Towards Provable (In)Secure Model Weight Release Schemes

Xing Yang [1] **Bingtao Wang** [2] **Yuhao Wang** [2] **Zimo Ji** [2] **Terry Jingchen Zhang** [3] **Wenyuan Jiang** [3]

Abstract

Recent secure weight release schemes claim to enable open-source model distribution while protecting model ownership and preventing misuse. However, these approaches lack rigorous security foundations and provide only informal security guarantees. Inspired by established works in cryptography, we formalize the security of weight release schemes by introducing several concrete security definitions. We then demonstrate our definition's utility through a case study of TaylorMLP, a prominent secure weight release scheme. Our analysis reveals vulnerabilities that allow parameter extraction thus showing that TaylorMLP fails to achieve its informal security goals. We hope this work will advocate for rigorous research at the intersection of machine learning and security communities and provide a blueprint for how future weight release schemes should be designed and evaluated.

1. Introduction

Deep learning models, especially large language models (LLMs), pose unique challenges for model weight release, the practice of sharing a model's learned parameters with users. Providers typically choose between closed API hosting, which preserves developer control but forces users to expose private data to the service (Achiam et al., 2023), and fully open-sourcing the weights, which protects user privacy but cedes ownership and control of the model (Touvron et al., 2023). This creates a dilemma: developers risk unauthorized extraction or repurposing of their proprietary weights if they release them openly, yet users face privacy and availability concerns when models are accessible only via APIs.

To address this, secure weight release schemes aim to enable local or offline inference without exposing raw weights, thereby balancing utility and protection. An ideal scheme would preserve accuracy and inference performance for legitimate users while making it difficult for an adversary to recover the original weights or to perform large-scale fine-tuning abuse. Such guarantees are crucial because model weights represent high-value intellectual property, often requiring millions of dollars in training investment (Refael et al., 2024).

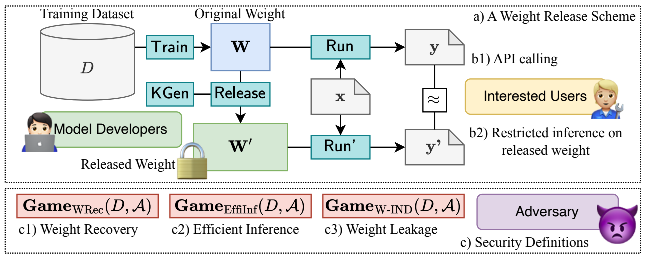

Figure 1. Overview of weight release schemes and our proposed security definitions. (a) Model developers transform original weights into a restricted released model; (b) users either call an API (b1) or run local inference on the restricted weights (b2); (c) proposed game-based security definitions: weight recovery (c1), efficient inference (c2) and weight leakage (c3).

However, current weight release schemes like TaylorMLP (Wang et al., 2024) lack a formal security foundation. Their security claims rest on informal hardness assumptions or empirical observations without reductions to well-studied hardness assumptions. Consequently, it remains unclear under what adversarial models and assumptions weight confidentiality truly holds, or how one might systematically evaluate a scheme's (in)security. Indeed, recent work shows that fine-tuning can often strip out empirical safeguards, hinting that ad hoc measures may be brittle (Tamirisa et al., 2024).

In this paper, we fill this gap by introducing a formal framework for model weight release security, as outlined in Figure 1. We define security properties, establish relationships among them, and show how they can guide scheme design. We then present a case study of TaylorMLP: we mount a parameter extraction attack that successfully recovers original weights, demonstrating that TaylorMLP falls short of its informal goals. Finally, we distill our insights into a blueprint for future schemes, outlining design principles and evaluation criteria to achieve provable security guarantees.

[1]Polytechnic Institute, Zhejiang University, Zhejiang, China [2]School of Software Engineering, Tongji University, Shanghai, China [3]ETH Zurich, Zurich, Switzerland. Correspondence to: Wenyuan Jiang <wenyjiang@ethz.ch>.

Our contribution in this paper is threefold.

- **Formal security properties of weight release schemes.** We defined several formal security properties of weight release schemes and proved some of their relations under this context.
- **A case study against TaylorMLP.** We performed an analysis on TaylorMLP and demonstrated that it is vulnerable to our parameter extraction attack, thus failing to deliver its claimed security goals.
- **A blueprint for future weight release schemes.** We provide a blueprint for how future weight release schemes should be designed and evaluated.

2. Related Work

Open-Source Models Open-source LLMs like LLaMA 2 (Touvron et al., 2023), Qwen (Bai et al., 2023), and DeepSeek (Liu et al., 2024) have democratized access by releasing full model weights under permissive licenses. While this fosters innovation and reproducibility, it also hands over complete control of the models to downstream users. Without embedded technical safeguards, these releases rely solely on legal agreements to prevent misuse.

Privacy-Aware Model Inference Cryptographic protocols such as MPC and homomorphic encryption enable inference without exposing either model weights or user inputs, as demonstrated by BOLT (Pang et al., 2024), BumbleBee (Lu et al., 2023), and PUMA (Dong et al., 2023). These systems offer provable security under standard assumptions but incur latency and resource costs orders of magnitude higher than native inference. Such overheads currently preclude their use for real-time or large-scale LLM deployment.

Weight Release Schemes TaylorMLP (Wang et al., 2024) publishes truncated Taylor-series coefficients of MLP weight matrices, allowing exact inference while obscuring the original parameters and introducing controllable throttling. Hardware obfuscation approaches embed DNNs in locked circuits that only function under specific configurations, preventing direct weight extraction (Goldstein et al., 2021). Its security is based on the presumed hardness of inverting the transformation, without formal reductions to established cryptographic problems.

Model Tracing Watermarking and fingerprinting embed hidden signals into model weights or outputs to detect unauthorized use. CoTGuard (Wen et al., 2025) and Double-*i* Watermark (Li et al., 2024) insert secret triggers into reasoning traces or fine-tuning, while Mark Your LLM (Xu et al., 2025) and ProFLingo (Jin et al., 2024) employ backdoor-based or query-based fingerprints. These techniques demonstrate empirical robustness against benign fine-tuning but lack strong guarantees against adaptive adversaries.

Model Abuse and Misuse Risks Open releases can be fine-tuned with minimal poisoned data to remove safety mitigations and repurpose models for harmful tasks. Surveys by Yan *et al.* (Yan et al., 2024) and Li *et al.* (Li et al., 2023) document how such attacks degrade alignment and enable sensitive data extraction. These risks motivate technical controls that enforce provable security properties in released models.

3. Security of Weight Release Schemes

In this section, we first formally define the syntax of a weight release scheme. Then we give several security properties for weight release schemes with formal definitions. We also discuss the relations of these security properties and possible hardness assumptions under the context of weight release schemes.

3.1. Weight release schemes

Consider a deep learning task where training the model on dataset $D \in \mathcal{D}$ gives the model weight $W \in \mathcal{W}$. The model is expected to take $x \in \mathcal{X}$ as the input and output $y \in \mathcal{Y}$ when running inference on weight W. We then define the syntax of a weight release scheme as follows.

Syntax. A weight release scheme is a tuple of 5 probabilistic algorithms $\Sigma = (\mathsf{Train}, \mathsf{Run}, \mathsf{KGen}, \mathsf{Release}, \mathsf{Run'})$, where

- $\mathsf{Train} : \mathcal{D} \to \mathcal{W}$ abstracts the original training process on dataset $D \in \mathcal{D}$ and produces a model weight $W \in \mathcal{W}$.
- $\mathsf{Run} : \mathcal{W} \times \mathcal{X} \to \mathcal{Y}$ abstracts the original inference process which runs $W \in \mathcal{W}$ on input $x \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$.
- $\mathsf{KGen} : \perp \to \mathcal{K}_{\mathsf{sk}} \times \mathcal{K}_{\mathsf{pk}}$ is similar to the public key cryptography case, which generates a pair of key $(\mathsf{sk}, \mathsf{pk})$ for future use. There are many cases where the weight release scheme requires no key, and in this case $\mathcal{K}_{\mathsf{sk}}$ and $\mathcal{K}_{\mathsf{pk}}$ can both be $\emptyset$.
- $\mathsf{Release} : \mathcal{K}_{\mathsf{sk}} \times \mathcal{W} \to \mathcal{W}'$ is the core of the weight release scheme which transforms the original weight $W \in \mathcal{W}$ to the released version $W' \in \mathcal{W}'$ under some private key sk. Note that depending on the security requirements, $\mathcal{W}'$ is generally not the same as $\mathcal{W}$, but there are cases like hidden watermarking where $\mathcal{W}' = \mathcal{W}$.
- $\mathsf{Run'} : \mathcal{K}_{\mathsf{pk}} \times \mathcal{W}' \times \mathcal{X} \to \mathcal{Y}$ models the user with the public key pk inferencing the released version of weight $W' \in \mathcal{W}'$ on input $x \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$.

Note that different from standard cryptographic practice where security parameters are included in the KGen parameter, the security parameters for a specific weight release scheme are often fixed and implicitly determined by the scheme (e.g., model weight size).

Game-based definition. We use game-based definitions for formally specifying the properties of weight release schemes through interactive experiments in this section, following standard cryptographic practice for defining security properties of protocols(Bellare & Rogaway, 2004; Shoup, 2004). A game is a probabilistic experiment, often written as a procedure, that captures the essential behavior of the system whose output indicates the result of the experiment. In our context, games allow us to rigorously compare the behavior of original and released model weights while accounting for the inherent randomness in neural network training and inference processes.

Correctness. Informally, a weight release scheme is correct if inferencing the released version of weight leads to *same* results as inferencing the original weight. Due to the intrinsic randomness in the inference process, defining *same* is not like defining *equality*. Here we consider $\mathsf{Dist}(a, b)$ as a distance function measuring how different $a, b \in \mathcal{S}$ are where $\mathcal{S}$ is the relevant domain for our definitions. We also consider $\mathsf{Same}(a, b)$ as $\mathsf{Dist}(a, b) \leq \epsilon$ for some practically meaningful threshold δ. For example, $\mathsf{Dist}(W_a, W_b)$ for $W_a, W_b \in \mathcal{W}$ can be defined as $\|W_a - W_b\|_p$ where $\|\cdot\|_p$ is the matrix p-norm, and $\mathsf{Same}(W_a, W_b)$ can be defined using $\delta = 10^{-3}$ for some specific model types. Then correctness is defined as follows.

Definition 3.1 (Correctness of a weight release scheme**)**. We define the following game.

```
$\mathbf{Game}_{\text{Correctness}}(D)$
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}()$
2 : $W \leftarrow\!\!\$\ \mathsf{Train}(D)$
3 : $W' \leftarrow\!\!\$\ \mathsf{Release}(\mathsf{sk}, W)$
4 : $x \leftarrow\!\!\$\ \mathcal{X}$
5 : $y \leftarrow\!\!\$\ \mathsf{Run}(W, x)$
6 : $y' \leftarrow\!\!\$\ \mathsf{Run'}(\mathsf{pk}, W', x)$
7 : $\mathbf{return}\ \mathsf{Same}(y', y)$
```

Then a weight release scheme Σ with $\Sigma = (\mathsf{Train}, \mathsf{Run}, \mathsf{KGen}, \mathsf{Release}, \mathsf{Run'})$ is correct if

$$\Pr[\mathbf{Game}_{\text{Correctness}}(D) \Rightarrow 1] = 1 \tag{1}$$

for given $D \in \mathcal{D}$.

3.2. Security properties

Informally, a secure weight release scheme aims to 1) protect the model ownership of developers while 2) allow users to perform inference on the released model weight with *reasonalble* efficiency. Additional security goals include preventing abuse and further unintended modification of the released model. To formally define these security properties, we choose several typical security goals and formulate them into game-based definitions. Each of the following security goals starts with an example scenario followed by a game and description of the adversary's abilities and constraints.

Preventing weight recovery. One natural security goal for secure weight release schemes is to prevent any adversary from recovering the original weight. This goal can be captured by the following game for weight release scheme Σ on a given training dataset $D \in \mathcal{D}$.

```
$\mathbf{Game}_{\text{WRec}}(D, \mathcal{A})$
1 : $W \leftarrow\!\!\$\ \mathsf{Train}(D)$
2 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}()$
3 : $W' \leftarrow\!\!\$\ \mathsf{Release}(\mathsf{sk}, W)$
4 : $W^* \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk}, W')$
5 : $\mathbf{return}\ \mathsf{Same}(W^*, W)$
```

The advantage of an adversary $\mathcal{A}$ in this game is defined as

$$\mathsf{Adv}_{\Sigma}^{\text{WRec}}(D, \mathcal{A}) = \Pr[\mathbf{Game}_{\text{WRec}}(D, \mathcal{A}) \Rightarrow 1] \tag{2}$$

Definition 3.2. Then for all efficient adversary $\mathcal{A}$ on some dataset $D \in \mathcal{D}$, if

$$\mathsf{Adv}_{\Sigma}^{\text{WRec}}(D, \mathcal{A}) \leq \epsilon \tag{3}$$

for some negligible ϵ with regard to the security paramter of the scheme Σ, then the weight release scheme Σ is considered to be (D, ϵ)-weight-recovery-secure (WRec-secure).

Note that since weight recovery is a very strong attack even for computational unbounded adversaries, because in $\mathbf{Game}_{\text{WRec}}(D, \mathcal{A})$, there is randomness in training and D is not given to $\mathcal{A}$ thus making it intuitively infeasible for $\mathcal{A}$ to recover the exact weight. Therefore, security property aiming at preventing this attack is relatively weak compared with other security properties defined in later parts of this section.

Only ensuring (D, ϵ)-weight-recovery-secure does not prevent practical attacks that do not rely on full weight recovery. However, while this security property is too weak to be practically useful for modeling real-world security, this can be useful for proving that some weight release schemes are blatantly insecure by constructing a valid and efficient weight recovery adversary. In our case study of TaylorMLP, we only used one released weight, but we can also define a stronger variant that allows $\mathcal{A}$ make multiple queries to $\mathsf{Release}(\mathsf{sk}, W)$, which would be convenient for proving insecurity with more than one released weights.

Preventing efficient inference. One of the underlying goals for weight recovery attacks is to improve the inference efficiency, as many current weight release schemes are designed to introduce an efficiency gap between the released weight and the original weight. For example, TaylorMLP aims to slow down the inference of released weight typically up to $8\times$ to prevent abuse and make a distinction between authorized and free versions of model weight.

Therefore, we introduce a stronger security property which is defined to prevent efficient inference on released model weights. To measure efficiency without loss of generality, here we consider $\mathsf{Cost}(F, x)$ as a function measuring the computational cost of running $F(x)$ where F is a procedure in our definitions. Cost can be measured both asymptotic or concrete according to a different context and is often instantiated with the running time of a procedure on a certain input.

We assume that weight release scheme Σ is designed such that $\mathsf{Cost}(\mathsf{Run'}, (\mathsf{pk}, W', x)) \gg \mathsf{Cost}(\mathsf{Run}, (W, x))$ for $W' \leftarrow\$ \mathsf{Release}(\mathsf{sk}, W)$ and $x \in \mathcal{X}$. Similarly, we introduce the following game for weight release scheme Σ on a given training dataset $D \in \mathcal{D}$.

```
Game_EffiInf(D, A)
1 :  $W \leftarrow\$ \mathsf{Train}(D)$
2 :  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}()$
3 :  $W' \leftarrow\$ \mathsf{Release}(\mathsf{sk}, W)$
4 :  $x \leftarrow\$ \mathcal{X}$
5 :  $y' \leftarrow\$ \mathsf{Run'}(\mathsf{pk}, W', x)$
6 :  $y^* \leftarrow\$ \mathcal{A}(\mathsf{pk}, W', x)$
7 :  return $\mathsf{Same}(y^*, y')$
```

The advantage of an adversary $\mathcal{A}$ in this game is defined as

$$\mathsf{Adv}_{\Sigma}^{\text{EffiInf}}(D, \mathcal{A}) = \Pr[\textbf{Game}_{\text{EffiInf}}(D, \mathcal{A}) \Rightarrow 1] \quad (4)$$

Definition 3.3. Then for all efficient adversary $\mathcal{A}$ that runs with computational cost t such that $t < \mathsf{Cost}(\mathsf{Run'}, (\mathsf{pk}, W', x))$ for $W' \leftarrow\$ \mathsf{Release}(\mathsf{sk}, W)$ and $x \in \mathcal{X}$ on some dataset $D \in \mathcal{D}$, if

$$\mathsf{Adv}_{\Sigma}^{\text{EffiInf}}(D, \mathcal{A}) \leq \epsilon \quad (5)$$

for some negligible ϵ with regard to the security paramter of the scheme Σ, then the weight release scheme Σ is considered to be (D, t, ϵ)-efficient-inference-secure (EffiInf-secure).

Different from (D, ϵ)-weight-recovery-secure, here we also explicitly consider the computational cost in the security definition. This is because if $\mathcal{A}$ is only polynomial time bounded, then $\mathcal{A}$ can just distill from the released weight into a more efficient weight space $\mathcal{W}^*$. Therefore, in the definition, we explicitly bound $t < \mathsf{Cost}(\mathsf{Run'}, (\mathsf{pk}, W', x))$ to ensure a valid adversary with less computation budget than direct inference. Similarly, we can also define a stronger variant that allows $\mathcal{A}$ to make multiple queries to $\mathsf{Release}(\mathsf{sk}, W)$ and add the query count as a parameter to the security definition.

Note that efficient-inference-secure is a stronger security property than weight-recovery-secure as performing efficient inference does not necessarily need weight recovery. For example, quantization of released weight can sometimes improve efficiency and thus can be considered a valid attack in this security notion. In §3.3 we will prove that efficient-inference-secure implies weight-recovery-secure.

Indistinguishability of released weights. Inspired by semantic security and IND-CPA properties in cryptographic works(Goldwasser & Micali, 1984), we can also define a similar security property in the form of indistinguishability, which is shown in the following game for weight release scheme Σ on a given training dataset $D \in \mathcal{D}$.

```
Game_W-IND(D, A)
1 :  $b \leftarrow\$ \{0, 1\}$
2 :  $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}()$
3 :  $W_0 \leftarrow\$ \mathsf{Train}(D)$
4 :  $W_1 \leftarrow\$ \mathsf{Train}(D)(\neg\mathsf{Same}(W_0, W_1))$
5 :  $W' \leftarrow\$ \mathsf{Release}(\mathsf{sk}, W_b)$
6 :  $b' \leftarrow\$ \mathcal{A}(\mathsf{pk}, W_0, W_1, W')$
7 :  return $b = b'$
```

The advantage of an adversary $\mathcal{A}$ in this game is defined as

$$\mathsf{Adv}_{\Sigma}^{\text{W-IND}}(D, \mathcal{A}) = 2\left|\Pr[\textbf{Game}_{\text{W-IND}}(D, \mathcal{A}) \Rightarrow 1] - \frac{1}{2}\right| \quad (6)$$

Definition 3.4. Then for all efficient adversary $\mathcal{A}$ runs within computation cost t on some dataset $D \in \mathcal{D}$, if

$$\mathsf{Adv}_{\Sigma}^{\text{W-IND}}(D, \mathcal{A}) \leq \epsilon \quad (7)$$

for some negligible ϵ with regard to the security paramter of the scheme Σ, then the weight release scheme Σ is considered to be (D, t, ϵ)-weight-indistinguishability-secure (W-IND-secure).

Intuitively, a good weight release scheme should leak no information about the original weight, and the game captures this property by letting an adversary distinguish the original weights of a released weight where the original weights are different in value but are trained on the same training data. If no adversary can win the game with a non-negligible advantage, that means that the scheme does not leak information about the original weight. Similarly, we can define a stronger variant that allows $\mathcal{A}$ to make multiple queries to different (W_0, W_1, W') and add the query count as a parameter to the security definition. By adding conditions on t, we can make (D, t, ϵ)-weight-indistinguishability-secure a stronger security property than weight-recovery-secure and efficient-inference-secure. We will show that weight-indistinguishability-secure implies weight-recovery-secure in §3.3.

3.3. Relations of the security properties

With the previously defined security properties, we show two implication relations between these properties under some assumption that fits the context. These relations are shown by contraposition, that is, if we want to show $A \to B$, then it is equivalent to showing $\neg B \to \neg A$ by constructing an adversary for A from an adversary for B. This is also known as security reductions in cryptographic works.

EffiInf-Secure implies WRec-Secure. By contraposition, we want to build an adversary $\mathcal{B}$ against EffiInf-Secure game for scheme Σ from an adversary $\mathcal{A}$ against WRec-secure game, given that weight release scheme Σ satisfies $\mathsf{Cost}(\mathsf{Run'}, (\mathsf{pk}, W', x)) \gg \mathsf{Cost}(\mathsf{Run}, (W, x)) + \mathsf{Cost}(\mathcal{A}, (\mathsf{pk}, W'))$ for $W' \leftarrow\$ \mathsf{Release}(\mathsf{sk}, W)$ and $x \in \mathcal{X}$. Adversary $\mathcal{B}$ is constructed as follows.

```
$\mathcal{B}_{\mathcal{A}}(\mathsf{pk}, W', x)$
1 : $W \leftarrow\$ \mathcal{A}(\mathsf{pk}, W')$
2 : return $\mathsf{Run}(W, x)$
```

The general idea is to first recover the original weight using $\mathcal{A}$ and then run inference on the recovered weight. We need to prove that $\mathcal{B}$ is both valid and efficient.

Proof. We know that $\mathcal{A}$ is valid, meaning that the recovered weight is the same as the original weight. Then by the correctness of weight release schemes, we have $\mathsf{Run}(W, x)$ is the same as $\mathsf{Run'}(\mathsf{pk}, W', x)$. Therefore $\mathcal{B}$ is valid. By the fact that $\mathsf{Cost}(\mathsf{Run'}, (\mathsf{pk}, W', x)) \gg \mathsf{Cost}(\mathsf{Run}, (W, x)) + \mathsf{Cost}(\mathcal{A}, (\mathsf{pk}, W'))$ we know that $\mathsf{Cost}(\mathcal{B}_{\mathcal{A}}, (\mathsf{pk}, W', x)) \approx \mathsf{Cost}(\mathcal{A}, (\mathsf{pk}, W')) + \mathsf{Cost}(\mathsf{Run}, (W, x)) \ll \mathsf{Cost}(\mathsf{Run'}, (\mathsf{pk}, W', x))$, which shows that $\mathcal{B}$ is efficient. Therefore $\mathcal{B}$ is both valid and efficient. □

From the proof we know that $\mathcal{B}$ wins whenever $\mathcal{A}$ wins, so we have

$$\mathsf{Adv}_{\Sigma}^{\mathrm{WRec}}(D, \mathcal{A}) \leq \mathsf{Adv}_{\Sigma}^{\mathrm{EffiInf}}(D, \mathcal{B}) \tag{8}$$

which gives EffiInf-Secure implies WRec-Secure.

W-IND-Secure implies WRec-Secure. Similarly by contraposition, we want to build an adversary $\mathcal{B}$ against W-IND-Secure game for scheme Σ from an efficient adversary $\mathcal{A}$ against WRec-Secure game under some feasible assumptions. Adversary $\mathcal{B}$ is constructed as follows.

```
$\mathcal{B}_{\mathcal{A}}(\mathsf{pk}, W_0, W_1, W')$
1 : $W \leftarrow\$ \mathcal{A}(\mathsf{pk}, W')$
2 : if $\mathsf{Same}(W, W_0)$ then return $0$
3 : else return $1$
```

The general idea is to first recover the original weight using $\mathcal{A}$ and then distinguish the original weight. Similarly, we need to prove that $\mathcal{B}$ is both valid and efficient.

Proof. We know that $\mathcal{A}$ is valid, meaning that the recovered weight is the same as the original weight. Because $\mathbf{Game}_{\text{W-IND}}$ ensures that $\neg\mathsf{Same}(W_0, W_1)$, so we have either $\mathsf{Same}(W_0, W)$ or $\mathsf{Same}(W_1, W)$. $\mathcal{B}$ runs with constant extra steps compared with $\mathcal{A}$. Since $\mathcal{A}$ is efficient, $\mathcal{B}$ is also efficient. Therefore $\mathcal{B}$ is both valid and efficient. □

From the proof we know that $\mathcal{B}$ wins whenever $\mathcal{A}$ wins, so we have

$$\mathsf{Adv}_{\Sigma}^{\mathrm{WRec}}(D, \mathcal{A}) \leq \mathsf{Adv}_{\Sigma}^{\mathrm{W\text{-}IND}}(D, \mathcal{B}) \tag{9}$$

which gives W-IND-Secure implies WRec-Secure.

4. Case Study: Insecurity of TaylorMLP

To show how our security definition can be applied to the analysis of real-world weight release schemes, we present a case study on a recent scheme named TaylorMLP. In this case study we will have a brief review of how TaylorMLP works and then we present an attack against the weight-recovery-security of TaylorMLP with experimental results. We then discuss the implications of this attack.

4.1. TaylorMLP

Based on syntax of weight release schemes described in §3.1, TaylorMLP can be seen as a weight release scheme $\Sigma = (\mathsf{Train}, \mathsf{Run}, \mathsf{KGen}, \mathsf{Release}, \mathsf{Run'})$, where

- Train is the original LLM training process on the training dataset.
- Run is the original LLM inference algorithm on the input prompt using the original model weight.
- KGen is not instantiated with a meaningful algorithm, and in this case $\mathcal{K}_{\mathsf{sk}}$ and $\mathcal{K}_{\mathsf{pk}}$ are both $\emptyset$.
- $\mathsf{Release}$ is the algorithm that converts MLP in LLM weight into TaylorMLP format as is shown in Algorithm 1.

Model	OPT-125M	OPT-1.3B	OPT-2.7B	OPT-6.7B	Llama2-7B	Llama2-13B
# TaylorMLP Parameters	28,311,552	402,653,184	838,860,800	2,147,483,648	1,442,840,576	2,831,155,200
# Recovered Parameters	28,310,784	401,649,665	837,347,840	2,144,915,459	1,442,840,576	2,831,155,200
Recovered Ratio	99.99%	99.76%	99.82%	99.88%	100.00%	100.00 %
Running Time	19.70 s	116.79 s	187.88 s	417.92 s	318.60 s	571.05 s
Attack Cost in USD	0.01	0.05	0.07	0.17	0.13	0.22

Table 1. Attack Performance. "# TaylorMLP Parameters" refers to the total number of weight parameters processed by TaylorMLP in the model, while "# Recovered Parameters" denotes the number of weight parameters successfully recovered by our attack. "Recovered Ratio" is defined as the proportion of weights that can be successfully recovered using our proposed method, whereas "successfully" is defined as the relative error of the recovered weights is less than 1%. Relative error is calculated as $|\mathbf{W}_{\text{rec}} - \mathbf{W}|/|\mathbf{W}| \cdot 100\%$ where $\mathbf{W}_{\text{rec}}$ denotes the recovered weights and $\mathbf{W}$ denotes the ground-truth weights.

Algorithm 1 Transforming MLP to TaylorMLP

```
Require: MLP($\bullet|\mathbf{V}, \mathbf{b}, \mathbf{W}, \mathbf{c}$) and $\mathbf{z}_0$
Ensure: TaylorMLP($\bullet|\mathbf{V}, \mathbf{z}_0, \{\Theta_{i,0}, \cdots, \Theta_{i,N}\}_{i=1}^{D}$)
1: for $i := 1$ to $D$ do
2:     $\mathbf{W}_i$ and $c_i$ take the $i$-th row and $i$-th element of $\mathbf{W}$ and $\mathbf{c}$, respectively.
3:     $\Theta_{i,0} = \mathbf{W}_i \odot \text{Act}(\mathbf{z}_0 + \mathbf{b}) + c_i$
4:     for $n := 1$ to $N$ do
5:         $\Theta_{i,n} = \mathbf{W}_i \odot \text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})(n!)^{-1}$
6:     end for
7: end for
```

- Run' is a LLM inference algorithm on input prompt using TaylorMLP format weight.

In short, TaylorMLP transforms $(\mathbf{b}, \mathbf{W}_i, c_i)$ into $(\mathbf{z}_0, [\Theta_{i,0}, \Theta_{i,1}, ..., \Theta_{i,N}])$ by calculating

$$\mathbf{z}_0 = \frac{\mathbf{z}_{\max} + \mathbf{z}_{\min}}{2} \tag{10}$$

$$\Theta_{i,n} = \mathbf{W}_i \odot \frac{\text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})}{n!} \tag{11}$$

where $\text{Act}^{(n)}$ is the n-th order derivative of activation function and $\mathbf{z}_{\max\,/\,\min}$ is the $\max/\min$ value collected on some test input. The correctness of TaylorMLP can be shown below.

$$y_i = \text{Act}(\mathbf{z} + \mathbf{b}) \cdot \mathbf{W}_i + c_i \tag{12}$$

$$\approx \left\langle \mathbf{W}_i, \sum_{n=0}^{N} \frac{\text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})}{n!} \odot (\mathbf{z} - \mathbf{z}_0)^n \right\rangle + c_i \tag{13}$$

$$= \sum_{n=0}^{N} \left\langle \mathbf{W}_i \odot \text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})(n!)^{-1}, (\mathbf{z} - \mathbf{z}_0)^n \right\rangle \tag{14}$$

$$= \sum_{n=0}^{N} \langle \Theta_{i,n}, (\mathbf{z} - \mathbf{z}_0)^n \rangle \tag{15}$$

4.2. The weight recovery attack

The general idea is that while TaylorMLP introduces randomness in $\mathbf{z}_0$, since most transformations are elementwise and more parameters are added, it is possible to solve equations from the released weight to recover the original weight. We present our attack for weight recovery as follows.

The attack. We start by transforming Equation 11, which gives

$$\mathbf{W}_i = (n!)\Theta_{i,n} \odot \frac{1}{\text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})} \tag{16}$$

We can see that as long as we can solve for the value of $\mathbf{b}$, we can solve $\mathbf{W}_i$ given $\text{Act}^{(n)}$ is invertible in some range. Note that we have multiple such equations. Consider a pair (a, b) satisfying $0 \leq a < b \leq N$, we have

$$\mathbf{W}_i = (a!)\Theta_{i,a} \odot \frac{1}{\text{Act}^{(a)}(\mathbf{z}_0 + \mathbf{b})} \tag{17}$$

$$= (b!)\Theta_{i,b} \odot \frac{1}{\text{Act}^{(b)}(\mathbf{z}_0 + \mathbf{b})} \tag{18}$$

After rearranging, we get

$$\frac{(a!)\Theta_{i,a}}{(b!)\Theta_{i,b}} = \frac{\text{Act}^{(a)}(\mathbf{z}_0 + \mathbf{b})}{\text{Act}^{(b)}(\mathbf{z}_0 + \mathbf{b})} \tag{19}$$

Since this equation holds elementwise, for each element of $(\mathbf{z}_0 + \mathbf{b})$, we can obtain one equation. To simplify the notation, we denote $f_{a,b}(x) = \frac{\text{Act}^{(a)}(x)}{\text{Act}^{(b)}(x)}$. Then we have

$$\frac{(a!)\Theta_{i,a}[j]}{(b!)\Theta_{i,b}[j]} = f_{a,b}((\mathbf{z}_0 + \mathbf{b})[j]) \tag{20}$$

for $j \in [D]$. Considering that for most pairs (a, b), $f_{a,b}(x)$ is expected to be invertible, we can recover $(\mathbf{z}_0 + \mathbf{b})$ with high probability, thus solving for the value of $\mathbf{b}$, and consequently recovering $\mathbf{W}_i$.

Efficiency. The attack is efficient given Act is efficiently invertible using numerical methods like Newton's method. For activation functions like SiLU and GeLU as used in the original TaylorMLP paper, the attack takes $O(N \cdot \#\text{Params})$ time and space, which is efficient.

Numerical stability. In the real scenario for activation functions like SiLU, $f_{a,b}(x) = \frac{\text{Act}^{(a)}(x)}{\text{Act}^{(b)}(x)}$ suffers from occasional problems with floating point numerical stability on some outlier weight values. This is partly due to the widespread use of float16 in LLM inference and switching to double-precision floating point numbers mitigates this issue. However, due to the fractional nature of $f_{a,b}$, some outlier weight values can not be reliably solved due to numerical stability issues, and this leads to a small portion ($< 1\%$) of weights that can not be recovered in practice, which is discussed in our experiments.

4.3. Experiments

Experiment settings. Following the evaluation in the original TaylorMLP paper, we choose OPT and Llama model family to evaluate the effectiveness of our attack. We process the MLP layers of each transformer block of the model weights to get the released weight. We then implemented our attack in Python 3 using numpy and scipy. We run the attack to recover the MLP layer weights using the released weight as input. All experiments were conducted on an x86 Linux machine equipped with 20 CPU cores and 128 GB of memory without GPU acceleration.

We measure the number and ratio of successfully recovered weight values as well as running time and estimated cost of the attack. A weight value is considered recovered successfully if its error compared with the original value is less than 1%. Estimated cost is calculated based on the price of instances from mainstream cloud providers that are comparable to the x86 Linux machine used in the experiment.

Results. The results are shown in Table 1. In our experiments, models from OPT and Llama of various sizes consistently achieved recovered rates very close to 100%, indicating that the relative error between $\mathbf{W}_{\text{rec}}$ and $\mathbf{W}$ is negligible. Furthermore, due to differences in architecture and weight distribution between the Llama-2 and OPT models, the relative error also varies. Specifically, the recovery results for Llama-2 are noticeably better than those for OPT.

We measured the time required to recover the entire set of model weights. As shown in our results, the OPT-125M model requires less than 20 seconds, while the largest model tested, Llama-2-13B, takes under 10 minutes. The recovery process is quite efficient, even on a standard CPU machine.

We then roughly estimated the monetary cost associated with the weight recovery process. Referring to `m8g.8xlarge` instance with a similar configuration on AWS, the price is approximately \$1.43616 per hour. Therefore the highest cost among tested models is only \$0.22 for a successful attack for computation. This demonstrates that the attack is cost-effective and feasible even for attackers with limited budget in practice.

Additionally, we visualize the relative error between the ground-truth weights and the recovered weights using a heat map. As shown in Figure 2, the majority of errors are extremely low, with only a few channels exhibiting relatively larger errors. This demonstrates the effectiveness of our recovery method, as it is able to accurately recover most of the elements.

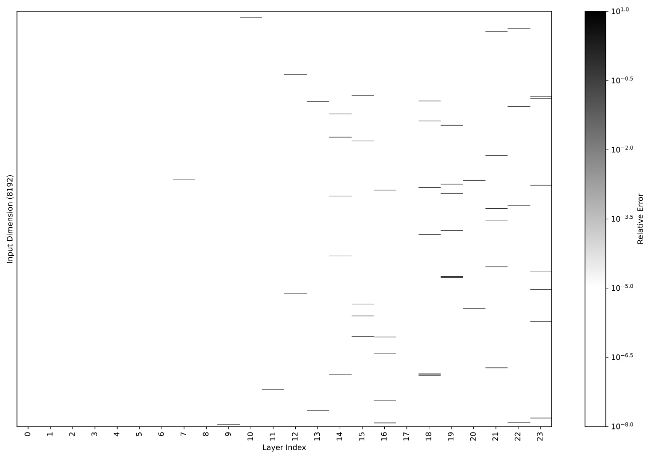

Figure 2. Relative Error for recovered weights in OPT-1.3B. We concatenate weights across all layers of the OPT-1.3B model along the x-axis for comprehensive visualization. To enhance visual clarity and emphasize variations, we apply a $\log_{10}$ scale to the relative error values.

4.4. Takeaways

The attack demonstrates that TaylorMLP is vulnerable to our weight recovery attack, thereby failing to provide weight-recovery-security. From the security relations established in §3.3, TaylorMLP consequently also fails to ensure efficient-inference-security and weight-indistinguishability-security. These results establish the insecurity of TaylorMLP as a weight release scheme designed to protect model ownership and prevent unauthorized use.

While the original TaylorMLP paper conducted detailed evaluations of correctness and efficiency gaps, and claimed that fine-tuning the released model is infeasible, the authors did not formulate formal security definitions or provide rigorous security proofs under reasonable assumptions. This lack of formal security analysis led to unnoticed design vulnerabilities that we successfully exploited in our attack.

5. Design Principles for Future Weight Release Schemes

Based on our security definitions and analysis of TaylorMLP, we outline key principles for designing and evaluating future weight release schemes.

Clear security goals. Current real-world weight release schemes are usually exclusive, meaning that two weight release schemes are generally not easily combined to hedge the risk. For example, if a developer decides to protect the model using TaylorMLP and has released the model in this format, he or she may not easily switch to another scheme given TaylorMLP is broken, thus the already released model is now under threat.

Depending on typical use cases for developers, users and adversaries, security goals can vary. While informal security notions are not enough against real-world attacks, they are a good starting point for sketching the security goals of a weight release scheme. Future schemes should begin with precise security definitions that specify the adversarial model, threat capabilities, and desired security properties, drawing from established cryptographic frameworks such as digital signature schemes(Guo et al., 2023).

Provable security. To ensure robust security guarantees, weight release schemes should provide formal security proofs. A security proof requires: 1) formal syntax defining the scheme's algorithms, 2) explicit computational hardness assumptions, 3) precise security definitions, and 4) rigorous proofs using established techniques such as security reductions. Such proofs relate the scheme's security to widely accepted hardness assumptions, providing theoretical foundations beyond informal arguments.

Offensive evaluation. Provable security alone is insufficient due to gaps between theoretical models and real-world implementations(Koblitz & Menezes, 2007). Schemes require thorough offensive evaluation to identify potential vulnerabilities before deployment. As demonstrated in our TaylorMLP analysis, successful attacks serve as "proofs of insecurity" that can invalidate informal security claims. This adversarial testing reveals practical limits and guides parameter selection for secure deployment.

Compatibility requirements. Beyond security, practical adoption requires compatibility with existing ML infrastructure. For example, the Run' algorithm may integrate seamlessly with mainstream inference frameworks to avoid imposing additional implementation burdens on users. Schemes that require specialized execution environments or exotic data formats can face significant adoption barriers regardless of their security properties.

6. Conclusion

We have taken a step toward establishing a rigorous theoretical foundation for secure weight release by formalizing security definitions for weight release schemes. Our case study of TaylorMLP reveals model weight recovery vulnerabilities, demonstrating that existing schemes fail to achieve their informal security claims and highlighting the gap between promised and actual security guarantees.

We hope this work will advocate for rigorous research at the intersection of machine learning and security communities. By providing concrete security definitions and demonstrating their application, we establish a blueprint for designing and evaluating future secured weight release schemes that provide meaningful security guarantees rather than false assurances.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Bellare, M. and Rogaway, P. Code-based game-playing proofs and the security of triple encryption. *Cryptology ePrint Archive*, 2004.

Dong, Y., Lu, W.-j., Zheng, Y., Wu, H., Zhao, D., Tan, J., Huang, Z., Hong, C., Wei, T., and Chen, W. Puma: Secure inference of llama-7b in five minutes. *arXiv preprint arXiv:2307.12533*, 2023.

Goldstein, B. F., Patil, V. C., Ferreira, V. C., Nery, A. S., França, F. M., and Kundu, S. Preventing dnn model ip theft via hardware obfuscation. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(2): 267–277, 2021.

Goldwasser, S. and Micali, S. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. ISSN 0022-0000. doi: https://doi.org/10.1016/0022-0000(84)90070-9. URL `https://www.sciencedirect.com/science/article/pii/0022000084900709`.

Guo, F., Susilo, W., Chen, X., Jiang, P., Lai, J., and Zhao, Z. Sok: Research motivations of public-key cryptography. *Cryptology ePrint Archive*, 2023.

Jin, H., Zhang, C., Shi, S., Lou, W., and Hou, Y. T. Proflingo: A fingerprinting-based intellectual property protection scheme for large language models. In *2024 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9. IEEE, 2024.

Koblitz, N. and Menezes, A. J. Another look at" provable security". *Journal of Cryptology*, 20:3–37, 2007.

Li, H., Chen, Y., Luo, J., Wang, J., Peng, H., Kang, Y., Zhang, X., Hu, Q., Chan, C., Xu, Z., et al. Privacy in large language models: Attacks, defenses and future directions. *arXiv preprint arXiv:2310.10383*, 2023.

Li, S., Yao, L., Gao, J., Zhang, L., and Li, Y. Double-i watermark: Protecting model copyright for llm fine-tuning. *arXiv preprint arXiv:2402.14883*, 2024.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Lu, W.-j., Huang, Z., Gu, Z., Li, J., Liu, J., Hong, C., Ren, K., Wei, T., and Chen, W. Bumblebee: Secure two-party inference framework for large transformers. *Cryptology ePrint Archive*, 2023.

Pang, Q., Zhu, J., Möllering, H., Zheng, W., and Schneider, T. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 4753–4771. IEEE, 2024.

Refael, Y., Hakim, A., Greenberg, L., Aviv, T., Lokam, S., Fishman, B., and Seidman, S. Slip: Securing llms ip using weights decomposition. *arXiv preprint arXiv:2407.10886*, 2024.

Shoup, V. Sequences of games: a tool for taming complexity in security proofs. *cryptology eprint archive*, 2004.

Tamirisa, R., Bharathi, B., Phan, L., Zhou, A., Gatti, A., Suresh, T., Lin, M., Wang, J., Wang, R., Arel, R., et al. Tamper-resistant safeguards for open-weight llms. *CoRR*, 2024.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Wang, G., Chuang, Y.-N., Tang, R., Zhong, S., Yuan, J., Jin, H., Liu, Z., Chaudhary, V., Xu, S., Caverlee, J., and Hu, X. Taylor unswift: Secured weight release for large language models via Taylor expansion. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6928–6941, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.393. URL https://aclanthology.org/2024.emnlp-main.393/.

Wen, Y., Guo, J., and Huang, H. Cotguard: Using chain-of-thought triggering for copyright protection in multi-agent llm systems. *arXiv preprint arXiv:2505.19405*, 2025.

Xu, Y., Liu, A., Hu, X., Wen, L., and Xiong, H. Mark your llm: Detecting the misuse of open-source large language models via watermarking. *arXiv preprint arXiv:2503.04636*, 2025.

Yan, B., Li, K., Xu, M., Dong, Y., Zhang, Y., Ren, Z., and Cheng, X. On protecting the data privacy of large language models (llms): A survey. *arXiv preprint arXiv:2403.05156*, 2024.

A. Analysis of numerical stability of SiLU/GeLU derivative.

The numerical stability of higher-order derivatives of the SiLU and GeLU activation function exhibits significant sensitivity when evaluating ratios of derivatives. As demonstrated in Fig.3 and Fig.4, ratios of SiLU derivatives within the interval $[-10, 10]$ frequently encounter numerical instability, often resulting in large fluctuations or undefined values at various points away from zero. While stability improves in the vicinity of zero due to more balanced numerical magnitudes, even slight deviations or outliers in the input distributions can cause pronounced instabilities. Importantly, since different network architectures and model sizes inherently produce distinct internal numerical distributions, the resulting numerical instabilities manifest differently across these models, leading to varying degrees of reconstruction errors and inaccuracies. Therefore, the numerical instability induced by varying distributions across different model architectures and scales inevitably introduces a certain degree of error into our reconstruction method, underscoring the necessity for careful numerical considerations when employing higher-order SiLU derivatives in analysis and optimization tasks.

B. Computational Infrastructure.

Table 2 provides details on the computational infrastructure and environment information.

Name	**Value**
CPU	Intel Core Ultra 7 265K
Memory	128GB
Data type	torch.bfloat16
OS	Debian GNU/Linux trixie/sid
Python	3.13.3
numpy	2.2.6
torch	2.7.1
transformers	4.52.4
scipy	1.15.3
matplotlib	3.10.3

Table 2. Configuration of experiment and computing infrastructure.

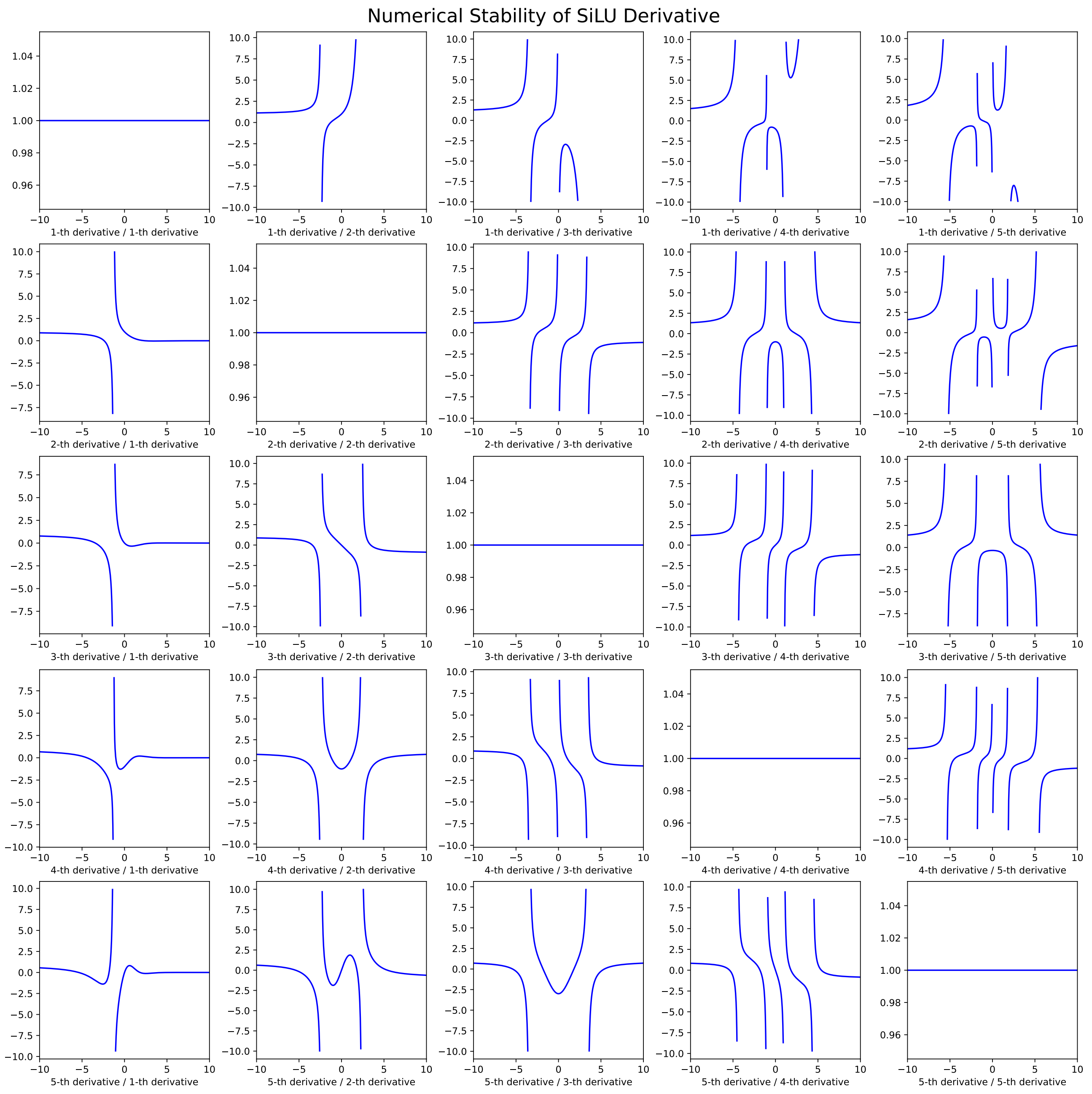

Figure 3. Numerical stability analysis of SiLU derivative ratios. Each subplot shows the ratio between SiLU derivatives from first to fifth order, evaluated over the interval. Diagonal entries represent ratios of derivatives with themselves and therefore equal 1. Many off-diagonal entries exhibit significant numerical instability, characterized by large fluctuations or extreme values, particularly away from zero. These instabilities highlight potential sources of errors and inaccuracies in practical computations involving higher-order SiLU derivatives.

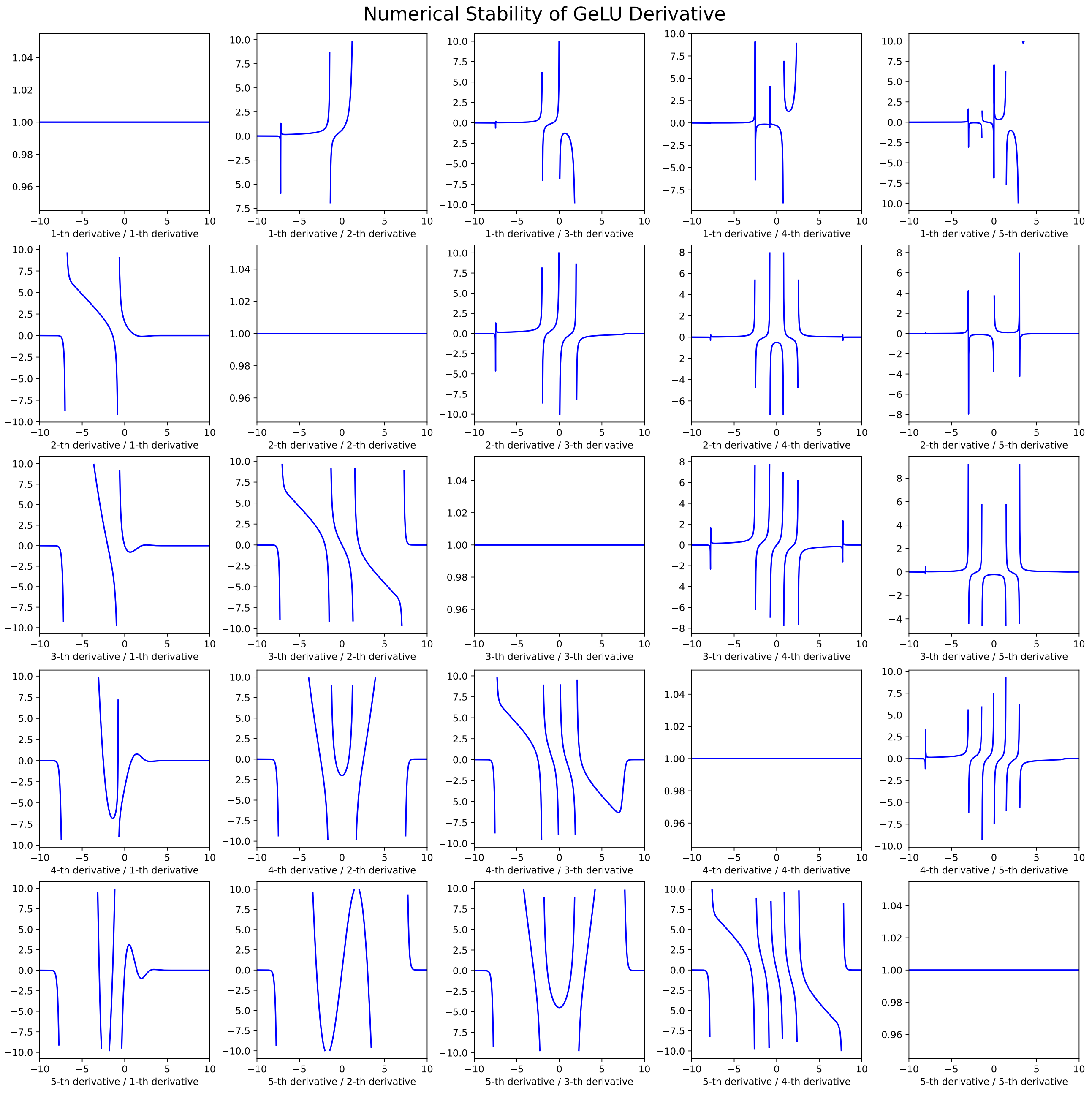

Figure 4. Numerical stability analysis of GeLU derivative ratios. Similar to SiLU, ratios between GeLU derivatives from first to fifth order show numerical instabilities characterized by large fluctuations and extreme values, particularly away from zero. Although minor differences exist, overall stability patterns closely resemble those observed for SiLU.