USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

# Depth Gives a False Sense of Privacy: LLM Internal States Inversion

Tian Dong and Yan Meng, *Shanghai Jiao Tong University;*
Shaofeng Li, *Southeast University;* Guoxing Chen, Zhen Liu,
and Haojin Zhu, *Shanghai Jiao Tong University*

https://www.usenix.org/conference/usenixsecurity25/presentation/dong-tian

This paper is included in the Proceedings of the
34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the
34th USENIX Security Symposium is sponsored by USENIX.

# Depth Gives a False Sense of Privacy: LLM Internal States Inversion

Tian Dong[1], Yan Meng[1,✉], Shaofeng Li[2], Guoxing Chen[1], Zhen Liu[1], and Haojin Zhu[1,✉]
[1]*Shanghai Jiao Tong University, {tian.dong, yan_meng, guoxingchen, zhu-hj}@sjtu.edu.cn*
[2]*Southeast University, shaofengli@seu.edu.cn*

## Abstract

Large Language Models (LLMs) are increasingly integrated into daily routines, yet they raise significant privacy and safety concerns. Recent research proposes collaborative inference, which outsources the early-layer inference to ensure data locality, and introduces model safety auditing based on inner neuron patterns. Both techniques expose the LLM's *Internal States* (ISs), which are traditionally considered irreversible to inputs due to optimization challenges and the highly abstract representations in deep layers. In this work, we challenge this assumption by proposing four inversion attacks that significantly improve the semantic similarity and token matching rate of inverted inputs. Specifically, we first develop *two white-box optimization-based attacks* tailored for low-depth and high-depth ISs. These attacks avoid local minima convergence, a limitation observed in prior work, through a two-phase inversion process. Then, we extend our optimization attack under more practical black-box weight access by leveraging the transferability between the source and the derived LLMs. Additionally, we introduce a *generation-based attack* that treats inversion as a translation task, employing an inversion model to reconstruct inputs. Extensive evaluation of short and long prompts from medical consulting and coding assistance datasets and 6 LLMs validates the effectiveness of our inversion attacks. Notably, a 4,112-token long medical consulting prompt can be nearly perfectly inverted with 86.88 F1 token matching from the middle layer of Llama-3 model. Finally, we evaluate four practical defenses that we found cannot perfectly prevent ISs inversion and draw conclusions for future mitigation design.

## 1 Introduction

Despite its widespread application, the large size of Large Language Models (LLMs) prohibits fast inference on local devices, forcing users to send their inputs (a.k.a., prompts) to the cloud and risk privacy leakage. This also impedes the application in sensitive domains and commercial cooperation [1]. Moreover, as the model scale continues to grow (e.g., Llama-3 has a size up to 405B [2]), a single server can merely load the model in one piece, let alone swift inference.

Therefore, collaborative inference [3, 4, 5] has been widely applied to enforce *data locality*, where the shallow layers are stored on the local device and only the Internal States (ISs) are transmitted to the cloud for continuous inference on rest layers. Meanwhile, to meet the requirements of trustworthy Artificial Intelligence (AI) [6], ISs can also be exposed to a third party for safety auditing, as ISs of deep layers can be leveraged to robustly identify factual errors [7, 8, 9, 10, 11], defend jailbreaks, backdoors [12, 13, 14], or manipulate internal representations of the model's concepts [15, 16, 17].

The potential exposure of increasingly used ISs raises our research question: *Can we invert the input query based on the ISs, even in highly deep LLMs?* Current embedding inversion [18] assigns trainable variables to each input token and selects the candidate tokens via optimization, which is proven effective on conventional Language Models (LMs) (e.g., BERT). Recent works show that text embeddings or model outputs [18, 19, 20] can be used to invert inputs. These attacks train generative inversion models conditioned on observed embeddings or outputs.

Yet, simple adoption cannot work well for ISs because of two new challenges. First, ISs are designed for subsequent inference and contain abstract logical representations [17], which are inherently different from previously studied embeddings or model outputs of high semantically relevance with inputs. Second, LLMs have significantly more layers, higher width, and larger dictionary than LMs studied in prior work, which further hinders the inversion, especially for ISs of deep layers because of feature loss based on the information bottleneck [21]. Therefore, we need more powerful inversion attacks to evaluate the privacy risk of ISs.

In this work, we are the first to explore the inversion feasibility of ISs by proposing both optimization-based and generation-based attacks adapting to white-box access and black-box access to model weights. Specifically, our white-

✉Yan Meng and Haojin Zhu are corresponding authors.

box attacks are designed for the adversary (e.g., curious-but-honest inference server) who can exploit the weights to optimize the input text with nearly exact and correctly ordered tokens without any assumption on input distribution. Our black-box attacks are suitable for a third-party adversary (e.g., LLM auditor) who can probe the ISs for analysis and can train an inversion model based on her own surrogate data of similar distribution to the victim's queries.

Since searching for the optimal token sequence through brute force is infeasible, we introduce a novel two-phase inversion for the optimization-based attack: we first invert the input embeddings and then recover the correct input tokens. For shallow layers, our attack, Embedding Recovery (ER), produces embeddings of candidate inputs by minimizing the distance of its ISs to the target. Then, the tokens with the closest input embedding to the optimized embedding are selected. This tackles the large-dictionary challenge by avoiding searching over significantly huge token combination space. For deep layers, ER can fail because of gradient explosion. We propose Token Basis Selection (TBS) that determines the optimal combination among base vectors of input embedding space as the inverted embeddings for further token inversion. This tackles the high-depth challenge by reducing optimized variables and avoiding local minima encountered in the previous solution [18].

Without access to the target model weights, we first extend our optimization attacks to the black-box setting by identifying whether the target is derived from adversary-known LLMs, based on our insights that a large number of LLMs are derived from existing ones instead of pretrained from scratch. For the generation-based attack, we regard the ISs as an encoded language and use the encoder-decoder models, which are commonly used in machine translation, for input inversion. To tackle the challenge of representation discrepancy between ISs and semantic meaning, we propose a projection module that aligns the ISs with the encoder for inversion with the decoder.

Our evaluations include 6 real-world high-ranking LLMs, both short-context prompts, as adopted in existing works, and additional long-context prompts on medical consulting and coding assistance. The results demonstrate the inversion effectiveness. For example, given ISs from the middle layer of `Llama-3-8B-Instruct`, our TBS attack can invert input of 4,112 tokens with 86.88 F1 token matching and 95.19 semantic similarity which cannot be reached by prior work. Our generation-based attack can also achieve 81.6 F1 score for inputs of medium length (i.e., $\sim$1k tokens) which is comparable to the white-box attack. Lastly, we test four defenses including quantization, dropout, noisy input embedding, and Differential Privacy (DP) through the Laplace mechanism. Our black-box attack cannot be mitigated without greatly deteriorating the model utility, calling for more effective defenses in the future.

In summary, our contributions are:

- We are the first to systematically investigate the input inversion risk of LLM ISs. Our work reveals that an attacker can successfully recover sensitive prompts of LLMs, spanning up to 4,112 tokens, from their ISs.
- To overcome the challenges of semantic spasticity and feature loss from high-depth layers, we propose four novel inversion attacks adapting to both white-box and black-box attack settings.
- We extensively evaluate our attacks on sensitive inputs including medical dialogues and coding assistance. We also evaluated DP-based defense and found our attack can still invert input of high semantic similarity even significantly sacrificing the downstream inference quality.

## 2 Preliminaries & Motivation

In this section, we first briefly introduce the modern LLM implementation, the risk of IS exposure and overview existing inversion techniques.

### 2.1 Language Modeling

We begin with a brief recap of how modern LLMs processes the texts. Formally, an input text (a.k.a., prompt) $x$ is first tokenized by the tokenizer $\mathcal{T}_f$ of an LLM $f$ into a string of tokens $\mathcal{T}_f(x) := \mathbf{t}^x = [t_i^x]_{\{i=0,\cdots,N_\mathcal{T}\}}$ where each token is marked with an ID $t_i$ bounded by the maximum token number $N_\mathcal{T}$, i.e., $t_i \in [0,\cdots,N_\mathcal{T}]$. The token IDs are then mapped by the input embedding layer $\mathcal{E}^f$ of weight $\mathbf{w}_{in}^f \in \mathbb{R}^{N_\mathcal{T} \times d_{in}}$ into the sub-matrix of corresponding row vectors $\mathcal{E}^f(\mathbf{t}^x) := [\mathbf{w}_{in}^f(t_i^x)]_{\{i=0,\cdots,N_t\}}$. We denote the first $l$ Transformer layers of an LLM $f$ is $\psi_l^f$. The ISs of the $l$-th layer of an LLM are $\mathbf{h}_l^f$. Put together, given input text $x$, the ISs at $l$-th layer of an LLM $f$ are $\mathbf{h}_l^f(x) = \psi_l^f(\mathcal{E}^f(\mathcal{T}_f(x)))$.

### 2.2 Motivation

In LLM service, maintaining the confidentiality of ISs is not always guaranteed. We identify two scenarios in which ISs may be exposed to an untrusted party for inversion.

**LLM Alignment & Concept Engineering.** LLMs are notorious for hallucination, which can deceive and misinform the user. Therefore, LLMs are persistently surveilled for safety reasons. Recently, a growing number of studies [8, 9, 10, 12, 17, 28] show that the ISs are robust indicators of hallucination. This provides LLM holders with a promising solution to correct model behavior in the runtime [11]. For instance, instead of directly examining the prompts, OpenAI may run an automated safety classifier to improve their services based on the classifier-generated metadata based on policy [29]. Besides, ISs are also required in representation engineering [15] to control the model concept.

Table 1: Comparison with previous inversion attacks against (large) language models.

| Inversion Target | Method | Weight Access | Inversion Goal | | | Evaluation | | |
|---|---|---|---|---|---|---|---|---|
| | | | Semantic-preserving | Token Matching | Attribute Inference | Max. Model Size[1] | Data Types | Max. Data Length |
| Embeddings [18, 19, 22, 23] | Generation | ● | ✓ | ✗ | ✗ | 110M | Chats, Medicine | 422 |
| | Optimization | ○ | ✗ | ✓ | ✓ | | | |
| Outputs [20, 24] | Generation | ● | ✓ | ✗ | ✗ | 7B | System Prompts, Chats | 256 |
| Gradients[2] [25, 26, 27] | Optimization | ○ | ✓ | ✓ | ✗ | 7B | Chats, Code, Math | 512 |
| Internal States (Ours) | Generation | ● | ✓ | ✗ | ✗ | 70B | Medicine, Code | 4,112 |
| | Optimization | ○ | ✗ | ✓ | | | | |

[1] Only for models with publicly known size. [2] Attacks targeting training data instead of input texts.

**Collaborative & TEE-shielded LLM Inference.** Several solutions [3, 4, 5, 30] have been proposed for layer-wise LLM splitting and partitioning to accelerate LLM serving capacity. For instance, EdgeShard [5] dynamically shards models onto edge devices for closer LLM deployment to the data source. PETAL [30] allows several servers to collaboratively infer or finetune models up to 405B through layer-wise model splitting. HELIX [4] exploits heterogeneous GPUs from different regions in the globe. Besides, the split model can be loaded in Trusted Execution Environment (TEE) to protect input privacy [31, 32]. In both settings, the party holding rest model layers receives and infers ISs.

## 2.3 Challenges

There has been a line of work studying how the LMs leak the user input texts. For example, the text embeddings are shown at risk of leaking the input text through inversion attacks in both white-box and black-box settings [18, 19]. Recently, it has been shown that the user's input prompt can be accurately reconstructed with only the LLM logits [24] or outputs [20]. The gradients in federated learning [25, 26, 27] can also leak the training texts by inversion attacks. Technically, the inversion either attempts to locate the candidate input tokens (without considering their relative positions) through optimization (Optimization-based), or trains a generative model with surrogate data for inversion conditioned on observed embeddings (Generation-based).

Our work is the first to explore the feasibility of input inversion through ISs as an LLM inner representation. Applying existing embedding inversion techniques on ISs can result in poor inversion due to the difference between text embeddings and LLM ISs. We identify two new challenges:

*Increased model scale and token dictionary size.* In general, LLMs have a deeper layered architecture, higher model width, and larger token dictionary than conventional pretrained LMs, making it almost impossible to get the exact input as the victim through the dictionary attack. For example, typical base BERT only contains 110M parameters, 12 layers with width 768 and around 30k dictionary size, while modern 8B Llama-3 contains 32 layers of width 4,096 and more than 120k dictionary tokens. The larger model scale significantly increases trainable parameters, causing previous optimization-based inversion [18] to fail because of falling into local minima on highly compressed ISs (see Section 4).

*Inference-oriented and complex representations.* Previous embedding inversion attacks focused on sentence embeddings which are typically optimized for semantic relevance based on mean-pooled encoder outputs [33]. On the contrary, in the context of LLMs, the ISs are generated for continuous inference, thus contain more sparse semantic features than semantically-enhanced embeddings. In addition, the ISs of deep layers contain a higher level of concept abstraction [15, 34] like reasoning, which further increases the difficulty of accurate input inversion.

In this work, we systematically analyze the inversion risk of ISs by addressing the aforementioned challenges through novel optimization-based and generation-based attacks. More importantly, we validate the effectiveness of these attacks in practical settings, as summarized in Table 1.

# 3 Threat Model

We consider a curious-but-honest adversary. For example, a malicious third-party auditor authorized to access ISs for safety auditing [7, 8] can stealthily store the observed ISs for the offline inversion. In the context of collaborative inference, the adversary server hosting middle layers or bottom layers of the deployed LLM, receives ISs of the splitting layer and sends results to the next party following the protocol with the client and other servers [4, 30]. The server can allocate partial computational power to invert the user queries based on knowledge of splitting layers and observed ISs.

**Adversary Goals.** The adversary $\mathcal{A}$ aims to invert the input texts/prompts of victim $\mathcal{V}$ based on the observed ISs $\mathbf{h}_l^{f_\mathcal{V}}(x)$ of ground truth inputs $x$ and her model $f_\mathcal{A}$. The inverted texts should preserve the semantics and exact tokens as the victim's input. Note that this goal is harder than privacy attribute inference and can be applied for further analysis (e.g., Personally Identifiable Information (PII) or user identification [35]). For-

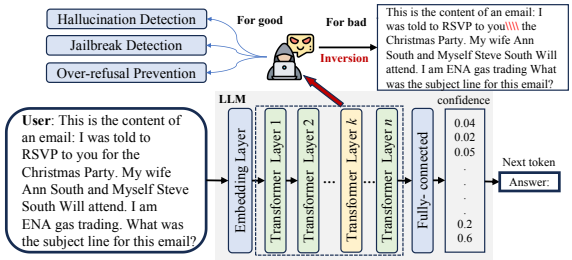


Figure 1: A curious-but-honest LLM safety auditor or collaborative inference party can observe ISs and recover the nearly *exact* user inputs even in deep layers (false inverted tokens are in red).

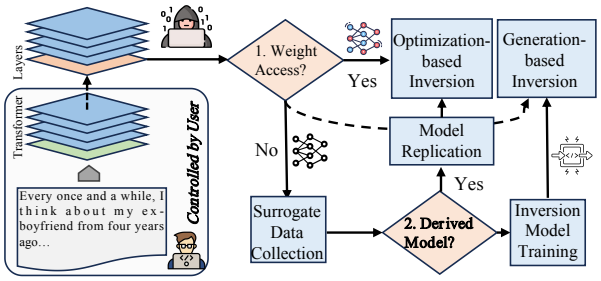


Figure 2: Overview procedure of our attacks. Depending on the model access, the adversary can adopt our optimization-based or generation-based attacks to achieve attack goals.

mally, the ISs inversion is

$$\hat{x} = \arg\min_{x' \in \mathrm{dom}\mathcal{T}} \mathrm{d}(\mathbf{h}_l^{f_{\mathcal{A}}}(x'), \mathbf{h}_l^{f_{\mathcal{V}}}(x)),$$
$$\text{s.t.} S(x', x) \geq \tau_S, \left|\mathcal{T}_{f_{\mathcal{A}}}(x') \cap \mathcal{T}_{f_{\mathcal{A}}}(x)\right| / \left|\mathcal{T}_{f_{\mathcal{A}}}(x)\right| \geq \tau_{tm} \quad (1)$$

where $\mathrm{dom}\mathcal{T}$ denotes the tokenizer's input domain, d is the distance metric between two ISs, $S$ evaluates semantic similarity, $\tau_S$ (resp., $\tau_{tm}$) is a threshold of $S$ (resp., token matching).

**Adversary Knowledge.** We assume the adversary knows the layer $l$ of ISs and consider two settings: 1) The adversary has white-box access to the weights (i.e., $f_{\mathcal{A}} = f_{\mathcal{V}}$). For instance, the collaborative inference PETAL [30] requires the adversary server to know the whole weights so as to select its layers during the load balancing [4], thus know the layer index for attacks; 2) The adversary has no knowledge (black-box access) of the model weights and can only observe the ISs. A typical example can be third-party model behavior auditing [7, 8, 9, 36] by probing ISs of specific layers[1] for the layer-specific detector training [37].

**Adversary Capacities.** We consider passive attacks thus the adversary cannot interact with user or manipulate the deployed model. However, the adversary can store the observed ISs $\mathbf{h}_l$ (we omit $f_{\mathcal{V}}$ and $x$ for simplicity) from the target model $f_{\mathcal{V}}$ and have computational resources enough for inversion but insufficient for brute force search. The adversary can also query the target model to obtain ISs of her owned data $\mathcal{X}_{\mathcal{A}}$ which is of distribution similar to but not exactly as the victim's query. For example, the adversary may have general instruction tuning data but the victim queries are from specific domains (e.g., coding), which are not known by the adversary because of no interaction.

## 4 Internal States Inversion

In this section, we first overview the attacks and clarify the terminology. Then, we elaborate our attack insight and method.

[1] https://github.com/microsoft/TaskTracker/blob/main/task_tracker/utils/activations.py

### 4.1 Overview

Our attack framework proceeds according to the adversary's knowledge to the target LLM. Figure 2 shows the overall workflow of our attacks based on the adversary's capacities. In particular, in cases of white-box access to the model weights, our *optimization-based attack* iteratively updates the inverted input by matching the observed target ISs. One important advantage is that, there is no assumption on the knowledge of victim's prompt domain and length (i.e., *data-agnostic*). We propose two attacks, Embedding Recovery (ER) and Token Basis Selection (TBS), targeting shallow and deep ISs respectively.

Without access to the weights, the adversary follows our black-box attacks. Due to high cost of pretraining, the target LLM is likely to be derived from public open-source models through finetuning or merging. As a result, the first step is to determine whether the target model is derived from any known base LLMs type. If it is, the adversary trains a surrogate model and apply our optimization-based attacks. If not, our *generation-based* attack trains an inversion model based on the ISs queried by the adversary's surrogate data. To tackle the challenge of semantic irrelevance, we introduce a projection module based on sparse encoder [20] and translation model to enhance the inversion. This can be weaker than previous white-box attacks because the knowledge of query data can limit the inversion accuracy in case of distinct data distribution. Note that the black-box generation-based attack can also work on open-sourced LLMs, as long as the adversary only needs to observe ISs to train inversion models.

### 4.2 White-box Inversion

We first introduce the strawman approach, then present our proposed two white-box attacks, Embedding Recovery (ER) and Token Basis Selection (TBS), for inverting ISs of shallow layers and deep layers, respectively.

**Strawman Approach.** Inspired by previous optimization-based embedding inversion [18, 38], the strawman attack, Token Selection (TS), typically assigns trainable variables

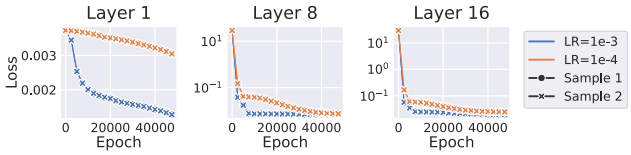


Figure 3: Evaluation of strawman attack TS on `Llama-3-8B-Instruct`. The strawman approach, TS, fails to converge on the deeper layers even under improved settings. Moreover, when attacking the first layer, the inverted texts contain no overlapping tokens with the input texts.

$\mathbf{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_{|\mathbf{h}_l|}] \in \mathbb{R}^{|\mathbf{h}_l| \times N_\mathcal{T}}$ of each embeddings to invert $i$-th input token of by $\arg\max \widehat{\mathbf{z}}_i$, where $\widehat{\mathbf{z}}_i$ are the rows of

$$\widehat{\mathbf{Z}} = \arg\max_{\mathbf{Z}} \left\| \psi_l(\text{softmax}(\mathbf{Z}/T) \cdot \mathbf{w}_{in}^{f_\mathcal{A}}) - \mathbf{h}_l^{f_\mathcal{V}} \right\|_2, \quad (2)$$

which is obtained through gradient-based optimization. Nevertheless, TS can fail to locate candidate tokens for LLMs with higher depth and larger token dictionary due to convergence into local minimum. In Figure 3, we apply TS on ISs from different layers of `Llama-3-8B-Instruct` and evaluate different settings. We observe that TS gets halted for deeper layers and output non-readable inverted texts (more quantitative evaluation in Section 5.2).

**Our Approach.** Instead of direct token inversion, our intuition is to first approximate the dummy input embeddings $\widehat{\mathbf{w}}$ that matches with the adversary-observed ISs $\mathbf{h}_l^{f_\mathcal{V}}$, and then invert the candidate tokens as those of highest cosine similarity with $\widehat{\mathbf{w}}$. Algorithm 1 shows the overview of ER and TBS attacks, where the blue (resp., red) blocks refer to the ER (resp., TBS) attack, and the rest is shared by two attacks. Note that the model weights are fixed during optimization.

*Embedding Recovery (ER).* As shown in Figure 6 of Section 5.2, for shallow layers, the gradients on dummy embedding $\nabla\widehat{\mathbf{w}}$ are of smaller magnitude, thus can avoid local minimum with stable convergence. Therefore, we directly optimize $\widehat{\mathbf{w}}$ for ISs matching with $\mathcal{L}_{im} := \mathrm{d}(\psi_l^{f_\mathcal{A}}(\widehat{\mathbf{w}}), \mathbf{h}_l^{f_\mathcal{V}})$. Common choices of d include $\mathcal{L}_p$ norm and cosine distance.

In experiments, we notice overfitting during inversion, which results in the optimized $\widehat{\mathbf{w}}$ of higher norm than embeddings from $\mathbf{w}_{in}^{f_\mathcal{A}}$ and causes incorrect token recovery in the second phase. Thus, to ensure $\widehat{\mathbf{w}}$ is similarly distributed to $\mathbf{w}_{in}^{f_\mathcal{A}}$, we introduce a penalty term based on distribution matching [39]: $\mathcal{L}_{dm} = \mathbb{E}_{\varphi\sim\mathcal{P}_\varphi} \left\| \overline{\varphi(\widehat{\mathbf{w}})} - \overline{\varphi(\mathbf{w}_{in}^{f_\mathcal{A}})} \right\|_2$, where $\varphi$ denotes random Gaussian neural network as universal feature extractor and the overline denotes averaging. In practice, we sample random embedding batches of equal size from $\widehat{\mathbf{w}}$ and $\mathbf{w}_{in}^{f_\mathcal{V}}$ before averaging in each step. In total, the inversion loss is:

$$\mathcal{L}_{inv} = \mathcal{L}_{im} + \lambda \cdot \mathcal{L}_{dm}, \quad (3)$$

**Algorithm 1:** ER ( blue ) and TBS ( red ) attacks

**Input:** The adversary's model $f_\mathcal{A}$ with input embedding weight $\mathbf{w}_{in}^{f_\mathcal{A}}$ and the first $l$ Transformer layers $\psi_l^{f_\mathcal{A}}$, the tokenizer $\mathcal{T}_{f_\mathcal{A}}$, target hidden states $\mathbf{h}_l^{f_\mathcal{A}}$ of length $\left|\mathbf{h}_l^{f_\mathcal{A}}\right|$, learning rate $\mu$, optimizer `optim`, distance metric `dist`, steps $E$

**Output:** Inverted prompt $\hat{x}$.

Initialize $\mathcal{L} \leftarrow []$, $\mathcal{Z} \leftarrow []$;

Initialize $\widehat{\mathbf{w}} \leftarrow \mathbf{0}$ where $\widehat{\mathbf{w}} \in \mathbb{R}^{\left|\mathbf{h}_l^{f_\mathcal{V}}\right| \times d_{in}}$;

$\mathbf{U}, \Delta, \mathbf{V}^\top \leftarrow \text{SVD}(\mathbf{w}_{in}^{f_\mathcal{A}})$ where $\mathbf{V} \in \mathbb{R}^{d_{in} \times d_{in}}$. Set $\mathbf{B} \leftarrow \mathcal{V}^\top$;

**if** *Apply Unbiased Basis* **then**

  $\mathbf{B} \leftarrow \mathcal{V}$;

Initialize projection weights $\mathbf{z} \leftarrow [\frac{1}{d_{in}}]$ and $\mathbf{z} \in \mathbb{R}^{\left|\mathbf{h}_l^{f_\mathcal{V}}\right| \times d_{in}}$;

`// 1. Optimization.`

**for** $i \leftarrow 1$ **to** $E$ **do**

  $\widehat{\mathbf{w}} \leftarrow \varphi_\mathbf{z}(\mathbf{z} \cdot \mathbf{B})$;

  Compute $\mathcal{L}_{inv}$ with Equation (3) and save as $\mathcal{L}[i]$.

  $\widehat{\mathbf{w}} \leftarrow \texttt{optim}(\mathcal{L}[i], \widehat{\mathbf{w}}, \mu)$; $\mathcal{Z}[i] \leftarrow \widehat{\mathbf{w}}$;

  $\mathbf{z} \leftarrow \texttt{optim}(\mathcal{L}[i], \mathbf{z}, \mu)$; $\mathcal{Z}[i] \leftarrow \mathbf{z}$;

`// 2. Prompt Inversion.`

$\widehat{\mathbf{w}} \leftarrow \mathcal{Z}[\arg\min_i \mathcal{L}]$;

$\widehat{\mathbf{w}} \leftarrow \varphi_\mathbf{z}(\mathcal{Z}[\arg\min_i \mathcal{L}] \cdot \mathbf{B})$;

$\widehat{\mathbf{w}} \leftarrow \mathcal{W}[\arg\min_i \mathcal{L}]$;

$\mathbf{t} \leftarrow \arg\max_{row}[\widehat{\mathbf{w}} \cdot (\mathbf{w}_{in}^{f_\mathcal{A}})^\top / (\|\widehat{\mathbf{w}}\|_{RN} \odot \left\|\mathbf{w}_{in}^{f_\mathcal{A}}\right\|_{RN})]$;

**return** $\mathcal{T}_{f_\mathcal{A}}$.`decode`$(\mathbf{t})$;

where $\lambda$ balances the inversion and the penalty.

*Token Basis Selection (TBS).* On deep layers, the gradients on the dummy embedding $\nabla\widehat{\mathbf{w}}$ have increased magnitude because of more back-propagated gradients from previous layers accumulated by the chain rule, which destabilizes the inversion. Therefore, our idea is to find the correct projection values $\widehat{\mathbf{z}}$ of an orthogonal basis of $\mathbf{w}_{in}^{f_\mathcal{A}}$ to compose $\widehat{\mathbf{w}}$, as illustrated in Figure 4. In contrast to TS, our TBS has much smaller search space. For example, on `Llama-3-8B-Instruct`, the search space is reduced by 30 times (i.e., 4k for TBS v.s. 120k+ for TS to recover one token). Compared with ER, the gradients on $\widehat{\mathbf{z}}$ is stabilized because of an additional gradient term from the chain rule $\partial\widehat{\mathbf{w}}/\partial\widehat{\mathbf{z}}$ which is close to the small-scaled basis. Inspired by TS, we also introduce a variable change function $\varphi_\mathbf{z}$ that bound the scale of $\mathbf{z}$. For example, in our experiments, we use arctan for $\varphi_\mathbf{z}$ to bound the $\mathbf{z} \cdot \mathbf{B}$. More detailed analysis can be found in Appendix A.

As the orthogonal basis is not unique, we propose two strategies: use the basis $\mathbf{V}$ from SVD decomposition of $\mathbf{w}_{in}^{f_\mathcal{A}}$ or the $\mathbf{V}^\top$ as an unbiased basis. Figure 5 compares the two strategies where we observe input embeddings are more cen-

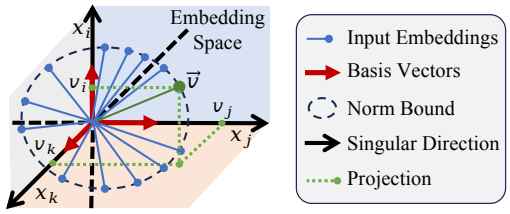


Figure 4: Intuition behind our TBS inversion attack. Instead of directly selecting candidate tokens from a large dictionary, our TBS attack optimizes weights among much fewer singular basis vectors (e.g., $v_i$, $v_j$ and $v_k$) to restore the candidate input embeddings (e.g., $\vec{v}$).

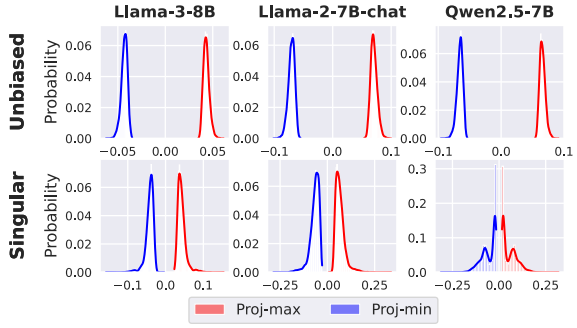


Figure 5: Comparison between unbiased basis and singular basis of LLM's input embedding matrix for TBS attack. We show the histogram of maximum and minimum projection values of input embedding on the basis and found the singular basis is more biased towards particular token groups because of higher projected values.

tered around certain basis vectors of $\mathbf{V}$ (i.e., higher projection values) where the projections of $\mathbf{w}_{in}^{f_{\mathcal{A}}}$ on unbiased basis $\mathbf{V}^{\top}$ are more uniformly distributed (i.e., maximum projection values are bounded around 0.1). We provide visualization to support the claim in Figure 17 of Appendix A, and further compare two basis choices in Section 5.2.

## 4.3 Black-box Inversion

The target LLM should be either finetuned or merged from an open-sourced base model or pretrained as a close-sourced model. The adversary first identifies the model type based on the observed ISs and then proceeds the extended optimization-based or generation-based inversion.

**Model Type Identification.** Our insight is that the ISs of derived models remain proximate to their base models because of minor weight perturbations during finetuning or merging. As evidenced in Figure 15, ISs exhibit tight intra-model clustering and clear inter-model separation across pretrained architectures. To realize fine-grained detection, we design an ensemble autoencoder to check if the deployed model is derived from adversary-known open-sourced ones. Specifically, the adversary queries a pretrained LLM $f$ with $\mathcal{X}_{\mathcal{A}}$ and obtains corresponding ISs $\mathbf{h}_l^f(\mathcal{X}_{\mathcal{A}})$. The adversary trains an autoencoder on $\mathbf{h}_l^f(\mathcal{X}_{\mathcal{A}})$ to detect this LLM type. By repeating this process on various pretrained models, the adversary trains a set of autoencoders $\{\phi_i\}_i$ for detecting LLMs $f_i$. In the test time, the adversary queries $f_{\mathcal{V}}$ with $\mathcal{X}_{\mathcal{A}}$ to obtain given the target ISs $\mathbf{h}_l^{f_{\mathcal{V}}}(\mathcal{X}_{\mathcal{A}})$ and harnesses the autoencoders $\{\phi_i\}_i$ to predict in an ensemble way:

$$y := \arg\min_i \{\phi_i(\mathbf{h}_l^{f_{\mathcal{V}}}(\mathcal{X}_{\mathcal{A}})) | \phi_i(\mathbf{h}_l^{f_{\mathcal{V}}}(\mathcal{X}_{\mathcal{A}})) \leq \tau\}. \quad (4)$$

There are two possible outcomes. First, the target LLM is derived from one of mainstream LLMs via finetuning, adapter or model merging. The adversary adopts this LLM for further attack. Second, there is no such LLM, and the model parameter is close-sourced. Then, the adversary can reuse $(\mathcal{X}_{\mathcal{A}}, \mathbf{h}(\mathcal{X}_{\mathcal{A}}))$ to train a generative inversion model.

**Extended Optimization-based Attack.** The adversary aims to replicate the target model with a surrogate LLM $f_{\mathcal{A}}$ that satisfies $\mathbf{h}_l^{f_{\mathcal{A}}}(x) \approx \mathbf{h}_l^{f_{\mathcal{V}}}(x)$ for any input $x$. Inspired by the model distillation, we propose to replicate the ISs with the pretrained base model $f_{base}$ of the identified type:

$$f_{\mathcal{A}} := \arg\min_{f_{base}} \left\| \mathbf{h}_l^{f_{base}}(x_i) - \mathbf{h}_l^{f_{\mathcal{V}}}(x_i) \right\|_2, \forall x_i \in \mathcal{X}_{\mathcal{A}}. \quad (5)$$

Then, the adversary then applies our white-box attacks (e.g., TBS) onto the replicated model $f_{\mathcal{A}}$ to invert input tokens of observed ISs $\mathbf{h}_l^{f_{\mathcal{V}}}$.

**Generation-based Attack.** In case where the model is drastically different from the known LLM types, such as the target LLM is close-sourced, we propose to train an inversion model that translates the observed ISs $\mathbf{h}_l^{f_{\mathcal{V}}}$ into inputs. Section 3 assumes the adversary data $\mathcal{X}_{\mathcal{A}} = \{x_i\}_i$ are of similar distribution to the victim's query, thus, before the attack, the adversary first trains an inversion model $\theta$ with previously obtained $(\mathcal{X}_{\mathcal{A}}, \mathbf{h}_l^{f_{\mathcal{V}}}(\mathcal{X}_{\mathcal{A}}))$.

We use an encoder-decoder model $\theta$ for inversion because of its wide usage for translation and that the inversion essentially translates the observed ISs into input tokens. However, the ISs are generated for continuous inference and deviate from the semantic meaning. Besides, the target ISs may have incompatible representation space with the encoder (e.g., different dimension). Therefore, we propose to use a projection module $\varphi_\theta : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{enc}}$ on top of the inversion model to project the ISs into the encoder's embedding space (width $d_{enc}$). In total, the inverted input is $\hat{x} \sim \theta(\varphi_\theta(\mathbf{h}_l^{f_{\mathcal{V}}}))$. Note that the projection module is necessary for aligning ISs. For instance, on the middle ISs of GPT-2, the projection module brings 32.81% inversion improvement (F1 score).

**Analysis from Bayesian Perspective.** To better understand our generation-based inversion, we analyze the inversion

Table 2: The statistics of LLM considered in this work.

| LLM | # Params (B) | # Token | Width | Depth |
|---|---|---|---|---|
| LLama-2-7B-chat | 6.74 | 32,000 | 4096 | 32 |
| Llama-3-8B-Instruct | 8.03 | 128,000 | 4096 | 32 |
| Llama-3.1-8B-Instruct | 8.03 | 128,000 | 4096 | 32 |
| Llama-3.3-70B-Instruct | 70.55 | 128,000 | 8192 | 80 |
| Qwen2.5-7B-Instruct | 7.62 | 151,643 | 3584 | 28 |
| Qwen2.5-Coder-7B-Instruct | 7.62 | 151,646 | 3584 | 28 |

based on Bayes' theorem. The inversion can be formulated as $\arg\max_x p(x|\mathbf{h},\theta)$, where we omit the projection $\varphi_\theta$ for simplicity. We can apply the Bayes's theorem and obtain

$$p(x|\mathbf{h},\theta) = \frac{p(\mathbf{h}|x,\theta)p(x|\theta)}{p(\mathbf{h}|\theta)}. \quad (6)$$

Notice that the observed ISs $\mathbf{h}$ and $\theta$ are known, thus the inversion can be formulated as:

$$\arg\max_x p(\mathbf{h}|x,\theta)p(x|\theta), \quad (7)$$

where $p(\mathbf{h}|x,\theta)$ means the posterior probability of $\mathbf{h}$ given prompt input $x$ thus depends on the deployed LLM. Therefore, the key is to maximize $p(x|\theta)$ where $\theta$ is dependent on the adversary's training data $\{(x_i,\mathbf{h}_i)\}_i$. Consequently, the adversary should minimize the distribution discrepancy between the adversary's data $\mathcal{X}_\mathcal{A}$ and the victim's prompt input.

# 5 Evaluation

In this section, we evaluate the effectiveness of our attack and compare with previous approaches. Then we investigate various attack settings. Finally, we study potential mitigation.

## 5.1 Experimental Setup.

**Models.** Throughout the evaluation, we mainly use the `Llama-3-8B-Instruct` (Llama-3) [2], `Qwen2.5-7B-Instruct` (Qwen2.5) [40] and `Qwen2.5-Coder-7B-Instruct` (Qwen2.5-Coder) as the most popular open-source LLMs. In addition, we also use `Llama-2-7B-Chat` (Llama-2) to fairly compare with prior work. To evaluate our black-box attacks, we adopt `Bio-Medical-Llama-3-8B` [41] (Llama-3-Doctor), which is a highly-downloaded model finetuned from `Llama-3-8B-Instruct` with medical instruction following data and `Llama-3.1-8B-Instruct` (Llama-3.1), which enhances Llama-3 in terms of multilingual capacities and context length via post-training [42]. For larger models, we use `Llama-3.3-70B-Instruct` (Llama-3.3-70B). The detailed statistics of base models are provided in Table 2.

**Metrics.** We compute Cosine Similarity (CS) between the embeddings of original and inverted sentences. To ensure reproducibility, we use an open-source embedding model `bge-large-en-v1.5` [43] of edge-cutting performance on Massive Text Embedding Benchmark [44, 45]. In addition, we use Exact Matching (EM) rate among the test dataset to evaluate the precision of our attacks. We also use F1 score to evaluate the matched tokens by Llama-3's tokenizer. Besides, we also consider widely used metrics BLEU and ROUGE to evaluate the semantic similarity.

Table 3: Comparison with previous attacks based on the ISs, logits and outputs. We report the mean for each metric and the standard error of the mean (SEM).

| Attack | CS | BLEU | ROUGE | EM | F1 |
|---|---|---|---|---|---|
| TS [18] (L=2) | 44.24± 0.32 | 0.00± 0.00 | 0.00 | 0.00± 0.00 | 0.00 ± 0.00 |
| logit2text [24] | 88.12± 0.57 | 56.74± 1.44 | 0.72 | 25.40± 0.02 | 75.22 ± 1.00 |
| output2text [20] | 93.26± 0.33 | 55.00± 1.46 | 0.77 | 16.21± 0.02 | 75.94 ± 0.99 |
| ER (L=2) | 94.24± 0.47 | 74.89± 1.56 | 0.87 | 52.61± 0.02 | 88.22 ± 0.90 |
| ER (L=8) | 95.74± 0.38 | 72.91± 0.94 | 0.89 | 6.05± 0.01 | 87.86 ± 0.69 |
| TBS (L=8) | 90.96± 0.82 | 77.38± 1.55 | 0.83 | 48.23± 0.02 | 82.61 ± 1.44 |
| TBS (L=16) | 83.70± 0.99 | 59.89± 1.88 | 0.70 | 31.73± 0.02 | 69.13 ± 1.72 |
| TBS (L=24) | 88.07± 0.83 | 65.47± 1.69 | 0.77 | 27.35± 0.02 | 75.32 ± 1.53 |
| TBS (L=32) | 81.34± 0.60 | 32.67± 1.19 | 0.62 | 3.97± 0.01 | 55.97 ± 1.02 |

## 5.2 White-box Inversion Attack

In this section, we first evaluate our white-box attack on the Instruction-2M test data and compare with prior work. Then, we show that our attacks can nearly fully invert the long prompt through case studies on long-context queries of medical consult and coding assistance.

**Attack Settings.** We consider the layers of index 2, 4, 8, 16, 24 for the 32-layered Llama-3 models and consider the layer of index 14 for the 28-layered Qwen2.5 models. We use the optimizer AdamW [46] with learning rate $\mu \in \{10^{-3}, 5\times 10^{-4}, 1\times 10^{-4}\}$. We set the penalty weight $\lambda \in \{0, 10^{-3}\}$. In default, we use the Mean Squared Error (MSE) to measure the distance between ISs and the unbiased basis for our TBS attack. We use scipy [47] to calculate the singular vectors in our TBS attack. In our attacks, we initialize the dummy input embeddings with zero for ER and with $1/d_{in}$ for the baseline TS and our TBS attacks. We also consider two distance metrics between ISs in the loss: MSE and cosine similarity (COS). For simplicity, the default setting uses MSE as the distance metric, sets $\mu = 5\times 10^{-4}$ and $\lambda = 0$, and $E = 5\times 10^4$. For TBS attack, we use unbiased basis in default and set $\varphi_\mathbf{z} = \alpha\arctan(\cdot)$ with $\alpha = 5/\pi$ to accelerate convergence. In later sections, we will see that the optimal setting may vary according to the input data and the deployed model.

### 5.2.1 Comparison

We first compare our white-box attacks with previous attacks and later compare with our Table 3 shows the results.

**Baselines.** We compare with output2text [20] and logit2text [24] as two recent state-of-the-art inversion

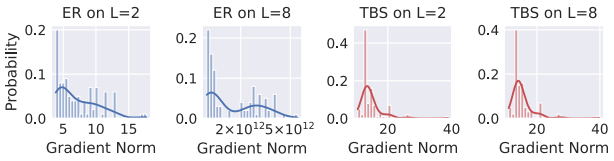


Figure 6: Distribution of gradient norm $\|\widehat{\mathbf{w}}\|_2$ and $\|\widehat{\mathbf{z}}\|_2$ for ER and TBS attacks, respectively.

attacks against LLMs, because the adversary (e.g., collaborative inference server) can also exploit the output and logits for inversion. For fair comparison, we use their test dataset of Instruction-2M and manually verify 479 common samples from the two official implementations [20, 24]. The maximum length of test samples is 63. Besides, we also compare with TS baseline [18].

In Table 3, we compare our proposed attacks with previous black-box and white-box attacks on different layer depths $L = 2, 8, 16, 24$ on a 32-layered Llama-2. Our ER attack can achieve high similarity scores on low layers and our TBS attack can remain effective even in deep layers like $L = 24$. We set $\mu = 0.001$ for layer index lower than 8 and $\mu = 0.01$ for the deeper layer to accelerate the convergence. Notably, our attacks achieve significant higher EM rates than prior work, indicating that the ISs can reveal (almost) identical text information as the raw input text.

*Cause Analysis.* We then proceed detailed analysis of inversion results. As discussed in Section 4, TS directly selects candidate tokens for inverted input text, which does not work as the similarity scores between inverted and real input texts are close to zero. In fact, the inversion loss converges around $10^{-2}$ and cannot be improved by tuning learning rates. On the other hand, previous black-box attacks, exploiting the output logits or texts, result in comparable inversion in terms of semantics and token matching, and slightly lower EM score, because of the randomness during inversion model generation and higher information loss in the model output.

As for our attacks, ER performs better at shallow layers and TBS can remain effective in deep or even the close-to-last layers. Notably, the ER attack can exactly invert more than half input texts on the shallow layer $L = 2$. This can be realistic in split learning with resource-constraint edge devices or small-sized enclave (e.g., SGX-v1) where the victim can only infer one or two shallow layers.

As for deep layers, the ISs contain less input text features and are more difficult to invert. Thus, even though the semantics are preserved, our ER attack's on deeper layer $L = 8$ has significantly lower EM rate because of more noisy tokens (e.g., inversion as "två017"" for "2/2/2017") added during inversion. From the optimization perspective, higher depth can lead to gradient exploding causing the directly recovered input embedding dissimilar to the actual token.

On the other hand, our TBS attack can stabilize the optimization curve by preserving the gradient magnitude. In Figure 6, we testify this through distribution of the gradient norm on the test texts, where we can observe that ER on deep layer (L=8) generates gradient of magnitude $10^{12}$ while our TBS on deep layers can maintain the gradient norm of the similar magnitude as on shallow layers. Therefore, the exploded gradients cause the convergence on local minimum and lead to low inversion quality. Note that the gradients of TBS on L=8 have sightly larger norm than L=2, which explains why inversion on deeper layer should use lower learning rates. As for qualitative results on the middle layer (L=16), on a random subset from Instruction-2M of 200 samples, ER only achieves CS score 50.35 and F1 score 7.56, which is significantly lower than our TBS attack shown in Table 3.

*Results on Middle Layer.* We note that the inversion by our TBS attack on the middle layer, although evaluated better in terms of EM rates, are worse than the deeper layer (L=24) in terms of semantic similarity scores and F1 scores. We investigate the inverted texts and found that there are also numerous noisy tokens from other languages (e.g., Russian and Korean) replaced for the original English words. More noisy tokens cause the readability degradation of inverted texts, thus lower the semantic similarity scores and token F1 score. Although we tried lower learning rate $\mu = 0.001$, the inversion on the middle layer does not get improved. We suspect the reason lies in the optimization dynamics: the TBS inversion on L=16 gets saturated after 20,000 steps but keeps improving until steps 30,000 for L=24. In future work, we will investigate real causes from the perspective of training dynamics.

*Inversion on Last Layer.* We apply TBS attack on the last layer L=32. As Table 3 shows, the depth further degrades inverted texts, which partially aligns with prior work [34] that deep layers capture more complicated concepts instead of simple features of input texts. Nonetheless, our attack validates a counterintuitive finding: high-depth latent layer can still leak the original inputs in entirety (EM≈4).

**Takeaway:** On short-context inputs, our optimization attacks outperform TS, achieve comparable inversion as previous generative inversion.

#### 5.2.2 Case Studies: Long-context Inversion

In the remainder of the paper, we evaluate how our inversion attack perform on long-context prompt through two case studies of privacy-sensitive tasks: healthcare consulting and coding assistance. We use Llama-3 for its support of longer context and choose the middle layer (i.e., L=16) as it is reported best to probe the LLM [7, 16] and balances the inference cost between two parts in collaborative inference. Thus,

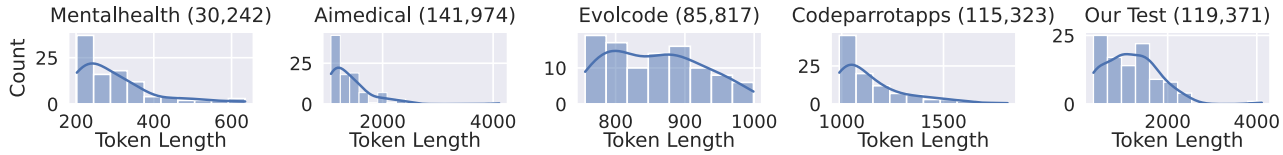


Figure 7: Distribution of token length from our test data (top 100 longest inputs of long-context benchmarks). The rightmost corresponds to the evaluation set for our TBS attack which contains longest 25 prompts from each benchmarks. We show the total token amount along in the figure title.

we mainly investigate our TBS attack. In addition, we investigate optimal optimization strategies (e.g., distance metric in loss) for different tasks.

**Datasets.** We consider datasets of two privacy-sensitive tasks: healthcare dialogue and coding assistance. For medical data, we use the symptom descriptions from Aimedical [48] and Mentalhealth (MH) [49]. For coding assistance, we use two coding problem datasets Evolcode (EC) [50] and CodeparrotApps (CA) [51] that contain prompts asking LLM to solve coding problem. Figure 7 plots the distribution of length of top 100 long prompts for the four benchmarks, from which we can find that Aimedical contains the longest prompt of 4,112 tokens and CodeparrotApps has longer coding prompts than Evolcode. We use the top 25 longest prompts from each dataset to evaluate our TBS (right-most of Figure 7).

**Our attack can scale to long-context inversion.** Figure 9 compares the similarity metrics for different attack settings. Note that due to long length and noisy inverted tokens, the EM rates are all zero, thus we omit the EM results to save space. Specifically, the optimal settings are reported for lower learning rate $\mu = 0.0001$ with COS as the distance metric, which leads to 99.4 CS and 97.88 F1 on Mentalhealth, and 98.12 CS and 96.7 F1 on Aimedical. Both result in 0.99 Rouge score. Next, we show examples of long-context inversion.

**Example: Healthcare Dialogue.** We begin with examples of healthcare dialogues. Figure 8 shows an inversion example of 384 tokens sampled from Mentalhealth. The missing tokens are highlighted in color. This example shows that our inversion attack can nearly invert the whole prompt text except for a small proportion of tokens.

**Comparison & Qualitative Analysis.** Prior work can fail for long-context inputs. For the above example in Figure 8, we test previous state-of-the-art output2text [20] pretrained on Llama-2 and unbridle the maximum sequence to 4,096. The output is "How do you manage the tension and tension that is causing you to go crazy?", which is completely different. On the contrary, ours only misses certain tokens.

We found that the missed tokens are synonym of the ground truth tokens and have similar input embeddings. Take the first missed token in Figure 8 as an example, the ground truth token is "friend" while the inversion is "boyfriend" because its embedding is the most similar to the inverted input embedding. After a manual checking, we found the embedding of the ground truth token "friend" is the third most similar token (which is apparently similar to the embedding of "boyfriend"), thus is missed during token generation. That is why smaller learning rates could benefit the inversion because of more refined updating. One potential improvement could be beam searching to cover all the possible paths.

*Learning Rates & Distance Metric.* To begin with, we investigate different learning rates and distance metrics of our TBS attack in Figure 9. The results are expected because the self-attention leverages inner product to compute attention and subsequent ISs, which makes COS more reliable distance metric than MSE. However, computing COS requires higher VRAM because the matrix computation has space complexity $O(n^2)$ for a $n$-length prompts. Besides, as mentioned earlier, we found smaller learning rates improve the inversion on deep layers. The potential reason can be more fine-grained input embedding inversion, which leads to more accurate candidate token recovery. Next, we explore to improve the attack performance by additional settings.

*Improved Attack Settings.* Table 4 shows inversion with application of our token distribution matching penalty and usage of SVD singular basis. We make two observations: 1) the penalty can improve the inversion at larger learning rate 0.0005 which may be useful to accelerate inversion through faster convergence (i.e., fewer steps). 2) usage of SVD singular basis can also improve the inversion performance regardless of learning rates, possibly because it enables more precise input embedding inversion than unbiased basis. Nevertheless, SVD singular basis may not work if applying optimization-based attack to derivatives under the black-box setting (see Section 5.3).

*Larger Models.* Our white-box attacks are size-agnostic thus can scale to larger models, at the expense of higher VRAM cost and optimization time. As shown in the Table 4, our TBS attack remains effective for Llama-3.3-70B on Mentalhealth. Notably, compared to smaller 7B Llamamodel, the inverted texts are closer to the ground truth as indicated by higher CS and F1 scores, possibly because of more information retained in wider ISs (i.e., 8192 for 70B and 4096 for 7B). However, as the TBS attack requires forwarding and backpropagation to iteratively update the inverted input, larger models can significantly increase the optimization time and GPU memory. For example, attacking 70B is 7 times slower and costs 10 times more memory than 7B.

| Input | Inversion |
|---|---|
| I am constantly having problems with the same two people who will always be in my life. I had a daughter with my ex-boyfriend. I am now married, and my husband's ex-girlfriend is involved with my ex-boyfriend. They also have a daughter together. My issue is that there is always drama. I am pregnant, and I told my ex-~~b~~boy~~frien~~friend~~d~~ that I don't want any drama or arguments. I want to get along as much as possible, and he agreed. However, we just had an incident where my ex-boyfriend started discussing drop-off details about my stepdaughter. I told him that he needed to ask my husband because I can't make decisions about my stepdaughter regarding the matter. That led to an argument. I told him all my concern is when I pick up my daughter. ~~M~~My stepdaughter's pick-up details are between my husband and his ex-girlfriend. I especially told him I di~~d~~n't want to be involved. Somehow, he turned it around and then ~~wa~~wante~~d~~ to change the schedule we agreed on. He threatened me and got ugly because I would~~n~~n't discuss my stepdaughter's matters with him. The point is there is so much drama. I try my best to get along with everyone. I don't understand where I went wrong (besides reply~~ing~~ back to his question). I feel like I'm going crazy because this is a constant battle where everyone's frustrations are taken out on each ~~other~~, and ~~i~~t's the children that are hurting. I had a party planned for my daughter's birthday, and my ex-boy~~friend~~ told me to cancel those plans because he would~~n~~n't let me have her. In my eyes, it's the child that is hurting. I was throwing a party for her birthday, and because of the problem with stupid pick-up details about my stepdaughter, which I have no control over, he took it out on our daughter. | I am constantly having problems with the same two people who will always be in my life. I had a daughter with my ex-boyfriend. I am now married, and my husband's ex-girlfriend is involved with my ex-boy boyfriend. They also have a daughter together. My issue is that there is always drama. I am pregnant, and I told my exSony>';\n\n that I don't want any drama or arguments. I want to get along as much as possible, and he agreed. However, we just had an incident where my ex-boy boyfriend started discussing drop-off details about my stepdaughter. I told him that he needed to ask my husband because I can't make decisions about my stepdaughter regarding the matter. That led to an argument. I told him all my concern is when I pick up my daughter. my stepdaughter's pick-up details are between my husband and his ex girlfriend. I especially told him I doesn't want to be involved. Somehow, he turned it around and then<\|begin_of_text\|> to change the schedule we agreed on. He threatened me and got ugly because I doesn't discuss my stepdaughter's matters with him.\n The point is there is so much drama. I try my best to get along with everyone. I don't understand where I went wrong (bes besides rep reply back to his question). I feel like I'm going crazy because this is a constant battle where everyone's frustrations are taken out on each birbir, and It's the children that are hurting. I had a party planned for my daughter's birthday, and my ex-boyildiği told me to cancel those plans because he didn't let me have her. In my eyes, it's the child that is hurting. I was throwing a party for her birthday, and because of the problem with stupid pick-up details about my stepdaughter, which I have no control over, he took it out on our daughter. |

Figure 8: An inversion example from Mentalhealth consisting of 384 tokens. The missed tokens are highlighted in color.

Table 4: Evaluation of different TBS attack settings for Llama-3 model.

| Dataset | Size | Basis | Penalty | $\mu$ | CS | BLEU | ROUGE | F1 |
|---|---|---|---|---|---|---|---|---|
| EC | 8B | Unbiased | 0 | 1e-4 | 95.49± 0.59 | 64.93± 2.90 | 0.90 | 76.22 ± 2.11 |
| | | | | 5e-4 | 88.79± 0.83 | 27.06± 2.04 | 0.66 | 49.56 ± 1.65 |
| | | Unbiased | 1e-3 | 1e-4 | 94.73± 1.28 | 58.09± 6.56 | 0.87 | 72.48 ± 3.70 |
| | | | | 5e-4 | 90.82± 1.05 | 34.20± 3.03 | 0.71 | 54.70 ± 2.29 |
| | | SVD | 0 | 1e-4 | 95.87± 0.95 | 64.34± 3.05 | 0.88 | 73.90 ± 2.60 |
| | | | | 5e-4 | 98.20± 0.49 | 72.09± 3.14 | 0.91 | 80.73 ± 2.22 |
| MH | 8B | Unbiased | 0 | 1e-4 | 95.02± 0.70 | 80.36± 1.47 | 0.95 | 90.63 ± 0.75 |
| | | | | 5e-4 | 88.59± 1.66 | 56.69± 3.04 | 0.87 | 78.53 ± 1.99 |
| | | Unbiased | 1e-3 | 1e-4 | 94.79± 0.77 | 77.55± 2.00 | 0.94 | 88.92 ± 1.20 |
| | | | | 5e-4 | 89.09± 0.79 | 59.03± 2.25 | 0.88 | 80.30 ± 1.08 |
| | | SVD | 0 | 1e-4 | 97.64± 0.39 | 86.70± 1.85 | 0.96 | 93.27 ± 1.05 |
| | | | | 5e-4 | 98.00± 0.57 | 90.24± 1.24 | 0.97 | 94.99 ± 0.61 |
| | 70B | Unbiased | 0 | 5e-4 | 95.80± 1.17 | 74.17± 9.44 | 0.94 | 88.34 ± 5.17 |

*Input Data Type.* We observe that the input data type can influence the inversion performance. In Figure 9, the coding data are more difficult to be inverted comparing to medical texts. Notably, the F1 score on coding datasets (92.05 and 94.50) are slightly lower than healthcare dialogue data under the optimal attack setting. After manual checking of inverted inputs, we found that the additional errors appear in the description of coding prompts instead of the main code. This may indicate that the model knowledge may also influence our optimization-based inversion attack because of the attention assignment to tokens of different topics is unequal. Therefore, in the following we investigate whether the coding-enhanced model can lead to higher inversion risk.

**Evaluation of Domain-specialized Models.** We consider Qwen2.5-Coder as the coding-specialized model because of its high ranking on the leaderboard [52]. Here we evaluate the top 50 longest coding prompts from two benchmarks. We also evaluate Qwen2.5 as a general-purpose model. Table 5 presents the results on two coding benchmarks. First, we observe that inversion is significantly better: the EM scores are around 50% for the top long prompts of both benchmarks and the inverted tokens F1 scores are around 99%. Second, compared to the general-purpose model Qwen2.5, the coding-specialized model Qwen2.5-Coder has ISs more susceptible to inversion attack because of higher EM rates.

Table 5: Evaluation results of Qwen2.5 and Qwen2.5-Coder.

| Dataset | Model | Distance | CS | BLEU | ROUGE | EM | F1 |
|---|---|---|---|---|---|---|---|
| EC | Qwen2.5 | MSE | 99.03± 0.26 | 98.90± 0.18 | 1.00 | 12.90± 0.06 | 98.90 ± 0.21 |
| | Qwen2.5-Coder | MSE | 99.78± 0.12 | 99.60± 0.13 | 1.00 | 64.00± 0.07 | 99.80 ± 0.06 |
| | Qwen2.5-Coder | COS | 99.96± 0.03 | 99.12± 0.34 | 1.00 | 45.00± 0.11 | 99.47 ± 0.14 |
| CA | Qwen2.5-Coder | MSE | 99.86± 0.08 | 99.78± 0.05 | 1.00 | 50.00± 0.07 | 99.82 ± 0.04 |
| | Qwen2.5-Coder | COS | 99.72± 0.28 | 99.46± 0.37 | 1.00 | 40.00± 0.16 | 99.76 ± 0.08 |

Surprisingly, we observe that the ISs of Qwen2.5 is easier to invert than Llama-3 models. We hypothesize no impact from their pretraining data and investigate the architecture difference. We exclude the causes from model width and trainable parameters up to the middle layer because they are similar for two models: Qwen2.5 is of width 3,584 and 3.8B parameters while Llama-3 is of width 4,096 and 4B parameters. After carefully examining their implementations, we found the main difference lies in the attention module: Qwen2.5 applies QKV bias [53] while Llama-3 does not by default. Therefore, we suspect that attention bias may amplify the feature representation of ISs thus, as a side effect, enable better inversion. Due to the huge cost of pretraining to obtain similar bias for Llama-3, we leave more detailed impact analysis of attention bias for future work.

**Takeaway:** On long-context inputs, our optimization-based attack TBS can invert nearly all tokens in the correct order to preserver semantics.

## 5.3 Black-box Inversion Attack

In this section, we evaluate and compare our replication-based and generation-based black-box inversion attacks.

**Attack Settings.** We use the prompts from Mentalhealth and Evolcode for test. For both black-box attacks, we assume the adversary uses NoRobots [54] as training data for replication model and inversion model training. NoRobots is composed

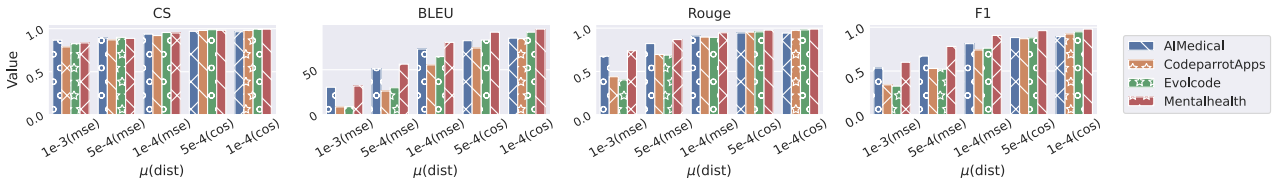


Figure 9: Evaluation of our TBS attack with different learning rates and distance functions (MSE and COS).

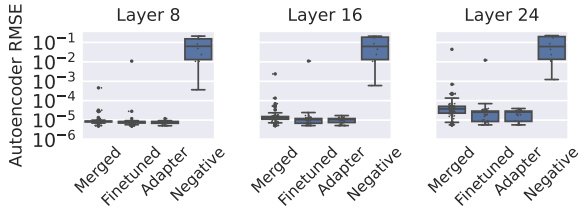


Figure 10: The distribution of autoencoder's reconstruction error to detect derivatives of Llama-3.

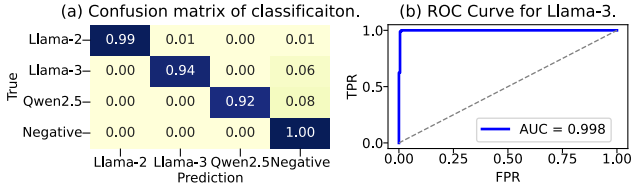


Figure 11: Evaluation of model type identification.

of 9.5k+ supervised finetuning samples from general topics, which aligns with our assumption that the adversary has data of similar distribution. We consider Llama-3 owned by the adversary and target Llama-3-Doctor and Llama-3.1 as two black-box deployed models. In this section we also test ISs of the middle layer as above.

**Model Type Identification.** Given unknown ISs, the adversary first identifies the deployed model type. To test the identify model type, we trained and crawled from Hugging Face 75 finetuned models, 70 merged models and 56 adapters of most downloading of Llama-3. In addition, we select 13 LLMs independent models (i.e., those not derived from Llama-3) as negatives, which are listed in the Appendix A.

Figure 10 demonstrates the distribution of autoencoder's Reconstruction MSE (RMSE) on three tested layers. For each tested layer, we use the test dataset of 500 samples of NoRobots as the probing data, and train the 3-layer autoencoder for 10 epochs. We found that most derivatives have separate error ranges to the independent models, which enables binary classifications by thresholding. Note that there are a small number of outliers that can be identified as independent model (False Negative). We check the false negatives and found they are caused by labeling errors based on crawled model name. Therefore, in the following test, we check the top downloading derivatives to remove potential errors.

To evaluate the identification of ensemble autoencoders, we additionally crawl the derivatives of Llama-2 and Qwen2.5 in a similar manner and ensemble the autoencoders of middle-layer ISs of three LLMs. Figure 11 exhibits the confusion matrix of model type classification (left) and the ROC curve with AUC score (right). We found our ensemble autoencoder can almost perfectly classify the target mode type. Therefore, the adversary can leverage the publicly available base LLM to apply the model replication-based inversion.

Table 6: Results of the replication-inversion attack.

| Test Dataset | Attack | Target | Basis | lr | CS | BLEU | ROUGE | F1 |
|---|---|---|---|---|---|---|---|---|
| MH | Transferred | Doctor | SVD | 1e-4 | 52.87± 1.30 | 0.98± 0.25 | 0.22 | 14.79 ± 1.77 |
| | | Llama-3.1 | | | 56.21± 1.24 | 1.85± 0.47 | 0.29 | 19.33 ± 1.66 |
| | | Doctor | Unbiased | 1e-4 | 64.50± 2.70 | 6.64± 1.62 | 0.45 | 32.12 ± 3.05 |
| | | | | 5e-4 | 58.36± 1.12 | 0.90± 0.08 | 0.23 | 17.91 ± 0.54 |
| | | Llama-3.1 | | 1e-4 | 61.19± 2.14 | 4.72± 0.76 | 0.40 | 27.53 ± 1.85 |
| | | | | 5e-4 | 63.60± 2.10 | 2.45± 0.51 | 0.30 | 22.10 ± 1.35 |
| | Replicated (Ours) | Doctor | Unbiased | 1e-4 | 72.32± 2.64 | 16.33± 2.23 | 0.58 | 43.35 ± 2.31 |
| | | | | 5e-4 | 66.74± 1.79 | 5.54± 1.30 | 0.42 | 30.59 ± 1.93 |
| | | Llama-3.1 | | 1e-4 | 68.33± 2.33 | 4.91± 0.71 | 0.45 | 30.76 ± 1.01 |
| | | | | 5e-4 | 61.90± 1.57 | 0.87± 0.16 | 0.25 | 17.53 ± 0.82 |
| EC | Transferred | Doctor | Unbiased | 1e-4 | 62.01±1.70 | 3.60±0.68 | 0.27 | 18.56±1.60 |
| | | Llama-3.1 | | 1e-4 | 63.05±3.08 | 2.13±0.46 | 0.23 | 15.63±1.60 |
| | Replicated (Ours) | Doctor | Unbiased | 1e-4 | 76.92±2.31 | 9.71±1.63 | 0.47 | 29.60±2.17 |
| | | Llama-3.1 | | 1e-4 | 62.22±2.00 | 1.91±0.43 | 0.26 | 16.74±1.21 |

**Replication-based Inversion.** We use Llama-3 replicate to the target models (ChatDoctor and Llama-3.1) on NoRobots for three epochs with Qlora [55] with learning rate 0.0001. As for the inversion attack, we use MSE as the loss distance. Table 6 presents the evaluation of our replication-based inversion on Mentalhealth and Evolcode, from which we can make four findings: 1) The use of SVD singular basis can degrade the attack because of the discrepancy between the inversion and the target input embedding spaces. 2) The replicated model with a low attack learning rate leads to better input inversion because of better input embedding alignment. 3) The post-training can offer better protection against inversion than finetuning, as Llama-3.1 is more difficult to invert input information (e.g., fewer matched tokens as indicated by lower F1). 4) The distribution gap between the adversary data and the victim's queries can degrade the inversion. On coding data, the improvement from model replication is lower than on medical texts, as shown under the optimal setting (ChatDoctor with unbiased basis and 1e-4).

In sum, the optimization-based attack performance is deteriorated by limited attack information in the black-box setting and can be difficult to improve. As discussed earlier, the token recovery can be sensitive to the cosine difference between the

Table 7: Black-box inversion on short-context input text.

| Dataset | Average Length | CS | BLEU | ROUGE | EM | F1 |
|---|---|---|---|---|---|---|
| Instruct-2M | 26.16 | 97.85± 0.24 | 92.73± 0.58 | 0.97 | 61.38± 0.02 | 96.87 ± 0.27 |
| Norobot-test | 89.04 | 83.90± 0.84 | 51.34± 1.55 | 0.64 | 13.00± 0.02 | 65.76 ± 1.44 |
| SyntheticGPT | 177.48 | 93.62± 0.61 | 46.98± 1.25 | 0.67 | 0.0 | 72.90 ± 0.88 |

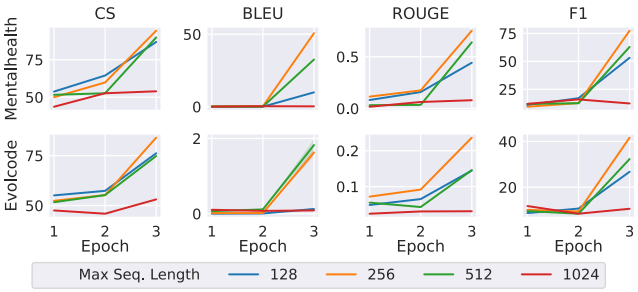


Figure 12: Generative inversion on long-context datasets.

inverted and the ground truth embeddings. The Transformer and input embedding weights of the derived model enlarge this inverted difference, resulting in worse token recovery. Next, we evaluate the generative model-based inversion attack which is based on observed ISs as context and can be more robust to the embedding gap.

**Generation-based Inversion.** As for inversion model, the adversary adopts the recent `T5-base` (T5) [56] as it is one of the most commonly used encoder-decoder model. We train T5 with learning rate 0.0002 and trains on Instruction-2M and NoRobots for short-context and long-context inputs, respectively. We also use SyntheticGPT [20] for short-context inversion evaluation. To align with replication-based black-box attack, the deployed model is Llama-3.1 as our target.

*Inversion of Short-context Input.* We train inversion model on Instruction-2M for 1 epoch. Table 7 presents the inversion evaluation on short-context datasets. The attack performs best on the same-distribution test dataset of Instruction-2M and achieves even better performance than our white-box optimization-based inversion attack. However, the downside for generative model is that it can hardly scale to longer prompts and the performance is susceptible to distribution shift. As the length increases, the semantic similarity scores drop significantly on longer SyntheticGPT dataset.

*Inversion of Long-context Inputs.* To understand the limit of input length for generative inversion, we train with NoRobots that contains long-context prompts of up to 3,384 tokens (detailed length distribution is shown in Figure 16). To investigate the impact of training sequence length, we set maximum sequence length to 128, 256, 512 and 1,024. As NoRobots only contains 9,485 samples, we train inversion model up to 3 epochs that counts for similar model update steps to Instruction-2M for ensuring model convergence.

After training, we evaluate on the top 100 longest prompts

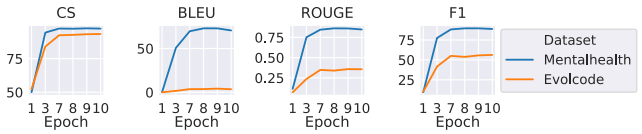


Figure 13: Evaluation of converged inversion models.

Table 8: Evaluation of generative inversion on 70B model.

| Dataset | Max Seq. Length | CS | BLEU | ROUGE | EM | F1 |
|---|---|---|---|---|---|---|
| MH | 256 | 87.92± 0.45 | 21.50± 0.57 | 0.50 | 0.0 | 55.56 ± 0.57 |
| | 512 | 97.76± 0.22 | 79.39± 1.84 | 0.88 | 2.00± 0.01 | 90.75 ± 0.90 |
| | 1024 | 97.44± 0.23 | 76.26± 1.82 | 0.87 | 1.00± 0.01 | 88.72 ± 0.90 |
| EC | 256 | 74.85± 0.81 | 3.69± 0.22 | 0.16 | 0.00 | 28.52 ± 0.89 |
| | 512 | 94.18± 0.42 | 20.98± 0.67 | 0.42 | 0.0 | 60.44 ± 1.15 |
| | 1024 | 91.74± 0.55 | 18.08± 0.81 | 0.38 | 0.0 | 55.62 ± 1.26 |

from Mentalhealth and Evolcode with greedy sampling of maximum length 4,096 to ensure full generation. Figure 12 shows the results from which we can make three observations. First, we found that the maximum sequence length 256 is the optimal setting for NoRobots dataset, where shorter or longer limits can result in worse inversion. Second, too long sequence limit 1024 can hinder the inversion quality. Third, the data type plays a central role to accurately invert ISs: the model cannot generate meaningful inverted texts on coding prompts (Evolcode) but achieve similar performance as optimization-based attack on the similar data (Mentalhealth).

*Longer Training Epochs.* We also observe that the inversion model only generates prompts until the third epoch, which may indicate that that more epochs can improve the performance. In Figure 13, we extend the epochs to 10 for model convergence while keeping maximum sequence length 256. We found that the converged performance on Mentalhealth is lower than the best performance of optimization-based inversion as shown in Figure 9 (e.g., 94.99 BLEU and 96.7 F1 on Mentalhealth). On the other hand, the inversion on coding data does not get improved as medical texts because of the distribution gap, which further highlights importance of the same distribution assumption. Appendix B includes more analysis on failure cases due to the distribution mismatch.

*Larger Models.* Table 8 validates the effectiveness of the generative inversion on Llama-3.3-70B model. Compared to previous smaller LLMs, the optimal maximum sequence length for inversion of 70B models is doubled: from 256 to 512. In particular, there are even exact matching cases (e.g., inversion trained with maximum sequence length 512). We observe that, despite deeper and wider ISs, more input tokens can be inverted as signified by higher F1 scores and CS scores. This result aligns with our previous white-box inversion evaluation on the 70B model.

*Comparison with Optimization-based Inversion.* We examine the generated inversion inputs and qualitatively compare them with those optimized by our white-box attack. Although

Table 9: Transferability of generative inversion to Llama-3.

| Dataset | Max. Seq. Length | CS | BLEU | ROUGE | F1 |
|---|---|---|---|---|---|
| Mentalhealth | 128 | 86.67± 0.43 | 10.35± 0.66 | 0.45 | 54.06 ± 0.68 |
| | 256 | 94.53± 0.30 | 53.12± 1.71 | 0.77 | 78.56 ± 0.74 |
| | 512 | 90.03± 0.36 | 32.71± 1.05 | 0.64 | 63.32 ± 0.62 |
| Evolcode | 128 | 76.97± 0.81 | 0.13± 0.05 | 0.15 | 28.37 ± 1.01 |
| | 256 | 84.04± 0.76 | 2.10± 0.19 | 0.23 | 40.63 ± 1.19 |
| | 512 | 74.78± 0.83 | 1.57± 0.16 | 0.15 | 31.45 ± 1.00 |

both achieve similar semantic similarity and token matching score, the reason for failed inversion tokens is different. As mentioned previously, the optimized inversion inputs may introduce or replace the true token with noisy tokens of different languages or Unicode which hinders the readability but also partially reveals the meaning of true inputs. On the other hand, generative inverted texts contain no noisy tokens but may miss entirely some sentences, especially those in the middle or the end of the original input. This is a disadvantage to the adversary because of potential key information loss.

*Transferability.* We explore whether the generative inversion model can directly transfer to the derived target model. We evaluate the T5 model trained on Llama-3.1's ISs and evaluate on Llama-3 (see Table 9). The inversion performance is close to the non-transfer case as shown in Figure 12. This validates our claim that the generation-based can better tolerate the embedding gap than the optimization-based attack.

## 5.4 Defense Evaluation

In this section, we test four practical defenses including DP, pruning, quantization, dropout and noisy input embedding. In Section 7, we discuss and provide mitigation suggestions.

**Setting.** We use Mentalhealth and consider the collaborative inference server who receives the ISs $\mathbf{h}_l^{f_\mathcal{V}}$ at the middle layer $l = 16$ for Llama-3-8B. The client applies defenses before the release of ISs for the first-round inference. For attack, we test the generation-based inversion attack trained for 10 epochs on NoRobots with the optimal maximum sequence length 256. In terms of utility, we evaluate the model with five-shot Massive Multitask Language Understanding (MMLU) [57], which refers to accuracy on the benchmark questions. We fix the random seed to ensure reproducibility for defenses involving randomness.

**Quantization.** We first test the 4-bit and 8-bit model quantization which, as a common technique to reduce memory cost, can alter the adversary-observed ISs through through forward-propagated error to degrade inversion performance. We test the model quantization and present the results in Table 10 of Appendix A. We note that only 4-bit slightly effect the MMLU but the inversion quality remains nearly unaffected, indicating that quantization cannot defend our attack.

**Dropout.** We apply the dropout on ISs of probability $p \in \{0.1, \cdots, 0.8\}$ that nullifies $p$ elements and scales the rest by $1/(1-p)$. As $p$ increases, the MMLU score will firstly decrease. Meanwhile, we can see that MMLU drops faster than the CS score. Among the highest dropout probabilities we tested $p \geq 0.7$, the MMLU score is close to the random guess ($\sim 0.25$) while we can still achieve inversion of CS score higher than 70. This indicates that dropout cannot fail our generative inversion while preserving the utility.

**Noisy Input Embedding.** The defender can blur the input embedding to obfuscate the ISs. Therefore, we add the noise following Gaussian distribution of variance $\sigma^2$, and show the evaluation results in the middle of Figure 14. For low noise scale $\sigma \leq 0.005$, there is negligible affect over the inference and inversion. However, we observe that $\sigma = 0.01$ is a turning point: the MMLU score become lower than 0.4 while the inversion is almost unaffected. Our attack can be effectively defended at high noise scale $\sigma \in \{0.05, 0.1\}$, but the MMLU also decreases to the random guess level. To wrap up, noisy input embedding cannot achieve a perfect trade-off between the inference utility and ISs privacy.

**Differential Privacy.** We add Laplacian noise to the IS to achieve $\varepsilon$-DP: $\delta \sim \text{Lap}(0, \varepsilon/\Delta_s)$, where $\Delta_s$ is the sensitivity. As the maximum value of ISs is not bounded, we clip the $\mathbf{h}_l^{f_\mathcal{V}}$ by $C_{\Delta_s} \in \{200, 500\}$, because we observe the highest ISs are within this range on Mentalhealth. This makes $\Delta_s = 2C_{\Delta_s}$.

The rightmost figure of Figure 14 shows the inversion performance (measured by CS) and model utility on different $\varepsilon$ and two clipping cases, where the horizontal dotted lines represent no DP protection. We observe that the blue curves (inverted text similarity) increase earlier than orange curves (model utility) in both clipping cases. This indicates that the adversary can gain advantage with low $\varepsilon$ that almost nullifies the model. In other words, with mild drop of model utility (e.g., $\varepsilon = 5,000$ for $C_{\Delta_s} = 200$), our generation-based attack remains equally effective.

> **Takeaway:** Our generation-based inversion can achieve accurate inversion on data of similar distribution and bypass practical defenses directly applied on ISs.

# 6 Related Work

In this section, we review applications of IS and inversion attacks in Natural Language Processing (NLP).

**Application of Internal States.** Recently, several works have shown that the IS are strong indicator of hallucination factual error and safety status, which provide a practical strategy for LLM holder and regulator to surveil the model behavior. Azaria and Mitchell first discovered that the ISs can indicate factual errors and found that deep layer's ISs can train

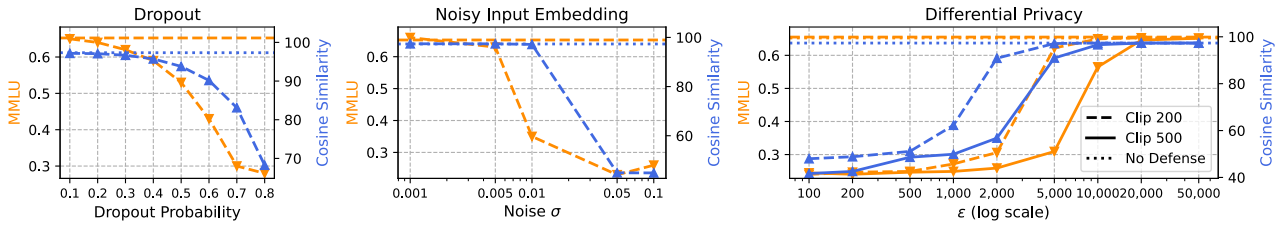


Figure 14: Evaluation of defenses including noisy input embedding, dropout and differential privacy with Laplace mechanism.

accurate hallucination detector [7]. The ISs-based hallucination detection can also be achieved by eigenvalues [8], unsupervised clustering [9] and linear probing [10] to stop the mistakes on the fly [11]. The middle layer has ISs that can indicate malicious and benign queries and can be used as safety layer to defend jailbreak attacks [12]. Additionally, ISs can reveal latent knowledge [16], membership privacy of query data [36], emotion [15, 28], internal symbolic calculations [17] and potential backdoors [13, 14]. As more applications of ISs emerge, the potential privacy risk should be carefully examined and our work takes the first step to testify the ISs inversion feasibility.

**Inversion Attacks.** In addition to the embedding [19] and outputs [20, 24], the risk of data inversion also exists for shared gradient in decentralized learning [58, 59], unlearning [60], outsourced shallow-layer inference [61] and KV cache [62] which are the most related works to ours. Nevertheless, our methods can extend to middle or last layers and can generalize to KV cache. Note that the side-channel attack to LLM serving [63] is also similar to our inversion, but leverages timing difference in accelerated inference. There is also finetuning-based defenses [38] but is limited to small LMs because of performance drop for LLMs. Recently, there are concurrent works [64, 65] attempting to invert prompt from the malicious server in collaborative learning. As for comparison, our work includes more comprehensive attacks (white-box and black-box) and has validated the attack effectiveness on long inputs (4k+ tokens) and large models (up to 70B).

## 7 Discussion & Conclusion

We conclude with discussion and future work.

**Practical Mitigation.** As we witnessed in section 5.4, directly protecting ISs is still limited to simultaneously offer both model utility and privacy. The fundamental reason is that the inference of subsequent layers depends on meaningful IS values, hence the noise-induced representation obfuscation must preserve sufficient utility by maintaining semantic coherence. To mitigate privacy leakage caused by ISs, it is essential to safeguard the entire model rather than concentrating solely on individual layers. One possible direction can be the exploitation of cryptographic tools or confidential computing. The homomorphic encryption is a promising solution, but current implementations generate unacceptable computational overhead for deployment [64]. Another countermeasure that can be immediately taken is exploiting the confidential virtual machine in CPU (e.g., AMD SEV [66]) and GPU (e.g., H100 [67]) which can provide confidentiality guarantee and allow addition obfuscation schemes inside the enclaves to hinder side channel attacks.

**Architecture-based Mitigation.** We suspect the equal-width architecture design may cause our privacy inversion attacks. Typically, the layer width in conventional classification models decreases with depth, enabling the compression of input data and contributing to the emergence of Harmless Space (HS) [68]. In contrast, Transformer-based language models have a uniform layer width. This architectural difference may explain why our attack methods are successful. Thus, a potential architecture-based defense could involve a model with varying sizes of input embeddings, intermediate states, and output embeddings, which, by exploiting information loss, could increase the difficulty of accurate inversion.

**Unavailable Ground Truth.** In Section 3, the adversary aims to recover the exact input, but original inputs are generally unavailable in practice, which can make it infeasible to check the inversion correctness. In practice, for the open-source models, the adversary can test with surrogate inputs as reference to ensure constraint. For closed-sourced models, the adversary can collect input data from their interested distributions for querying and training the inversion model. The convergence can guarantee the inversion for in-distribution data.

**Limitation.** Our optimization-attack is sensitive to the hyper-parameter and added noises. For example, TBS cannot recover meaningful inputs even under the highest ε we tested because of noise sensitivity. Future work can enhance this attack by denoising on DP-protected ISs. Another limitation is the linear complexity (i.e., $O(E)$) during optimization, which can cause longer attack time for larger models. Also, it is possible to adopt random initialization for optimization-based attacks to avoid local optima. Appendix B includes more detailed analysis of failure cases and boundary conditions. To improve the attack, one of the future direction is to come up with black-box inversion attack that can also be as context-free and length-free as the white-box optimization-based attack.

## 8 Ethical considerations

Our attacks exploit ISs of models to invert nearly original sensitive input, posing significant threats to user privacy and data security. As our research do not involve human participants and only use public medical dialogue, Mentalhealth and coding datasets, the IRB of authors' intuition, after our consultation, determined that our research does not require further review. We acknowledge that exemption from IRB review may not fully address all ethical considerations, and have therefore taken the following steps.

**Ensuring no harm is caused to stakeholders.** The potential stakeholders include LLM user and LLM service provider and broader privacy community. To mitigate potential harms to individuals, by following existing privacy work, we only select public datasets that are highly-downloaded and appropriately anonymized. In addition, we also manually checked all datasets used in our research to ensure there is no sensitive information (e.g., PII) exposed in our research. We did not attack real-world collaborative inference systems, so no individual privacy is leaked by our research. As for the models, we only used opensource models including Llama, Qwenand T5 in our work for compliance with model-use license.

**Responsible disclose to service providers.** We also share our findings with main cloud LLM service providers that might deploy collaborative inference including OpenAI, Meta AI and Qwen. We recognize that this cannot perfectly mitigate privacy risks, because our attacks, if known to the public, may still be misused by the actual collaborative inference server or IS auditor to recover the user inputs. However, we believe the benefits of publicizing our attacks outweight the potential harm. As our work demonstrates the potential privacy risks of LLM ISs, it will draw increased attention to, not only the ISs inversion threat, but also the general privacy concerns surrounding LLM service systems. This will encourage the LLM practitioners to proactively implement well-established solutions like confidential virtual machine within their systems to offer better privacy protection.

## 9 Open Science

In compliance with the USENIX Security's Open Science policy, we commit to publicly releasing the source code to implement the attacks, pretrained inversion models on non-sensitive data in our study upon acceptance of this paper and inversion logs of main results. As the datasets used in our paper are all downloadable from Hugging Face, in our artifact we direct users to the datasets downloading link and provide processing scripts. We also provide instructions on how to test our attacks on user's own data, detailed configuration, program scripts, hyperparameters and hardware requirements. Our code is released at https://doi.org/10.5281/zenodo.15605325.


## Acknowledgments

We thank our shepherd and anonymous reviewers for insightful feedback. We also thank Yiming Wang for insightful discussions that inspired this project. The work has been supported in part by the National Natural Science Foundation of China (62325207, 62132013, 62302298). Shaofeng Li is supported by the Start-up Research Fund of Southeast University (No. RF1028624178).


## References


[1] Robert Hart. X fact checks elon musk after he blasts apple-openai partnership as 'creepy' privacy nightmare. https://www.forbes.com/sites/roberthart/2024/06/11/x-fact-checks-elon-musk-after-he-blasts-apple-openai-partnership-as-creepy-privacy-nightmare/, 2024.

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[3] Zheng Lin, Xuanjie Hu, Yuxin Zhang, Zhe Chen, Zihan Fang, Xianhao Chen, Ang Li, Praneeth Vepakomma, and Yue Gao. Splitlora: A split parameter-efficient fine-tuning framework for large language models. *arXiv preprint arXiv:2407.00952*, 2024.

[4] Yixuan Mei, Yonghao Zhuang, Xupeng Miao, Juncheng Yang, Zhihao Jia, and Rashmi Vinayak. Helix: Distributed serving of large language models via max-flow on heterogeneous gpus. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2025*, 2025.

[5] Mingjin Zhang, Jiannong Cao, Xiaoming Shen, and Zeyang Cui. Edgeshard: Efficient llm inference via collaborative edge computing. *arXiv preprint arXiv:2405.14371*, 2024.

[6] Fact sheet: President biden issues executive order on safe, secure, and trustworthy artificial intelligence. https://www.whitehouse.gov/briefing-room/statements-releases/2023/10/30/fact-sheet-president-biden-issues-executive-order-on-safe-secure-and-trustworthy-artificial-intelligence/, 2023.

[7] Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, December 2023. doi: 10.18653/v1/2023.findings-emnlp.68.

[8] Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. INSIDE: LLMs' internal states retain the power of hallucination detection. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, Yujia Zhou, and Yiqun Liu. Unsupervised real-time hallucination detection based on the internal states of large

language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 14379–14391, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics.

[10] Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth A. Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms. *CoRR*, abs/2406.15927, 2024. doi: 10.48550/ARXIV.2406.15927.

[11] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *arXiv preprint arXiv:2407.20311*, 2024.

[12] Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. Safety layers of aligned large language models: The key to llm security. *arXiv preprint arXiv:2408.17003*, 2024.

[13] M Lamparth and A Reuel. Analyzing and editing inner mechanisms of backdoored language 353 models. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 354, 2024.

[14] Tian Dong, Minhui Xue, Guoxing Chen, Rayne Holland, Yan Meng, Shaofeng Li, Zhen Liu, and Haojin Zhu. The philosopher's stone: Trojaning plugins of large language models. In *Network and Distributed System Security Symposium, NDSS 2025*. The Internet Society, 2025.

[15] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

[16] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*, 2023.

[17] Junhao Chen, Shengding Hu, Zhiyuan Liu, and Maosong Sun. States hidden in hidden states: Llms emerge discrete state representations implicitly. *arXiv preprint arXiv:2407.11421*, 2024.

[18] Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 377–390, 2020.

[19] John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.765.

[20] Collin Zhang, John X. Morris, and Vitaly Shmatikov. Extracting prompts by inverting LLM outputs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*, pages 14753–14777. Association for Computational Linguistics, 2024.

[21] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2015. doi: 10.1109/ITW.2015.7133169.

[22] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331, 2020. doi: 10.1109/SP40000.2020.00095.

[23] Haoran Li, Mingshi Xu, and Yangqiu Song. Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence. *arXiv preprint arXiv:2305.03010*, 2023.

[24] John Xavier Morris, Wenting Zhao, Justin T Chiu, Vitaly Shmatikov, and Alexander M Rush. Language model inversion. In *The Twelfth International Conference on Learning Representations*, 2024.

[25] Jieren Deng, Yijue Wang, Ji Li, Chenghong Wang, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. TAG: gradient attack on transformer-based language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3600–3610, 2021. doi: 10.18653/V1/2021.FINDINGS-EMNLP.305.

[26] Xinguo Feng, Zhongkui Ma, Zihan Wang, Eu Joe Chegne, Mengyao Ma, Alsharif Abuadbba, and Guangdong Bai. Uncovering gradient inversion risks in practical language model training. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3525–3539, 2024.

[27] Guanzhong Chen, Zhenghan Qin, Mingxin Yang, Yajie Zhou, Tao Fan, Tianyu Du, and Zenglin Xu. Unveiling the vulnerability of private fine-tuning in split-based frameworks for large language models: A bidirectionally enhanced attack. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, page 2904–2918. Association for Computing Machinery, 2024. ISBN 9798400706363. doi: 10.1145/3658644.3690295.

[28] Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, and Yongbin Li. How alignment and jailbreak work: Explain LLM safety through intermediate hidden states. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2461–2488, November 2024. doi: 10.18653/v1/2024.findings-emnlp.139.

[29] OpenAI. Enterprise privacy at openai - does openai review my business data? https://openai.com/enterprise-privacy/, 2024.

[30] Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Maksim Riabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. Petals: Collaborative inference and fine-tuning of large models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 558–568, July 2023. doi: 10.18653/v1/2023.acl-demo.54.

[31] 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, 2023. doi: 10.1109/SP46215.2023.10179382.

[32] Z. Zhang, C. Gong, Y. Cai, Y. Yuan, B. Liu, D. Li, Y. Guo, and X. 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 55–55, may 2024.

[33] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*, 2021.

[34] Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. Exploring concept depth: How large language models acquire knowledge at different layers? In *Proceedings of the 2025 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2025)*, 2025.

[35] Baixiang Huang, Canyu Chen, and Kai Shu. Authorship attribution in the era of llms: Problems, methodologies, and challenges. *arXiv preprint arXiv:2408.08946*, 2024.

[36] Ziwei Ji, Delong Chen, Etsuko Ishii, Samuel Cahyawijaya, Yejin Bang, Bryan Wilie, and Pascale Fung. LLM internal states reveal hallucination risk faced with a query. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 88–104, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.6.

[37] Sahar Abdelnabi, Aideen Fay, Giovanni Cherubin, Ahmed Salem, Mario Fritz, and Andrew Paverd. Get my drift? catching llm task drift with activation deltas. In *SaTML*, 2025.

[38] Minxin Du, Xiang Yue, Sherman S. M. Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023*, pages 2665–2679. ACM, 2023.

[39] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2023*, 2023.

[40] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

[41] ContactDoctor. Bio-medical: A high-performance biomedical language model. https://huggingface.co/ContactDoctor/Bio-Medical-Llama-3-8B, 2024.

[42] Meta AI. Introducing llama 3.1: Our most capable models to date. https://ai.meta.com/blog/meta-llama-3-1/, 2024.

[43] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

[44] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022. doi: 10.48550/ARXIV.2210.07316.

[45] MTEB Leaderboard. Massive text embedding benchmark (mteb) leaderboard. https://huggingface.co/spaces/mteb/leaderboard, 2024.

[46] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[47] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[48] Ruslan Magana Vsevolodovna. Ai medical dataset, 2023. URL https://github.com/ruslanmv/ai-medical-chatbot.

[49] Amod. mental_health_counseling_conversations (revision 9015341), 2024. URL https://huggingface.co/datasets/Amod/mental_health_counseling_conversations.

[50] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct, 2023.

[51] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps. *NeurIPS*, 2021.

[52] bigcode. Big code models leaderboard. https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard, 2025.

[53] Jianlin Su. The magical effect of the bias term: Rope + bias = better length extrapolation. https://spaces.ac.cn/archives/9577, 2023.

[54] Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. No robots. https://huggingface.co/datasets/HuggingFaceH4/no_robots, 2023.

[55] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[56] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[57] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

[58] Ziang Li, Mengda Yang, Yaxin Liu, Juan Wang, Hongxin Hu, Wenzhe Yi, and Xiaoyang Xu. GAN you see me? enhanced data reconstruction attacks against split inference. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, 2023.

[59] Guanzhong Chen, Zhenhan Qin, Mingxin Yang, Yajie Zhou, Tao Fan, Tianyu Du, and Zenglin Xu. Unveiling the vulnerability of private fine-tuning in split-based frameworks for large language models: A bidirectionally enhanced attack. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, CCS 2024*. ACM, 2024.

[60] Hongsheng Hu, Shuo Wang, Tian Dong, and Minhui Xue. Learn what you want to unlearn: Unlearning inversion attacks against machine unlearning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3257–3275, may 2024. doi: 10.1109/SP54263.2024.00248.

[61] Fei Zheng. Input reconstruction attack against vertical federated large language models. *arXiv preprint arXiv:2311.07585*, 2023.

[62] Huan Yang, Deyu Zhang, Yudong Zhao, Yuanchun Li, and Yunxin Liu. A first look at efficient and secure on-device llm inference against kv leakage, 2024.

[63] Guanlong Wu, Zheng Zhang, Weili Wang, Jianyu Niu, Yao Zhang, Ye Wu, and Yinqian Zhang. I know what you asked: Prompt leakage via kv-cache sharing in multi-tenant llm serving. In *Network and Distributed System Security Symposium, NDSS 2025*. The Internet Society, 2025.

[64] Wenjie Qu, Yuguang Zhou, Yongji Wu, Tingsong Xiao, Binhang Yuan, Yiming Li, and Jiaheng Zhang. Prompt Inversion Attack against Collaborative Inference of Large Language Models . In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1602–1619. IEEE Computer Society, May 2025. doi: 10.1109/SP61157.2025.00160.

[65] Xinjian Luo, Ting Yu, and Xiaokui Xiao. Prompt inference attack on distributed large language model inference frameworks. *arXiv preprint arXiv:2503.09291*, 2025.

[66] AMD Sev-Snp. Strengthening vm isolation with integrity protection and more. *White Paper, January*, 53:1450–1465, 2020.

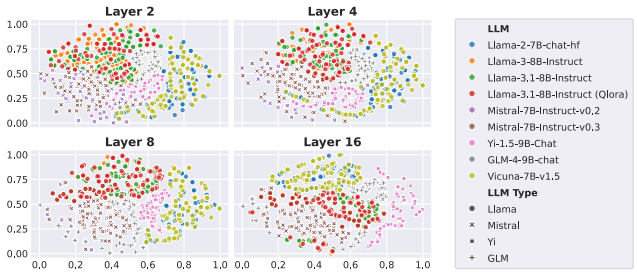


Figure 15: Visualisation of internal states for mainstream LLMs. The selected LLMs have the same width 4096 for dimension reduction using t-SNE.

Table 10: Evaluation of quantization as defense.

| Defense | MMLU | CS | F1 |
|---|---|---|---|
| No Quant | 0.65 | 97.33± 0.25 | 88.77 ± 0.83 |
| 8-bit | 0.65 | 97.40± 0.24 | 88.84±0.83 |
| 4-bit | 0.62 | 97.22± 0.24 | 88.57±0.83 |

[67] Rob Nertney. Confidential Compute on NVIDIA Hopper H100. https://images.nvidia.cn/aem-dam/en-zz/Solutions/data-center/HCC-Whitepaper-v1.0.pdf, 2023.

[68] Lu Chen, Shaofeng Li, Benhao Huang, Fan Yang, Zheng Li, Jie Li, and Yuan Luo. Seeing is not always believing: The space of harmless perturbations. *arXiv preprint arXiv:2402.02095*, 2024.

# A Additional Results

**Visualization of Internal States.** We applies the t-SNE onto the ISs of various open-source LLMs across different scales. As visualized in Figure 15, LLMs rooted from a common pretrained LLM, have similar ISs. For example, Llama-3 and Llama-2 are two pretrained models and have separable ISs. That said, it is possible to train classify the target LLM type simply using the ISs. To evaluate model type identification of Llama-3, we select the total 13 LLMs open-sourced before or after Llama-3:Vicuna-7B-v1.5, GLM-4-9B-chat, Qwen2.5-14B-Instruct, Meta-Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, Qwen2.5-3B-Instruct, Meta-Llama-3.1-8B (Qlora), gemma-2-9B-it, Llama-3.2-3B-Instruct, Yi-1.5-9B-Chat, Mistral-7B-Instruct-v0.3, Llama-2-7b-chat-hf, and Mistral-7B-Instruct-v0.2.

**Token length of NoRobots.** Figure 16 shows the histogram of token length of NoRobots. Compared to Instruction2M, we note there are certain long-context training samples which can be indispensable to train long-context inversion model, at expense of increased training time and memory usage.

**Quantization Defense.** Table 10 shows the evaluation of our generative inversion on quantized model. We use `bitsandbytes` for LLM quantization.

**Theoretical Analysis.** We investigate the gradient magnitude for ER and TSB. To simply notation, we denote the target IS of layer $l$, $\mathbf{h}_l^{f_\psi}(x))$ by $\mathbf{h}$, and consider the adversary optimizes $\widehat{\mathbf{w}}$ to approximate

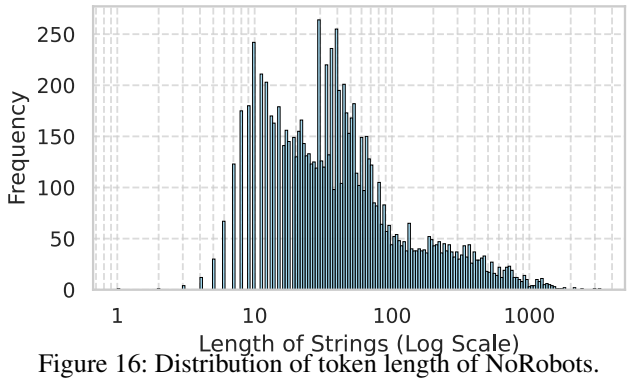

Figure 16: Distribution of token length of NoRobots.

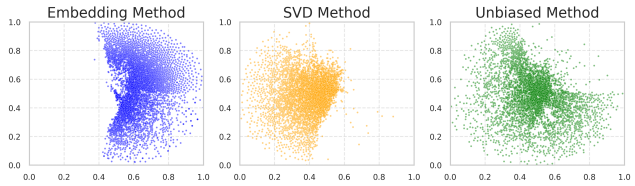

Figure 17: Visualization of sampled input embeddings from Llama-3-8B (left), the singular basis obtained through SVD decomposition of input embeddings (middle) and unbiased basis (right).

$\psi_l(\widehat{\mathbf{w}})$ to $\mathbf{h}$. Without loss of generality, we consider the squared $\mathcal{L}_2$ norm for the distance d and $\widehat{\mathbf{w}} \in \mathbb{R}^d_{in}$. The loss function is:

$$\mathcal{L} = \mathrm{d}(\psi_l(\widehat{\mathbf{w}}), \mathbf{h}) = \|\psi_l(\widehat{\mathbf{w}}) - \mathbf{h}\|_2^2. \tag{8}$$

The gradient with respect to $\widehat{\mathbf{w}}$ is:

$$\frac{\partial \mathcal{L}}{\partial \widehat{\mathbf{w}}} = 2\left(\psi_l(\widehat{\mathbf{w}}) - \mathbf{h}\right)^\top \cdot \frac{\partial \psi_l}{\partial \widehat{\mathbf{w}}}. \tag{9}$$

Here, the gradient magnitude of ER depends directly on the residual error $\|\psi_l(\widehat{\mathbf{w}}) - \mathbf{h}\|$ and the Jacobian $\partial\psi_l/\partial\widehat{\mathbf{w}}$ of the first $l$ Transformer layers. For TBS, as $\widehat{\mathbf{w}} = \varphi_{\mathbf{z}}(\widehat{\mathbf{z}} \cdot \mathbf{B})$, the gradient with respect to $\mathbf{z}$ becomes:

$$\frac{\partial \mathcal{L}}{\partial \widehat{\mathbf{z}}} = \frac{\partial \mathcal{L}}{\partial \widehat{\mathbf{w}}} \cdot \frac{\partial \widehat{\mathbf{w}}}{\partial \widehat{\mathbf{z}}} = \frac{\partial \mathcal{L}}{\partial \widehat{\mathbf{w}}} \cdot \frac{\partial \varphi_{\mathbf{z}}(\widehat{\mathbf{z}} \cdot \mathbf{B})}{\partial \widehat{\mathbf{z}}}. \tag{10}$$

In the experiments, we set $\varphi_{\mathbf{z}}(\widehat{\mathbf{z}} \cdot \mathbf{B}) = \alpha \arctan(\widehat{\mathbf{z}} \cdot \mathbf{B})$, thus the derivative is

$$\frac{\partial \widehat{\mathbf{w}}}{\partial \widehat{\mathbf{z}}} = \frac{\alpha}{1 + (\widehat{\mathbf{z}}\mathbf{B})^2} \cdot \mathbf{B}^\top. \tag{11}$$

For small $\mathbf{z}$, the scaling factor $\alpha/(1 + (\widehat{\mathbf{z}}\mathbf{B})^2) \approx \alpha$, leading to larger gradient magnitudes compared to the baseline. As $\mathbf{z}$ increases, the factor $\alpha/(1 + (\widehat{\mathbf{z}}\mathbf{B})^2) \to 0$, which stabilizes the gradient on $\widehat{bz}$, especially for deep model where the Jacobian can vary greatly.

In conclusion, TBS introduces a *nonlinear scaling* of gradients through $\varphi_{\mathbf{z}}$, creating a dynamic where:

$$\left\| \frac{\partial \mathcal{L}}{\partial \widehat{\mathbf{z}}} \right\| \propto \frac{1}{1 + (\widehat{\mathbf{z}}\mathbf{B})^2} \cdot \left\| \frac{\partial \mathcal{L}}{\partial \widehat{\mathbf{w}}} \right\|. \tag{12}$$

**Visualization of Input Embedding and Basis.** To support the findings of Figure 5, we also visualize the vectors through uniform t-SNE dimension reduction. Figure 17 demonstrates that the singular basis vectors and input embeddings exhibit tight clustering in the representation space, whereas the unbiased basis displays significantly greater dispersion.

## B Analysis of Failure Cases

In this section, we analyze the failure cases caused by non-convergence for our optimization-based attack TBS and by the distribution mismatch for our generation-based inversion attack.

The failure cases for our white-box attack are commonly caused by inappropriate settings. For example, too high learning rate may accelerate the optimization at the beginning but fall into local minimum afterwards, as validated by the worse inversion quality in Figure 9. A failure example of high learning rate 0.001 is shown in Figure 18, where we can see most inverted tokens are unreadable. Even there are some keywords recovered, the adversary cannot guess original inputs.

Besides, due to the complexity of optimization space, we also observe failure example even under appropriate setting. In Figure 20, we provide the loss curves for the first four samples of Mentalhealth during TBS on Llama-3.3-70B. We observe that the Sample 3 encountered loss divergence, leading to inversion F1 score 72.91. We observe that Sample 3 exhibits loss divergence, resulting in the F1 score of 72.91. In contrast, the remaining samples successfully escape local optima despite similar initial loss increases, ultimately achieving smooth convergence. Given our fixed initialization scheme, we explore random initialization for as a potential alternative. Empirical evaluation reveals that standard Gaussian noise initialization degrades convergence speed and ultimately yields inferior inversion performance. Future research directions include developing bounded random initialization strategies or incorporating dynamic noise injection during optimization.

Figure 19 illustrates a failure case of generation-based inversion arising from distributional mismatch. The target text contains SQL table code that lies outside the inversion model's training distribution, resulting in uninterpretable "<unk>" tokens. This demonstrates that a necessary condition for successful inversion is comprehensive training data coverage of all target input tokens. Furthermore, the discrepancy in token distributions between the inversion model's training data and target inputs may lead to suboptimal inversion performance, particularly when significant frequency mismatches exist for critical tokens.

| | |
|---|---|
| **Input** | ... in this negative environment with my child and keep our family together? Do I move away with my child and have my relationship end? I do not want to take him out of either of his kids' lives. What do I do? |
| **Inversion** | ... in this negative environment withyouралася and keep�! family together?DoelementGuidIdалася.АраласяалаСя Child and Have my relationship end !اص 경우 not want(cuda take himDECREF يشكلeither благод his kids's\tdfs▌▌▌▌▌▌▌▌ What получения_SL \t\t\n imkân вияв |

Figure 18: Failure inversion example of TBS attack in MentalHealth.

| Input | Inversion |
|---|---|
| Generate an SQL query to find the average score for each student, but exclude any students whose average score is below 6. The result should be ordered in decreasing order of the average score. Table: student_scores \| id <unk> student_id <unk> score <unk> <unk> <unk>