35 (1.00×)	678.47	616.85	377.11 (2.04×)	32.21	Total	6.04 (1.00×)	4.22	4.68	1.49 (4.05×)	0.17

We also evaluate the inference and computation overhead for ViT-Large and OPT-2.7B, in order to demonstrate AegisGuard can be applied in different type of models. As shown in Table.8, AegisGuard achieves the best efficiency both in ViT-Large and OPT-2.7B, accelerating 2.04× and 4.05×, respectively. Additionally, we provide the transfer intermediate features maps of NLP models (OPT-2.7B, LLaMA-7B), their feature dimension are 4096. The results are presented in Table.9.

Table 9: Comparison of transfer flops/size between original and compressed hidden dimensions (30%) across different batch sizes. The original hidden size for LLMs are all 4096.

Batch Size	Original Size (MB)	Compressed Size (MB)	Original Flops	Compressed Flops
16	0.25	0.17	65,536	45,875
32	0.5	0.35	131,072	91,750
64	1	0.7	262,144	183,500
128	2	1.4	524,288	367,001

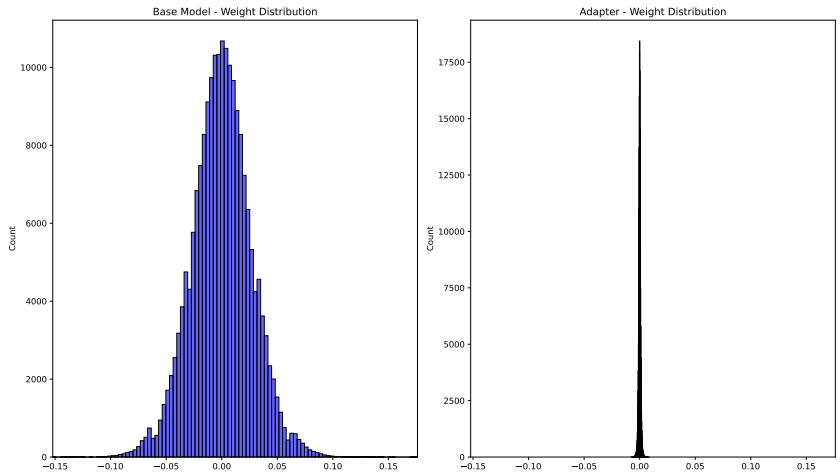

Figure 5: Weight Value Distribution Comparison between Base Model and LoRA Adapter

D.3 Gaussian Noise Analysis

Fig. 5 shows the weight magnitude distribution of base model ViT-Base and LoRA Adapter, which is W and $B \cdot A$. Obviously, the weight of W has larger value than fine-tuned adapters. Therefore, the noise setting based on adapters' weights has a negligible impact on base model weights.

In Fig.6, we present all sensitivity scores of ViT-Base for all layers under various vision datasets. The results present that different layer has domain-specific sensitivity scores, which demonstrates shielding fixed number of layers are impractical for real on device LM protection.

D.4 Evaluation of Surrogate Model

As shown in Table.10, we present the MS attack performance of ViT-Large and OPT-2.7B, in order to demonstrate `AegisGuard` can also apply to different LM architectures. The results demonstrate `AegisGuard` has comparable defensive effectiveness against black-box level protections. But for ViT-Large, the surrogate model has better attack performance than ViT-Base, cause the base model (open-sourced weights) already has good generalization ability in classify CIFAR and UTKFace datasets.

Table 10: Performance of MS attack accuracy. A lower attack accuracy (surrogate model) represents a higher defense effectiveness. indicate the best defensive capability.

	Model Stealing Attack (ViT-Large)						Model Stealing Attack (OPT-2.7B)				
Dataset	No-Shield	Phantom	TEESlice	`AegisGuard`	Shield-LoRA	**Dataset**	No-Shield	Phantom	TEESlice	`AegisGuard`	Shield-LoRA
C10	99.01	96.48	94.12	93.58	93.47	ARC-E	52.27	45.38	38.01	37.12	36.39
C100	91.05	48.63	27.25	26.86	27.01	PIQA	71.82	61.83	50.27	50.39	51.47
UTKFace	88.25	67.37	54.79	55.58	49.67	WinoGrande	59.82	50.34	45.23	45.74	45.81
Average	92.77	70.83	58.72	58.67	56.72	Average	61.30	52.52	44.50	44.42	44.56

Additionally, we conduct experiments of selecting random number of layers on standard fine tuned ViT-Large model. The results are shown in Fig.7. Obviously, standard fine tuned ViT-Large requires more shielded layers to protect its privacy. Averagely, the least number of shielded layers should

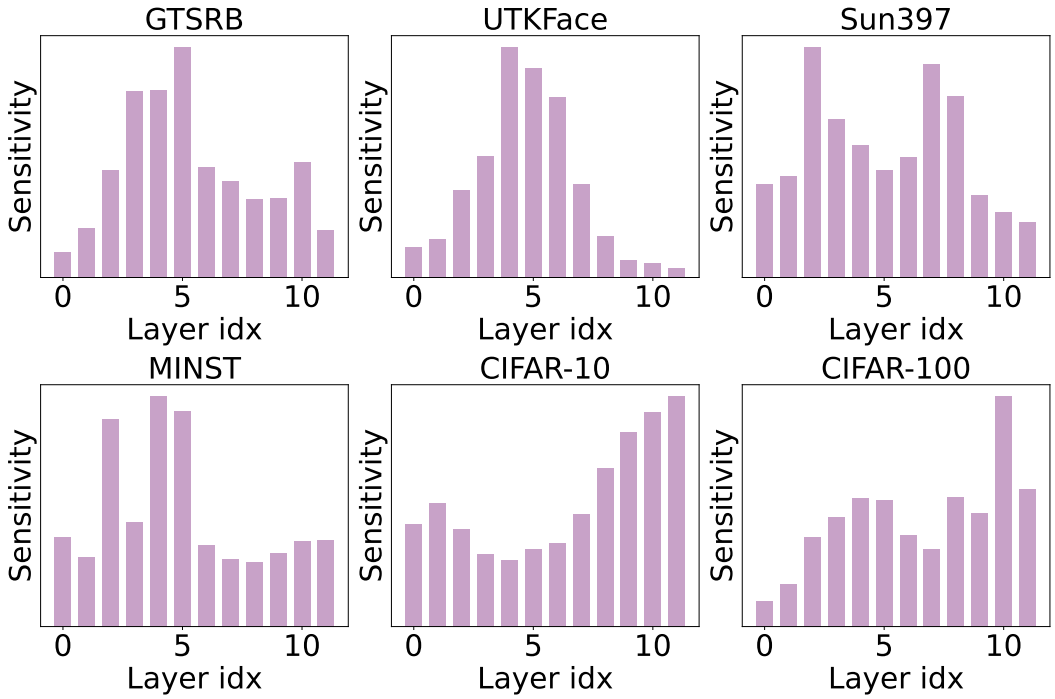

Figure 6: Sensitivity scores distribution of all layers for ViT-Base with six vision task benchmarks

greater than 15 while `AegisGuard` could achieve black-box level protection with approximately 1/3 shielded layers.

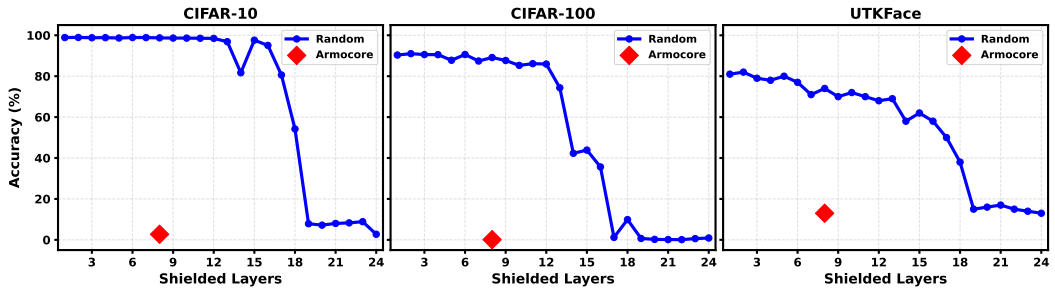

Figure 7: Accuracy comparison of Random Shielded Layers Selection and `AegisGuard`

E Implementation of Baselines

Given the lack of previous works' implementation and experiments on TEE shielded fine tuned LMs, especially with low-rank adaptation, there is no ready-to-use tools for baseline evaluation. To provide a comprehensive demonstration of the effectiveness of `AegisGuard`, we adapt most recent TEE-shielded Deep Neural Networks methods for evaluation.

E.1 Phantom Description

Phantom is a recent method proposed at USENIX Security 2025, which leverages reinforcement learning-based architecture search combined with top-K layer-wise sensitivity analysis to identify the most sensitive layers for inserting redundant modules. These redundant layers are designed to degrade model performance for unauthorized users, while maintaining full functionality for authorized users who possess valid decryption keys. During inference, authorized users must verify their keys within a

Trusted Execution Environment (TEE) to obtain correct predictions; otherwise, the model's forward pass becomes corrupted, leading to erroneous outputs.

However, Phantom is tailored specifically for models trained from scratch and cannot be directly applied to large models (LMs) fine-tuned with Low-Rank Adaptation (LoRA). To enable a fair comparison, we assume that adversaries lack knowledge of the actual forward computation path of a LoRA-based LLM. Accordingly, we propose to insert a subset of redundant LoRA adapters into the model. Our objective is to maximize the loss of LoRA-fine-tuned models in the presence of these redundant adapters. Following the experimental setup in Phantom, we insert redundant adapters into the three most sensitive layers, as determined by layer-wise sensitivity analysis. The corresponding results are presented in Appendix D.1.

E.2 TEESlice Description

Although TEESlice are specifically designed for models, such as ResNet-18[12], VGG[31], it has similar model partition mechanism as adapter-based fine tuning. TEESlice consists of two stages: model slice extraction and hybrid model deployment. The slice extraction stage automatically finds the "sweet spot" by minimizing the utility cost while maintaining accuracy and security. TEESlice trains M_{dense} from the public backbone and then prunes M_{dense} to get M_{sparse}. In the hybrid model deployment stage, M_{sparse} is deployed across the TEE and GPU. Model slices are deployed inside TEEs, whereas the other part of the backbone are offloaded on GPUs.

In our experiments, we regard lora adapter as model slices proposed in TEESlice. The objective of this method is dropping as more as adapters with minimal accuracy loss following the iterative slice prune mechanism in TEESlice. However, the pre-defined threshold δ defined as 1% governs the tolerable accuracy loss during pruning is too small for LMs to drop adapters. Then we set threshold δ as 5% as tolerable accuracy loss. The main difference between TEESlice and `AegisGuard` is that our method actively selects model layers with higher sensitivity, intentionally gathering more private information into the layers should be deployed within the TEEs. In contrast, TEESlice adopts a passive selection strategy, merely determining whether removing a certain layer would negatively impact the final performance. The experimental results and analysis are presented Appendix D.1.

F Limitations

Although the architecture presented in this paper achieves competitive performance in terms of inference efficiency, defensive effectiveness, and accuracy, it has the following limitations. First, even though we compare `AegisGuard` with most state-of-the-art baselines (TEESlice[47], Phantom[4]), published in top-tier security conferences. We do not include methods from leading AI conferences. There is an ICML 2024 paper, namely GroupCover[45], but it targets convolutional layers and cannot be applied to the Transformer models used in our setting. Second, the baselines focus on smaller models such as ResNet18 [12] and VGG16 [31], whereas we evaluate on large-scale models. Although their techniques could in principle be adapted to larger architectures, they may not achieve the same level of effectiveness in that context.