FILOsofer: A TEE-Shielded Model Partitioning Framework based on Fisher Information-Guided LoRA Obfuscation

Fan Zhang
UCLA ECE
Los Angeles, CA 90095
fanz@ucla.edu

Ziqi Zhang
UIUC CS
Urbana, IL 61801
ziqi24@illinois.edu

Hossein Khalili
UCLA ECE
Los Angeles, CA 90095
hkhalili@ucla.edu

Neiro Cabrera
UCLA ECE
Los Angeles, CA 90095
neirocabrera7@g.ucla.edu

Jonathan Xue
UCLA ECE
Los Angeles, CA 90095
jonathanxue@ucla.edu

Nader Sehatbakhsh
UCLA ECE
Los Angeles, CA 90095
nsehat@ucla.edu

Abstract

On-device machine learning exposes deep neural networks as white-box artifacts, making them vulnerable to model-stealing attacks. Trusted Execution Environments (TEEs) mitigate this risk by isolating model execution, but running entire models inside TEEs incurs prohibitive overhead. To balance security and efficiency, prior work proposes TEE-Shielded DNN Partitioning (TSDP), which executes privacy-insensitive components on GPUs while confining sensitive layers to TEEs. We demonstrate that existing TSDP schemes remain vulnerable because exposed GPU weights provide an effective warm start. This allows adversaries to exploit inevitable information leakage and reconstruct high-fidelity surrogates using only a fraction of the training data. To address this vulnerability, we propose FILOsofer, a principled defense that strategically obfuscates exposed weights using Fisher Information. By deliberately rendering leaked weights misleading, FILOsofer steers an adversary's initialization away from the true optimum. To recover user-side accuracy, we introduce a novel cross-layer LoRA mechanism that efficiently restores performance while storing only lightweight LoRA parameters inside the TEE. Extensive evaluation in real-world settings shows that FILOsofer achieves black-box–equivalent security in the worst case, while reducing computational overhead by more than $50\times$ compared to prior approaches.

1 Introduction

On-device machine learning improves latency and privacy by processing data locally, but it also exposes models to new threats such as unauthorized access, model stealing attacks, and membership inference attacks [Zhu et al., 2021, Yan et al., 2020, Rakin et al., 2022, Mehnaz et al., 2022, Zhang et al., 2023]. The model stealing attacks essentially aim to clone the victim model's functionality without authorized access to its original training data or parameters. Prior work [Zhang et al., 2024a, Yuan et al., 2024, Rakin et al., 2022] shows that white-box access to GPU-deployed models allows adversaries to efficiently steal models by replicating weights and parameters, achieving high accuracy with minimal computational cost [Orekondy et al., 2020, Juuti et al., 2019, Hanzlik et al., 2021].

To mitigate these security risks, researchers have explored two defense strategies: *(i) Cryptographic approaches:* Methods such as Multi-Party Computation (MPC) [Juvekar et al., 2018] and Homomorphic Encryption (HE) [Gilad-Bachrach et al., 2016, Kim et al., 2022] aim to safeguard both

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

input data and model parameters through algorithmic guarantees. Despite their strong theoretical protection, these techniques remain impractical for mobile and IoT deployment due to excessive computational overhead and non-trivial accuracy degradation. *(ii) Hardware-based defenses:* By leveraging Trusted Execution Environments (TEEs) [Zhang et al., 2024a, Hu et al., 2023], these methods achieve substantially lower overhead than cryptographic techniques. However, executing entire DNNs within TEEs is generally infeasible, as their computational performance is 50× lower than that of GPU-based rich execution environments (REEs).

To balance security and efficiency, recent work proposes **TEE-Shielded DNN Partitioning (TSDP)** [Zhang et al., 2024a, Mo et al., 2020, Sun et al., 2020], which protects privacy-sensitive portions inside TEEs while offloading the rest to REEs. To realize this idea, prior work explores different partitioning strategies. Some shield layers are based on depth (shallow, deep, or intermediate layers) [Shen et al., 2022, Elgamal and Nahrstedt, 2020, Mo et al., 2020] while others focus on non-linear layers [Sun et al., 2020, Zhang et al., 2024a]. In addition to layer-based partitioning, obfuscation techniques keep obfuscated or quantized weights within TEEs to protect model confidentiality [Zhou et al., 2023, Sun et al., 2024].

Despite these advances, existing TSDP methods still suffer from a ***critical limitation:*** although selected layers and parameters are protected inside TEEs, the partitioned model executed on GPUs remains highly accurate. This residual accuracy enables adversaries to bootstrap surrogate models using correct architectures and well-aligned weight initializations. To address this limitation, we propose **FILOsofer** (Fisher Information-Guided LoRA Obfuscation). FILOsofer leverages Fisher Information to identify the most information-dense parameters within the GPU-resident model. By selectively perturbing this minimal subset of critical weights, FILOsofer renders the exposed parameters quantitatively inaccurate. This effectively corrupts the adversary's surrogate model initialization, depriving them of a valid starting point and forcing them to require an infeasible amount of query data to refine the model.

To recover utility, we introduce an adaptive, cross-layer LoRA branch securely hosted within the TEE. This lightweight module learns to compensate for the induced perturbations, restoring near-original performance without incurring the prohibitive runtime overhead of reloading obfuscated weights Zhou et al. [2023]. We further propose a constraint-aware joint training algorithm that explicitly balances the degree of weight obfuscation against the size of the LoRA branch, achieving strong security guarantees with minimal latency overhead. Under this design, the adversary's capabilities are effectively *reduced* to those of a ***fully black-box setting***, where only correct input–output pairs are observable. Crucially, the exposed weights are not merely unhelpful but actively misleading, steering adversarial optimization away from the true solution and yielding worse outcomes than random initialization. The contributions of this paper are summarized as follows:

- We conduct a systematic evaluation of existing TSDP approaches, demonstrating that they fail to provide effective protection as the query budget increases.
- We propose FILOsofer, a novel TSDP framework that impedes model recovery by rendering exposed weights deceptive, while supporting low-latency inference on edge devices. FILOsofer combines Fisher-guided obfuscation with a lightweight cross-layer LoRA recovery branch, jointly misguiding the adversary, preserving predictive accuracy, and incurring minimal overhead.
- We comprehensively evaluate FILOsofer on both experimental and real-world devices (Jetson Orin Nano), demonstrating a 2× improvement in security against model stealing attacks with 50× lower computational overhead.

2 Background

Trusted Execution Environments (TEEs) TEEs offer strong confidentiality and integrity guarantees in untrusted environments by providing two key features: *execution isolation* and *code/data protection* [Costan and Devadas, 2016]. Isolation is achieved through the physical separation of hardware and memory between protected and untrusted worlds. Code and data protection rely on cryptographic techniques such as encryption and message authentication codes (MACs). Both features depend on a distinct hardware/software runtime environment, known as the trusted computing base (TCB), which operates correctly even under a fully compromised operating system (OS).

TEE-Shielded Secure Inference. To balance security and efficiency, TSDP selectively protects only parts of a DNN model within the TEE. Existing TSDP strategies generally fall into two categories: layer-based partitioning and weight-based obfuscation. Layer-based methods focus on confining specific components, including shallow Elgamal and Nahrstedt [2020], deep Mo et al. [2020], intermediate Zhang et al. [2024a], Shen et al. [2022], and non-linear layers Sun et al. [2023], Hou et al. [2021], within the TEE. Alternatively, weight-based obfuscation schemes Zhou et al. [2023], Zhang et al. [2024b], Sun et al. [2024] perturb critical parameters and leverage the TEE to securely manage the obfuscation mechanisms or compute the recovery logic. However, these approaches suffer from a critical limitation: the partitioned or obfuscated models remaining on the GPU still provide high residual accuracy. This inadvertently grants adversaries an effective *warm start* to reconstruct high-fidelity surrogates via model-stealing attacks. Consistent with the TEESlice setup, we evaluate six representative baselines across these categories.

Low-Rank Adaptation. LoRA [Hu et al., 2022, Dettmers et al., 2023] is a parameter-efficient tuning method that adapts pre-trained models by injecting trainable low-rank matrices. Formally, consider a weight matrix $W_0 \in \mathbb{R}^{d\times k}$ in a neural network layer, where d is the output dimension and k is the input dimension. Traditional fine-tuning would update all parameters in W_0, resulting in $\mathcal{O}(dk)$ trainable parameters. In contrast, LoRA freezes W_0 and injects a learnable update in the form of a low-rank decomposition: $W = W_0 + \Delta W = W_0 + BA$, where $A \in \mathbb{R}^{r\times k}$, $B \in \mathbb{R}^{d\times r}$, and $r \ll \min(d, k)$ is the rank of the decomposition. The matrix A projects the input into a lower-dimensional space of rank r (the parameter tested in our experiment), and B maps it back to the original output dimension. Only A and B are trained, reducing the number of trainable parameters from $\mathcal{O}(dk)$ to $\mathcal{O}(r(d + k))$, which is significantly smaller. Thus, LoRA achieves fine-tuning with minimal additional memory, compute, and storage cost, making it highly suitable for large-scale and resource-constrained scenarios.

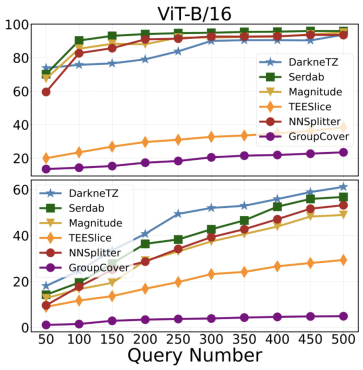

Figure 1: **Surrogate accuracy results across architectures and datasets.**

3 Threat Model

Model Stealing. We consider a deep neural network (DNN) deployed on resource-constrained edge devices equipped with Trusted Execution Environments (TEE). In this scenario, the attacker attempts to steal the victim model (M_{vic}) by exploiting access to its predictions and any unprotected components within the Rich Execution Environment (REE; e.g., GPU). Consistent with prior TSDP work and real-world deployments [Zhang et al., 2024a, Zhou et al., 2023, Zhang et al., 2024b], we assume that deployed models provide users with label-only outputs, an assumption further supported by a comprehensive survey of on-device ML systems [Sun et al., 2021].

Adversary's Capabilities. We consider the adversary's capabilities in three aspects. 1) The adversary first infers the protected model's architecture and weights from publicly available models (M_{pub}) in the REE, then initializes a surrogate model with these priors. 2) The attacker issues limited queries on carefully selected inputs and records the corresponding outputs to approximate the victim model's behavior. 3) The collected input–output pairs are then used to train the surrogate model. However, the portion of training data available for constructing such queries is restricted to fewer than 5% of the original training set, and query budgets are also restricted based on previous settings [Zhang et al., 2024a, Zhou et al., 2023, Orekondy et al., 2019].

4 Systematic Study and Insights

In this section, we conduct a systematic analysis of the limitations inherent in existing TSDP methods against model stealing attacks. By critically examining their empirical performance, we highlight key vulnerabilities and main gaps.

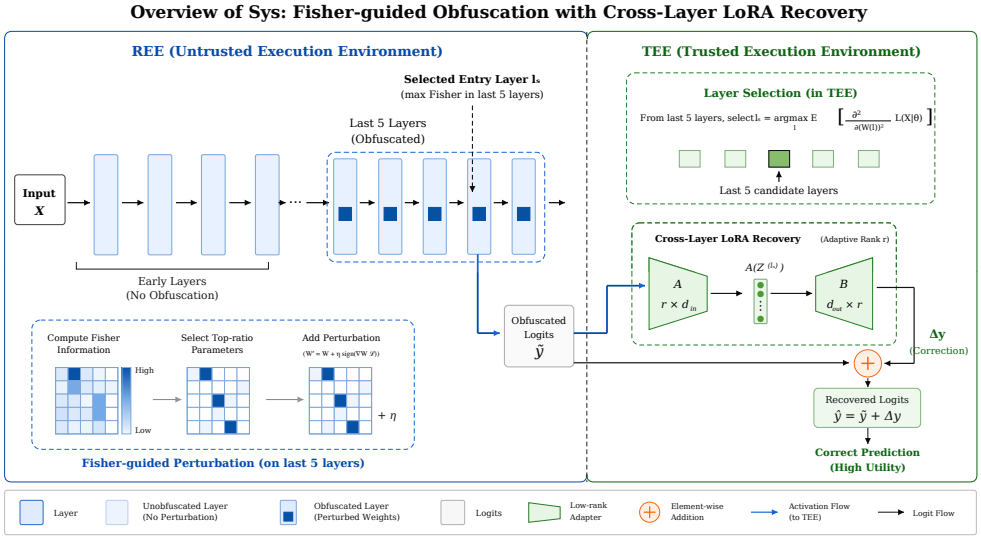

Figure 2: **Overview of FILOsofer**. The system combines Fisher-guided obfuscation to resist model stealing with cross-layer LoRA to restore utility via adaptive rank updates. The obfuscated model runs in the REE, while the LoRA branch executes inside the TEE.

For the attack algorithm, we follow prior work [Zhang et al., 2024a, Sun et al., 2020] and adopt the KnockoffNet [Orekondy et al., 2019] as our representative modified model stealing attack. Figure 1 shows that all defenses fail against model stealing as the number of queries increases. Our empirical analysis reveals a **fundamental weakness** in existing TSDP schemes: they inadvertently provide attackers with a warm start. While TSDP methods aim to hide specific layers or weights, the bulk of the model parameters residing on the GPU remain accurate. Attackers can directly extract these parameters to **initialize** their surrogate models. Although the model is partially shielded or obfuscated, the labels inevitably leak information to users. With a highly accurate initialization derived from the GPU-exposed weights, attackers can effectively utilize this label leakage to fine-tune their surrogate models, achieving high accuracy with a very small query budget. **Therefore, the key to defense is not just hiding layers but corrupting this initialization to drastically inflate the query cost required for convergence.**

5 FILOsofer

The overall system is shown in Figure 2. Our method integrates two components: Fisher-guided obfuscation, which perturbs key weights in critical layers to degrade backbone accuracy, and cross-layer LoRA, which restores task utility with adaptive rank updates. A constraint-aware joint-training algorithm balances these modules, ensuring obfuscation resists trivial recovery while LoRA maintains performance, thus achieving a trade-off between security and utility. During inference, we deploy the cross-layer LoRA in the TEE and the obfuscated model in the REE, without reloading the obfuscated weights back to the REE.

5.1 Fisher Information Guided Target Obfuscation

The goal of obfuscation is to safeguard the edge-deployed model M_{vic} by rendering its parameters functionally degenerate, thereby denying adversaries an effective initialization. While minimizing parameter perturbation to preserve recoverability, we must ensure the exposed weights capture *no* functional knowledge of the original task. Formally, let W denote the victim model's weights and ΔW denote the perturbation. To enforce functional collapse, our objective is to select ΔW that minimizes the mutual information between the original and obfuscated representations, effectively decoupling the outputs from the inputs:

$$\Delta W^* = \arg \min_{\Delta W \in \mathcal{R}} I\big(f(x; W); f(x; W + \Delta W)\big), \tag{1}$$

where $I(\cdot;\cdot)$ denotes the mutual information over the input distribution $\mathcal{D}$ and $\mathcal{R}$ constrains the perturbation (e.g., sparsity, magnitude). Let $f(x;W)$ and $f(x;W+\Delta W)$ denote the outputs of the original and obfuscated models. The mutual information is defined as:

$$I(f(x;W);f(x;W+\Delta W))=\sum_{y,z} p(y,z)\log\frac{p(y,z)}{p(y)p(z)}, \tag{2}$$

where $y=f(x;W)$, $z=f(x;W+\Delta W)$, and $p(y,z)$ denote the joint probability. Under the condition of functional collapse, where $f(x;W+\Delta W)$ is independent of x, the joint distribution factorizes as $p(y,z)=p(y)p(z)$, indicating zero mutual information. This derivation provides a theoretical objective for corrupting initialization: by optimizing ΔW to drive this mutual information to zero (i.e., forcing input-independent outputs), we effectively push the model parameters $W+\Delta W$ into a degenerate state. Consequently, these weights become functionally orthogonal to the target task, preventing their exploitation as an effective initialization point.

Fisher Information, F, has been widely applied to evaluate the importance of parameters [Rissanen, 1996]. Specifically, given a model M with input X and parameters θ_T, the Fisher information can be calculated as $F=\mathbb{E}\left[-\frac{\partial^2}{\partial\theta_T^2}\mathcal{L}(X|\theta_T)\right]$, where $\mathcal{L}$ is the loss function for the model. Intuitively, Fisher Information measures how sensitively the model output responds to small changes in its parameters; the more sensitive it is, the higher the Fisher Information. To enforce input-independent outputs, we perturb the weights that most strongly drive the model's predictions toward the target label L_t. Given input x and target L_t, the Fisher information can be calculated as:

$$F_{L_t}=\mathbb{E}\left[\left(\frac{\partial\mathcal{L}(x,W)}{\partial W}\right)^2\Big|y=L_t\right]. \tag{3}$$

We quantify perturbation intensity using the modification ratio r, defined as the fraction of weights obfuscated out of the total. To steer the model toward consistently predicting L_t, we perturb the key weights along the gradient of the target loss:

$$W\leftarrow W+\eta\cdot\nabla_W\mathcal{L}(x,W), \tag{4}$$

where η is the scale factor. Experiments show that even small η (e.g., $\leq 1e^{-4}$) are sufficient to reliably bias the model toward the target label. The complete algorithm is in Appendix 1. The full proof is provided in Appendix C.17.

5.2 Cross-Layer LoRA-based Recovery

While input-independent outputs effectively prevent accurate initialization, they inevitably degrade usability. To restore model utility for authorized users, we propose a Cross-Layer LoRA Recovery scheme. Unlike standard LoRA, which attaches low-rank modules to specific weight matrices within each layer, our approach constructs a single, global residual branch residing in the TEE.

We define the TEE recovery function $f_{\text{TEE}}(Z^{(\ell_s)};A,B)$ as a lightweight mapping that spans from a selected entry layer ℓ_s directly to the final output layer L. We term this 'Cross-Layer' because the module effectively creates a shortcut that bypasses the subsequent obfuscated layers $\ell_{s+1},\ldots,L$. The architecture consists of two low-rank matrices $A\in\mathbb{R}^{r\times d_{in}}$ and $B\in\mathbb{R}^{d_{out}\times r}$ (where r is the rank). The input is the activation map $Z^{(\ell_s)}$ from the entry layer, and the output is a correction vector in the logit space. The entry layer ℓ_s is selected from the last five layers based on maximum Fisher information, ensuring the branch originates from the most information-dense feature space:

$$\ell_s=\arg\max_{\ell}\ \mathbb{E}\Big[-\frac{\partial^2}{\partial(W^{(\ell)})^2}\mathcal{L}(X|\theta)\Big]. \tag{5}$$

Inference Flow. During inference, the GPU (REE) executes the obfuscated backbone, producing two quantities: the specific entry-layer activation $Z^{(\ell_s)}$ and the final (erroneous) prediction $\tilde{y}=f_{\text{REE}}(X;W')$. The TEE then accepts $Z^{(\ell_s)}$ and computes the recovery term. Crucially, instead of modifying intermediate weights, f_{TEE} learns to predict the precise error vector required to correct the final logits:

$$\hat{y}=\underbrace{\tilde{y}}_{\text{Obfuscated Logits}}+\underbrace{B(A(Z^{(\ell_s)}))}_{\text{Secure Correction}}. \tag{6}$$

Table 1: **The accuracy of the surrogate model.** Green and Red boxes highlight the lowest and highest accuracy, respectively.

Model	Data	No-Shield	DarkneTZ		Serdab		Magnitude		NNSplitter		TEESlice		GroupCover		Ours		Blackbox	
		/	50	5000	50	5000	50	5000	50	5000	50	5000	50	5000	50	5000	50	5000
AlexNet	C10	81.58	66.01	80.97	74.08	80.26	68.42	78.08	58.90	75.26	24.79	66.51	13.60	47.70	10.00	12.17	19.37	64.20
	C100	55.97	13.60	52.19	42.59	54.25	30.64	50.40	19.92	42.76	3.98	35.20	1.00	1.00	1.00	1.00	2.89	30.75
	IN200	47.70	4.12	33.16	31.29	34.44	14.92	35.83	3.67	29.73	0.65	13.39	0.50	0.50	0.50	0.50	0.66	13.46
ResNet18	C10	93.07	86.84	92.79	86.86	92.25	73.26	88.63	78.30	90.66	23.87	65.21	10.00	57.60	10.00	18.51	23.12	55.92
	C100	81.50	26.36	79.23	77.93	80.79	59.55	77.02	47.86	78.78	5.31	58.01	1.00	21.90	1.00	1.00	3.25	31.53
	IN200	65.68	61.22	63.50	5.96	59.10	43.08	58.39	25.24	42.12	2.27	48.08	0.50	6.30	0.50	0.50	1.16	34.06
VGG19	C10	91.42	89.34	91.45	91.44	91.42	83.93	90.15	79.87	89.51	40.52	89.62	10.00	21.70	10.00	17.25	40.61	71.05
	C100	70.39	22.63	67.71	69.07	69.85	59.95	65.34	25.50	59.62	7.31	48.97	1.00	1.30	1.00	1.00	7.09	39.34
	IN200	63.23	60.89	61.17	5.24	52.66	24.67	45.20	13.28	49.26	2.83	43.72	0.50	2.90	0.50	0.50	2.54	32.09
ViT-B/16	C10	97.69	67.96	97.12	65.54	94.99	95.26	97.92	59.55	93.56	22.64	97.64	12.60	39.70	10.31	14.87	20.63	85.02
	C100	86.58	24.38	78.17	14.70	80.84	15.64	85.48	9.63	53.17	12.69	86.89	1.60	11.50	1.41	6.83	11.62	74.36
	IN200	81.99	12.94	78.32	9.02	80.10	72.62	85.91	14.68	82.74	10.18	81.71	0.80	4.30	0.85	2.21	8.82	70.50

By superimposing this secure corrective vector onto the erroneous preliminary result $\tilde{y}$, the system successfully reconstructs the accurate label $\hat{y}$. This architectural decoupling separates utility from the bulk parameters: the REE executes the heavy, obfuscated computation, while the TEE handles the lightweight, cross-layer recovery logic. Consequently, this design prevents the leakage of functional weights while avoiding the high latency of full-model TEE execution.

5.3 Obfuscation and Recovery Trade-off

Obfuscation degrades backbone accuracy for security, while LoRA fine-tuning restores utility, creating a trade-off: excessive distortion hinders recovery and increases adaptation cost. To address this trade-off, we propose Constraint-Aware Obfuscation under Resource-Limited Adaptation (Details in Appendix, Algorithm 2). The algorithm iteratively maximizes obfuscation on the most sensitive layers while applying a resource-constrained cross-layer LoRA branch (e.g., limited in rank or parameter budget) to restore task performance. A rollback mechanism ensures that the LoRA-recovered accuracy never falls below a predefined threshold, guaranteeing recoverability. This procedure provides a realistic framework for maximizing model obfuscation under practical adaptation limits, particularly in edge device deployments, where recovery modules are inherently resource-constrained, and highlights the security–utility trade-off that arises in such constrained environments.

6 Experiments

Configuration. Following the methodology outlined in TEESlice [Zhang et al., 2024a], we evaluate feasible configurations for the benchmarks introduced in Section 2. Specifically, for DarkneTZ [Mo et al., 2020] and Serdab [Elgamal and Nahrstedt, 2020], we vary the number of consecutive layers and report the results for the last three layers and first three layers, respectively. For Magnitude [Hou et al., 2021], we test configuration parameter 'mag_ratio' among $\{0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 1\}$, where 0.01 is the recommended setting. For TEESlice [Zhang et al., 2024a], NNSplitter [Zhou et al., 2023] and GroupCover [Zhang et al., 2024b], we adopt the default configuration.

Utility Cost Metric. To evaluate the efficiency implications of different TSDP configurations, we adopt FLOPs as the primary utility cost metric. Following the setting proposed by TEESlice [Zhang et al., 2024a], %FLOPs is defined as the proportion of total floating-point operations (FLOPs) executed within the TEE, relative to the overall FLOPs of the full DNN model.

6.1 Security Guarantees and Utility Cost

Defense against Model Stealing. Table 1 presents the results of model stealing on four model architectures across three datasets under two attack budgets (50 and 5000 queries). The 'No-Shield' column denotes the baseline without any defense, reflecting that the surrogate model can directly copy the victim model. The 'Black-Box' setting assumes the attacker has no access to the model's weights and architecture but can use the input-output pairs to train the surrogate model.

Moreover, our proposed method consistently achieves superior protection across diverse datasets and architectures. As illustrated in Table 1, our approach limits the surrogate model to worse-than-random accuracy levels, significantly outperforming competing defenses and effectively mitigating model extraction threats. Crucially, our method outperforms the true black-box baseline (last two columns), indicating that the exposed weights are not merely unhelpful but actively misleading. Consequently, in the ***worst case***, our approach provides security equivalent to a black-box setting. We emphasize that our goal is not to prevent black-box model stealing under arbitrarily large query budgets, but rather to eliminate the *warm-start advantage* enabled by exposed weights. Residual vulnerabilities under large query budgets can be further mitigated through standard system-level defenses, such as user rate limiting, i.e., enforcing a cap on the total number of queries per user. In fact, this can be securely implemented within our TEE (see Appendix C.3 for details).

Cross-Layer LoRA-based Recovery. We evaluate the impact of LoRA rank on the efficacy of our cross-layer recovery mechanism. As shown in Table 6.1, increasing the LoRA rank generally improves recovery, yet even low ranks (e.g., rank 4) are sufficient to nearly restore the original accuracy. The recovery behavior varies across settings: while shallow models benefit noticeably from increased ranks, deeper models or more complex datasets (e.g., VGG19 on ImageNet200) require higher ranks to achieve equivalent restoration efficacy. In contrast, ViT-B/16 exhibits strong robustness and efficient recovery across datasets; higher ranks even surpass the original accuracy, suggesting that LoRA facilitates both utility restoration and performance enhancement.

Table 2: **The recovery accuracy of cross-layer LoRA among varying ranks**.

Model	Dataset	Original	Obfuscated	LRank 4	LRank 8	LRank 16	LRank 32
AlexNet	C10	81.85	10.00	80.77	80.95	81.32	81.68
	C100	55.97	1.00	55.93	55.95	56.30	56.92
	ImageNet200	47.70	0.50	47.89	48.08	48.96	48.90
ResNet18	C10	93.07	10.00	92.92	92.98	93.04	93.42
	C100	81.50	1.00	80.57	80.63	80.50	80.92
	ImageNet200	65.68	0.50	50.03	52.13	53.82	55.20
VGG19	C10	91.42	10.00	90.85	90.94	91.48	91.65
	C100	70.39	1.00	69.01	69.14	69.67	70.03
	ImageNet200	63.23	0.50	61.10	61.77	62.66	63.32
ViT-B16	C10	97.69	10.00	97.73	97.67	97.80	97.96
	C100	87.58	1.00	88.14	86.36	87.96	88.50
	ImageNet200	81.99	0.50	83.27	83.43	83.58	84.16

Efficiency. We further compare the computational efficiency and latency of our proposed method with existing approaches under a query size of 500. Following the definition of the Utility Cost Merit outlined in Section 6, we estimate the number of floating-point operations (FLOPs) required for each method. Following prior work [Zhang et al., 2024a], we define Utility(C) as the fraction of FLOPs that must run inside the TEE to match the security of the black-box baseline. NNSplitter [Zhou et al., 2023] uses an RL controller to select layers, making the protection level and TEE cost hard to quantify, since the modified weight ratio is not directly tunable.

As shown in Table 3, our method consistently achieves the lowest utility cost across all tested models and datasets, significantly outperforming state-of-the-art TEE-based defenses (notably against *GroupCover*, which achieves competitive accuracy results but has *orders of magnitude* higher utility cost). We also observe that utility cost increases with dataset complexity, especially for large datasets such as ImageNet. This demonstrates that storing layers without importance selection is inefficient, whereas our Fisher information–based selection and cross-layer recovery scale robustly without sacrificing security, making it practical for deployment on resource-constrained edge devices.

6.2 Ablation Studies

Motivation for Fisher-based weight selection vs. random selection. We conducted a direct comparison on ResNet18/CIFAR-100 (original accuracy: 81.52%, LoRA rank 4, scale=0.006, weight ratio=10^{-4}, obfuscating layer4+fc). Fisher-top obfuscates the weights with the largest Fisher information, while Fisher-bottom targets the smallest. As shown in Table 5, the results reveal a critical distinction. While all strategies achieve similar obfuscation accuracy ($\sim$1.00%), only Fisher-top provides an effective defense against model stealing. Both Fisher-bottom and Random strategies achieve high accuracy at a budget of 5000 queries. Fisher-top targets the most information-dense parameters that maximally influence the decision boundary. Perturbing these parameters corrupts the functional structure of the model, making the exposed weights actively misleading (i.e., providing a

Table 3: **Utility (%FLOPs) of prior works and FILOsofer**. Lower values imply lower utility cost, with %FLOPs being 0% for the white-box baseline and 100% for the black-box baseline.

	Resnet18			VGG19			Alexnet			ViT		
	C10	C100	ImageNet	C10	C100	ImageNet	C10	C100	ImageNet	C10	C100	ImageNet
DarkneTZ	100.00	100.00	72.16	98.85	100.00	80.70	100.00	100.00	83.23	91.07	91.07	75.13
Serdab	100.00	100.00	96.54	100.00	100.00	98.62	100.00	100.00	95.72	91.73	100	83.40
Magnitude	100.00	94.71	78.43	100.00	87.43	75.57	81.18	90.58	71.82	100.00	72.20	66.91
TEESlice	3.80	5.33	3.80	0.34	0.37	0.31	12.48	12.48	8.75	7.24	8.51	8.92
GroupCover	2.18	2.18	2.42	2.75	2.75	2.85	3.33	3.33	1.39	0.24	0.24	0.24
Ours	0.0027	0.0027	0.0013	0.0032	0.0032	0.0021	0.0013	0.0013	0.0017	0.0069	0.0069	0.0069

Table 4: **Ablation on Layer Selection.** Left: Obfuscating individual layer groups shows position invariance. Right: Expanding obfuscation scope with fixed LoRA parameters (layer4+fc) degrades recovery.

Position	Obf. Acc.	Q@50	Q@5000	Recovery
layer1	1.00	1.00	1.25	79.38
layer2	1.00	1.00	2.87	75.88
layer3	1.00	1.00	2.00	73.82
layer4	1.00	1.00	3.07	78.97

Scope	Obf. Acc.	Recovery
layer4+fc	1.00	80.39
layer3+4+fc	1.00	46.73
layer2+3+4+fc	1.00	37.20
layer1∼fc	1.00	14.86

misleading initialization). In contrast, perturbing low-Fisher or random weights leaves the critical decision-making parameters intact, preserving the warm-start advantage for the attacker.

Table 5: **Effectiveness of Fisher Information Guidance.** Comparison of selection strategies on ResNet18/CIFAR-100. Only Fisher-top successfully derails the surrogate fine-tuning.

Strategy	Obfuscation Acc.	Query@50	Query@5000	LoRA Recovery
Fisher-top (ours)	1.00	1.00	2.07	80.22
Fisher-bottom	1.03	54.92	77.96	80.51
Random (mean±std)	1.00±0.01	54.63±0.37	77.80±0.04	80.44±0.08

Sensitivity of layer selection for recovery. We conducted two ablation studies on ResNet18/CIFAR-100 to evaluate the spatial impact of obfuscation. First, we applied FILOsofer to each layer group individually (Table 4, Left). All positions achieve effective obfuscation and comparable surrogate resistance. Since obfuscation is effective regardless of layer position, we select the last layers for a practical reason: placing obfuscation near the output allows the LoRA branch within the TEE to directly correct the final logits while keeping the final outputs hidden. Second, we kept the LoRA parameters fixed across layer 4 and the fully connected layer (scale factor: 0.05, weight ratio: $1e^{-4}$, rank: 4) to evaluate the impact of expanding the obfuscated layers (Table 4, Right). Recovery degrades monotonically as more layers are corrupted beyond the LoRA's scope. This demonstrates that: (1) the LoRA branch is a lightweight correction module, not a standalone classifier, and cannot compensate for corruption in layers it does not cover; (2) the recovery accuracy is directly contingent on receiving meaningful features from uncorrupted preceding layers.

6.3 Memory Footprint and Fisher Preparation Cost.

Beyond inference latency, we evaluate the memory footprint of FILOsofer and the computational cost of Fisher-guided obfuscation. As shown in Table 6, the TEE-resident LoRA branch requires only 33.6 KB to 2.37 MB of parameter storage, substantially smaller than the corresponding REE backbones. The peak inference memory ranges from 69.7 MB to 352.6 MB at batch size one.

Fisher Information is computed offline during model preparation and introduces no additional computation during inference. To quantify its cost, we compare a Fisher computation pass with a standard training step (forward, loss, and backward) using identical batch sizes (100) and input dimensions. The Fisher accumulator requires 9.20 to 108.32 MB, corresponding to 0.9% to 5.8% of the training-step peak memory. The measured Fisher computation time ranges from 10.05 ms to 676.05 ms per batch, comparable to a standard training step. These results demonstrate that FILOsofer maintains a compact TEE-resident recovery module with modest offline preparation overhead.

Table 6: Memory footprint and Fisher preparation cost. REE and TEE columns report parameter storage. Inference peak memory is measured at batch size 1, while Fisher preparation uses batch size 100. Fisher time is measured per batch.

Model	REE (MB)	TEE (KB)	Infer. (MB)	Fisher (MB)	Train (MB)	Fisher (ms)
ResNet18	42.8	257.6	69.7	32.21	555.6	17.92
VGG19	76.6	33.6	103.3	9.20	320.7	10.05
ViT-B/16	327.6	2365.6	352.6	108.32	11890.8	676.05

6.4 Performance on Real-World Devices

To evaluate the practical performance of our methods, we deploy them on a NVIDIA Jetson Orin Nano, a widely used edge AI platform featuring an Ampere GPU with 32 Tensor Cores, and 8 GB of LPDDR4x RAM. In addition to AI acceleration, Jetson provides hardware-level security features by ARM TrustZone, which enables secure execution by isolating trusted operations on ARM Cortex-A CPUs. We leverage OP-TEE to run trusted applications (e.g., our LoRA branch) within TrustZone.

We choose one ImageNet image as the input. As shown in Table 8, the inference latency of the backbone models executed on Jetson varies significantly by architecture, ranging from 12.4ms for AlexNet to 91.7ms for ViT-B/16, reflecting the growing computational demand of more complex models. For end-to-end latency (includes REE, TEE, and the communication between them), our evaluation shows a variation between 13.8–16.7 ms for AlexNet and 93.2–96.1 ms for ViT-B/16. In contrast, the latency of the corresponding LoRA recovery branches deployed within TrustZone remains consistently low across all models, below 1 ms in every case. This demonstrates that our LoRA-based design imposes minimal runtime overhead while providing robust model recovery.

7 Resilience to Adaptive Attacks

Table 7: **FisherPatch results on ViT-B/16, CIFAR100.** Impact of varying scale factor (with fixed obfuscation ratio 5×10^{-5}) and weight ratio (with fixed scale factor 0.1).

Parameter	Value	Surrogate Acc. (Top-k)			Cross-LoRA Acc. (LRank)		
		1k	10k	50k	16	32	64
Scale	0.05	87.97	85.17	83.97	80.59	83.44	85.91
	0.1	2.40	2.14	1.95	79.40	81.93	81.95
	0.5	1.30	1.00	1.00	77.18	82.15	82.85
Ratio	1e-5	3.17	69.67	84.26	79.37	82.82	83.17
	5e-5	2.40	2.14	1.95	79.40	81.93	81.95
	1e-4	1.01	1.00	1.00	79.24	82.24	82.21

Norm Clipping. Following NNSplitter [Zhou et al., 2023], norm clipping [Yu et al., 2021] can be adapted to the weight level, where the adversary constrains weight perturbations within a scaled range of the modified parameters. The clipping interval is computed by scaling the minimum and maximum of $W+\Delta W'$ with a factor $t\in[0,1]$, thereby effectively compressing the range to suppress outliers:$w_i \leftarrow \text{clip}(w_i,\, t\cdot\min(W+\Delta W'),\, t\cdot\max(W+\Delta W'))$.

As shown in Table 9, norm clipping fails across all threshold values t: large t fails to clip the modified weights, whereas small t excessively clips weights, significantly degrading classification performance. In contrast, norm clipping improves accuracy for NNSplitter [Zhou et al., 2023], as its magnitude-based obfuscation does not target specific directions; the clipped weights naturally revert toward the original decision boundary, partially restoring performance.

The ineffectiveness of this defense can be attributed to two main causes: 1) **Sparse directional perturbations** are resilient to norm bounds. Since only a few weights are altered, most values remain close to the original W, preserving the obfuscation even after clipping, especially when the perturbation scale η is small (e.g., 1×10^{-4}). 2) **Semantic bias** is directional rather than magnitude-based. Perturbations align with the decision boundary of the target class. Even if clipping reduces their magnitude, the directional effect in weight space remains, sustaining misclassification.

FisherPatch. We also propose a novel adaptive attack in which the adversary is assumed to be aware that Fisher Information is used for obfuscation, but remains unaware of which specific layers are

Table 8: **Inference latency (per image) on Jetson Orin Nano**. The first row reports the inference latency (ms) of the obfuscated models executed on the GPU, the second row shows the latency (ms) of the LoRA branch deployed within TrustZone.

Model	**AlexNet**	**ResNet18**	**VGG19**	**ViT-B/16**
GPU	12.4	22.1	48.9	91.7
TrustZone	0.84	0.86	0.85	0.87
Overhead	6.3%	3.7%	1.7%	.9%

Table 9: Accuracy of obfuscated models before $\rightarrow$ after applying norm clipping with varying t from 0.1 to 0.9. For different t, the clipping accuracy remains low, which means clipping fails to restore performance across datasets.

Model	C10	C100	ImageNet200
Alexnet	10.0→10.0	1.0→1.0	0.5→0.5
Resnet18	10.0→10.0	1.0→1.0	0.5→0.5
VGG19	10.0→10.0	1.0→1.0	0.5→0.5
Vit-B/16	10.0→10.0	1.0→1.0	0.5→0.5

targeted. Consequently, the adversary computes Fisher information over the entire model, ranks the parameters, and fine-tunes only the top-k weights using 5% of the training set. We evaluate obfuscation hyperparameters (scale factor η and modified-weight ratio r), the choice of k (number of retrained parameters), and the cross-layer LoRA used for recovery, reporting both surrogate model and recovery accuracies to quantify attack success and defense robustness.

Based on Table 7, we observe a trade-off between utility and security. Minor obfuscation can be easily recovered by the adaptive attacker due to small weight perturbations. As obfuscation intensifies, the attacker's ability to recover the model progressively diminishes. In contrast, our recovery method remains robust: while heavier obfuscation requires a larger LoRA branch, recovery accuracy stabilizes once the branch reaches a sufficient size (e.g., 32).

Full-Logit Supervision. We further consider a stronger attacker with access to the full corrected logits rather than only predicted labels. On ViT-B/16 with CIFAR-100, we evaluate surrogate training using soft-label knowledge distillation with temperatures $T = 1$ and $T = 4$, alongside hard-label cross-entropy as a reference. As shown in Table 10, full-logit supervision provides limited additional benefit under the evaluated query budgets. At 5,000 queries, the strongest distillation attack achieves 2.73% surrogate accuracy, compared with 1.00% under hard-label supervision. These results suggest that FILOsofer remains resistant to extraction even when the attacker receives richer output information in this setting.

Table 10: Surrogate accuracy (%) under hard-label and full-logit supervision.

Query Budget	Hard-label CE	KD ($T = 1$)	KD ($T = 4$)
50	1.02	1.02	0.92
500	1.07	1.01	1.00
5000	1.00	2.71	2.73

8 Conclusion

We proposed FILOsofer, a TSDP framework that achieves robust protection against model stealing attacks. By leveraging a Fisher-guided obfuscation strategy, FILOsofer selectively perturbs a critical subset of weights to induce functional collapse, thereby depriving adversaries of an effective warm start. To reconcile security with utility, we introduced a novel Cross-Layer LoRA branch hosted within the TEE, which restores model performance without leaking sensitive weight information. Extensive evaluation on both experimental and real-world devices (Jetson Orin Nano) demonstrates that FILOsofer increases resistance to model stealing by 2× while reducing computational overhead by 50×. We further demonstrate the system's robustness against three adaptive attacks.

Acknowledgment

We thank our reviewers for their guidance and feedback. We also thank Nicholas Johnson and Michael Bunte for helping us with the initial TEE setup. This work was supported in part by NSF grants CNS-2303115, CNS-2312089, and CNS-2338623. The views and findings in this paper are those of the authors and do not necessarily reflect the views of NSF. The authors declare no competing interests.

References

Yuankun Zhu, Yueqiang Cheng, Husheng Zhou, and Yantao Lu. Hermes attack: Steal {DNN} models with lossless inference accuracy. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

Mengjia Yan, Christopher W Fletcher, and Josep Torrellas. Cache telepathy: Leveraging shared resource attacks to learn {DNN} architectures. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2003–2020, 2020.

Adnan Siraj Rakin, Md Hafizul Islam Chowdhuryy, Fan Yao, and Deliang Fan. Deepsteal: Advanced model extractions leveraging efficient weight stealing in memories. In *2022 IEEE symposium on security and privacy (SP)*, pages 1157–1174. IEEE, 2022.

Shagufta Mehnaz, Sayanton V Dibbo, Roberta De Viti, Ehsanul Kabir, Björn B Brandenburg, Stefan Mangard, Ninghui Li, Elisa Bertino, Michael Backes, Emiliano De Cristofaro, et al. Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4579–4596, 2022.

Ziqi Zhang, Yuanchun Li, Bingyan Liu, Yifeng Cai, Ding Li, Yao Guo, and Xiangqun Chen. Fedslice: Protecting federated learning models from malicious participants with model slicing. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 460–472. IEEE, 2023.

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, 2024a.

Yuanyuan Yuan, Zhibo Liu, Sen Deng, Yanzuo Chen, Shuai Wang, Yinqian Zhang, and Zhendong Su. Hypertheft: Thieving model weights from tee-shielded neural networks via ciphertext side channels. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4346–4360, 2024.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Utility-constrained defenses against model stealing attacks. In *International Conference on Representation Learning (ICLR) 2020*, 2020.

Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019.

Lucjan Hanzlik, Yang Zhang, Kathrin Grosse, Ahmed Salem, Maximilian Augustin, Michael Backes, and Mario Fritz. Mlcapsule: Guarded offline deployment of machine learning as a service. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3300–3309, 2021.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th USENIX security symposium (USENIX security 18)*, pages 1651–1669, 2018.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.

Sangpyo Kim, Jongmin Kim, Michael Jaemin Kim, Wonkyung Jung, John Kim, Minsoo Rhu, and Jung Ho Ahn. Bts: An accelerator for bootstrappable fully homomorphic encryption. In *Proceedings of the 49th annual international symposium on computer architecture*, pages 711–725, 2022.

Bin Hu, Yan Wang, Jerry Cheng, Tianming Zhao, Yucheng Xie, Xiaonan Guo, and Yingying Chen. Secure and efficient mobile dnn using trusted execution environments. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 274–285, 2023.

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.

Zhichuang Sun, Ruimin Sun, Changming Liu, Amrita Roy Chowdhury, Somesh Jha, and Long Lu. Shadownet: A secure and efficient system for on-device model inference. *arXiv preprint arXiv:2011.05905*, 2020.

Tianxiang Shen, Ji Qi, Jianyu Jiang, Xian Wang, Siyuan Wen, Xusheng Chen, Shixiong Zhao, Sen Wang, Li Chen, Xiapu Luo, et al. {SOTER}: Guarding black-box inference for general neural networks at the edge. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 723–738, 2022.

Tarek Elgamal and Klara Nahrstedt. Serdab: An iot framework for partitioning neural networks computation across multiple enclaves. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 519–528. IEEE, 2020.

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.

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.

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

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.

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.

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*, 2024b.

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.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Zhichuang Sun, Ruimin Sun, Long Lu, and Alan Mislove. Mind your weight (s): A large-scale study on insufficient machine learning model protection in mobile apps. In *30th USENIX security symposium (USENIX security 21)*, pages 1955–1972, 2021.

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.

Jorma J Rissanen. Fisher information and stochastic complexity. *IEEE transactions on information theory*, 42(1):40–47, 1996.

Cheng Yu, Jiansheng Chen, Youze Xue, Yuyang Liu, Weitao Wan, Jiayu Bao, and Huimin Ma. Defending against universal adversarial patches by clipping feature norms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16434–16442, 2021.

Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. Ppfl: Privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th annual international conference on mobile systems, applications, and services*, pages 94–108, 2021.

Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Ali Jalali, Dean Tullsen, and Hadi Esmaeilzadeh. Shredder: Learning noise distributions to protect inference privacy. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 3–18, 2020.

Yecheng Xiang, Yidi Wang, Hyunjong Choi, Mohsen Karimi, and Hyoseung Kim. Aegisdnn: Dependable and timely execution of dnn tasks with sgx. In *2021 IEEE Real-Time Systems Symposium (RTSS)*, pages 68–81. IEEE, 2021.

Hanieh Hashemi, Yongqin Wang, and Murali Annavaram. Darknight: An accelerated framework for privacy and integrity preserving deep learning using trusted hardware. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 212–224, 2021.

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

Shijun Zhao, Qianying Zhang, Yu Qin, Wei Feng, and Dengguo Feng. Sectee: A software-based approach to secure enclave architecture using tee. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1723–1740, 2019.

A Appendix

A.1 The Use of Large Language Model

We employed a large language model (GPT-5) to assist in proofreading and enhancing the clarity of the manuscript. Specifically, the LLM was utilized for grammar checking, sentence restructuring, and minor language polishing to improve readability and linguistic precision. All scientific content, including the formulation of hypotheses, experimental design, data analysis, and interpretation of results, was entirely executed, and verified by the authors.

B Accessibility

To facilitate reproducibility, we have uploaded the full implementation of our method, including training scripts, evaluation code, and configuration files, to a github repository [1].

C Impact Statement

This work uses only publicly available datasets and does not involve human subjects or any private or sensitive information. We strictly follow all licensing terms and usage guidelines associated with the datasets employed. Our experiments are conducted in a controlled research setting, ensuring that no confidential or personally identifiable data is exposed or utilized. The contributions of this study focus entirely on methodological improvements in existing TSDP methods, aiming to enhance security and efficiency in machine learning systems. Moreover, all code and evaluations are intended for academic and scientific purposes, promoting reproducibility and responsible research.

C.1 Related Work

C.2 Algorithm

Fisher-guided Obfuscation We summarize FILOsofer obfuscation mechanism in Algorithm 1. The process takes three steps: (1) computing the gradients with respect to the target class t, (2) estimating the Fisher information based on these gradients, and (3) updating the weights with the highest Fisher information.

Constraint-Aware Obfuscation under Resource-Limited Adaptation Algorithm 2 implements a systematic procedure to maximize model obfuscation while respecting the resource constraints of the adaptation module. Joint training uses the same setup as the cross-layer LoRA finetuning

[1] The link is available at: `https://github.com/ssysarch/FILOsofer`

Table 11: We categorize prior work relevant to TSDP and highlight the representative studies that are empirically evaluated in this paper.

Category	Name	Venue	Methods	Weakness
Shallow Layers	Serdab Elgamal and Nahrstedt [2020]	CCGRID 2020	Put the layers closer to the input inside the TEE	No protection on other layers and outputs
Deep Layers	PPFLMo et al. [2021]	MobiSys 2021	Put the layers closer to the output inside the TEE	No protection on other layers and outputs
	DarkneTZ Mo et al. [2020]	MobiSys2020		
	Shredder Mireshghallah et al. [2020]	ASPLOS 2020		
Intermediate Layers	AegisDNN Xiang et al. [2021]	RTSS 2021	Choose intermediate layers inside the TEE	No protection on other layers and outputs
	SOTER Shen et al. [2022]	ATC 2022		
	TEESlice Zhang et al. [2024a]	S&P 2024		
Non-Linear Layers	ShadowNet Sun et al. [2023]	S&P 2023	Put non-linear layers, such as activation layers, inside the TEE	No protection on other layers and outputs
	Magnitude Hou et al. [2021]	TDSC 2021		
	DarKnight Hashemi et al. [2021]	MICRO 2021		
	Slalom Tramer and Boneh	ICLR 2018		
Model Obfuscation	NNSplitter Zhou et al. [2023]	ICML 2023	Perturbs critical weights and store them inside the TEE	Obfuscated weights need to be reload to REE for inference, no protection on the output.
	GroupCover Zhang et al. [2024b]	ICML 2024		
	TSQP Sun et al. [2024]	S&P 2025		
	TensorSheild	CCS 2025		

Algorithm 1 Model Obfuscation with Fisher Information

Require: Loss function $\mathcal{L}$, model parameters W, dataset $\mathcal{D}$, target class L_t, selection ratio r, scale factor η
Ensure: Updated (obfuscated) model parameters W'
1: **for** each $x \in \mathcal{D}$ **do**
2: compute loss for target class: $\mathcal{L}(x, W)$
3: compute gradient: $g(x) \leftarrow \nabla_W \mathcal{L}(x, W)$
4: accumulate squared gradients for Fisher estimate: $F_W = g(x) \odot g(x)$
5: **end for**
6: normalize Fisher estimate: $F_W \leftarrow F_W / |\mathcal{D}|$
7: **for** each parameter tensor/block w in W **do**
8: select top-r fraction indices by F_W: $\mathcal{I} \leftarrow \mathrm{TopK}(F_W, r)$
9: compute perturbation on selected indices: $\Delta w_{\mathcal{I}} \leftarrow \eta \cdot g_{\mathcal{I}}$
10: apply perturbation: $w' \leftarrow w + \Delta w$
11: **end for**
12: Return W'

stage. For the obfuscation component, we only perform a single forward pass to calculate the Fisher information of each weight, which introduces negligible overhead. The LoRA branch requires finetuning and therefore needs access to the corresponding training dataset (e.g., CIFAR-100). To maintain efficiency while preserving effectiveness, rather than attaching a separate LoRA module to every layer, we design a cross-layer LoRA branch that spans all obfuscated layers, significantly reducing both parameters and training cost.

Line 1–5: Layer Sensitivity Selection. We first compute the Fisher information for each layer to measure its sensitivity (Line 2). The entry layer ℓ_s is chosen as the most sensitive layer, and all subsequent layers $\ell \geq \ell_s$ are defined as target layers L_t for obfuscation. This ensures that the perturbation focuses on layers critical to model performance.

Line 7–8: Accuracy Evaluation and Stopping Criterion. After recovery, the LoRA-recovered accuracy Acc_L is evaluated. The iteration continues until Acc_L falls below a predefined threshold, ensuring that obfuscation is maximized without exceeding the adaptation capacity.

Line 10–16: Iterative Obfuscation and Recovery. For each iteration, the obfuscation function F_{obf} is applied to target layers L_t, progressively increasing the perturbation magnitude via parameters (r_{obf}, η). The resource-constrained LoRA branch is then applied across L_t to restore task utility under the given adaptation budget.

Core Insight. This constraint-driven loop reveals that, under limited adaptation resources, one can systematically explore the maximum obfuscation a model can tolerate. By decoupling obfuscation strength from adaptation capacity, the algorithm balances security (through progressive perturbation) and utility (through resource-limited recovery), providing a principled mechanism to probe the security–utility trade-off. Compared with prior obfuscation methods such as TEESlice, NNSplitter, and GroupCover, our computational cost is substantially lower, and the overall process is more stable. TEESlice requires iterative slice pruning and repeatedly training the pruned model, while NNSplitter

Algorithm 2 Constraint-Aware Obfuscation under Resource-Limited Adaptation

Require: Backbone model M, dataset D, weight obfuscation function F_{obf}, cross-layer LoRA branch $h_{\text{LoRA}}(A, B)$, iterations T, initial obfuscation ratio r_{obf}, scaling factor η, fixed LoRA rank r_l, LoRA accuracy threshold Acc_{LoRA}

Ensure: Obfuscated model M_{obf} and LoRA branch h_{LoRA}

for each layer ℓ with $|\ell, \ldots, L| < 5$ **do**
  Compute $F_\ell = \mathbb{E}\Big[- \frac{\partial^2 \mathcal{L}(X|\theta)}{\partial (W^{(\ell)})^2} \Big]$ {Layer selection via Fisher Information}
end for
Select entry layer $\ell_s = \arg\max_\ell F_\ell$
Define target layers $L_t = \{\ell \geq \ell_s\}$
for iteration $t = 1$ to T **do**
  if $\text{Acc}_L < Acc_{\text{LoRA}}$ **then**

    return $M_{\text{obf}}, h_{\text{LoRA}}$ {Rollback IF threshold violated}
  else
    $r_{\text{obf}} \leftarrow r_{\text{obf}} \cdot \beta$, $\eta \leftarrow \eta \cdot \beta$ {Increase obfuscation}
    for each layer $\ell \in L_t$ **do**
      $W^{(\ell)} \leftarrow F_{\text{obf}}(W^{(\ell)}; r_{\text{obf}}, \eta)$ {Weight Obfuscation}
    end for
    Apply $h_{\text{LoRA}}(A, B)$ with fixed rank r_l across L_t {LoRA Recovery}
    Store M_{obf} and $h_{LoRA}(A, B)$
    $\text{Acc}_L = M_{\text{obf+LoRA}}(D)$ {Evaluate LoRA-recovered accuracy}
  end if
end for

relies on reinforcement learning to identify layers and weights, often requiring many search rounds. GroupCover applies both randomization strategies and mutual covering obfuscation, and needs to calculate the mask process and nonlinear parts in TEE. In contrast, our approach needs only one Fisher pass plus lightweight cross-layer LoRA finetuning, making it significantly more efficient.

C.3 Authorized Access and TEE Implementation

We explain how authorized access (e.g., for LLMs) and/or rate limitation (to limit the number of queries) is achieved in our setup.

To ensure security, we consider a *provisioning* step, where a remote trusted gateway and TEE agree on a "token" and a "session key" (uk) Zhao et al. [2019]. The assumption is that users negotiate with such a trusted gateway (which knows the license key), and once proper authentications are made, the trusted gateway provisions the new token and session key and shares them with the authenticated user Zhao et al. [2019].

The session key is generated by leveraging a symmetric *license* key, k, using established cryptographic algorithms Zhao et al. [2019]. All communication after this point (including communication required for token generation) is cryptographically protected (integrity and confidentiality) by uk.

The token is created by also leveraging the license, and can be defined as: $user_{id}, credits, expiry||HMAC_k(...)$. The "credits" and "expiry" are optional but can be set if this is a pay-per-inference service.

During the inference phase, the remote user can directly query the model by creating an ARM TrustZone SDK call (i.e., Secure Monitor Call, SMC) with the token. Note that all communications are encrypted and authenticated using uk. The trusted app (TA) then verifies the token and accepts the request if credits remain and are unexpired. The TA then performs the inference and returns a response. Under this model, an unauthorized user, whether local or remote, cannot successfully query the model, as they lack access to the session key and valid tokens.

C.4 Evaluation on MIA

We also evaluate membership inference attacks as downstream threats to demonstrate the effectiveness of our protection. Membership inference attacks (MIA) test whether an input sample x belongs to the training dataset D_{train}. Formally, given query access to a target model f_θ, the adversary constructs

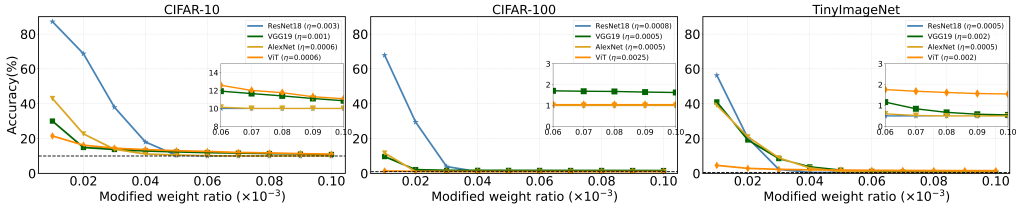

Figure 3: The influence of modified weight ratio with target label 3. For different models, different η are applied. The inserted figure shows an amplified view of the x-axis in the range [0.06, 0.10].

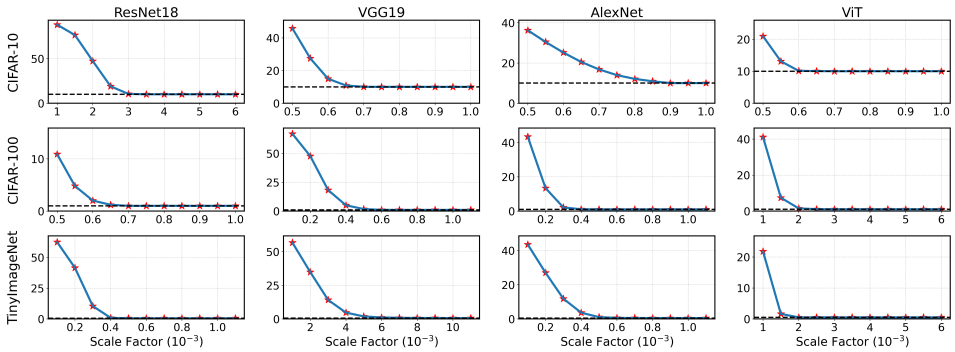

Figure 4: The influence of scale factor η on different datasets and models.

a hypothesis test between $H_0 : x \notin D_{train}$ and $H_1 : x \in D_{train}$, often leveraging prediction confidence or loss values.

Table 12: **Results of membership inference attack**.

Dataset/Model	Serdab	DarkneTZ	Magnitude	NNsplitter	TEESlice	Ours	Blackbox
C10/ResNet18	66.06	65.13	59.28	50.00	50.00	50.00	50.00
C10/VGG19	63.87	64.03	58.82	50.00	50.00	50.00	50.00
C100/ResNet18	91.81	85.47	61.88	50.00	50.00	50.00	50.00
C100/VGG19	87.80	84.66	71.48	50.00	50.00	50.00	50.00

The results show that obfuscated methods, like NNSplitter, is effective against membership inference attacks (MIA). By perturbing parameters, obfuscation reduces overfitting and diminishes the statistical gap between members and non-members in the output distribution $p_\theta(y|x)$, thereby weakening the adversary's likelihood test advantage.

C.5 Impact of Obfuscation Parameters

We further explore the factors that affect the effectiveness of model obfuscation. In particular, we examine the influence of the scaling factor η, weight modification ratio r, and target label L_t. Based on the experiment, we have the following findings:

Scale Factor η. Figure 4 shows the impact of the scale factor η across different models (Resnet18, VGG19, AlexNet, and ViT) and datasets (CIFAR10, CIFAR100 and TinyImageNet200). As shown in the figure, different models and datasets favor different scaling factors; the optimal range for ResNet18 on CIFAR-10 is around $1e^{-3}$, while for ViT on the more complex TinyImageNet, it is in the much smaller range of approximately $5e^{-4}$. All effective values remain very small, a characteristic that is advantageous for obfuscation as it makes the corresponding modifications to the input data difficult to detect.

Weight Ratio r. A small set of weights heavily influences predictions, but their identities vary across models and datasets. As shown in Figure 3, modifying key weights in ViT models reduces accuracy on CIFAR-10 and TinyImageNet, but the output does not consistently converge to the target label. When the scale factor is too small, even widespread perturbations fail to induce consistent

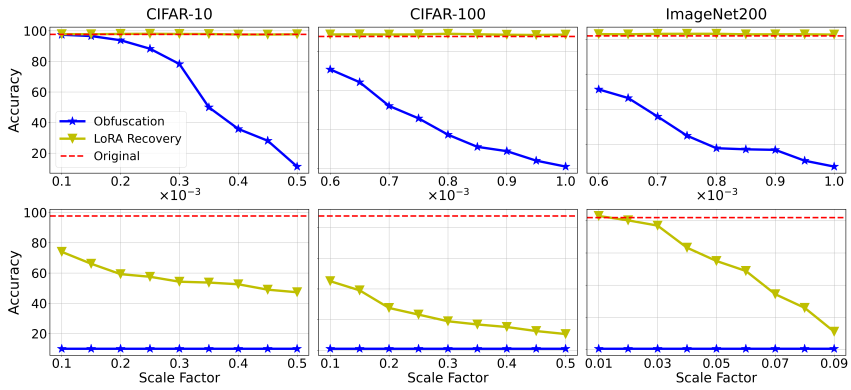

Figure 5: **The trand-off between Fisher-guided obfuscation and Cross-layer LoRA-based fine-tuning.**

misclassification. This underscores the importance of selecting an appropriate scale and confirms that only a small fraction of weights are truly critical to model behavior.

Target Class: Table 13 illustrates how obfuscation accuracy varies with different target labels, under the setting of $r = 0.00005$ and $\eta = 0.003$ for ResNet18, $\eta = 0.0007$ for VGG19, and $\eta = 0.001$ for AlexNet. By slightly increasing η, the accuracy drops to 10% (all output target labels) across all classes. Therefore, we carefully select η to highlight the differences between target labels. The results show only minor differences across target labels, suggesting that the model's sensitivity is largely uniform regardless of the target.

Table 13: The evaluation of obfuscated target classes on CIFAR-10.

Class	ResNet18	VGG19	AlexNet
0	10.38	10.25	10.40
1	14.90	10.24	11.33
2	18.67	10.00	10.42
3	10.15	10.00	10.55
4	10.38	10.00	12.24
5	11.81	10.00	12.30
6	12.85	10.00	12.10
7	10.85	10.00	12.41
8	12.72	10.25	12.66
9	18.91	10.02	11.21

C.6 Trade-offs Between Fisher Information Obfuscation and Cross-layer LoRA-based Fine-tuning

We also evaluate the trade-off between obfuscation and LoRA-based fine-tuning, which is shown in Fig 5. We set $r = 0.00005$, target label three, and LoRA rank two. By varying η, we control the degree of model obfuscation and observe the extent to which the LoRA branch fails to recover.

Across Datasets. The tolerance for obfuscation varies significantly by dataset complexity. On CIFAR-10, even aggressive perturbations (e.g., reducing accuracy to 10%) still allow LoRA to recover over 97% performance, showing strong robustness in simpler tasks. In contrast, CIFAR-100 and ImageNet200 exhibit much steeper trade-offs: small increases in obfuscation strength rapidly degrade recoverability, reflecting their higher label granularity and reliance on fine-grained features.

Across Models. ViT achieves better LoRA recovery than CNN-based models under the same obfuscation conditions, especially on CIFAR-100 and ImageNet200. This suggests that Transformer architectures offer more adaptable representations, even when key weights are perturbed.

Joint Training Advantage. These results validate the effectiveness of our joint training algorithm, which dynamically balances obfuscation strength and LoRA capacity.

C.7 Comparison with Cryptographic Parameter Protection

We evaluate a Slalom-style additive-masking baseline on ResNet18/CIFAR-100 to examine whether protecting model parameters alone removes the attacker's ability to reconstruct model functionality. The baseline masks the weights of all linear operators, comprising 20 convolutional layers and one fully connected layer. For each operator, the TEE samples a secret mask R and exposes only $W + R$ to the REE. Given the runtime input x, the REE computes $(W + R)x$, and the TEE subtracts Rx to

Table 14: Surrogate accuracy against the additive-masking baseline on ResNet18/CIFAR-100 under different query budgets.

Method	50 queries	500 queries	5,000 queries
Crypto additive masking	1.71%	9.03%	48.51%

Table 15: Trusted-execution cost on the measured RTX A6000 setup for the additive-masking baseline and FILOsofer. Latency is not a Jetson TrustZone end-to-end measurement.

Method	TEE (MMACs)	Memory (MB)	Latency (ms)
Crypto additive masking	555.47	62.77	0.592
FILOsofer	0.015	8.64	0.053

obtain the intended linear output Wx. The correction term Rx depends on the input and cannot be precomputed once for arbitrary inference inputs. Moreover, the REE cannot compute the correction without knowing the secret mask. This implementation therefore requires corresponding linear computation inside the TEE on each inference.

Table 14 reports the observed model-extraction accuracy. Although masking protects the original weight values, surrogate accuracy reaches 48.51% after 5,000 queries under the evaluated attack. This illustrates that parameter confidentiality alone does not necessarily prevent functional model extraction within the stated threat model. FILOsofer instead aims to reduce the optimization value of the weights exposed to the attacker by deliberately introducing a misleading initialization. Table 15 reports the trusted-computation cost of the two implemented schemes. The comparison reflects this specific additive-masking construction, rather than a universal claim about all cryptographic protection techniques.

C.8 Kerckhoffs-Style Adaptive Recovery Attacks

We consider an adaptive adversary who knows the defense architecture and Fisher-guided obfuscation procedure, obtains the exposed obfuscated weights, and searches candidate entry layers for a replacement recovery mechanism. The genuine TEE-resident recovery parameters remain inaccessible. On ResNet18/CIFAR-100, the last block is obfuscated to approximately 1% exposed-backbone accuracy. We evaluate entry layers 1 through 4, a data budget of 5%, and query budgets of 50, 500, and 5,000. Two strategies are tested under this shared setup: (i) fitting a rank-4 cross-layer LoRA bypass from the chosen entry layer to the output and (ii) freezing the prefix and fine-tuning the obfuscated tail (SFT).

Tables 16 and 17 report the resulting surrogate accuracies. The strongest LoRA-based reconstruction achieves 45.80% at 5,000 queries, while tail fine-tuning reaches 57.56%. Recovery is therefore sensitive to both entry location and attack strategy. These results evaluate a stronger, method-aware adversary rather than assuming that the attacker is unaware of the perturbation or recovery architecture; they do not establish that all adaptive recovery attempts remain near chance.

C.9 Extraction under Larger Query Budgets

We extend model-extraction experiments to larger query budgets to examine whether obfuscated-weight initialization eventually becomes useful as more labeled data become available. Table 18 reports the highest surrogate accuracies observed across the trials in the rebuttal experiments. At 10,000 queries, accuracy from the exposed obfuscated initialization is 6.83%, compared with 87.39% for the black-box reference. At 50,000 queries the gap narrows but remains substantial (70.62% versus 90.85%). With access to the entire training set, the respective accuracies are 83.73% and 90.87%.

These results indicate that larger budgets improve model extraction, but that the tested obfuscated initialization does not outperform the black-box reference over the reported budgets. The defense is not claimed to prevent extraction under unlimited data or computational resources. The "All Data" column denotes access to the full training set, not a fixed number of queries. In interpreting the

Table 16: Adaptive cross-layer rank-4 LoRA recovery accuracy (%) on ResNet18/CIFAR-100.

Entry layer	50 queries	500 queries	5,000 queries
Layer 1	1.04	29.72	38.77
Layer 2	1.00	30.62	40.22
Layer 3	1.00	31.27	40.67
Layer 4	1.00	39.18	45.80

Table 17: Adaptive obfuscated-tail fine-tuning (SFT) accuracy (%) on ResNet18/CIFAR-100.

Entry layer	50 queries	500 queries	5,000 queries
Layer 1	1.00	1.00	1.22
Layer 2	1.00	0.63	1.66
Layer 3	1.00	1.11	3.93
Layer 4	1.00	48.91	57.56

comparisons, the initialization and query/data protocols should be kept matched within each evaluated configuration.

C.10 Scalability to Larger Vision Architectures

We evaluate FILOsofer on two larger vision classifiers, ViT-L/16 and Swin-L, using CIFAR-100. For ViT-L/16, we obfuscate the final two transformer blocks (22 and 23) and the classification head with scale 0.03 and modification ratio 10^{-4}. For Swin-L, we obfuscate stage 4 and the head with scale 0.1 and the same ratio. A rank-4 cross-layer LoRA branch is used for recovery in both settings.

Table 19 reports accuracy before obfuscation, after legitimate recovery, and for surrogate attackers trained from exposed obfuscated weights. The recovered ViT-L/16 and Swin-L achieve 88.59% and 93.19% accuracy, respectively; at 5,000 queries the corresponding surrogate accuracies remain at 4.71% and 5.13%. We also examine parameter-selection strategies at this scale. Table 20 shows that top-Fisher selection collapses exposed backbone accuracy to 1%, whereas random and bottom-Fisher selection leave substantially higher accuracies. These experiments support the practical relevance of Fisher-guided selection on the evaluated larger vision classifiers, without establishing scalability to arbitrary model sizes or modalities.

C.11 REE–TEE Interface and Adversary Visibility

Our primary model-stealing threat model grants the adversary access to the exposed REE model weights, knowledge of the model architecture and defense procedure, and label-only queries to the deployed victim. It does not grant access to the private TEE-resident LoRA parameters or arbitrary observation of intermediate runtime activations. During inference, the obfuscated backbone in the REE computes the entry activation $Z^{(\ell_s)}$ and the obfuscated output; the recovery branch in the TEE uses the designated activation to generate the correction, while the authorized interface returns the final predicted label. The recovery matrices and internal correction are protected within the enclave under the assumed TEE confidentiality and integrity boundary.

It is important to distinguish information present on a REE–TEE interface from information made observable to the modeled attacker. In particular, the label-only application interface alone does not imply that an adversary with arbitrary REE memory access is unable to inspect intermediate activations. Such arbitrary runtime observation is not included in the evaluated adversary capabilities. If an attacker could collect clean entry activations together with sufficiently informative correct outputs, the resulting pairs could enable training a substitute recovery module. Resistance to that stronger attack is therefore not established by weight obfuscation alone, and confidentiality of the relevant recovery state and interface data must be understood as part of the deployment assumptions. System-level authorization and rate limiting can further restrict queries but are separate from the protection offered by the obfuscated initialization.

Table 18: Maximum observed surrogate accuracy (%) at larger extraction budgets, as reported in the rebuttal.

Method	10k	20k	30k	50k	All Data
Obfuscated initialization	6.83	17.09	57.16	70.62	83.73
Black-box reference	87.39	88.84	90.25	90.85	90.87

Table 19: FILOsofer on larger vision classifiers using CIFAR-100. All accuracy values are percentages.

Model	Victim	Defender	Atk@50	Atk@500	Atk@5000
ViT-L/16	90.77	88.59	0.86	1.28	4.71
Swin-L	93.38	93.19	1.00	1.04	5.13

C.12 Side Channels and Leakage of Recovery Parameters

FILOsofer relies on the confidentiality and integrity of the trusted computing base (TCB). In the implemented design, the recovery branch follows a fixed computational structure and uses the same REE–TEE communication pattern across inputs; the obfuscation itself does not require an additional input-dependent choice of recovery location. Nevertheless, this does not establish side-channel resistance. Cache, timing, power, electromagnetic, memory-disclosure, fault-injection, and other microarchitectural attacks against the TEE or its interface fall outside the evaluated threat model and require TCB-level analysis or countermeasures.

The consequences of recovery-parameter leakage also depend on the compromised information. FILOsofer stores compact low-rank recovery parameters in the TEE rather than full sensitive backbone layers. This reduces the protected parameter footprint relative to schemes that place larger layers inside the enclave. However, disclosure of the complete recovery branch, in combination with the already exposed obfuscated backbone and the necessary feature interface, could allow an attacker to reconstruct the authorized model's functionality. FILOsofer therefore makes no security claim under complete TEE compromise. Nor does the small size of the protected branch alone guarantee robustness to partial leakage; the effect depends on which parameters and auxiliary information are disclosed.

C.13 Scope and Limitations Beyond Classification

The current formulation and experiments target image classification models with fixed output classes. In particular, target-directed obfuscation attempts to bias predictions toward a designated class, and extraction is evaluated through accuracy under bounded label queries. These assumptions do not immediately transfer to autoregressive language models: at each generation step, the relevant output is a context-dependent distribution over a large vocabulary, and model extraction is typically evaluated through sequence-level generation or distribution distillation rather than classification accuracy. Extending the obfuscation objective and the trusted recovery interface to generative models therefore requires additional algorithmic development and evaluation and is left for future work.

C.14 Cross-layer LoRA Adaptive Attack Discussion

We conducted a systematic evaluation of the impact of both the LoRA rank and the amount of available training data on the adaptive attack performance, results shown in Tab 21. Our findings suggest that these two factors exhibit a strong interdependence. Specifically, for a fixed LoRA rank, increasing the proportion of training data consistently leads to improved accuracy, as the model benefits from more representative and diverse training signals. For example, by increasing the training data from 5% to 10%, ViT-B/16 accuracy becomes higher for all dataset.

However, the relationship between LoRA rank and performance is more nuanced. Contrary to the intuition that higher-rank adaptations might yield better results due to increased capacity, we observe that excessively high ranks can lead to suboptimal performance, particularly when the training data is limited. In such scenarios, large LoRA branches introduce a greater number of trainable parameters,

Table 20: Exposed-backbone accuracy (%) under different parameter-selection strategies on CIFAR-100.

Model	Fisher-top	Random	Fisher-bottom
ViT-L/16	1.00	39.35	88.60
Swin-L	1.00	70.16	93.38

Table 21: The accuracy of the whole model cross-layer LoRA fine-tuning with different LoRA ranks and fine-tuning data.

	Percentage(%)	AlexNet			VGG19			ResNet18			ViT-B/16		
	LoRA Rank	1	4	16	1	4	16	1	4	16	1	4	16
C10	5%	17.62	17.06	16.10	19.48	23.72	22.61	17.53	17.25	19.78	13.38	14.03	12.62
	10%	17.22	17.61	16.53	20.45	23.44	22.78	17.76	18.59	20.93	13.63	14.73	13.28
C100	5%	11.47	12.95	10.45	12.20	12.85	12.84	11.05	11.83	12.03	22.49	15.62	13.67
	10%	12.98	13.78	10.63	14.09	14.54	14.73	12.54	12.22	12.79	24.22	22.41	21.62
ImageNet	5%	4.05	3.08	3.20	6.11	5.60	4.83	5.93	4.25	3.40	21.32	22.37	22.57
	10%	8.78	3.83	3.25	6.36	5.69	5.76	9.39	6.42	5.24	21.49	23.64	29.36

which may not be adequately optimized given the data constraints. Also, larger parameter spaces lead to more complex loss surfaces, making training more sensitive to initialization and learning rates.

These findings reveal a fundamental challenge in the attacker's recovery strategy. Despite increasing the LoRA rank or leveraging a moderate amount of training data, the obfuscated base model imposes a structural bottleneck that restricts information flow. Consequently, even high-capacity LoRA branches struggle to compensate for the intentionally degraded base model, resulting in a persistent gap from the original performance. This supports the robustness of the FILOsofer technique against adaptive fine-tuning attacks.

We observe that when knowing exactly which layers have been obfuscated, applying cross-layer LoRA directly to these layers enables effective recovery of the original model behavior (as demonstrated by the recovery methods summarized in Tab. 6.1). However, in the adaptive attack scenario, where the attacker knows nothing about the target layers and attaches a large LoRA branch only at the input and output of the model, recovery performance significantly degrades. This contrast reveals several key insights.

1) **Lack of Access to Obfuscated Semantics.** The obfuscation targets the last few layers of the model, where task-specific semantics reside. LoRA branches attached only at the input/output cannot directly influence or correct these corrupted internal representations, making recovery ineffective. 2) **Gradient Misalignment.** When fine-tuning is performed without targeting the actual obfuscated layers, the gradients flow through a corrupted backbone. This leads to poor alignment between the loss signal and the parameters that need adaptation, severely limiting learning efficiency. 3) **Input-Level Adaptation is Too Weak.** Adapting only at the input/output level essentially treats the backbone as a fixed black box. Without modifying the internal transformations, the model cannot recover class-separability or generalization, especially when its outputs are collapsed to a single label.

C.15 The Choice of Obfuscated Weights Analysis

We present the top five most sensitive weights of ResNet18 across different datasets and target labels in Table 22. The results indicate that the specific sensitive weights vary significantly depending on both the model's training data and the chosen target class.

C.16 Future Discussion

Scalability of large language models A promising direction for future research is further exploring the applicability of FILOsofer to protect large language models (LLMs), which pose unique challenges beyond those addressed in our current work. First, the definition and evaluation of model stealing in the context of LLMs remain underexplored and ambiguous. Unlike classification models with clear prediction labels, LLMs operate in open-ended generation settings such as dialogue, summarization, or instruction following, making it difficult to measure what constitutes a successful attack. Second, our current approach is tailored to classification tasks and does not account for the nuanced and

Table 22: Top-5 gradient-sensitive parameters per class and dataset. Each entry shows Layer[Index] of the most sensitive parameters.

Label	CIFAR10	CIFAR100	TinyImageNet200
0	layer4.1.bn1.bias[318] layer4.1.bn1.bias[480] layer2.0.bn2.bias[40] layer2.0.downsample.1.bias[40] layer4.1.bn1.bias[38]	layer4.1.bn1.bias[196] fc.weight[17] fc.weight[182] layer4.1.bn1.weight[489] fc.weight[310]	fc.weight[255] fc.weight[280] layer2.0.bn2.bias[55] layer2.0.downsample.1.bias[55] fc.weight[182]
1	layer4.1.bn1.bias[38] layer4.1.bn1.bias[132] layer4.1.bn1.weight[132] layer4.1.bn1.bias[318] layer4.1.bn1.weight[38]	layer2.0.bn2.bias[121] layer2.0.downsample.1.bias[121] layer1.1.bn1.bias[55] layer2.1.bn2.bias[121] fc.weight[529]	layer4.1.bn1.bias[461] fc.weight[767] fc.weight[792] layer4.1.bn1.bias[259] fc.weight[694]

context-dependent outputs of LLMs. Obfuscating the model in such a way that it consistently degrades the utility of stolen outputs without harming legitimate usage requires more sophisticated techniques. Developing mechanisms that generalize to the diverse interaction modes of LLMs will be critical for securing them in real-world applications.

Distributed deployment scenarios. Another important direction for future exploration is the protection of models in distributed deployment scenarios, where a single model is partitioned and deployed across multiple edge devices. In such settings, different segments of the model are executed on separate devices, potentially increasing the system's vulnerability surface. Attackers may attempt to compromise a subset of devices to reconstruct the behavior of the partial model or launch collaborative attacks. Our current framework, FILOsofer, is designed under the assumption of a single-device deployment and does not yet consider inter-device communication or consistency under adversarial interference. Adapting FILOsofer to support secure distributed inference requires addressing challenges such as secure partition coordination, synchronization of obfuscation effects across devices, and minimizing communication overhead, all while maintaining strong security guarantees. Future work could explore integrating lightweight secure multi-party inference protocols or developing partition-aware obfuscation strategies tailored to distributed edge environments.

C.17 Fisher Information Proof

Notation and setup. Let $x \sim \mathcal{D}$ denote inputs and consider a conditional model $p(y \mid x; W)$. We perturb parameters W to $W + \Delta W$ with $\|\Delta W\|$ small. Denote the perturbed conditional output distribution by $p_{W+\Delta W}(z \mid x)$ and its marginal by $p_{W+\Delta W}(z) = \int p_{W+\Delta W}(z \mid x)p(x)\,dx$. We use $g_{L_t}(x) := \nabla_W \log p(L_t \mid x; W)$ and the target-class score $s(x; W) := \log p(L_t \mid x; W)$.

Assumption 1 (Smoothness). The mapping $W \mapsto p(y \mid x; W)$ is twice continuously differentiable for each x.

Assumption 2 (Local Validity). The perturbation ΔW is sufficiently small so that Taylor expansions are valid and higher-order terms are negligible.

Assumption 3 (Regularity). Score functions have bounded second moments and satisfy standard regularity conditions ensuring the interchange of expectation and differentiation (thereby ensuring the Fisher information is well-defined).

[Mutual information identity] For any joint distribution $p(x, z)$,

$$I(X; Z) = \mathbb{E}_{x\sim\mathcal{D}}\big[D_{\mathrm{KL}}\big(p(z \mid x) \,\big\|\, p(z)\big)\big].$$

[Local KL expansion; conditional Fisher] Under (A1)–(A3), for small ΔW,

$$D_{\mathrm{KL}}\big(p(\cdot \mid x; W) \,\big\|\, p(\cdot \mid x; W + \Delta W)\big) = \tfrac{1}{2}\,\Delta W^\top F(x; W)\,\Delta W + o(\|\Delta W\|^2),$$

where

$$F(x; W) := \mathbb{E}_{z\sim p(\cdot|x;W)}\big[\nabla_W \log p(z \mid x; W)\,\nabla_W \log p(z \mid x; W)^\top\big].$$

Proof. By Taylor expansion of the log-likelihood and using the score zero-mean property, the first-order term cancels, and the leading term is quadratic in ΔW; standard deviations in asymptotic statistics produce the displayed form. □

[Local quadratic approximation of mutual information] Under (A1)–(A3), for sufficiently small ΔW,

$$I_{W+\Delta W}(X;Z) = I_W(X;Z) + \tfrac{1}{2}\,\mathbb{E}_{x\sim\mathcal{D}}\big[\Delta W^\top F(x;W)\Delta W\big] + o(\|\Delta W\|^2).$$

Proof. By Lemma 1,

$$I_{W+\Delta W}(X;Z) = \mathbb{E}_x\big[D_{\mathrm{KL}}(p(\cdot \mid x; W+\Delta W)\|p_{W+\Delta W}(\cdot))\big].$$

One can expand the integrand around W taking into account that both the conditional $p(\cdot \mid x; W + \Delta W)$ and the marginal $p_{W+\Delta W}(\cdot)$ vary with ΔW. Careful bookkeeping of first- and second-order terms, and using Lemma C.17 for the conditional contribution, yields the stated quadratic term as the dominant second-order contribution. The remainder is $o(\|\Delta W\|^2)$. □

[Optimal local perturbation under a Fisher (KL) budget] Define the population-averaged conditional Fisher $\overline{F} := \mathbb{E}_{x\sim\mathcal{D}}[F(x;W)]$. Consider the constrained problem (quadratic-budget approximation)

$$\max_{\Delta W} \quad \mathbb{E}_x[g_{L_t}(x)]^\top \Delta W \qquad \text{s.t.} \quad \Delta W^\top \overline{F}\,\Delta W \le \varepsilon.$$

If $\overline{F}$ is positive definite, the optimal direction is

$$\Delta W^\star \propto \overline{F}^{-1}\,\mathbb{E}_x[g_{L_t}(x)].$$

Proof. This is a standard linear objective with quadratic constraint problem. The Lagrangian is $L(\Delta W, \lambda) = \mathbb{E}_x[g_{L_t}(x)]^\top \Delta W - \lambda(\Delta W^\top \overline{F}\Delta W - \varepsilon)$. Stationarity yields $\mathbb{E}_x[g_{L_t}(x)] = 2\lambda\overline{F}\Delta W$. For $\lambda > 0$ and invertible $\overline{F}$, the result follows. □

Remarks on Metric Consistency. It is crucial to distinguish between metrics for different objectives. For quantifying distributional shifts (KL divergence), the population-averaged matrix $\overline{F}$ is the theoretically rigorous metric. Using the uncentered approximation $F_{L_t}^{(\text{raw})} = \mathbb{E}[g_{L_t} g_{L_t}^\top]$ is generally inconsistent with the KL expansion unless specific assumptions hold. In contrast, if the optimization objective is to minimize the sample-wise variance of the target score, the correct quadratic cost is $\Delta W^\top \tilde{F}_{L_t} \Delta W$. Here, $\tilde{F}_{L_t}$represents the centered class-covariance, defined as $\mathbb{E}_x[(g_{L_t}(x) - \bar{g})(g_{L_t}(x) - \bar{g})^\top]$ with $\bar{g} = \mathbb{E}_x[g_{L_t}(x)]$.

Feasibility and practical implementation.

1. For a finite representative dataset $\{x_i\}_{i=1}^m$, enforcing $s(x_i; W + \Delta W) = c$ for all i under the first-order model reduces to a linear system $G\Delta W = b$ with $G_{i,:} = g_{L_t}(x_i)^\top$. If G has full row rank and the parameter dimension p is large, a solution exists (minimum-norm solution $G^+ b$).
2. For population-level exact independence $p_{W+\Delta W}(z \mid x) = p_{W+\Delta W}(z)$ for all x is generically impossible with finite-dimensional ΔW; thus one aims at minimizing distributional proxies (variance, mutual information, empirical KL) instead of exact equality.
3. In practice, $\overline{F}$ and $\mathbb{E}_x[g_{L_t}(x)]$ are replaced by empirical estimates and F^{-1} by approximations.

Targeted Fisher for Obfuscation To steer the model toward a target label L_t and reduce input dependence, we define the gradient-based measure:

$$F_{L_t} = \mathbb{E}\Big[\Big(\frac{\partial \mathcal{L}(x,W)}{\partial W}\Big)^2 \Big| y = L_t\Big]. \tag{7}$$

This is a non-standard, heuristic Fisher matrix that captures which weights most strongly influence the output toward L_t. Selecting the top weights according to F_{L_t} ensures that perturbations are applied where they are most effective in controlling the output.

Perturbation via Gradient Update The perturbation is applied along the gradient of the target loss:

$$W \leftarrow W + \eta \cdot \nabla_W \mathcal{L}(x, W), \tag{8}$$

where η is a scale factor. By first-order Taylor expansion:

$$s(x; W + \Delta W) \approx s(x; W) + \nabla_W s(x; W)^\top \Delta W, \tag{9}$$

the perturbation increases the target class score while reducing output variance across inputs, approximately decreasing mutual information.

Rationale and Limitations This strategy is justified based on three key points:

1. **Fisher-guided selection:** Perturbing weights with high F_{L_t} effectively targets the most sensitive parameters that control the output, consistent with information-theoretic intuition.
2. **Gradient alignment:** Applying $\Delta W \propto \nabla_W \mathcal{L}(x, W)$ aligns the perturbation with the direction that maximally increases the target score, which locally reduces output variance across x.
3. **Approximate input-independence:** While exact input-independence cannot be guaranteed (because different x have different gradients and the model is nonlinear), iterative or multi-sample perturbations can significantly reduce the output's sensitivity to inputs, decreasing mutual information in expectation.

Therefore, the perturbation strategy is theoretically justified as an *approximate mutual information minimization* scheme guided by Fisher information.

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: We explicitly list our main claims and contributions at the end of Section 1 (Introduction). The theoretical design of the proposed framework is detailed in Section 5.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations of FILOsofer are discussed in Appendices C.9 and C.11–C.13, including larger query budgets, REE and TEE interface visibility, side-channel threats, and generalization beyond image classification. Future extensions are discussed in Appendix C.16.

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: Section 5.1 and Appendix C.17 provide the information-theoretic motivation and local Fisher-based analysis.

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: The implementation of FILOsofer is available at `https://github.com/ssysarch/FILOsofer`. Sections 5–7 and Appendix C describe the methodology, experimental configurations, evaluation protocols, and implementation details.

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: The implementation of FILOsofer, including training and evaluation scripts and configuration files, is available at `https://github.com/ssysarch/FILOsofer`.

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) necessary to understand the results?

   Answer: [Yes]

   Justification: Details already provided in experiments part.

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: Details already provided in experiments part.

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: Section 6.3 reports memory and Fisher preparation costs, while Section 6.4 describes deployment on the Jetson Orin Nano.

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: All code and evaluations are intended for academic and scientific purposes, promoting reproducibility and responsible research.

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

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: The name of the license (CC-BY 4.0) is included for each asset.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

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: [N/A]

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

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: [N/A]

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

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 rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We provided the use of language model in Appendix A.1.