LoRO: Real-Time on-Device Secure Inference for LLMs via TEE-Based Low Rank Obfuscation

Gaojian Xiong, Yu Sun,* Jianhua Liu, Jian Cui, Jianwei Liu
School of Cyber Science and Technology, Beihang University, China
{xionggaojian, sunyv, jason_hua, cuijianw, liujianwei}@buaa.edu.cn

Abstract

While Large Language Models (LLMs) have gained remarkable success, they are consistently at risk of being stolen when deployed on untrusted edge devices. As a solution, TEE-based secure inference has been proposed to protect valuable model property. However, we identify a statistical vulnerability in existing protection methods, and furtherly compromise their security guarantees by proposed Model Stealing Attack with Prior. To eliminate this vulnerability, LoRO is presented in this paper, which leverages dense mask to completely obfuscate parameters. LoRO includes two innovations: (1) Low Rank Mask, which uses low-rank factors to generate dense masks efficiently. The computing complexity in TEE is hence reduced by an exponential amount to achieve inference speed up, while providing robust model confidentiality. (2) Factors Multiplexing, which reuses several cornerstone factors to generate masks for all layers. Compared to one-mask-per-layer, the secure memory requirement is reduced from GB-level to tens of MB, hence avoiding the hundred-fold latency introduced by secure memory paging. Experimental results indicate that LoRO achieve a $0.94\times$ Model Stealing (MS) accuracy, while SOTA methods presents $3.37\times$ at least. The averaged inference latency of LoRO is only $1.49\times$, compared to the $112\times$ of TEE-shielded inference. Moreover, LoRO results no accuracy loss, and requires no re-training and structure modification. LoRO can solve the concerns regarding model thefts on edge devices in an efficient and secure manner, facilitating the wide edge application of LLMs.

1 Introduction

Large Language Models have demonstrated considerable ability in various domains [1]. However, the predominant cloud-based deployment exposes critical limitations: users have to tolerate additional network latency and upload their private data. In domains requiring real-time responsiveness and private data protection, e.g. autonomous driving [2] and personal smart agent [3], there is growing demand to deploy LLMs on user's edge devices with accelerators [4, 5]. Alarmingly, on-device models are usually white-box vulnerable to be stolen [6, 7]. Since LLMs are extremely expensive to train, it is expected that *users can benefit from edge inference in an efficient but black-box manner*, i.e. enjoy high-quality and fast inference with no knowledge about private parameters.

Trusted Execution Environment (TEE)-based secure inference [8, 9, 10, 11, 12, 13, 14] have been proposed to safeguard valuable on-device models against theft. Advanced TEE-based methods obfuscate and offload model parameters to Rich Execution Environment (REE) for acceleration using GPUs. The de-obfuscation keys are shielded in TEE to restore the inference results accurately. Compared to cryptograph-based secure inference that relies on Homomorphic Encryption [15] or Multi-Party Computation [16], TEE-based methods offer more practical solutions due to their real-time performance [9, 11, 14] and elimination of network bandwidth occupation.

*Corresponding author.

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

Table 1: Model Stealing Accuracy (↓) against proposed LoRO and SOTA TEE-based secure inference methods. The averaged accuracy of each defense relative to black-box is reported in the last row. The best performance is in green, and the worst is in red.

Model	Model Size	Dataset	No-Shield	TLG[14]	ShadowNet[11]	Magnitude[9]	SOTER[10]	Ours	Black-Box
ViT	Base	Food101	81.57%	81.51%	71.90%	80.05%	75.52%	16.05%	10.28%
		CIFAR100	89.86%	89.67%	79.27%	87.28%	88.30%	15.03%	14.57%
RoBERTa	Base	SQuAD	81.84%	47.44%	34.04%	50.32%	45.51%	6.35%	7.01%
		MRPC	87.99%	69.39%	70.01%	86.88%	82.06%	31.61%	33.90%
		SST-2	93.58%	90.86%	87.78%	91.42%	93.20%	73.43%	69.45%
		MNLI	85.33%	84.25%	82.84%	79.43%	84.00%	45.18%	44.02%
BART	Large	SQuAD	83.15%	49.21%	37.32%	49.84%	42.79%	10.50%	7.36%
		MRPC	87.01%	75.93%	68.89%	79.96%	82.11%	31.61%	31.84%
		SST-2	95.30%	90.90%	88.53%	91.05%	94.57%	67.99%	68.50%
		MNLI	88.15%	86.05%	79.50%	81.62%	85.49%	49.90%	49.96%
Qwen2	1.5B	SQuAD	68.45%	29.33%	21.22%	26.86%	24.05%	5.07%	11.20%
	3B	GSM8K	70.96%	40.13%	28.27%	34.15%	32.83%	1.70%	5.85%
	7B	Spider	17.99%	13.55%	10.15%	10.34%	9.69%	2.32%	2.50%
LLaMA3	1.5B	SQuAD	59.20%	30.62%	23.53%	28.72%	26.80%	7.90%	9.73%
	3B	GSM8K	53.12%	14.57%	12.31%	11.68%	12.47%	1.56%	2.31%
	8B	Spider	34.62%	12.97%	10.70%	9.15%	11.94%	3.90%	3.82%
	Average		7.03×	4.12×	3.37×	3.85×	3.75×	0.94×	1.00×

Nevertheless, we identify a fundamental statistical vulnerability in the state-of-the-art protection methods, and undermine their security guarantees. As shown in Figure 2, since LLMs typically undergo retraining from public available pretrained models, there remains a statistical similarity in parameters or intermediate results [13]. Regretfully, current protection schemes fail to effectively conceal this inherent correlation. Based on this vulnerability, we propose a Model Stealing attack with Prior knowledge (MSP), which enables adversary to approximate the de-obfuscate keys using public prior knowledge, and recover private parameters through minimal fine-tuning. As demonstrated in Table 1, our MSP attack can achieve a high accuracy near original model, indicating that LLMs protected by existing methods could still be stolen. The details of MSP is introduced in Section 4.

To address aforementioned problem, we present LoRO (Low-Rank-Obfuscation), an efficient secure inference framework. To eliminate statistical vulnerability, dense masks are leveraged to completely obfuscate private parameters. But two challenges are raised by dense mask: *(C1) high computing complexity in TEE* and *(C2) GB-level secure memory requirement*, which can cause unacceptable hundred-fold latency. To solve (C1), *Low Rank Mask* is designed to reduce the computing complexity significantly. We employ low-rank factors to randomly generate dense masks, which reduces complexity from $\mathrm{O}(n^3)$ of full matrix multiplication to $\mathrm{O}(n^2)$, enabling real-time inference without compromising security. To solve (C2), *Factor Multiplexing* is designed to optimize the secure memory requirement. Factor Multiplexing generates dense masks through efficient random linear combinations of several cornerstone factors. This optimizes one-mask-per-layer to only several cornerstone factors. For 7B models, the secure memory can be reduced from 1.02 GB to only 26 MB.

Experiments are conducted on Intel SGX and Arm TrustZone platforms to evaluate proposed LoRO, focusing on three aspects: (1) For model confidentiality, LoRO effectively defends against Model Stealing (MS) attacks with near black-box performance (0.94×), outperforming existing methods that exhibit 3.37× at least. (2) For inference efficiency, LoRO's lightweight design maintains only a 1.49× inference latency on average (compared to REE inference), achieving significant improvement over existing approaches the reaches 112× at least. (3) For model accuracy, LoRO introduce zero accuracy loss. Moreover, no training effort and architecture modification is required.

Proposed LoRO effectively protect edge-deployed LLMs as black-box model, while providing real-time and accurate inference service. Moreover, LoRO is plug-and-play without modifying the model structure or retraining. We believe that LoRO can solve the concerns regarding model thefts on edge devices, and facilitate the wide application of LLMs.

The code for implementing LoRO is available in https://github.com/D1aoBoomm/LoRO. Our contributions could be summarized as follows:

- We reveal a statistical vulnerability in existing TEE-based secure inference methods. Based on this, we propose Model Stealing attack with prior, which empirically demonstrates the SOTA methods inadequately deliver their security guarantee when protecting LLMs.

Table 2: Representative methods of TEE-based secure inference compatible with LLMs. ○ means no access to REE, ◐ means only part of parameters is accelerated by REE, and ● means full acceleration.

Category	Method	REE Acceleration	No Retraining Requirement	No Architecture Modification	Resistance to MSP attacks	Obfuscation Scheme
Full Shielding	MLCapsule	○	✔	✔	✔	None
	Penetralium	○	✔	✔	✔	None
Partial Shielding	AegisDNN	◐	✔	✔	✘	None
	TEESlice	◐	✘	✘	✔	None
De-Obfuscation Shielding	Magnitude	●	✔	✔	✘	Additive
	SOTER	◐	✔	✔	✘	Multiplicative
	ShadowNet	●	✔	✔	✘	Add & Mul
	NNSplitter	●	✘	✔	✘	Additive
	TLG	●	✔	✔	✘	Permutation
	LoRO	●	✔	✔	✔	Additive

- We present LoRO, a TEE-based efficient secure inference framework, which consists of two components: (1) Low Rank Mask to shield valuable LLMs from thefts, while providing real-time and accurate inference performance on edge. (2) Factor Multiplexing to eliminate the intensive secure memory requirement of protecting LLMs, which avoids the hundred-fold latency introduced by frequent secure memory paging.
- Experiments on both Intel SGX and ARM TrustZone demonstrate that MS attack accuracy is reduced to black-box level ($0.94\times$) from existing $3.37\times$. Moreover, LoRO introduces only $1.49\times$ inference latency, compared to the $112\times$ of TEE-shielded inference. Notably, LoRO introduces no accuracy loss, and requires no re-training or architecture modification.

2 Related Work

2.1 Trusted Execution Environment

Trusted Execution Environments (TEEs) employ memory isolation to ensure the confidentiality and integrity of application inside. Prominent edge TEE implementations include Intel SGX [17] and Arm TrustZone [18]. However, edge TEEs are originally designed for small critical tasks, e.g. key exchange. They typically equip with limited computational resources and secure memory capacities restricted to 128 MB. Deploying LLMs within TEEs induces excessive secure memory paging, which necessitates additional cryptographic operations to maintain data security. Consequently, inference latency could escalate by hundred-fold [19, 20, 21], which is unacceptable to users.

2.2 TEE-based Secure Inference

TEE-Based Secure Inference aims to protect edge-deployed valuable models from being stolen. Advanced methods leverage TEE to safeguard critical neural network components, e.g. selective parameter subsets and de-obfuscation keys. The computation-intensive major parameters [11, 22] are offloaded to REE, enabling access to hardware accelerators (GPU/NPUs) for accelerations. Without the secret parts in TEE, adversary can only access the model with performance degraded to nearly random guessing. Compared to cryptographic-based approaches, TEE-based solutions demonstrate superior practicality by reducing the inference latency and eliminating the network occupation. As shown in Table 2, existing methods can be categorized as follows:

Full shielding. Some research focus on shielding the whole inference process in TEE. As a typical method, MLCapsule [23] decrypts model layer-by-layer in TEE to avoid secure memory paging. Similarly, Penetralium [20] breaks down the models in parts by a novel adaptive decomposition algorithm, achieving efficient memory allocation. Such methods protect all parameters in TEE, preventing any possible privacy leakage. However, the absence of REE acceleration would result in hundred-fold inference latency, rendering they impractical for real-time tasks.

Partial shielding. To facilitate REE acceleration, several studies [24, 25, 26] have proposed partial model shielding strategies. For example, AegisDNN [27] designs a dynamic programming algorithm

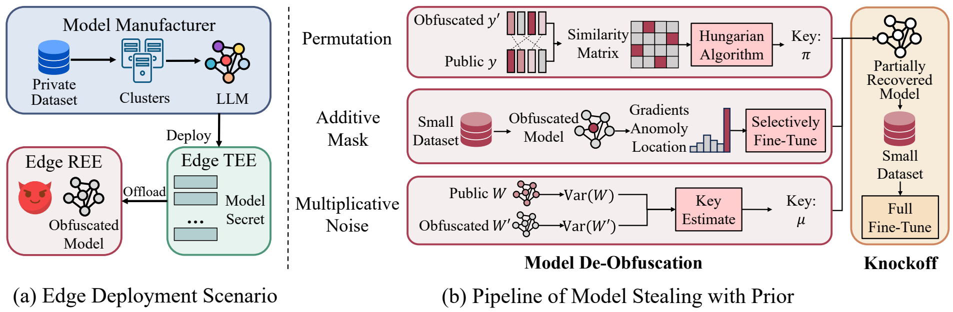


Figure 1: (a) Edge Deployment Scenario (b) Pipeline of proposed Model Stealing attack with Prior.

to identify and protect only the most critical layers. Likewise, TEESlice [13] suggests to freeze the backbone, confining all privacy in shielded model slices. These approaches achieve a security-efficiency balance in small models, but face severe latency delay in LLMs.

De-obfuscation shielding. Recent studies have increasingly focused on parameter obfuscation methods. The parameters are obfuscated and offloaded to REE for acceleration [12, 28]. Subsequently, intermediate results are de-obfuscated in TEE using shielded keys. According to obfuscation mechanism, current methods can be classified into three categories: permutation [14], additive mask [9, 11, 12] and multiplicative noise [10, 11]. Since the major computation (commonly over 95% [11]) is offloaded to REE, obfuscation-based solutions provide remarkable inference efficiency.

We introduce the protection mechanism of each methods, and analyze their vulnerability in Section 4.

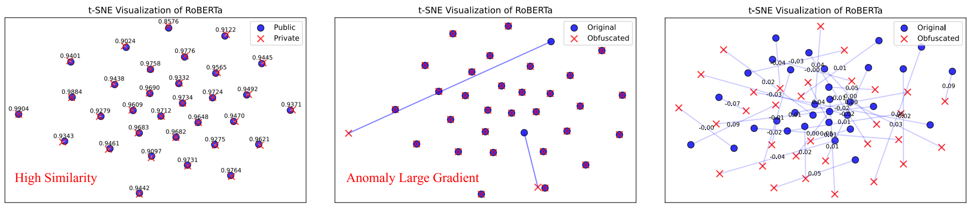
3 Threat Model

Edge environment. As shown in Figure 1 (a), we consider an edge environment, where edge devices are equipped with TEE and accelerators. The TEE is considered absolutely secure, preventing any data leakage and integrity compromising. Side channel attacks [29, 30], which could cause privacy leakage in TEE, is not considered in this study.

Adversary. The adversaries try to achieve a surrogate model that exhibits similar performance to the edge-deployed LLMs. These adversaries possess powerful capabilities, controlling the entire environment external to the TEE, including the operating system and hardware. They are also well-versed in artificial intelligence. To enhance MS attacks, adversaries can deduce the complete model architecture from information available in the REE and utilize publicly available models from the Internet. A small dataset (about 10% size of training set) is held by adversaries to launch attacks.

Design goal. The goal of model manufacturer is to protect the edge-deployed LLMs from thefts, while providing fast and accurate inference service to users. To achieve this goal, the following objectives should be met simultaneously:

- **Model Confidentiality**: Without authorization to TEE, adversaries cannot obtain the parameters of deployed LLMs through MS attacks or analysis attacks. The ideal situation is that models are protected as black-box.
- **Inference Efficiency**: Inference latency of deployed models should be reduced as possible. The ideal situation is that inference latency is comparable to that in REE.
- **Model Accuracy**: The model accuracy should be maintained to provide high-quality service. The ideal situation is that no loss in accuracy is introduced.



(a) TLG. (b) Magnitude. (c) Proposed LoRO.

Figure 2: T-SNE visualization of parameters on representative methods. Each point represents the a column vector from weight matrix. Number in (a),(c) indicates the cosine similarities , and line in (b),(c) means the shift distance of obfuscated parameters.

4 Model Stealing Attack with Prior

In this section, we introduce how MSP is designed in detail. For each methods category, we first analyze the potential risk, and then introduce our attack pipeline. Finally, we experimentally evaluate our attack performance. The overall pipeline of MSP attack is depicted in Figure 1 (b).

4.1 Permutation Methods.

Risk analysis. This approach leverages permutation to obfuscate model parameters [14]. Given a permutation matrix π, in which elements are $\in \{0, 1\}$ and $\pi\pi^T = I$, the input x and parameters W of linear layer are obfuscated as follows: $x' = x\pi, W' = \pi^T W$. Although the parameters are obfuscated, the output y is still accurate, since $y = x'W' = x\pi\pi^T W = xW$.

However, a significant drawback of this approach lies in *the shared obfuscation key between intermediate results and model parameters*. Once adversaries successfully restore the intermediate results, they can directly recover the plain-text model parameters. Furthermore, permutation-based obfuscation fails to conceal the statistical distribution of columns in the intermediate results. As shown in Figure 2 (a), adversaries can exploit the high distribution similarity between private and public models to estimate π, and then de-obfuscate model without training effort.

Attack pipeline. Based on the risk analyzed above, we first estimate the permutation key based on the intermediate results distribution. Specifically, *we formulate the attack as an optimal matching problem for maximal similarity of intermediate results columns*. First, a batch of input x_r is randomly chosen, and sent to both public pretrained model layer W_p and obfuscated layer W' to get batched output $y_p = x_r W_p$ and $y' = x_r W'$. Given the $columns$ of intermediate results, a similarity matrix S is computed over the columns of y and y' as follows:

$$S = [s_{ij} = \text{cosine\_similarity}(y_i, y'_j) \quad | \quad 0 \le i, j \le columns - 1], \tag{1}$$

where y_i and y'_i represent the i-th column of y and y' respectively, which is averaged across the batch. Then, based on S, we leverage Hungarian Algorithm [31] to resolve the optimal column matching. The permutation key π can be nearly perfectly recovered, and utilized to de-obfuscate parameters. Finally, we perform a fine-tuning using the limited dataset, to correct potential minor mismatches.

4.2 Additive Mask Methods.

Risk analysis. These methods apply additive mask r on parameters as: $W' = W + r$, and apply One-Time-Pad (OTP) o on input: $x' = x + o$. The results y is de-obfuscated in TEE as:

$$y = \underbrace{x'W'}_{\text{REE}} - \underbrace{xr}_{\text{TEE}} - \underbrace{(oW + or)}_{\text{Pre-Computed}} = xW, \tag{2}$$

where xW' is accelerated in REE, xr is securely computed in TEE, and $oW + or$ is pre-computed and stored. Notably, xr holds the same heavyweight complexity with xW, which results in unacceptable latency in TEE. To overcome this, existing methods leverage sparse mask r_s to downgrade the complexity. For instance, Magnitude [9] only obfuscate 1% largest weights, and NNSplitter [12] selects extremely sparse parameters to protect by reinforcement learning.

However, under the constraint of mask sparsity, only limited number of parameters are obfuscated. To reduce the model accuracy to random guessing, the magnitude of changes in the obfuscated parameters is significantly large [28, 32]. Based on this insight, we observe a *gradient anomaly phenomenon*: compared to the normal counterparts in the same layer, obfuscated parameters exhibit substantially larger gradient values on the training data. As shown in Figure 2 (b), obfuscated parameters can be easily located, and then selectively fine-tuned for recovery.

Attack pipeline. Based on the identified gradient anomaly phenomenon, we initially localize the obfuscated parameters. By iterating through the attacker's small dataset, we identify the top 5% of parameters with the largest averaged gradient as obfuscated parameters. Subsequently, these identified parameters are set to be trainable, while the remaining parameters are frozen, enabling selective fine-tuning. Empirical observations reveal that the localization accuracy is not entirely precise, as a small subset of obfuscated parameters still exhibits minimal variation. Consequently, a few additional rounds of full fine-tuning are ultimately required.

4.3 Multiplicative Noise.

Risk analysis. These methods apply multiplicative factor to obfuscate parameters. For instance, SOTER [10] obfuscate a layer using one factor μ: $W' = \mu W$, and de-obfuscate the results as: $y = \mu^{-1} x W' = xW$. Similarly, ShadowNet [11] applies one factor for each columns.

However, as indicated in [13], multiplicative noise struggles to conceal the statistical distribution of parameters. The value of applied factor have significant impact on the variance of parameters. As a result, the adversary can leverage the variance of public models to estimate the applied factor, and then de-obfuscate the parameters.

Attack pipeline. Given the variance $\text{var}(W)$ of public models, and the variance $\text{var}(W')$, we follow [13] tofirst estimate multiplicative noise as: $\mu' = \sqrt{\text{var}(W')/\text{var}(W)}$. Then we de-obfuscate parameters using μ', and fine-tune the model with adversary's small dataset.

4.4 Attack Results

As demonstrated in Table 1, we conduct experimental evaluations regarding the attack performance of proposed MSP. To estimate the strongest adversary, the highest result in five individual experiments are reported. Notably, the MS accuracy against existing methods reaches an average of $3.37\times$ to $4.12\times$, even comparable to the original model in some scenarios, indicating that current approaches fail to safeguard high-value LLMs from thefts.

5 TEE-Based Low Rank Obfuscation

In this section, we introduce LoRO in detail. As shown in Figure 3, it consists of two components: Factor Multiplexing to reuse the cornerstone factors in deployment stage, and Low Rank Mask to safeguard the valuable LLM from thefts in inference stage. These two components are described respectively as follows, and we provide persuade code in Appendix E.

5.1 Low Rank Mask

Efficiency-Confidentiality dilemma. Edge TEEs face significant resource constraints, particularly when protecting high-complexity computations, which can introduce a hundred-fold increase in inference latency [8, 20]. Consequently, the core of TEE-based secure inference lies in maximizing inference efficiency while maintaining model confidentiality. On the one hand, full shielding methods can provide robust security, but the unacceptable latency renders them impractical. On the other hand, while existing methods offer substantial efficiency improvements, they fail to fully obscure the statistical distribution (as detailed in Section 4). This compromise from confidentiality to efficiency significantly increase the risk of model theft, leading them vulnerable to proposed MSP attack. Hence, it's challenge to achieve robust confidentiality in an efficient way.

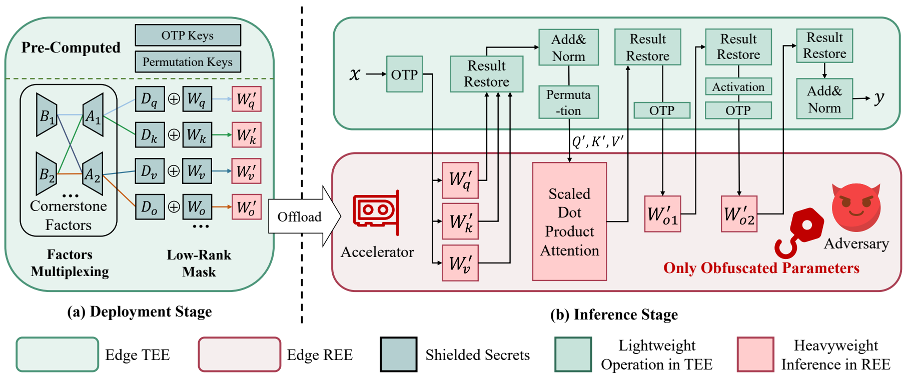


Figure 3: Illustration of proposed LoRO, which incorporates two stages: (a) Deployment Stage: to pre-computing keys and obfuscate parameters in TEE. (b) Inference Stage: inference with obfuscated parameters in REE and restore the results in TEE.

Our solution. To address the dilemma introduced above, our solution is to obfuscate parameters using dense additive mask, which can fully obscure the statistical distribution and provide high security [11, 13]. As shown in Figure 3, we focus on protecting parameters of linear layer. LoRO is compatible to popular LLMs since they are almost based linear layers. Given the dense mask D, the parameter W is obfuscated as $W' = W + D$. The inference results y is described as follows:

$$y = \underbrace{xW'}_{\text{REE}} - \underbrace{xD}_{\text{TEE}} = xW. \tag{3}$$

However, one main limitation is that the dense mask is too heavyweight to be shielded. Given the shape $[n \times n]$ of W, dense mask holds a high computation complexity of $\mathrm{O}(n^3)$. This is same with full shielding methods, which could result in an unacceptable latency in TEE. To tackle this problem and achieve real-time efficiency, *our key insight is to generate dense mask using low-rank factors*. Given two low-rank factors $B^{n\times d}$, $A^{d\times n}$ and generated dense mask $D = BA$, the inference results are re-formulated as follows:

$$y = \underbrace{xW'}_{\text{REE}} - \underbrace{xBA}_{\text{TEE}} = xW. \tag{4}$$

In conclusion, proposed methods address the efficiency-confidentiality dilemma in following aspects. For efficiency, the complexity in TEE is reduced from complex $\mathrm{O}(n^3)$ to $\mathrm{O}(n^2 \times d) = \mathrm{O}(n^2)$, where $d \ll n$. Hence, inference efficiency is significantly improved. For confidentiality, each elements in W are obfuscated randomly. As shown in Figure 2 (c), since statistical distribution are fully obscured, adversaries cannot access W without obtaining shielded D.

Intermediate results protection. To protect masks from analysis attacks, the intermediate results should also be protected. For the input x and of a linear layer, OTP o is employed as: $x' = x + o$. Same to Equation 2, the applied noise can be accurately removed. Specifically, for the scaled dot production of attention module [33], since OTP is not applicable [34] (as detailed in Appendix D, we leverage permutation matrix π to protect intermediate results as follows:

$$Q' = \pi Q, K' = \pi K, V' = \pi V, \quad y' = \text{softmax}\left(\frac{Q'K'^T}{\sqrt{k}}\right)V' = \pi\text{softmax}\left(\frac{QK^T}{\sqrt{k}}\right)V, \tag{5}$$

where k is the hyper-parameter that has been set in model training stage. To accommodate model variants, the potential RoPE [35] and other activation function is computed in TEE. Then the result is restored in TEE as: $y = \pi^T y'$. Notably, since our secret D is independent from π, the risk we discovered in TLG (detailed in Section 4) is avoided. Due to page limitation, more details of design and hyper-parameters can be found in Appendix A.

5.2 Factor Multiplexing

Limited secure memory challenge. Edge TEEs are constrained by limited secure memory, typically capped at 128 MB [17, 36]. However, to shield a pair of low-rank factors per layer for 7B LLMs, the secure memory could reach approximately 1.02 GB. The introduced memory paging issue leads to a unacceptable increase in inference latency and potential security risk [37], severely compromising the practicality. Hence, a critical challenge lies in efficiently protecting low-rank factors within such restricted secure memory.

Our solution. Factor Multiplexing is proposed to address this challenge. Our insight lies in the random combination of several limited low-rank factors, which enables reusing factors to save secure memory. Instead of preparing a pair of factors per layer, we randomly choose from cornerstone factors and generate dense mask. As shown in Figure 3 (a), given a cornerstone factors set $\{A_1, ..., A_n, B_1, ...B_n\}$, a small set $S \subseteq \{A_1, ..., A_n\} \times \{B_1, ...B_n\}$ is randomly chosen,where the $|S| = m$. Then a mask D is generated as follows:

$$D = \sum_{(A_i, B_j) \in P} \alpha B_j A_i, \tag{6}$$

where α is the random weight. In this way, only the cornerstone factors are required in memory, instead of the heavyweight one mask per layer. In real-time scenario, m is set to two for best efficiency, and the low computation complexity is maintained. Moreover, Factor Multiplexing enables updating the obfuscated parameters using new masks periodically. The secure memory requirement of one-mask-per-layer and Factor Multiplexing is reported in Appendix B.

6 Experiments

In this section, LoRO is evaluated experimentally. According to design goal (in Section 3), we aim to answer the following questions:

RQ1: How is the defense effectiveness of LoRO? (in Section 6.2)
RQ2: What is the inference speed of LoRO compared to existing methods? (in Section 6.3)
RQ3: What is the model accuracy of LoRO compared to original models? (in Section 6.4)

6.1 Experiment Setup.

Benchmarks. Proposed LoRO is evaluated on several famous benchmarks, including SQuAD (reading comprehension) [38], GSM8k (mathematics) [39], Spider (code generation) [40] and representative standard GLUE [41]. The accuracy is reported as metric. For generation tasks, only exact matched answer is considered correct. We also evaluate LoRO on Computer Vision tasks, including CIFAR100 [42] and Food101 [43].

Models. Representative models of various scales and structure are selected, including RoBERTa [44, 45], BART [46], ViT [47] Qwen [48] and LLaMA [49, 50].

Hardware devices. Experiments regarding model confidentiality and accuracy are conducted on a server equipped with two NVIDIA RTX 4090 GPUs. Inference efficiency are evaluated on two TEE platforms. For TrustZone, NVIDIA Jetson Orin NX board equipped with 6-core ARM Cortex-A78AE CPU and 1024-core NVIDIA Ampere architecture GPU is employed, and OP-TEE [51] is leveraged as TEE OS. For Intel SGX, a laptop equipped with Intel Core I9-10885H CPU and Quadro T2000 GPU is adopted, and Gramine-SGX [52] is the basic TEE OS.

Due to page limitation, more details and experiments can be found in Appendix A and B.

6.2 Model Confidentiality

As shown in Table 1, we report the MSP accuracy against LoRO and various defense methods. The significantly high MS accuracy (3.75× to 4.12×) demonstrate the vulnerability of current defense

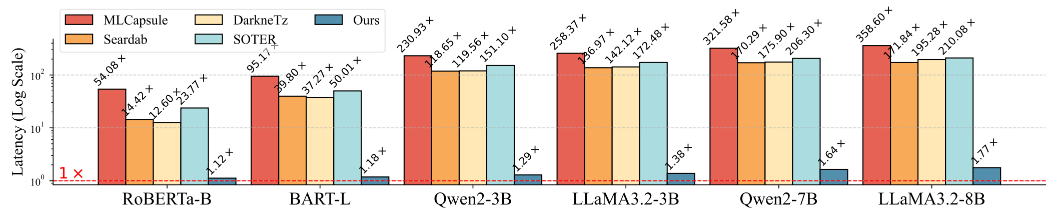


Figure 4: Inference latency of various methods in SGX environment. All results are normalized to REE inference latency (red line, $1\times$). We depict in $\log_{10}$ scale for better visualization.

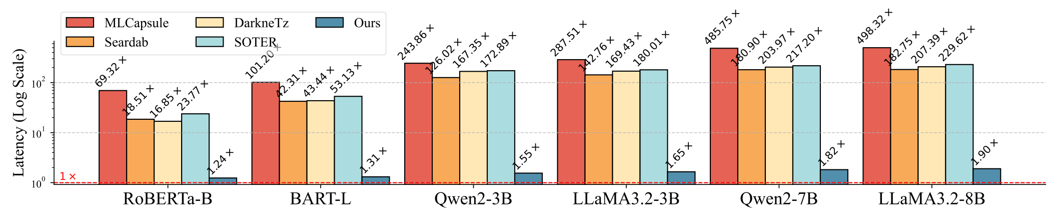


Figure 5: Inference latency of various methods on TrustZone environment. All results are normalized to REE inference latency (red line, $1\times$). We depict in $\log_{10}$ scale for better visualization.

approaches to our proposed MSP attack. Notably, adversary can achieve near-complete recovery in several scenarios. In contrast, LoRO maintains a robust defense performance of only 0.94×, comparable to the 1.00× baseline in black-box settings. These results indicate that LoRO provides effective protection on LLMs against MS attempts.

Answer to RQ1: Proposed LoRO effectively downgrades private LLMs to black-box, providing robust security to valuable intellectual property protection.

6.3 Inference Efficiency

Experiments are conducted on popular TEEs to evaluate the inference efficiency of LoRO, including Intel SGX and ARM TrustZone. As depicted in Figure 4 and Figure 5, proposed LoRO introduces only $1.49\times$ latency on average, compared to the $112.10\times$ to $250.39\times$ of existing methods. This demonstrates that *our methods provides real-time inference and is practical for real-world application*. Notably, as shown in table 6, with the raise of model scale, the computing complexity in TEE is significantly increased due to larger intermediate size, resulting a higher latency.

Answer to RQ2: Compared to the averaged $112.10\times$ of existing methods, proposed LoRO introduces only $1.49\times$ latency, achieving real-time inference efficiency.

6.4 Model Accuracy

As shown in Table 3, we report the model accuracy of original model, proposed LoRO method and the obfuscated model exposed in REE. Only a 0.01% drop is observed in BART on MRPC, which is caused by float precision error. It is demonstrated that the high accuracy of original model is maintained. Moreover, the obfuscated model is completely downgraded to random guessing level.

Answer to RQ3: Proposed LoRO introduces no accuracy loss, maintaining the high model accuracy. The obfuscated model exposed in REE is downgraded to unusable.

Table 3: Model accuracy of original model (Ori.), LoRO and obfuscated model in REE (Obf.).

		SQuAD	MRPC	SST-2	MNLI	GSM8K	Spider
RoBERTa	Ori.	81.84%	87.99%	93.58%	85.33%	-	-
	LoRO (↑)	81.84%	87.99%	93.58%	85.33%	-	-
	Obf. (↓)	0.64%	31.61%	50.91%	35.45%	-	-
BART	Ori.	83.15%	87.01%	95.30%	88.15%	-	-
	LoRO (↑)	83.15%	87.00%	95.30%	88.15%	-	-
	Obf. (↓)	0.76%	31.60%	49.08%	35.45%	-	-
Qwen2	Ori.	48.45%	-	-	-	70.96%	17.99%
	LoRO (↑)	48.45%	-	-	-	70.96%	17.99%
	Obf. (↓)	0.00%	-	-	-	0.00%	0.00%
LLaMA3	Ori.	49.20%	-	-	-	53.12%	34.62%
	LoRO (↑)	49.20%	-	-	-	53.12%	34.62%
	Obf. (↓)	0.00%	-	-	-	0.00%	0.00%

7 Discussion

Side channel Attacks. Side channel attacks [29, 30] can cause leakage from TEE potentially. Fortunately, our LoRO is compatible with existing advanced defense schemes [53, 54] to mitigate such attacks. Hence, defending against side channel attacks is considered a sole research domain regarding TEE security, and is out of our scope.

Trusted GPU. Some advanced GPU provides Confidential Computing [55, 56]. Nevertheless, the substantial cost of GPUs with confidential computing (CC) capabilities presents a significant barrier to their widespread adoption in edge computing environments. More affordable alternatives, including Intel Software SGX or ARM TrustZone integrated with dedicated AI acceleration hardware, are expected to remain the predominant solution for edge devices in the foreseeable future. Consequently, our investigation into secure inference for edge computing systems does not currently incorporate Trusted GPUs within its scope.

8 Conclusion

In this paper, we first identify a statistical vulnerability in existing TEE-based secure inference methods, and break their security by designed Model Stealing attack with Prior. To protect on-device LLMs, LoRO is proposed, which incorporates two components: (1) Low Rank Mask to completely obfuscate model parameters, while achieving fast inference via lightweight low-rank computing in TEE. Moreover, no accuracy loss is introduced, and no retraining and model modification is required. (2) Factor Multiplexing to significantly reduce the secure memory requirement, avoiding additional latency introduced by secure memory paging. Proposed LoRO downgrade on-device LLMs to black-box level ($0.94\times$ MS accuracy), while providing real-time inference service (only $1.49\times$ latency, compared to the $112\times$ of existing SOTA). LoRO can solve the concern regarding theft on untrusted edge devices, hence greatly facilitating the wide adoption of LLMs.

9 Acknowledgement

We would express our gratitude to all anonymous reviewers and chairs for their valuable dedication. This research is supported by the Natural Science Foundation of Hangzhou (under Grant 2025SZRJJ1400) and the CCF-Phytium Fund (under Grant CCF-Phytium 202306).

References

[1] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

[2] Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. Llm4drive: A survey of large language models for autonomous driving. *arXiv preprint arXiv:2311.01043*, 2023.

[3] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

[4] Tianxiang Tan and Guohong Cao. Deep learning on mobile devices through neural processing units and edge computing. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1209–1218. IEEE, 2022.

[5] Yu Sun, Jianhua Liu, Gaojian Xiong, Qinglin Song, Jianwei Liu, Gang Wang, and Rui Wang. Towards trusted 6g mobile edge computing: A secure batch large language models deployment framework. *IEEE Transactions on Mobile Computing*, 2025.

[6] 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.

[7] Pengcheng Ren, Chaoshun Zuo, Xiaofeng Liu, Wenrui Diao, Qingchuan Zhao, and Shanqing Guo. Demistify: Identifying on-device machine learning models stealing and reuse vulnerabilities in mobile apps. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13, 2024.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] Qinfeng Li, Zhiqiang Shen, Zhenghan Qin, Yangfan Xie, Xuhong Zhang, Tianyu Du, Sheng Cheng, Xun Wang, and Jianwei Yin. Translinkguard: Safeguarding transformer models against model stealing in edge deployment. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3479–3488, 2024.

[15] 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.

[16] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 325–342, 2020.

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

[18] Sandro Pinto and Nuno Santos. Demystifying arm trustzone: A comprehensive survey. *ACM computing surveys (CSUR)*, 51(6):1–36, 2019.

[19] Yuepeng Li, Deze Zeng, Lin Gu, Quan Chen, Song Guo, Albert Zomaya, and Minyi Guo. Efficient and secure deep learning inference in trusted processor enabled edge clouds. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4311–4325, 2022.

[20] Mengda Yang, Wenzhe Yi, Juan Wang, Hongxin Hu, Xiaoyang Xu, and Ziang Li. Penetralium: Privacy-preserving and memory-efficient neural network inference at the edge. *Future Generation Computer Systems*, 156:30–41, 2024.

[21] Yu Sun, Gaojian Xiong, Xiao Liu, and Yan Li. A survey on trusted execution environment based secure inference. *Netinfo Security*, 24(12):1799–1818, 2024.

[22] Yu Sun, Gaojian Xiong, Xianxun Yao, Kailang Ma, and Jian Cui. Gi-pip: Do we require impractical auxiliary dataset for gradient inversion attacks? In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4675–4679. IEEE, 2024.

[23] 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.

[24] Alexander Schlögl and Rainer Böhme. ennclave: Offline inference with model confidentiality. In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, pages 93–104, 2020.

[25] 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.

[26] 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.

[27] 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.

[28] 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*, 2024.

[29] Alexander Nilsson, Pegah Nikbakht Bideh, and Joakim Brorsson. A survey of published attacks on intel sgx. *arXiv preprint arXiv:2006.13598*, 2020.

[30] Jo Van Bulck, Frank Piessens, and Raoul Strackx. Nemesis: Studying microarchitectural timing leaks in rudimentary cpu interrupt logic. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 178–195, 2018.

[31] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[32] Jingtao Li, Adnan Siraj Rakin, Yan Xiong, Liangliang Chang, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. Defending bit-flip attack through dnn weight reconstruction. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[34] Ding Li, Ziqi Zhang, Mengyu Yao, Yifeng Cai, Yao Guo, and Xiangqun Chen. Teeslice: Protecting sensitive neural network models in trusted execution environments when attackers have pre-trained models. *ACM Transactions on Software Engineering and Methodology*, 2024.

[35] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[36] Wei Zheng, Ying Wu, Xiaoxue Wu, Chen Feng, Yulei Sui, Xiapu Luo, and Yajin Zhou. A survey of intel sgx and its applications. *Frontiers of Computer Science*, 15:1–15, 2021.

[37] Yunkai Bai, Peinan Li, Yubiao Huang, Shiwen Wang, Xingbin Wang, Dan Meng, and Rui Hou. Secpaging: Secure enclave paging with hardware-enforced protection against controlled-channel attacks. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, 2024.

[38] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[39] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

[40] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

[41] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[42] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[43] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[44] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.

[45] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[46] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[47] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[48] Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*, 2024.

[49] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

[50] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[51] Linaro. Op-tee trusted os. `https://github.com/OP-TEE/optee_os`, 2024.

[52] Chia-Che Tsai, Donald E Porter, and Mona Vij. {Graphene-SGX}: A practical library {OS} for unmodified applications on {SGX}. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 645–658, 2017.

[53] Ghada Dessouky, Tommaso Frassetto, and Ahmad-Reza Sadeghi. {HybCache}: Hybrid {Side-Channel-Resilient} caches for trusted execution environments. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 451–468, 2020.

[54] Sajin Sasy, Sergey Gorbunov, and Christopher W Fletcher. Zerotrace: Oblivious memory primitives from intel sgx. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018*. The Internet Society, 2018.

[55] Yongqin Wang, Rachit Rajat, Jonghyun Lee, Tingting Tang, and Murali Annavaram. Fastrack: Fast io for secure ml using gpu tees. *arXiv preprint arXiv:2410.15240*, 2024.

[56] Jianwei Zhu, Hang Yin, Peng Deng, Aline Almeida, and Shunfan Zhou. Confidential computing on nvidia hopper gpus: A performance benchmark study. *arXiv preprint arXiv:2409.03992*, 2024.

[57] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. {ML-Doctor}: Holistic risk assessment of inference attacks against machine learning models. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4525–4542, 2022.

[58] 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.

[59] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[60] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

Table 4: Hyper-parameters of knockoff attacks.

	RoBERTa		BART		Qwen2		LLaMA3	
	Epoch	Learning Rate	Epoch	Learning Rate	Epoch	Learning Rate	Epoch	Learning Rate
TLG	1	3e-5	2	3e-5	2	5e-5	2	5e-5
ShadowNet	2	1e-5	3	1e-5	4	1e-5	4	1e-5
SOTER	2	1e-5	3	1e-5	4	1e-5	4	1e-5
Magnitude	3	5e-5	4	1e-5	5	3e-5	5	3e-5

Table 5: Detailed hyper-parameters of LoRO.

	RoBERTa	BART	Qwen2			LLaMA3		
Model Size	Base	Large	1.5B	3B	7B	1.5B	3B	8B
Rank	12	16	24	30	36	24	30	36

A More Details

Attack details. We introduce our attack details here. For each method, the adversary first reason the model structure from exposed model parts in REE, and get public pretrained model from Internet.

Then, the adversary conducts model de-obfuscation (as shown in Figure 1). In this stage, the batch size we leveraged to attack permutation methods is 16, and the inputs are randomly selected from datasets. Notably, we empirically find the batch size can't be too large, since the averaged intermediate results would be very similar and difficult to be matched. For additive noise, we iterate all training data in small dataset to accumulate the gradients. We freeze major parameters, and only 5% largest parameters are fine-tuned for 2 epoches. For multiplicative noise methods, our methods is as same as [8, 13], and no other details need to be clarified. The knockoff setting of each method are shown in Table 4. The ShadowNet is considered as the combination of additive mask and multiplicative noise. We follow [13] to first remove the mask in REE, and then estimate the multiplicative key.

In the final knockoff stage, we develop our code based on the ML-Doctor [57]. Since model have been approximately recovered before, and only 10% of dataset are leveraged to conduct attacks, we only train models with a low learning-rate as 5e-4 in 2 epoches to avoid over-fitting. The results reported in Table 1 is taken from the best of five individual experiments.

LoRO details. The details and hyper-parameters of LoRO is introduced here. All the random noise and low-rank factors are randomly sampled from Gaussian Noise. We set the m as 2 to achieve fast inference speed. According to LoRA [58], if the rank d is able to effectively adapt the model to specific domain, we thought the d is enough to obfuscate models well. The detailed hyper-parameters of our experiments is shown in Table 5.

Implementation optimization. For TrustZone, we mainly conduct two optimizations. First, take inspiration from ShadowNet [11], we design matrix multiplication based on NEON instruction set to accelerate computation in TEE. This achieves about $3\times$ speed up when the dimension of matrix is over 768. Second, to reduce the data transfer time between REE and TEE, we allocate a pinned shared memory between them as data exchange zone. While this may raise inconvenience in coding, but the transfer latency is significantly reduced. Notably, it is necessary to compile the OP-TEE with larger secure memory (at least 128MB) since LLMs are significantly large. For methods like MLCapsule [23], the required memory could reach tens of GB.

Compatibility for small models. Our LoRO is designed to obfuscate the linear parameters (including both linear and convolutional layers). Previous work targeting small models [9, 10, 11, 13, 27] also focuses on protecting linear layers. Hence, proposed LoRO is inherently compatible to small models, where the linear operations major the parameters and inference computing.

Table 6: Latency breakdown of LoRO on Intel SGX.

	RoBERTa-B	BART-L	Qwen2-3B	LLaMA3.2-3B	Qwen2-7B	LLaMA3.2-8B
TEE	45.72%	47.20%	38.71%	29.92%	40.57%	37.82%
REE	37.82%	37.03%	55.62%	56.09%	43.93%	49.20%
Transfer	16.45%	15.76%	15.66%	13.98%	15.50%	12.96%

Table 7: Accuracy of LoRO on MATH.

Model	Category	MATH
Gemma 9B	Ori.	36.63%
Gemma 9B	LoRO (↑)	36.62%
Gemma 9B	Obf. (↓)	0.00%

Table 8: Accuracy of LoRO on ViT.

		Food101	CIFAR100
	Ori.	81.57%	89.86%
ViT	LoRO (↑)	81.57%	89.85%
	Obf. (↓)	0.99%	1.00%

B More Experiments on LoRO

Latency breakdown. As presented in Table 6, we analyze the latency distribution among TEE, REE, and data transfer. For smaller LLMs such as RoBERTa-B and BART-L, the computational efficiency is primarily limited by TEE processing and data transfer latency. However, in larger models like LLaMA and Qwen, the REE latency becomes dominant due to the high dimension full matrix computations. This clearly demonstrates the lightweight advantage achieved by LoRO.

Accuracy of LoRO on ViT. We also test the accuracy of LoRO on ViT, as shown in Table 8. Only a small drop of 0.01% is observed on CIFAR100, which is caused by float error. This demonstrate the high accuracy of LoRO on vision tasks.

Accuracy of LoRO on MATH. LoRO is also evaluated on a challenging dataset MATH [59]. As shown in the Table 7, LoRO maintains near-identical accuracy, with the 0.01% drop attributable to floating-point error. This can be solved in 32-bit models. Crucially, the obfuscated model becomes unusable (0% accuracy), confirming our method's robustness to adversary.

Secure memory requirement. As shown in Table 9, we report the secure memory requirement of preserving secrets in TEE. Compared to one-mask-per-layer paradigm, proposed Factor Multiplexing effectively reduce the secure memory to MB-level compatible to edge TEE. The high latency and potential risk introduced by secure memory page is hence avoided.

More latency comparison. As shown in Table 10, LoRO's latency outperforms most of existing methods. For comprehensive comparison, we only report the de-obfuscation latency of SOTER. While TLG performs similarly to LoRO in inference efficiency, TLG cannot provide black-box security (as demonstrated in Section 6.2). We also notice that MLCapsule suffers critical slowdowns from the secure paging overhead, and Magnitude is slow since the sparse computation in TEE is too heavyweight for large scale LLMs.

C More experiments on MSP

MSP under low auxiliary data. In Section 3, about 10% of training data is assumed to be held by the adversary to launch MSP attacks. The 10% data assumption allows us to design defenses robust against a powerful adversary.

We evaluate the attack performance under low data ratio (1%-10% of SQuAD on Qwen2). As shown in Table 11, our LoRO consistently surpasses black-box performance even at 1% data volume, while existing SOTA TLG fails to provide adequate protection. Even under low data ratio conditions, the proposed MSP poses a serious threat to existing methods.

MSP on CV models. We also evaluate Model Stealing with Prior (MSP) experiments on ViT and ResNet18. For model prior knowledge, we used model trained on CIFAR10. We did not adapt TLG to CNNs. As shown in the table 12, MSP remains effective for CV models like ViT and ResNet18.

Table 9: Secure memory requirement of Factor Multiplexing and naive one-mask-per-layer.

	RoBERTa	BART	Qwen-7B	LLaMA-8B
One-Mask-Per-Layer	232 MB	594 MB	1.02 GB	1.14 GB
Factor Multiplexing	3.84 MB	11.88 MB	26.30 MB	28.28 MB

Table 10: More experiments on latency comparison.

Model	MLCapsule	SOTER	ShadowNet	Magnitude	TLG	LoRO
RoBERTa-B	54.08×	1.34×	2.35×	8.92×	1.10×	1.12×
BART-L	95.17×	1.62×	2.48×	10.33×	1.14×	1.18×
Qwen2-7B	321.58×	2.01×	3.60×	35.84×	1.59×	1.64×
LaMA3.2-8B	358.60×	2.10×	3.75×	37.67×	1.70×	1.77×

D Limitation of OTP on Attention

We analyze why the OTP cannot be applied to secure intermediates of Attention module here. During forward inference, Attention block requires to compute QK^T. When OTP are applied to secure Q and K as $Q' = Q + o_q, K' = K + o_k$, the multiplication can be formulated as follows:

$$Q'K'^T = QK^T + \underline{Qo_k^T + o_qK^T} + o_qo_k^T. \tag{7}$$

The term $o_qo_k^T$ can be pre-computed and removed in a lightweight manner, the term $Qo_k^T + o_qK^T$ must be computed in a real-time manner. Since o_q and o_k shares the same shape of Q and K, applying OTP will introduce unacceptable computation in TEE. Due to the limited computational resources in TEE, this will lead to prohibitive inference latency. This issue is also exists in protecting V.

The TOSEM version of TEESlice [34] suggests that this issue can be solved by using Linear Attention [60]. But the Linear Attention may have a bit influence on model performance, and is still not widely adopted in popular LLMs. In this paper, we propose to use permutation to secure the intermediate results of Attention Block, while the remained parts should be protected by OTP.

E Algorithm Description

For better comprehension, we present the persuade code of deployment stage in Algorithm 1, and the inference stage in Algorithm 2.

Table 11: MSP performance under low auxiliary data.

Dataset Ratio	1%	3%	5%	10%	Average
TLG	14.20%	19.78%	24.95%	29.33%	2.92x
LoRO	4.08%	4.22%	4.38%	5.07%	0.63x
Black-Box	4.37%	6.95%	8.40%	11.20%	1.00x

Table 12: MSP performance on Computer Vision models.

Model	Dataset	TLG	SOTER	Magnitude	ShadowNet	Black-Box
ViT-Base	CIFAR100	89.67%	88.30%	87.28%	79.27%	14.57%
ViT-Base	Food101	81.51%	75.52%	80.05%	71.90%	10.28%
ResNet18	CIFAR100	N/A	67.30%	73.55%	65.42%	14.74%

Algorithm 1: LoRO Deployment Stage.

Input: Original Parameters W
Output: Obfuscated Parameters W'
$Key, Secret, W' \leftarrow [], [], []$;
$B, A \leftarrow RandomNoise()$; ▷ Randomly generate cornerstone low-rank factors
$i \leftarrow 0$;
$len = \text{len}(W)$;
while $i < len$ **do**
    $\alpha, B\_index, A\_index = \text{RandomIndex}()$; ▷ Randomly choose factors
    $W'[i] = W[i] + \alpha B[B\_index[i]]A[A\_index[i]]$; ▷ Parameters obfuscation
    Secret$[i]$ = $[B[B\_index[i]], A[A\_index[i]], \alpha]$;
    $Key[i]$ =Precompute(Secret$[i]$); ▷ Precompute de-obfuscation keys.
end
W'.to(REE); ▷ Offload obfuscated parameters to REE
$Secret$.to(TEE); ▷ Protect secrets and keys in TEE
Key.to(TEE);
return W';

Algorithm 2: LoRO Inference Stage.

Input: Inference Input x
Output: Inference Output y
$Key, Secret \leftarrow$ TEE_Prepare();
$W' \leftarrow$ REE_Prepare(); ▷ Load in Memory
$i \leftarrow 0$;
$len = \text{len}(W)$;
while $i < len$ **do**
    $y_{REE} = \text{Inference}(W'[i], x.to(\text{REE})$; ▷ Inference with obfuscated parameters in REE
    $y_{TEE} = \text{Low\_Rank\_Inference}(Secret[i], x.to(\text{TEE}))$; ▷ De-obfuscation in TEE
    $x = y_{REE}.to(\text{TEE}) + y_{TEE}$; ▷ Result restore in TEE
    if $i == len - 1$ **then**
        $y = x.to(\text{REE})$; ▷ Return result when finish
        break;
    end
    if *next_is_scaled_dot_attention* **then**
        $x = \text{permutation}(x, Key[i])$; ▷ Intermediate result protection
    else
        $x = \text{OTP}(x, Key[i])$; ▷ Intermediate result protection
    end
end
return y;

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: The main contributions have been concluded in the final of Introduction (Section 1).

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitaions have been discussed in Section 7.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

Justification: Our paper does not include any theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We have included all the required information.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Our code is available in https://github.com/D1aoBoomm/LoRO, and all datasets are publicly available.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: All details have been introduced in Section 4, Section 5, Section 6 and Appendix A.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Since our work is about model property protection, error bars are hard to reflect the best ability of adversaries. Instead, we report the highest results in five individual experiments, which is more meaningful.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We have included the computing environments in Section 6.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We follow the NeurIPS code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: Our work focuses on protecting model property, and has no negative societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: No data or models are released.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All the existing assets in this work are publicly available for research usage. The license can all be found in huggingface.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: No new assets are released.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: No crowdsourcing nor research with human subjects are involved.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: No crowdsourcing nor research with human subjects are involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [NA]

    Justification: Focusing on LLM models protection, We have not involved LLMs as components in our methods.

    Guidelines:

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