# CORE: IDENTIFYING AND PROTECTING CRITICAL OUTLIERS IN LLMS AGAINST FUNCTIONALITY STEALING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Protecting proprietary large language models (LLMs) on user-controlled devices requires preventing unauthorized reuse while preserving normal inference. We investigate which components to prioritize based on their importance to model functionality and how to hinder their reconstruction from observable input–output pairs. We find that removing a small subset of outlier-associated weights from the MLP of one specific decoder layer severely degrades model performance, whereas removing the same number from other layers has much less effect. This finding motivates prioritizing the corresponding critical MLP for protection. We propose CORE, a learning-based defense algorithm that combines fixed encodings with learned selective recovery to hinder reconstruction of this component. With the original LLM parameters frozen, the recovery module is trained to support normal inference, while occasional output modifications shift the MSE-optimal surrogate prediction away from the output used in most executions. The fixed encodings and trained module can be reused without preparing fresh masks and corresponding recovery values for each inference forward pass. Across three LLMs, we show that the proposed CORE retains 96.0–98.8% of the original average zero-shot accuracy, while an MSE-trained surrogate replacement performs 17.8–31.2% below the protected model.

## 1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of applications (Touvron et al., 2023a; Workshop, 2024; Noy & Zhang, 2023; Brown et al., 2020). The substantial investment in developing these models gives model owners a strong incentive to protect them from unauthorized copying and reuse (Cottier et al., 2024). Such protection is particularly important when proprietary LLMs run on user-controlled devices, where attackers may gain access to their parameters (Li et al., 2025a; Wang et al., 2025). Trusted execution environments (TEEs), for example, can isolate the selected computation from untrusted host software (Costan & Devadas, 2016; Alves & Felton, 2004). To reduce the overhead of protecting the entire model, existing approaches adopt *selective protection*, keeping selected parameters and computations private while allowing the remaining computation to run outside the protected environment (Mo et al., 2020; Li et al., 2025a;b). The choice of what to protect is central to these approaches: the protected computation must be important enough that the accessible parts alone cannot provide the model's intended functionality.

However, keeping selected parameters private does not necessarily prevent an attacker from reproducing their computation. When the inputs and outputs of the protected portion remain observable, an attacker can collect these pairs to train a surrogate and combine it with the accessible parts of the model, enabling *model functionality stealing* (Li et al., 2025a). Work on LLM compression illustrates the feasibility of this approach by learning lightweight replacements for selected layers from their input–output representations using a mean-squared-error (MSE) objective (Chen et al., 2025). To defend against such reconstruction, a common approach in selective protection is to restrict access to these intermediate inputs and outputs. For example, existing methods conceal intermediate features through *per-inference masking*, using fresh masks and corresponding recovery values to preserve correct computation (Sun et al., 2023; Li et al., 2025a). Although applying and removing masks can be inexpensive online, preparing and supplying fresh masks and their corresponding recovery



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

values for each inference forward pass adds computation, storage, or communication overhead to the protected interface.

These considerations raise two questions for selective protection of LLMs: *which components should be prioritized based on their importance to model functionality, and how can we hinder their reconstruction from observable input–output pairs while preserving inference performance?*

First, inspired by the outlier activation phenomenon in LLMs, where a small number of activations have unusually large magnitudes (Dettmers et al., 2022; Sun et al., 2024a), we investigate whether the importance of their associated weights varies across layers. We conduct a layer-wise study across four LLMs, removing equally sized subsets of these weights from one layer at a time. For each model, removing weights accounting for approximately $0.02\%$ of its parameters from one specific decoder layer causes severe performance degradation or even model collapse, whereas removing the same number from other layers has much less effect. We refer to the outliers associated with these weights as *critical outliers*, and the MLP containing those weights as the *critical MLP*. This finding motivates prioritizing the critical MLP for protection, as removing even a small subset of its weights substantially compromises the model's functionality.

Second, we propose *encoding with learned selective recovery* to make the protected component's outputs less useful for surrogate training while preserving their usefulness for inference. In practice, we first apply two fixed rotations to encode the component's original output, then train a shared recovery module to recover useful outputs from one encoding while keeping the original LLM parameters frozen. The recovered output supports normal inference, while occasional outputs from the other encoding shift the prediction favored by MSE-based surrogate training. Unlike per-inference masking methods, the trained recovery module and fixed encodings can be reused without preparing fresh masks and corresponding recovery values for each inference forward pass.

Our main contributions are:

- **Identifying critical outliers for selective protection.** Through layer-wise weight-removal experiments, we identify specific MLPs whose outlier-associated weights are particularly important to LLM functionality, providing a basis for prioritizing these components for protection.
- **Encoding with learned selective recovery against model functionality stealing.** We introduce CORE, a defense algorithm that combines fixed encodings with a recovery module trained to selectively recover representations needed for normal inference. The proposed method constructs outputs that hinder surrogate reconstruction without requiring fresh masks and corresponding recovery values for each inference forward pass. Our analysis further shows that occasional output modifications shift the MSE-optimal prediction away from the output used in most inference executions.
- **Experimental validation across three LLMs.** We show that the proposed CORE preserves normal inference performance while substantially degrading the functionality recovered by the evaluated model-stealing attacks. On LLaMA-2-7B (Touvron et al., 2023b), LLaMA-3-8B (Grattafiori et al., 2024), and Qwen3-8B (Yang et al., 2025), the protected models retain $98.8\%$, $96.0\%$, and $98.1\%$ of their original average zero-shot accuracy, respectively. In contrast, replacing the protected component with an MSE-trained surrogate reduces this accuracy by $47.3\%$, $42.7\%$, and $26.0\%$ relative to the corresponding protected models.

## 2 Background and Related Work

### 2.1 Protecting Models against Functionality Stealing

**Selective Model Protection.** Existing selective protection methods differ in what they keep inside the trusted execution environments (TEEs). Partition-based methods protect selected layers (Mo et al., 2020), restructure network architecture to introduce lightweight protected branches (Liu et al., 2023; 2024), or train private slices alongside a public backbone (Li et al., 2025a). To protect offloaded weights, existing approaches use linear transformations (Sun et al., 2023), permutations coupled with authorization (Li et al., 2025b; 2024), or transformations that obscure weight-vector directions (Wang et al., 2025). To keep the boundary inputs and outputs of protected components from supervising surrogate training, ShadowNet and TEESlice conceal transmitted features with one-time pad (OTP)

masking and recover the computation results within the TEE (Sun et al., 2023; Li et al., 2025a). Although online masking and unmasking are lightweight, these protocols require fresh masks and associated recovery terms, introducing preprocessing and storage or communication overhead (Sun et al., 2023; Li et al., 2025a). CORE takes a complementary approach. Rather than introducing an additional protocol to conceal these boundary values, we modify the released outputs to reduce their effectiveness as supervision for MSE-based surrogate reconstruction while preserving their utility for inference in the frozen downstream LLM layers. Because its design does not depend on how the protected component is selected, it could in principle be applied on top of existing methods.

**Output-modification defenses** A separate line of work defends against model stealing and distillation by modifying the outputs a learner observes, so they remain useful for normal use but less effective as training targets. Prediction Poisoning perturbs prediction distributions to interfere with attacker training (Orekondy et al., 2019), whereas Adaptive Misinformation selectively returns misleading predictions for out-of-distribution queries (Kariyappa & Qureshi, 2020). Nasty Teacher impairs knowledge transfer through specialized teacher training (Ma et al., 2021), while Antidistillation Sampling modifies next-token sampling to reduce the value of generated traces for distillation (Savani et al., 2025). Our work shares this goal but operates at an internal interface rather than through final prediction distributions or generated sequences. The output representation is directly sent to the downstream layers, so the defense must preserve its inference utility without relying on those layers to adapt to the modification. We address this constraint by encoding with selective recovery and by randomly selecting between primary and secondary outputs. Unlike Adaptive Misinformation, our branch selection is input-independent and does not rely on detecting out-of-distribution queries. Unlike Nasty Teacher, we train only an added recovery module while keeping all original LLM parameters frozen.

### 2.2 Outlier Activations in LLMs

*Outlier activations* (Dettmers et al., 2022; Sun et al., 2024a) have been widely observed in Large Language Models (LLMs), referring to the phenomenon where an extremely small subset of activation values exhibit magnitudes significantly larger than the average across the same layer. These extreme values, while sparse, disproportionately influence model behavior and pose unique challenges for model compression techniques, particularly quantization and pruning. In quantization, outliers increase the dynamic range, leading to severe rounding or clipping errors that degrade model accuracy. To address this, researchers have proposed several outlier-aware quantization methods. For example, SmoothQuant (Xiao et al., 2023) rescales activations and weights to align their magnitudes and reduce quantization error. In the context of pruning outliers complicate magnitude-based importance estimation, as naively pruning them can significantly degrade performance. To address this, pruning methods such as Wanda (Sun et al., 2024b) incorporate activation statistics to identify important weights to preserve. We use outlier-associated weight sensitivity to motivate selecting a functionally sensitive interface. We study how the functional importance of outlier-associated weights varies across layers and use this variation to prioritize what to protect (Section 3.1). We also adopt QuaRot's rotation idea for a different purpose: to encode the protected component's output before selective recovery (Section 3.2).

## 3 CORE

### 3.1 Identify and Protect Critical Outlier-Associated Weights

We begin by investigating whether an LLM's functionality critically depends on a small subset of its parameters. A promising starting point is the outlier activation phenomenon in LLMs: studies on LLM quantization Dettmers et al. (2022) and pruning (Sun et al., 2024b) highlight the importance of preserving the contributions of weights associated with unusually large activations to maintain model performance. Recent work on bit-flip attacks further illustrates the sensitivity of these weights, as they exploit outlier activations to identify weights whose perturbation can substantially disrupt model behavior Almalky et al. (2025). These findings motivate us to examine how removing outlier-associated weights affects LLM performance and whether the resulting degradation varies across layers.

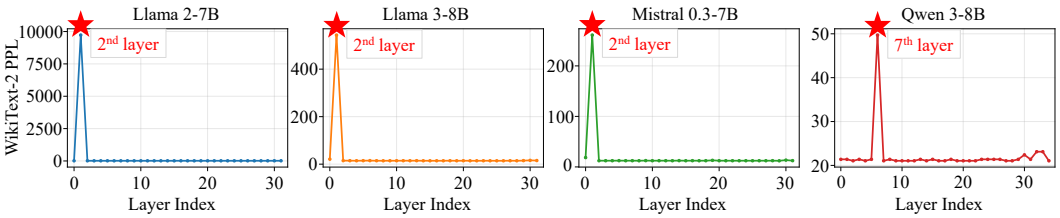



Figure 1: We evaluate the WikiText-2 PPL on four different LLMs by removing the same number of the highest outliers from each layer.

To examine how the functional importance of outlier-associated weights varies across layers, we conduct a layer-wise ablation study on four different LLMs, evaluating model performance after removing these weights from individual layers. Specifically, we adopt Wanda (Sun et al., 2024b), an activation-aware pruning method that scores weights using the product of weight magnitude and the corresponding activation norm. For each model, we remove an equally sized subset of high-scoring weights from one layer at a time, with each subset accounting for approximately $0.02\%$ of the model's parameters. As shown in Figure 1, we observe a striking difference across layers:

*Removal of outlier-associated weights in one specific layer induces significant performance degradation or even model collapse, while similar removal from other layers has less effect.*

We further observe that the layers most sensitive to weight removal (i.e., the $2^{nd}$ layer in Llama2/3 and Mistral models, and the $7^{th}$ layer in Qwen3 model) are precisely where outlier activations first emerge. We refer to the outliers in these layers as ***critical outliers***. Since the weights associated with critical outliers are located in the MLP of the corresponding decoder layer, we refer to this MLP as the ***critical MLP***. This finding motivates prioritizing these weights for protection: although they constitute only a small fraction of the model's parameters, removing them severely compromises the remaining model's functionality.

### 3.2 Protecting Critical Outlier-Associated Weights Against Model Stealing

While critical outliers are important for model functionality, protecting their associated weights still leaves the protected component's inputs and outputs observable, allowing an attacker to train a surrogate that replaces it (Li et al., 2025a), as illustrated in Figure 2. Defending against model functionality stealing while maintaining inference performance is challenging: preserving accurate outputs provides useful supervision for surrogate training, whereas perturbing these outputs can impair normal inference.

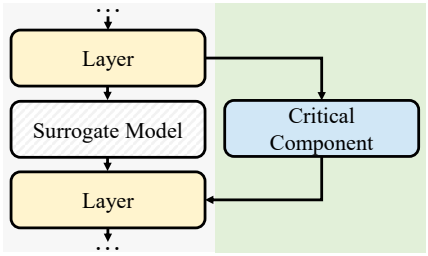



Figure 2: *Model functionality stealing:* The attacker trains a surrogate layer on the critical component's inputs and outputs using an MSE loss, bypassing the protected layers.

To address this challenge, we perturb the outputs in the observable input–output pairs to hinder model functionality stealing while preserving the protected LLM's inference performance. Specifically, rather than returning the normal output on every execution, the protected component occasionally returns a modified output. For an input $r$, let $y_0(r)$ denote the *primary output* that preserves normal inference, and let $y_1(r)$ denote the *secondary output* that deviates from the primary output to alter the attacker's training targets. The component returns

$$Y(r) = \begin{cases} y_0(r), & \text{with probability } 1-\epsilon, \\ y_1(r), & \text{with probability } \epsilon, \end{cases} \tag{1}$$

where $\epsilon$ controls the probability of returning the secondary output. The primary output is selected with high probability $(1-\epsilon)$ to maintain the protected LLM's inference performance, while the secondary output is selected with low probability $\epsilon$ to shift the surrogate's MSE training target. For each input sequence, we randomly select an output according to Eq. 1. The attacker observes the output but not which type was selected. Thus, the secondary output affects only a small fraction

of the protected LLM's inference executions, but can influence the surrogate's predictions through training. We quantify this effect in Section 3.2.2.

A key consideration in constructing the two outputs in Eq. 1 is is that the secondary output should influence surrogate training while the primary output preserves the protected LLM's inference utility. To this end, we propose the **encoding with selective recovery** method to construct the two outputs. We first encode the critical component's original output in two different ways, then process both encodings through a shared learnable recovery module, with the aim of maintaining similar output characteristics. Unlike methods that directly perturb prediction outputs (Orekondy et al., 2019), our method generates both outputs through the same recovery module. We train this module only on the primary encoding to preserve the protected LLM's inference performance. The secondary encoding passes through the trained module without a corresponding recovery constraint, allowing its output to deviate from the primary output.

### 3.2.1 Output Construction via Encoding and Selective Recovery

**Encoding via Orthogonal Rotations.** Inspired by QuaRot (Ashkboos et al., 2024), which uses rotations to smooth activation outliers for quantization, we apply orthogonal rotations to spread large activation values across activation dimensions while preserving the representation norm. This changes the original output without discarding the information needed for recovery.

To obtain two encodings, we independently sample two random sign masks $s_0, s_1 \in \{-1, +1\}^d$ and combine each with the same normalized Hadamard matrix $\bar{H}$. For the critical component's original output $h(r) \in \mathbb{R}^d$, we compute

$$Q_c = \bar{H}\,\mathrm{diag}(s_c), \qquad z_c(r) = Q_c h(r), \qquad c \in \{0, 1\}, \tag{2}$$

where $z_0(r)$ and $z_1(r)$ are the primary and secondary encodings, respectively. Each sign mask first changes the signs of selected activation values, and the Hadamard transform then mixes them across dimensions. Both rotations satisfy $Q_c^\top Q_c = I$, so they are invertible and preserve the norm of $h(r)$. The two sign masks remain fixed after initialization.

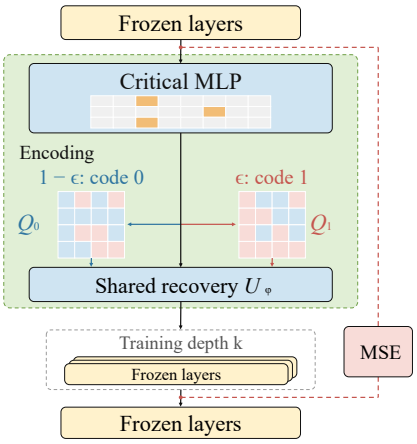



Figure 3: *Critical-interface protection in CORE.* The red dashed path depicts the recovery-supervision objective used to train $U_\phi$.

**Selective Recovery.** After encoding, we use a shared learnable MLP $U_\phi$ to produce the output passed to the remaining LLM layers:

$$y_c(r) = U_\phi(z_c(r)), \qquad c \in \{0, 1\}. \tag{3}$$

We train $U_\phi$ only on the primary encoding $z_0(r)$ so that the primary output $y_0(r)$ preserves the protected LLM's inference performance. The secondary encoding $z_1(r)$ passes through the same MLP but is not included in recovery training, allowing the secondary output $y_1(r)$ to differ from the primary output.

To train the recovery MLP, we compare the primary output with the original output after both are processed by the next $k$ decoder layers. We compute the recovery loss after the frozen mapping $G_k$ and update only the parameters of the shared recovery MLP $U_\phi$. Let $G_k$ denote these layers. We minimize

$$\mathcal{L}_{\mathrm{rec}}^{(k)}(\phi) = \mathbb{E}_r \left\| G_k(U_\phi(z_0(r))) - G_k(h(r)) \right\|_{\mathrm{MSE}}^2, \tag{4}$$

where the squared error is averaged over tokens and activation dimensions. When $k = 0$, $G_0$ is the identity, and training directly matches the primary output to the original output. When $k > 0$, training matches their outputs after the next $k$ decoder layers, encouraging the primary output to preserve the computation performed by those layers. Only the recovery MLP's parameters $\phi$ are updated; all original LLM parameters remain frozen. This objective trains only the recovery module. The attacker's surrogate is trained separately from the observable input–output pairs.

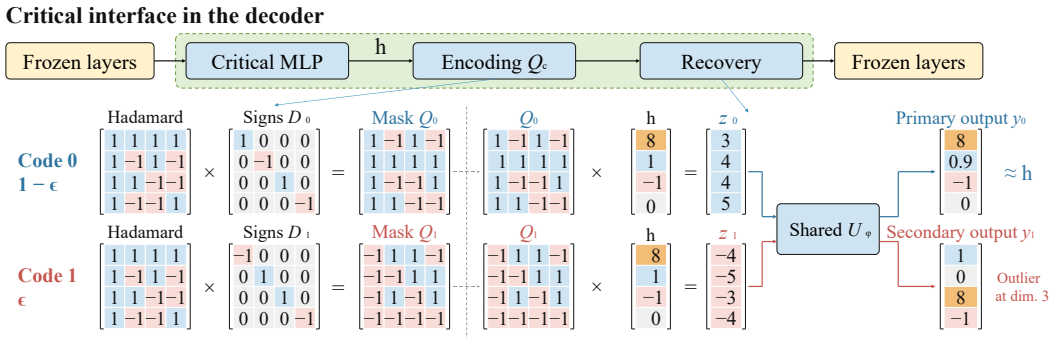



Figure 4: Sign–Hadamard encoding and selective recovery at the critical interface. Numerical values are illustrative.

During inference, we first compute the critical component's original output and select the primary or secondary encoding according to Eq. 1. We then apply the selected rotation and pass the encoded output through the trained recovery MLP. The resulting output is passed to the remaining LLM layers and is observable to the attacker, while the original output, encoded output, and encoding selection remain inside the protected component.

### 3.2.2 Analysis of Surrogate Training under Mixed Outputs

We analyze how the primary and secondary outputs affect surrogate training. The protected LLM uses the secondary output only when it is selected. During surrogate training, however, secondary outputs contribute to the learned prediction function together with primary outputs. The following proposition characterizes this effect under an MSE objective.

**Proposition 1** (Prediction bias under mixed outputs)**.** *Let $y_0(r)$ and $y_1(r)$ be the primary and secondary outputs, with finite second moments. Suppose the protected component selects the secondary output with probability $\epsilon$, independently of the input $r$, and does not reveal the selection to the attacker. Among deterministic prediction functions, the population MSE*

$$\mathcal{L}(S) = \mathbb{E}_{r,c} \left\| S(r) - y_c(r) \right\|_{\mathrm{MSE}}^2 \tag{5}$$

*is minimized by*

$$S^*(r) = (1-\epsilon) y_0(r) + \epsilon y_1(r). \tag{6}$$

*Let*

$$D = \mathbb{E}_r \left\| y_1(r) - y_0(r) \right\|_{\mathrm{MSE}}^2 \tag{7}$$

*denotes the average squared difference between the two outputs. The optimal prediction then deviates from the primary output by*

$$\mathbb{E}_r \left\| S^*(r) - y_0(r) \right\|_{\mathrm{MSE}}^2 = \epsilon^2 D. \tag{8}$$

*Proof.* Let $\mu(r) = \mathbb{E}[y_c(r) \mid r]$. Since the output selection is independent of $r$, $\mu(r) = (1-\epsilon) y_0(r) + \epsilon y_1(r)$. The expected squared error decomposes as

$$\mathcal{L}(S) = \mathbb{E}_r \left\| S(r) - \mu(r) \right\|_{\mathrm{MSE}}^2 + \epsilon(1-\epsilon) D. \tag{9}$$

The second term does not depend on $S$, so the minimum is attained at $S^* = \mu$. Substituting $S^*(r) - y_0(r) = \epsilon(y_1(r) - y_0(r))$ gives Eq. 8. □

Proposition 1 shows that fitting the observed outputs does not generally recover the primary output alone. Instead, the secondary output shifts the surrogate's optimal prediction. The size of this shift depends on both its selection probability $\epsilon$ and the output difference $D$. A low $\epsilon$ limits how often the protected LLM uses the secondary output, but also reduces its influence on the surrogate's optimal prediction. Our design therefore seeks a sufficiently large $D$ to maintain a substantial prediction shift at a low secondary-output probability. The proposition characterizes the population MSE optimum; the prediction of a trained surrogate also depends on its capacity, training data, and optimization.

**Limitation of Exact Recovery.** Since the encodings are orthogonal, a straightforward alternative to learned recovery is to apply the inverse of the primary encoding. Setting $U = Q_0^\top$ exactly recovers the original output on the primary path:

$$y_0(r) = h(r), \qquad y_1(r) = Q_0^\top Q_1 h(r). \tag{10}$$

However, this choice restricts the difference between the two outputs. For our sign–Hadamard encodings,

$$Q_0^\top Q_1 = \mathrm{diag}(s_0 \odot s_1). \tag{11}$$

The MSE-optimal surrogate therefore predicts

$$S^*_{\mathrm{inv}}(r) = \mathrm{diag}\left((1-\epsilon)\mathbf{1} + \epsilon(s_0 \odot s_1)\right) h(r). \tag{12}$$

Each activation value is multiplied by either 1 or $1 - 2\epsilon$. With $\epsilon = 0.01$, these factors are 1 and $0.98$, respectively. Thus, exact recovery leaves the surrogate's optimal prediction close to the original output, with each activation magnitude changed by at most $2\%$.

**Role of Selective Recovery.** The learned recovery MLP is not restricted to this coordinate-wise scaling. Training on the primary encoding constrains the primary output to preserve the protected LLM's inference performance, while the secondary output is not constrained by the recovery objective. This allows the two outputs to differ beyond the changes permitted by exact recovery, potentially increasing $D$ without increasing $\epsilon$.

The resulting design separates the frequency of secondary outputs during inference from the magnitude of their effect on the surrogate's learning target. The protected LLM uses primary outputs in most executions, whereas the surrogate is trained toward a prediction influenced by both outputs. Because this prediction supplies the activations used by the remaining LLM layers, its deviation from the primary output can impair the stolen model's performance. The analysis quantifies this deviation; its effect on model functionality, together with the protected LLM's inference performance, is assessed experimentally.

# 4 Experiments

## 4.1 Experimental Setup

We evaluate the effectiveness of CORE by assessing both the protected model's inference performance and the performance of surrogate models (i.e., PPL) trained by the attacker, as discussed in Section 3.2. Specifically, we conduct experiments using three open-source LLMs: LLaMA 2-7B (Touvron et al., 2023b), LLaMA 3-8B (Grattafiori et al., 2024), Qwen 3-8B (Yang et al., 2025). Following a similar experimental setup as prior work (Chen et al., 2025), we use 1% of the SlimPajama-6B dataset (DKYoon/SlimPajama-6B (Soboleva et al., 2023)) to train the attacker's surrogate model and our learned recovery MLP. This corresponds to 29995 samples, each containing 2048 tokens, totaling around 0.06 billion tokens. Appendix A.1 provides detailed optimization hyperparameters and hardware configurations.

We evaluate three configurations: (1) **Base**, the original model; (2) **Reference**, the protected model using encodings and the learned recovery MLP, which marks the final performance to the user; (3) **Attacked**, the model obtained by replacing the protected component with an MSE-trained surrogate. For the main results, we set the secondary-output probability to $\epsilon = 0.01$ and use recovery-supervision depths $k = 0$, 5, and 4 for LLaMA-2-7B, LLaMA-3-8B, and Qwen3-8B, respectively. We further study the effect of recovery-supervision depth in Section 4.3, and various secondary-output probabilities in Appendix A.2. We evaluate model functionality using three metrics: perplexity (PPL) on WikiText-2 (Merity et al., 2016) (Wiki2), perplexity on Penn Treebank (Marcus et al., 1993) (PTB), and zero-shot accuracy on the CommonsenseQA (CMQA) benchmark suite including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy (Clark et al., 2018), ARC-Challenge (Clark et al., 2018), and OpenbookQA (Mihaylov et al., 2018), applied through the EleutherAI LM Harness pipeline (Gao et al., 2024).

## 4.2 Main Results

Table 1 shows that the protected reference models retain most of the base models' average zero-shot accuracy across all three LLMs. WikiText and PTB perplexity increase moderately, with a more

Table 1: Main results. Base is the unmodified model, Reference is the protected model using fixed encodings and learned recovery, and Attacked uses an MSE-trained surrogate in place of the protected component.

| Model | Mode | PPL (Wiki2)↓ | PPL (PTB)↓ | CMQA (%)↑ |
|---|---|---|---|---|
| LLaMA-2-7B | Base | 12.19 | 47.21 | 66.79 |
| | Reference | 13.75 | 53.20 | 65.98 |
| | Attacked | 28055.57 | 66762.53 | 34.78 |
| LLaMA-3-8B | Base | 14.14 | 27.98 | 70.17 |
| | Reference | 22.10 | 43.09 | 67.33 |
| | Attacked | 533.44 | 728.17 | 38.61 |
| Qwen3-8B | Base | 21.13 | 42.36 | 69.67 |
| | Reference | 24.35 | 46.21 | 68.32 |
| | Attacked | 84.56 | 119.25 | 50.56 |

pronounced increase on LLaMA-3-8B than on the other models. Thus, CORE largely preserves performance on the evaluated zero-shot tasks, while its impact on language-modeling performance remains model-dependent.

In contrast, replacing the protected component with an MSE-trained surrogate results in substantially higher perplexity on both datasets and lower CMQA accuracy than the corresponding reference model. The performance loss after surrogate replacement is substantially greater than the loss introduced by protection. These results demonstrate a clear utility gap between the protected reference and the evaluated surrogate replacement, consistent with the objective of preserving authorized inference while disrupting functional recovery through MSE distillation.

### 4.3 Ablation Studies

**Exact inversion versus learned recovery.** Table 2 compares exact inversion, $U = Q_0^\top$, with the learned recovery MLP $U_\phi$ on LLaMA-2-7B. Although exact inversion provides slightly better reference utility, as expected since it reconstructs $h(r)$ without error, it offers substantially weaker protection. Relative to its own reference, the attacked model's WikiText-2 perplexity increases by $6.8\%$ and its CMQA accuracy drops by $2.9\%$, although PTB perplexity nearly doubles. With learned recovery, the same attack increases WikiText-2 perplexity by about $2{,}040\times$ and reduces CMQA accuracy by $47.3\%$. This contrast supports the rationale in Sec. 3.2.2. Under exact inversion, the surrogate's MSE-optimal target stays a globally linear function of the clean representation, and for the shared Hadamard construction, its coordinates are scaled only by $1$ or $1 - 2\epsilon$ ($0.98$ at a 99/1 ratio). A poorly performing off-code branch therefore need not substantially distort the student's optimal target. The learned nonlinear recovery module is not restricted to this relation and can respond selectively to the two encoding distributions.

Table 2: Exact inverse versus learned recovery on LLaMA-2-7B. We compare the fixed exact inverse $Q_0^\top$ with the learned recovery MLP $U_\phi$ trained only on code 0.

| Mode | Recovery | PPL (Wiki2)↓ | PPL (PTB)↓ | CMQA (%)↑ |
|---|---|---|---|---|
| Base | – | 12.19 | 47.21 | 66.79 |
| Reference | Exact inverse $Q_0^\top$ | 13.09 | 50.01 | 66.37 |
| | Learned recovery $U_\phi$ | 13.75 | 53.20 | 65.98 |
| Attacked | Exact inverse $Q_0^\top$ | 13.98 | 97.56 | 64.45 |
| | Learned recovery $U_\phi$ | 28055.57 | 66762.53 | 34.78 |

**Effect of Recovery-Supervision Depth** Table 3 examines the effect of the supervision depth $k$ in equation 4. At $k = 0$, the recovery loss is applied directly at the protected interface. For $k > 0$, the primary and original outputs are first propagated through the next $k$ frozen decoder layers. The

Table 3: Recovery-supervision ablation. Depth $k$ denotes the number of frozen decoder layers in $G_k$.

| Model | Mode | Depth $k$ | PPL (Wiki2)↓ | PPL (PTB)↓ | CMQA (%)↑ |
|---|---|---|---|---|---|
| LLaMA-2-7B | Base | – | 12.19 | 47.21 | 66.79 |
| | Reference | 0 | 13.75 | 53.20 | 65.98 |
| | | 2 | 14.44 | 53.60 | 64.65 |
| | | 4 | 14.36 | 53.10 | 64.69 |
| | Attacked | 0 | 28055.57 | 66762.53 | 34.78 |
| | | 2 | 37340.11 | 96977.01 | 38.44 |
| | | 4 | 28326.11 | 56293.24 | 35.07 |
| LLaMA-3-8B | Base | – | 14.14 | 27.98 | 70.17 |
| | Reference | 0 | 311.62 | 263.81 | 37.13 |
| | | 2 | 48.36 | 64.04 | 60.82 |
| | | 4 | 23.84 | 45.46 | 66.56 |
| | | 5 | 22.10 | 43.09 | 67.33 |
| | Attacked | 0 | 847.56 | 824.18 | 36.52 |
| | | 2 | 620.13 | 777.53 | 36.62 |
| | | 4 | 762.53 | 1092.39 | 37.31 |
| | | 5 | 533.44 | 728.17 | 38.61 |
| Qwen3-8B | Base | – | 21.13 | 42.36 | 69.67 |
| | Reference | 0 | 25.37 | 47.24 | 68.34 |
| | | 2 | 24.38 | 46.24 | 67.62 |
| | | 4 | 24.35 | 46.21 | 68.32 |
| | Attacked | 0 | 24.61 | 45.26 | 69.55 |
| | | 2 | 45.98 | 81.96 | 64.06 |
| | | 4 | 84.56 | 119.25 | 50.56 |

effect of $k$ depends on the model. On LLaMA-2-7B, $k = 0$ already preserves reference utility and provides strong protection, while deeper supervision brings no further benefit. On LLaMA-3-8B, increasing $k$ from 0 to 5 improves reference CMQA accuracy by $30.20$ percentage points and reduces WikiText-2 perplexity by $92.9\%$, indicating that downstream supervision substantially improves recovery. On Qwen3-8B, increasing $k$ from 0 to 4 reduces surrogate CMQA accuracy by $18.99$ percentage points, while reference accuracy decreases by only $0.02$ percentage points. These results support model-specific depth choices, while deeper supervision primarily improves reference utility on LLaMA-3-8B and protection against the evaluated attack on Qwen3-8B.

## 5 Conclusion

We identified critical MLPs whose outlier-associated weights are particularly important to LLM functionality, providing a basis for selective protection. Building on this finding, we proposed CORE, which combines fixed encodings with learned selective recovery to hinder MSE-based reconstruction from observable input–output pairs. Experiments across three LLMs show that CORE retains most of the original average zero-shot accuracy while substantially degrading the evaluated surrogate replacements.

**Limitations and Future Work.** Our analysis and experiments consider surrogates trained with MSE, a standard objective for learning replacements of LLM layers from hidden-state pairs (Chen et al., 2025). Proposition 1 assumes the optimal predictor is the conditional mean, so objectives with other optimal predictors fall outside our analysis. Evaluating CORE against adaptive attackers, such as those that discard suspected secondary outputs or train the surrogate end-to-end through the frozen downstream layers, is a direction for future work.

**Reproducibility Statement.** We provide an anonymous implementation of CORE in the supplementary material, including scripts for critical-MLP identification, recovery-module training, surrogate training, and evaluation. Additional details, such as hyperparameters and experimental settings, are described in the main text and appendix to ensure full reproducibility.

### AI USE STATEMENT

In accordance with the ICLR 2027 policy on large language model (LLM) usage, we disclose that we used ChatGPT to polish the writing and improve clarity. We did not use it for research ideation, experimental design, data analysis, or other substantive contributions. The authors are responsible for all scientific content, results, and conclusions.

## REFERENCES

Abeer Matar A Almalky, Ziyan Wang, Mohaiminul Al Nahian, Li Yang, and Adnan Siraj Rakin. Invisible hands: Gray-box bit flip attack for steering llms without knowledge of gradients, data, and weights. *arXiv preprint arXiv:2511.22700*, 2025.

Tiago Alves and Don Felton. Trustzone: Integrated hardware and software security. ARM White Paper, 2004.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. Streamlining redundant layers to compress large language models. In *International Conference on Learning Representations*, volume 2025, pp. 30362–30383, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

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

Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, Tamay Besiroglu, and David Owen. The rising costs of training frontier ai models. *arXiv preprint arXiv:2405.21015*, 2024.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

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.

Sanjay Kariyappa and Moinuddin K Qureshi. Defending against model stealing attacks with adaptive misinformation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 767–775. IEEE, 2020.

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*, 34(6):1–49, 2025a.

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*, pp. 3479–3488, 2024.

Qinfeng Li, Tianyue Luo, Xuhong Zhang, Yangfan Xie, Zhiqiang Shen, Lijun Zhang, Yier Jin, Hao Peng, Xinkui Zhao, XianWei Zhu, and Jianwei Yin. Coreguard: Safeguarding foundational capabilities of llms against model stealing in edge deployment. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 69221–69243. Curran Associates, Inc., 2025b. doi: 10.52202/085713-2327. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/6447714b83edcbed61dbe10371dd7ae5-Paper-Conference.pdf.

Ziyu Liu, Yukui Luo, Shijin Duan, Tong Zhou, and Xiaolin Xu. Mirrornet: A tee-friendly framework for secure on-device dnn inference. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–9. IEEE, 2023.

Ziyu Liu, Tong Zhou, Yukui Luo, and Xiaolin Xu. Tbnet: A neural architectural defense framework facilitating dnn model protection in trusted execution environments. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pp. 1–6, 2024.

Haoyu Ma, Tianlong Chen, Ting-Kuei Hu, Chenyu You, Xiaohui Xie, and Zhangyang Wang. Undistillable: Making a nasty teacher that cannot teach students. *arXiv preprint arXiv:2105.07381*, 2021.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

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*, pp. 161–174, 2020.

Shakked Noy and Whitney Zhang. Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654):187–192, 2023.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. *arXiv preprint arXiv:1906.10908*, 2019.

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

Yash Savani, Asher Trockman, Zhili Feng, Yixuan Xu, Avi Schwarzschild, Alexander Robey, Marc Finzi, and Zico Kolter. Antidistillation sampling. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 117800–117827. Curran Associates, Inc., 2025. doi: 10.52202/085713-3927. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/aad5e2a483869d9ba3fab491686c3bf2-Paper-Conference.pdf.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024a.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024b.

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)*, pp. 1596–1612. IEEE, 2023.

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

Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models, 2023b. URL https://arxiv.org/abs/2307.09288.

Pengli Wang, Bingyou Dong, Yifeng Cai, Zheng Zhang, Junlin Liu, Huanran Xue, Ye Wu, Yao Zhang, and Ziqi Zhang. Game of arrows: On the (In-)Security of weight obfuscation for On-Device TEE-Shielded LLM partition algorithms. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 279–298, Seattle, WA, August 2025. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/wang-pengli.

BigScience Workshop. Bloom: A 176b-parameter open-access multilingual language model. *Journal of Machine Learning Research*, 25(422):1–74, 2024. URL http://jmlr.org/papers/v25/23-0581.html.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.

An Yang et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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

# A Appendix

## A.1 Detailed Experimental Setup

To conduct the retraining attack, a 1% subset of SlimPajama-6B is used, consisting of 29995 training samples (each with 2048 tokens, approximately 0.06B tokens) and 1026 evaluation samples of the same length. Training is conducted for 20 epochs with a batch size of 32. The learning rate is set to 1e-3, with a minimum learning rate of 2.5e-5 and a maximum of 1e-3. Weight decay is configured to 1e-4. We adopt a cosine scheduler to perform learning rate scheduling. We have a 3% warmup step in the first epochs. All model parameters are set to bfloat16 precision by default if no other notification exists. All experiments are conducted on a cloud computing server with an AMD EPYC 9554 CPU, 318.6 GB of memory, 400GB NVME SSD, and one Nvidia H100 80GB GPU.

## A.2 Secondary-output probability

Table 4 studies the effect of the secondary-output probability $\epsilon$ on LLaMA-2-7B. Increasing $\epsilon$ from 0.01 to 0.05 raises reference WikiText-2 perplexity and PTB perplexity, reflecting the utility cost of returning secondary outputs more frequently. The attacked models also exhibit higher perplexity, but their CMQA accuracy does not decrease monotonically. Thus, increasing the secondary-output probability does not uniformly improve protection across the evaluated metrics. The default setting $\epsilon = 0.01$ provides the best reference utility among the tested settings while already substantially reducing the functionality recovered by the evaluated MSE-based attack.

Table 4: Effect of the secondary-output probability on LLaMA-2-7B. The protected component returns the primary output with probability $1 - \epsilon$ and the secondary output with probability $\epsilon$.

| **Mode** | **Primary/Secondary** | **PPL (Wiki2)↓** | **PPL (PTB)↓** | **CMQA (%)↑** |
|---|---|---|---|---|
| Base | – | 12.19 | 47.21 | 66.79 |
| Reference | 95/5 | 18.74 | 70.63 | 64.73 |
| | 96/4 | 17.35 | 65.80 | 65.04 |
| | 97/3 | 16.05 | 61.29 | 65.35 |
| | 98/2 | 14.86 | 57.10 | 65.67 |
| | 99/1 | 13.75 | 53.20 | 65.98 |
| Attacked | 95/5 | 135856.06 | 172888.62 | 37.20 |
| | 96/4 | 130069.67 | 150067.63 | 37.01 |
| | 97/3 | 90936.11 | 137047.39 | 38.51 |
| | 98/2 | 65759.51 | 115643.45 | 38.02 |
| | 99/1 | 28055.57 | 66762.53 | 34.78 |