Dual Defense: Enhancing Privacy and Mitigating Poisoning Attacks in Federated Learning

Runhua Xu
Beihang University
runhua@buaa.edu.cn

Shiqi Gao
Beihang University
gaoshiqi@buaa.edu.cn

Chao Li
Beijing Jiaotong University
li.chao@bjtu.edu.cn

James Joshi
University of Pittsburgh
jjoshi@pitt.edu

Jianxin Li[*]
Beihang University and Zhongguancun Laboratory
lijx@buaa.edu.cn

Abstract

Federated learning (FL) is inherently susceptible to privacy breaches and poisoning attacks. To tackle these challenges, researchers have separately devised secure aggregation mechanisms to protect data privacy and robust aggregation methods that withstand poisoning attacks. However, simultaneously addressing both concerns is challenging; secure aggregation facilitates poisoning attacks as most anomaly detection techniques require access to unencrypted local model updates, which are obscured by secure aggregation. Few recent efforts to simultaneously tackle both challenges offen depend on impractical assumption of non-colluding two-server setups that disrupt FL's topology, or three-party computation which introduces scalability issues, complicating deployment and application. To overcome this dilemma, this paper introduce a **D**ual **D**efense **Fed**erated learning (*DDFed*) framework. *DDFed* simultaneously boosts privacy protection and mitigates poisoning attacks, without introducing new participant roles or disrupting the existing FL topology. *DDFed* initially leverages cutting-edge fully homomorphic encryption (FHE) to securely aggregate model updates, without the impractical requirement for non-colluding two-server setups and ensures strong privacy protection. Additionally, we proposes a unique two-phase anomaly detection mechanism for encrypted model updates, featuring secure similarity computation and feedback-driven collaborative selection, with additional measures to prevent potential privacy breaches from Byzantine clients incorporated into the detection process. We conducted extensive experiments on various model poisoning attacks and FL scenarios, including both cross-device and cross-silo FL. Experiments on publicly available datasets demonstrate that *DDFed* successfully protects model privacy and effectively defends against model poisoning threats.

1 Introduction

Federated learning (FL)[18] is gaining popularity as a collaborative model training paradigm that provides primary privacy protection by eliminating the need of sharing private training data. Based on the participants' scale, FL is typically divided into two categories: cross-silo FL and cross-device FL[17]. Cross-device FL typically involves numerous similar devices, while cross-silo FL usually includes fewer participants like organizations. Recent studies show that FL mainly confronts two types of threats: privacy risks from curious adversaries attempting to compromise data privacy

[*] corresponding author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

through methods like membership inference and model inversion attacks, and security risks from Byzantine adversaries looking to damage the final model's integrity with backdoors or by lowering its accuracy [2, 24, 14, 11, 3, 1, 34].

To mitigate privacy risks in FL, researchers have developed a range of techniques to bolster privacy. These encompass differential privacy-based aggregation [32], as well as secure aggregation approaches using homomorphic encryption[41], functional encryption[35], and secure multi-party computation[6, 43]. Aside from privacy concerns, many studies have proposed strategies to identify and mitigate potentially harmful updates during the model aggregation phase, thereby safeguarding the global model against adversarial attacks. Notable Byzantine-resistant aggregation mechanisms encompass the Krum fusion method[5], cosine defense aggregation mechanism[29, 38], and median/mean-based strategies like clipping median and trimmed mean strategies [40]. Research in these two areas has been conducted separately, and addressing both issues at once continues to be challenging. This difficulty arises because secure aggregation makes it easier for adversarial attacks to occur, as most anomaly detection methods need access to "unencrypted" local model updates that secure aggregation protects.

Few recent efforts [39, 13, 16, 43, 15, 23, 9, 20] to tackle both challenges simultaneously often depend on differential privacy techniques [39, 13, 16, 22, 12], which can degrade model performance due to added noise, or rely on impractical non-colluded two-server assumption that disrupts FL's topology[43, 15, 23, 9, 20], complicating its deployment and application. In light of these limitations, a critical yet overlooked question is *how to create a straightforward dual defense strategy that simultaneously strengthens privacy protection and mitigates poisoning attacks without introducing new participant roles or altering the single-server multiple-clients structure?*

To address this dilemma, this paper proposes a **D**ual **D**efense approach that simultaneously enhances privacy protection and combats poisoning attacks in **Fed**erated learning (*DDFed*), without changing the structure of current FL frameworks. *DDFed* initially leverages cutting-edge cryptographic technology, specifically fully homomorphic encryption (FHE), to securely aggregate model updates without the impractical assumption of non-colluding two-server setups and ensures strong privacy protection by permitting only the aggregation server to perform secure aggregation in the dark. To tackle the challenge of detecting malicious models within encrypted model updates, *DDFed* introduces a novel two-phase anomaly detection mechanism. This approach enables cosine similarity computation over encrypted models and incorporates a feedback-driven collaborative selection process, with additional measures to prevent potential privacy breaches from Byzantine clients incorporated into the detection mechanism. Our main contributions are summarized as follows:

- We introduce a dual defense strategy that simultaneously boosts privacy and combats poisoning attacks in federated learning. This is achieved by integrating FHE-based secure aggregation with a mechanism for detecting malicious encrypted models based on similarity.
- To effectively detect malicious models in encrypted updates, we propose a novel two-phase anomaly detection mechanism with extra safeguards against potential privacy breaches by Byzantine clients during the detection process. Additionally, we introduce a clipping technique to bolster defenses against diverse poisoning attacks.
- We carried out comprehensive experiments on multiple model poisoning attacks and federated learning scenarios, covering both cross-device FL and cross-silo FL. Our experiments with publicly accessible datasets demonstrate *DDFed*'s effectiveness in safeguarding model privacy and robustly defending against model poisoning threats.

2 Related Works

Privacy Risks and Countermeasures in FL The fundamental design of FL ensures that all training data stays with its owner, offering basic privacy. However, it still exposes vulnerabilities to inference attacks, which allow adversaries to extract information about the training data used by each party [24, 27, 2, 24, 14, 11]. In some cases, the risk of private information leakage may be unacceptable. Therefore, several defenses have been suggested to mitigate these risks, including differential privacy (DP) and secure aggregation (SA), based on various cryptographic primitives such as (partial) homomorphic encryption [21, 41], threshold Paillier [30], functional encryption [36], and pairwise masking protocols [6].

Poisoning Risks and Countermeasures in FL. Besides privacy inference attacks, FL is also susceptible to poisoning attacks, where adversaries can compromise certain clients and manipulate their data or models to intentionally worsen the global model's performance by introducing corrupted updates during training. This paper focuses on untargeted model attacks, whose goal is to significantly diminish the effectiveness of the global model through methods such as Inner Product Manipulation (IPM) attack [34], scaling attack[1], and "a little is enough" (ALIE) attack [3]. Several strategies have been developed to counteract the impact of attacks, ensuring they don't compromise model performance. These strategies fall into two categories: client-side and server-side defenses. Client-side defenses adjust the local training algorithm with a focus on secure client updates[28], whereas server-side defenses [5, 29, 38, 40] either reduce the influence of updates from malicious clients through adjusted aggregation weights or use clustering techniques to aggregate updates from trustworthy clients only. However, these defense strategies typically operate under the assumption that model updates are not encrypted, which contradicts the objectives of privacy-focused secure aggregation defense strategies.

Private and Robust Federated Learning. In privacy-preserving FL, identifying poisoning attacks is harder because of the need to balance local model privacy with the detection of harmful models. Only a few existing studies like those mentioned in [39, 13, 16, 15] employ Byzantine-resilient aggregation through differential-privacy techniques. This approach necessitates a compromise between privacy and model accuracy. Additionally, recent initiatives have been launched to address this problem through diverse methods by using various secure computation technologies. These include 3PC[9], which faces scalability limitations; an oblivious random grouping method constrained by its design for partial parameter disclosure[43]; and both additive secret sharing[20] and two-trapdoor homomorphic encryption[23], which depend on the impractical assumption of non-colluding dual servers.

3 Dual Defense Federated Learning Framework

3.1 Formulation and Assumption

Formulation. A typical FL framework involves m clients, $\mathcal{C}_1, ..., \mathcal{C}_m$, and a single aggregation server $\mathcal{A}$. Each client $\mathcal{C}_i$ possesses its own dataset D_i. The overarching goal in FL across these m clients is to minimize the global objective function:

$$\min_{\boldsymbol{W}_1,...,\boldsymbol{W}_m} \frac{1}{m} \sum_{i=1}^{m} \frac{|D_i|}{\sum_{i=1}^{m} |D_i|} L_i(\boldsymbol{W}_i; D_i). \tag{1}$$

Here, L_i represents the local loss function for each client's data, and $\boldsymbol{W}_i$ are the local model parameters specific to client $\mathcal{C}_i$. The term D_i refers to the private dataset of client i, with $|D_i|$ indicating its size in terms of sample count. In short, the goal of general FL is to learn an optimal global model $\boldsymbol{W}_G$ across m clients. This is achieved by periodically synchronizing the model parameters from all clients using specified fusion algorithms like *FedAvg* and its variants, with the aggregation server $\mathcal{A}$ over several training rounds.

Due to various malicious activities, including inference attacks that aim to steal private information from legitimate clients and poisoning attacks designed to undermine model integrity by degrading its performance, existing privacy-preserving FL often relies on a secure aggregation mechanism[21, 41, 30, 36, 6]. Typically, without loss of generality, during the t-th federated learning training round, each client $\mathcal{C}_i$ secures its local model update $\boldsymbol{W}_i$ - referred to as $[\![\boldsymbol{W}_i]\!]$ throughout this paper - before transmitting it to the aggregation server. This is achieved by using various privacy-enhancing technologies such as homomorphic encryption and secure multi-party computation.

Threat Assumption. *DDFed* tolerates an adversary, capable of corrupting any subset of local clients at a specified ratio $r_{\text{ATTACK}}, s.t., r_{\text{ATTACK}} < 0.5$, to carry out model poisoning attacks that degrade the global model's performance. Additionally, we assume the aggregation server $\mathcal{A}$ is semi-honest (honest-but-curious), meaning it adheres to the protocol but seeks to glean as much private information as possible. Similarly, the compromised clients $\mathcal{C}_i^{\text{ADV}}$ can conduct privacy inference attacks like those performed by $\mathcal{A}$. In summary, regarding privacy preservation, both the inquisitive $\mathcal{A}$ and the corrupted client subset aim to extract private information from benign clients; however,

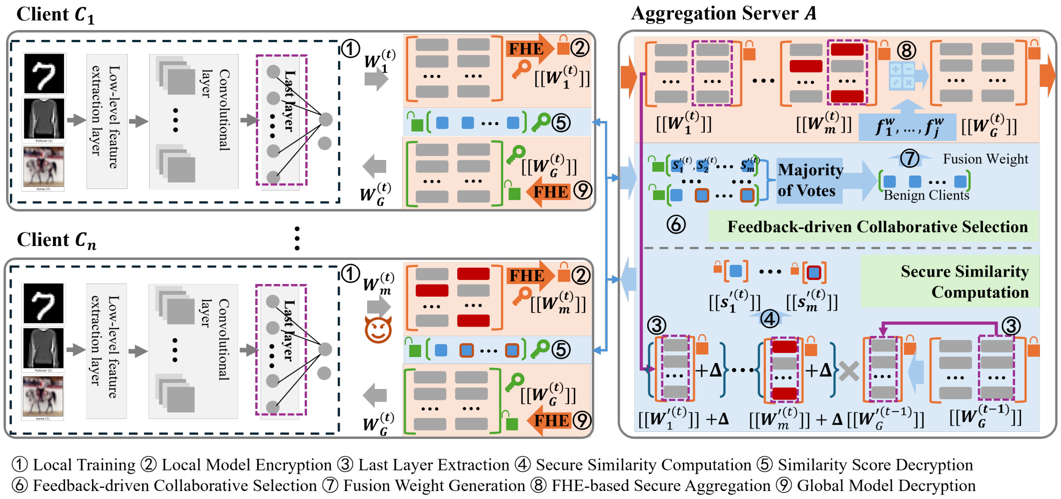
Client C_1
Low-level feature extraction layer
Convolutional layer
Last layer
①
$W_1^{(t)}$
FHE
②
$[[W_1^{(t)}]]$
⑤
$[[W_G^{(t)}]]$
FHE
⑨
$W_G^{(t)}$
Client C_n
$W_m^{(t)}$
$[[W_m^{(t)}]]$
Aggregation Server A
$[[W_1^{(t)}]]$
$[[W_m^{(t)}]]$
⑧
$f_1^w, \dots, f_j^w$
$[[W_G^{(t)}]]$
$s_1^{(t)}, s_2^{(t)} \cdots s_m^{(t)}$
Majority of Votes
⑦ Fusion Weight
Benign Clients
⑥ Feedback-driven Collaborative Selection
Secure Similarity Computation
$[[s_1'^{(t)}]]$ $[[s_m'^{(t)}]]$
③ ④ ③
+Δ +Δ
$[[W_1'^{(t)}]] + \Delta$ $[[W_m'^{(t)}]] + \Delta$ $[[W_G'^{(t-1)}]]$ $[[W_G^{(t-1)}]]$
① Local Training ② Local Model Encryption ③ Last Layer Extraction ④ Secure Similarity Computation ⑤ Similarity Score Decryption
⑥ Feedback-driven Collaborative Selection ⑦ Fusion Weight Generation ⑧ FHE-based Secure Aggregation ⑨ Global Model Decryption

Figure 1: Overview of *DDFed* framework and illustration of a single round *DDFed* training.

only the corrupted client subset will also initiate model poisoning attacks to undermine the global model.

3.2 Framework Details

Objective of DDFed. *DDFed* is designed to bolster privacy protection and mitigate model poisoning attacks seamlessly within the existing FL framework. Unlike existing private and robust approaches [39, 13, 16, 43, 15, 23, 9, 20] that add new participant roles or depend on differential privacy, which may compromise model performance, *DDFed* maintains effectiveness efficiently. *DDFed* introduces a dual defense strategy that combines fully homomorphic encryption (FHE) for secure data aggregation with an optimized similarity-based mechanism to detect malicious models, ensuring unparalleled privacy protection and security against model poisoning attacks.

Similarity-based methods are commonly used in existing studies for anomaly detection models [29, 38]. Specifically, it computes the cosine similarity between each local model update of training round t and the global model from the previous round $t-1$:

$$cos(\alpha_i) = \frac{\langle \boldsymbol{W}_i^{(t)}, \boldsymbol{W}_G^{(t-1)} \rangle}{\|\boldsymbol{W}_i^{(t)}\|_2 \cdot \|\boldsymbol{W}_G^{(t-1)}\|_2}, \tag{2}$$

where α_i denotes the angle between global model weights $\boldsymbol{W}_G^{(t-1)}$ and local model update $\boldsymbol{W}_i^{(t)}$ of client $\mathcal{C}_i$. However, existing similarity-based mechanisms [29, 38] offer no privacy protection for local model updates, and integrating FHE into them poses significant challenges. These challenges arise from FHE's limitations in performing division and comparison operations, which are essential for identifying benign clients in these methods.

Framework Overview and Training Process. Figure 1 provides an overview of *DDFed* framework, which includes several clients $\mathcal{C}_1, ..., \mathcal{C}_m$ and a single aggregation server $\mathcal{A}$, consistent with the architecture of most existing FL frameworks. In the following section, we demonstrate the *DDFed* training process. Due to space limitations, the formal algorithm pseudocode is provided solely in Appendix A.1.

Before the FL training begins, each client $\mathcal{C}_i$ is equipped with an FHE key pair (PK, SK). During the FL training phase, let's assume that in the t-th round, each client $\mathcal{C}_i$ trains a local model $\boldsymbol{W}_i^{(t)}$ (①) and performs the normalization and encryption as $\llbracket \boldsymbol{W}_i^{(t)} \rrbracket = \text{FHE.ENC}_{\text{PK}}(\boldsymbol{W}_i^{(t)})$ with public key PK (②). Upon receiving encrypted local models, $\{\llbracket \boldsymbol{W}_i^{(t)} \rrbracket\}_{i \in [1,...,m]}$, $\mathcal{A}$ starts to detect anomaly model updates over all encrypted local models. Specifically, $\mathcal{A}$ first extracts the last layer, denoted as $\{\llbracket \boldsymbol{W}_i^{'(t)} \rrbracket\}_{i \in [1,...,m]}$, which remains encrypted (③), and adds a perturbation $\Delta^{(t)}$ to safeguard against potential privacy attacks by malicious clients. Next, it retrieves the last layer of the encrypted global model from the previous training round ($[\![\boldsymbol{W}_G^{'(t-1)}]\!]$), The method for adding perturbations will be discussed in Section 3.2. Then, $\mathcal{A}$ performs secure inner-product between each perturbed $[\![\boldsymbol{W}_i^{'(t)}]\!] + \Delta^{(t)}$ and $[\![\boldsymbol{W}_G^{'(t-1)}]\!]$ to derive encrypted similarity score, denoted as $[\![\boldsymbol{s}^{'(t)}]\!] = ([\![s_1^{'(t)}]\!], ..., [\![s_m^{'(t)}]\!])$, and query each client (④). After receiving $[\![\boldsymbol{s}^{'(t)}]\!]$, each client $\mathcal{C}_i$ decrypts it to obtain the plaintext scores $\boldsymbol{s}_i^{'(t)}$. Subsequently, each client submits their list of similarity scores (⑤). It's important to note that at this stage, malicious clients may tamper with their similarity scores in an attempt to prevent detection of their compromised models. Since a benign client will honestly and accurately decrypt and select trustworthy clients group via threshold-based filter, and hence their results should be consistent. Therefore, $\mathcal{A}$ uses a majority voting strategy to acquire the final client score list, i.e., the voted $\boldsymbol{s}^{(t)}$ (⑥). Next, $\mathcal{A}$ normalizes $\boldsymbol{s}^{(t)}$ and generates the fusion weight (⑦). Here, *DDFed* employs *FedAvg*'s approach by weighting the aggregation according to dataset size proportions in current training round (⑧). Finally, each client $\mathcal{C}_i$ receives the aggregated global model $[\![\boldsymbol{W}_G^{(t)}]\!]$, decrypts it, and initiates the $(t+1)$-th round of *DDFed* training (⑨).

Private and Robust Malicious Model Detection. As observed in [38], the distribution of local data labels can be more effectively represented in the weights of the last layer than in other layers. Consequently, *DDFed* employs a similar approach to enhance the efficiency of detecting anomalies, as it requires performing similarity computation on encrypted model updates. Given that FHE supports only basic mathematical operations, and the similarity-based anomaly model detection mechanism needs complex operations like division (as shown in equation 2), comparison and sorting operations, *DDFed* breaks it down into two stages: *secure similarity computation* and *feedback-driven collaborative selection*. In the rest of the paper and during our experimental evaluation, we adhere to the layer section settings described in [38]. However, *DDFed* can be easily extended to support strategies for detecting malicious models using full layers. Additional experiments are detailed in Appendix A.2.4 to demonstrate the impact of layer sections on the *DDFed* framework.

Secure Similarity Computation. To circumvent division operations, *DDFed* necessitates that all clients pre-process their inputs for normalization and shifts the task of comparing similarity scores to the client side. This is because clients possess the FHE private key, allowing them to obtain the similarity score in plaintext. Formally, we have the following:

$$[\![cos(\alpha_i)]\!] = \frac{\langle [\![\boldsymbol{W}_i^{(t)}]\!], [\![\boldsymbol{W}_G^{(t-1)}]\!] \rangle}{\|[\![\boldsymbol{W}_i^{(t)}]\!]\|_2 \cdot \|[\![\boldsymbol{W}_G^{(t-1)}]\!]\|_2} = \langle [\![\frac{\boldsymbol{W}_i^{(t)}}{\|\boldsymbol{W}_i^{(t)}\|_2}]\!], [\![\frac{\boldsymbol{W}_G^{(t-1)}}{\|\boldsymbol{W}_G^{(t-1)}\|_2}]\!] \rangle, \tag{3}$$

where each client $\mathcal{C}_i$ prepares the $\frac{\boldsymbol{W}_i^{(t)}}{\|\boldsymbol{W}_i^{(t)}\|_2}$ and $\frac{\boldsymbol{W}_G^{(t-1)}}{\|\boldsymbol{W}_G^{(t-1)}\|_2}$ in advance, and then encrypts them using FHE encryption algorithm. Next, the aggregation server $\mathcal{S}$ verifies received $[\![\frac{\boldsymbol{W}_G^{(t-1)}}{\|\boldsymbol{W}_G^{(t-1)}\|_2}]\!]$ and perturbs local inputs and conducts secure inner-product computation as follows:

$$[\![\boldsymbol{s}^{'(t)}]\!] = \langle [\![\frac{\boldsymbol{W}_i^{(t)}}{\|\boldsymbol{W}_i^{(t)}\|_2}]\!] + \Delta^{(t)}, [\![\frac{\boldsymbol{W}_G^{(t-1)}}{\|\boldsymbol{W}_G^{(t-1)}\|_2}]\!] \rangle. \tag{4}$$

Motivation of Similarity Score Perturbation. *DDFed* aims to simultaneously address privacy and poisoning risks. This means it not only considers model poisoning attacks but also prevents adversarial clients from inferring private information from other benign clients by exploiting decrypted similarity scores and previous global models. To mitigate this privacy risk, *DDFed* improves secure inner-product computation by introducing perturbations into each normalized and encrypted model update. Specifically, *DDFed* uses (ε, δ)-differential privacy with a Gaussian mechanism as its method of perturbation, $\Delta^{(t)} = \mathcal{N}(0, \sigma^2), \sigma = \frac{\Delta f \sqrt{2\ln(1.25/\delta)}}{\varepsilon}$, where (ε, δ) represents the parameters of the DP mechanism and Δf denotes sensitivity.

It's important to note that our perturbation affects only the anomaly detection phase and does not change the encrypted model updates that are to be aggregated. Consequently, the final aggregated model retains its accuracy, just as it would with a standard aggregation mechanism. Furthermore, our experiments indicate that the perturbation noise does not affect the effectiveness of anomaly detection.

Even at $\varepsilon = 0.01$, which offers strong privacy protection, *DDFed* still performs well and delivers good model performance.

Feedback-driven Collaborative Selection. As shown in the threat model, *DDFed* tolerates less than 50% malicious clients, indicating that over half of the clients are benign and will execute the steps honestly and correctly as designed. *DDFed* employs a feedback-driven collaborative selection approach to filter out potentially malicious models. Specifically, upon receiving the encrypted $[\![\boldsymbol{s}'^{(t)}]\!]$, each client $\mathcal{C}_i$ first decrypts to acquire $\boldsymbol{s}_i'^{(t)}$ using the FHE private key $\textsc{sk}$. Next, each client $\mathcal{C}_i$ independently decrypts the similarity scores, sorts them, and selects trustworthy clients $\boldsymbol{s}_i^{(t)}$ for the current training round based on a threshold. *DDFed* uses only the mean value of similarity scores as its filtering threshold. Subsequent experiments have demonstrated its effectiveness. Additionally, *DDFed* is open and compatible with alternative methods for setting thresholds. After each client returns their decision on the group of benign clients ($\boldsymbol{s}_i^{(t)}$), the aggregation server uses a majority of vote strategy to decide the final aggregation group ($\boldsymbol{s}^{(t)}$) for the current training round. Next, similar to *FedAvg*, *DDFed* applies a data size-based fusion weight strategy to calculate each client's fusion weight $\boldsymbol{f}^{\boldsymbol{W}^{(t)}}_{\boldsymbol{s}^{(t)}}$ in the aggregation group, where $f_j^{(t)} = \frac{|D_j|}{\sum_{j \in \boldsymbol{s}^{(t)}} |D_j|}$.

FHE-based Secure Aggregation with Clipping. *DDFed*'s secure aggregation leverages the FHE cryptosystem, specifically the CKKS instance[8], which excels in arithmetic operations on encrypted real or complex numbers and stands as one of the most efficient methods for computing with encrypted data. Formally, the aggregation server performs secure aggregation as $[\![\boldsymbol{W}_G^{(t)}]\!] = \langle [\![\boldsymbol{W}^{(t)}]\!], \boldsymbol{f}^{\boldsymbol{W}^{(t)}}_{\boldsymbol{s}^{(t)}} \rangle$. Once receiving the aggregated global model $[\![\boldsymbol{W}_G^{(t)}]\!]$, each client $\mathcal{C}_i$ uses their private key to decrypt it, obtaining the final global model $\boldsymbol{W}_G^{(t)}$ in plaintext via the FHE decryption algorithm. In contrast to current approaches in private and robust FL, *DDFed* uniquely enables each benign client to execute a clipping operation before the next training round. This enhancement is designed to counteract more sophisticated model poisoning attacks that conventional similarity-based methods [29, 38] fail to address, as will be shown in the experiments section.

3.3 Analysis on Privacy and Robustness

Based on the threat model discussed earlier, *DDFed* prevents an honest-but-curious aggregation server from potentially inferring private information from accessible model updates. Additionally, it also withstands a subset of local clients, compromised by an adversary, to launch model poisoning attacks and attempt to infer private information from other benign clients during the anomaly model detection phase.

In terms of privacy risks, *DDFed* utilizes FHE primitives to ensure cryptographic-level privacy protection. This means the aggregation server processes each operation without any insight into the model update (in the dark), eliminating any chance of inferring private information from local model updates. Furthermore, to counter potential inferences by corrupted clients exploiting decrypted similarity scores, *DDFed* incorporates a perturbation method where DP noise is added during the secure similarity computation phase. Due to space limitations, the formal DP-enhanced perturbation analysis is provided solely in Appendix A.3.

Regarding the risk of poisoning attacks, *DDFed* adopts similarity-based anomaly detection technologies with additional optimizations such as perturbation-based similarity computation and post-aggregation clipping. These enhancements bolster the robustness of its aggregation mechanism. Our experiments demonstrate that *DDFed* effectively resists a range of continuous poisoning attacks, including IPM, SCALING, and ALIE attacks, which will be elaborated in Section 4.

4 Experiments

4.1 Experimental Setup

Datasets and Implementation. We assessed our proposed *DDFed* framework using publicly available benchmark datasets: MNIST[19], a collection of handwritten digits, and Fashion-MNIST (FMNIST)[33], which includes images of various clothing items, offering a more challenging and

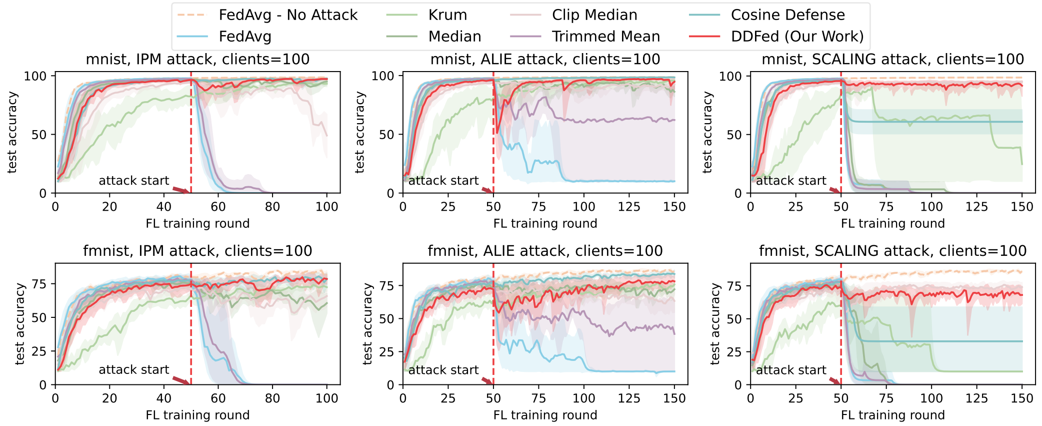

Figure 2: Comparison of defense effectiveness across various defense approaches, evaluated on MNIST (top) and FMNIST(bottom), under IPM attack (left), ALIE attack (middle), and SCALING attack (right).

diverse dataset for federated learning tasks. We create non-iid partitions for all datasets based on previous research [38, 43], using a default q value of 0.5, where a higher q reflects greater degrees of non-iid. We assess the framework's performance using a nine-layer CNN model with 225k parameters, secured by the FHE cryptosystem in each training round. This secure aggregation is implemented through TenSEAL library [4]. The experimental *DDFed* is available on the GitHub repository.

Baselines and Default Setting. We compare our proposed method ***DDFed*** with well-known FL fusion algorithms and robust aggregation methods, including ***Krum*** [5], ***Cos Defense*** [38], and median/mean-based approaches like ***Median***, ***Clipping Median***, and ***Trimmed Mean*** strategies[40]. We exclude baselines such as FLTrust[7] or RFFL[37] because they require server-side validation data or are incompatible with client sampling, making them impractical for real-world applications. Additionally, we omit secure robust approaches[9, 20, 23, 43] that depend on complex secure aggregation techniques due to their requirement for additional non-colluding participants, which alters the original structure of the federated learning framework. Note that the core contribution of this paper is not to propose new model poisoning defense approaches, but to enhance existing popular defenses with privacy features—specifically, server-side similarity-based defenses. Therefore, the experiments aim to evaluate how these privacy-preserving features affect the original defense methods, rather than defending against recent attack techniques and strategies as shown in works like [26, 31, 42, 10, 25].

To assess defense performance, we evaluated the proposed work against popular model poisoning attacks: Inner Product Manipulation (***IPM***) attack [34], ***scaling*** attack[1], and the "a little is enough" (***ALIE***) attack[3]. Unless otherwise mentioned, we assume a default attacker ratio of 0.3 among all participants as malicious clients. The attacks commence at the 50th round and persist until training ends. The default FL training involves 10 clients randomly chosen from 100 for each communication round. Furthermore, we employ a batch size of 64 with each client conducting local training over three epochs per round using an SGD optimizer with a momentum of 0.9 and a learning rate of 0.01. Our *DDFed* implementation's default epsilon (ε) value is set to 0.01 unless specified differently.

4.2 Performance Evaluation

Performance of Defense Effectiveness under Various Attacks. Figure 2 demonstrates the effectiveness of our *DDFed* method compared to baseline methods in countering three prevalent model poisoning attacks, with an attacker ratio set at 0.3. The attack commences at the 50th round and continues until training concludes. Under the IPM attack scenario, aside from FedAvg, Trimmed Mean, and Clipping Median mechanisms, our approach along with other defense strategies performs well (nearly as model accuracy as without any model poisoning attack) in defending against the IPM attack. The same conclusion also holds true in the ALIE attack. However, only *DDFed* and Clip Median successfully withstand SCALING attacks with minor and acceptable losses in model

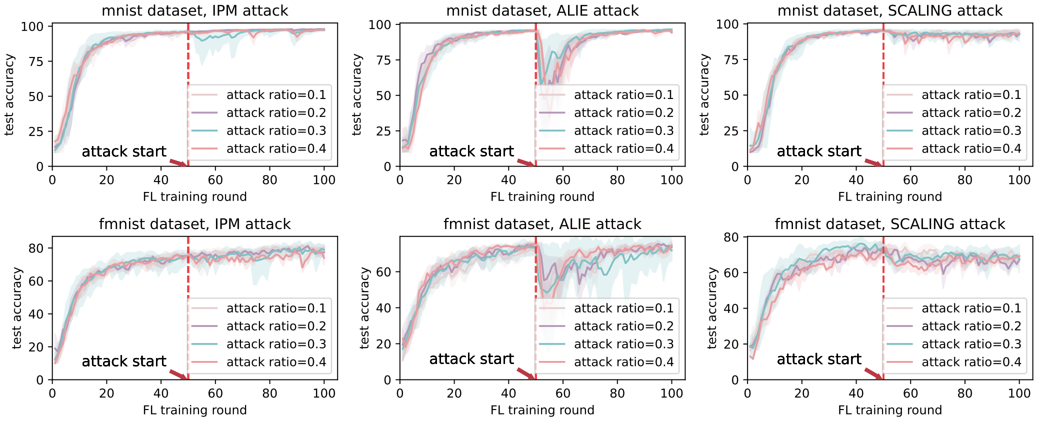

Figure 3: Comparison of *DDFed* effectiveness across different attack ratios, evaluated on MNIST (top) and FMNIST (bottom), under IPM attack (left), ALIE attack (middle), and SCALING attack (right).

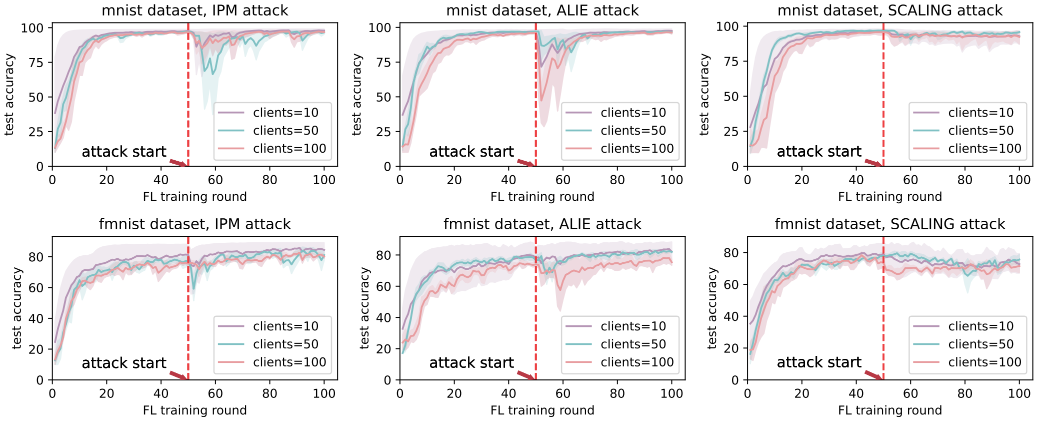

Figure 4: Comparison of *DDFed* effectiveness across different client numbers, evaluated on MNIST (top) and FMNIST (bottom), under IPM attack (left), ALIE attack (middle), and SCALING attack (right).

performance. Note that *DDFed* remains robust even when attackers target the system from the start of training. Due to space constraints, we present the defense effectiveness against various cold-start attacks in Appendix A.2.3. In summary, our *DDFed* method achieves the best comprehensive defense performance.

Impact of Attacker Ratio. To further investigate the impact of attacker ratio in the *DDFed* framework, we conducted experiments with various attacker ratio settings. It's important to note that *DDFed* operates under the security assumption that at least half of the participants must be benign (i.e., $r_{attacker} < 0.5$), therefore, in our experiments, the attacker ratio setting is ranged from 0.1 to 0.4. As shown in Figure 3, the proportion of attackers among all clients does not significantly affect our proposed *DDFed* method. This suggests that it can effectively counter three types of model poisoning attacks. Additionally, we observed that under an ALIE attack scenario, our method may require approximately 10-20 training rounds to recover from the continuous attack, depending on the dataset evaluated.

Compatibility with Cross-device and Cross-silo FL Scenarios. To explore how the number of clients affects our *DDFed* framework and to confirm its compatibility with two common federated learning scenarios, i.e., cross-device and cross-silo, we conducted multiple experiments. These experiments had an attacker ratio fixed at 0.3, with client counts varying from 10 to 100. In cross-silo FL, client numbers are typically small, often ranging from a few to several dozen; however, for simulating the cross-device FL scenario in our study, we used 100 clients due to their generally larger population. As illustrated in Figure 4, our *DDFed* framework effectively defends against all three attacks across various client number settings. This suggests that the performance of *DDFed* is not significantly affected by the number of clients, indicating its suitability for both cross-silo and cross-device FL scenarios. Furthermore, a higher number of client settings may result in relatively large fluctuations during training rounds immediately following the attack; however, the model training ultimately converges steadily, unaffected by the continuous attack.

Table 1: Time cost per training round of various defense approaches.

Approaches	MNIST, IPM attack		FMNIST, IPM attack	
	avg (s)	var (s)	avg (s)	var (s)
FedAvg	10.26	0.07	10.47	0.01
Krum	10.32	0.03	10.26	0.01
Median	10.32	0.01	10.28	0.02
Clipping Median	10.31	0.01	10.32	0.01
Trimmed Mean	10.32	0.02	10.30	0.01
Cos Defense	10.25	0.01	10.26	0.02
DDFed (Our Work)	12.43	0.01	12.14	0.01

Time Cost of Secure Aggregation. To assess the additional time cost incurred by integrating FHE-based secure similarity computation and secure aggregation into *DDFed*, we measured the time cost of each training round and compared it with the baseline methods mentioned earlier. All experiments were carried out using the default settings described above. Due to space constraints, we only present the defense approach's time cost per training round when under an IPM attack and have included further results in Appendix A.2.2.

As shown in Table 1, compared to other robust aggregation mechanisms that lack privacy-preserving features, our *DDFed* solution incurs additional time costs due to the integration of FHE-based secure similarity computation and secure aggregation. Across experiments on various datasets and under different attacks, our *DDFed* generally requires an extra 2 seconds compared to the usual 10-second training round, resulting in a 20% increase in time per training round. However, our *DDFed* is capable of defending against model poisoning attacks while also offering strong privacy guarantees. Note that the time-related experiments were conducted on a MacOS platform with an Apple M2 Max chip and 96GB of memory.

Impact of Epsilon Setting. To better understand the effect of the hyperparameter ε setting on *DDFed*'s perturbation-based secure similarity computation phase, we conducted several experiments with different ε settings, ranging from 0.01 to 0.1. Here, we only demonstrate the results from 10 clients here, with additional results in Appendix A.2.1.

As shown in Figure 5, the ε setting has a negligible impact on performance with the MNIST dataset. However, higher ε values, which indicate stronger DP protection, cause relatively larger fluctuations in performance on the FMNIST dataset. Therefore, we believe that the optimal ε setting depends on the specific task at hand and leave it as an open question for future research.

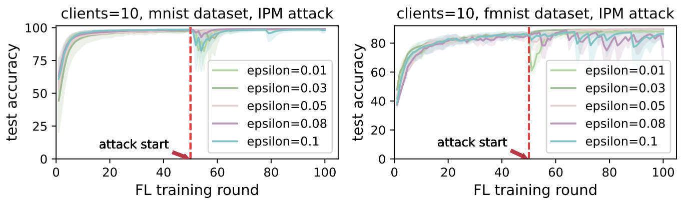

Figure 5: Impact of hyper-parameter ϵ of differential privacy based perturbation at secure similarity computation phase, evaluated on MNIST (left) and FMNIST (right), under IPM attack.

4.3 Discussion and Limitation

To the best of our knowledge, *DDFed* offers a dual defense strategy that simultaneously boosts privacy protection and fights against poisoning attacks in FL, without altering the existing FL framework's architecture. *DDFed* utilizes FHE for top-notch privacy, enabling the aggregation server to perform similarity calculations and aggregation without directly accessing model updates. Additionally, *DDFed* introduces perturbation techniques to block attempts by malicious clients to infer information from similarity scores. It further employs similarity-based anomaly detection, enhanced with strategies like perturbation and post-aggregation clipping, to protect against various types of poisoning attacks. However, *DDFed* has not fully explored two related questions: how can we relax the attacker ratio restriction (i.e., $r_{\text{ATTACK}} < 0.5$) while still ensuring effective dual defense? And how can we adapt *DDFed* to more complex FL scenarios, such as dropout and dynamic participant groups? We leave these questions open for future research. Currently, *DDFed* only enhances existing popular defenses, such as similarity-based strategies with privacy features. Extending *DDFed* to support other or more recent defense strategies remains an open question.

5 Conclusion

To tackle the dual challenges of privacy risks and model poisoning in federated learning, we introduce *DDFed*, a comprehensive approach that strengthens privacy protections and counters model poisoning attacks. *DDFed* enhances privacy by using an FHE-based secure aggregation mechanism and addresses encrypted poisoned model detection through an innovative secure similarity-based anomaly filtering method. This method includes secure similarity computation with perturbation and feedback-driven selection process to distinguish safe model updates from potentially harmful ones. Our approach has been rigorously tested against well-known attacks on diverse datasets, demonstrating its effectiveness. We believe our work sets a solid foundation for future advancements in secure and robust federated learning.

Acknowledgments and Disclosure of Funding

This work is funded by the National Natural Science Foundation of China, under grants No.62302022, No.62225202, No.62202038. We sincerely thank the anonymous reviewers for their insightful comments and constructive feedback, which have greatly improved this paper. Their suggestions were invaluable in refining our analysis and presentation, as well as guiding future research questions related to this work.

References

[1] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*, pages 2938–2948. PMLR, 2020.

[2] N. Baracaldo and R. Xu. Protecting against data leakage in federated learning: What approach should you choose? In *Federated Learning: A Comprehensive Overview of Methods and Applications*, pages 281–312. Springer, 2022.

[3] G. Baruch, M. Baruch, and Y. Goldberg. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption, 2021.

[5] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

[6] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[7] X. Cao, M. Fang, J. Liu, and N. Z. Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2021.

[8] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

[9] C. Dong, J. Weng, M. Li, J.-N. Liu, Z. Liu, Y. Cheng, and S. Yu. Privacy-preserving and byzantine-robust federated learning. *IEEE Transactions on Dependable and Secure Computing*, 2023.

[10] M. Fang, X. Cao, J. Jia, and N. Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[11] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

[12] H. Guo, H. Wang, T. Song, Y. H. R. Ma, X. Jin, Z. Xue, and H. Guan. Siren+: Robust federated learning with proactive alarming and differential privacy. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[13] M. T. Hossain, S. Islam, S. Badsha, and H. Shen. Desmp: Differential privacy-exploited stealthy model poisoning attacks in federated learning. In *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, pages 167–174. IEEE, 2021.

[14] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in Neural Information Processing Systems*, 34:7232–7241, 2021.

[15] Y. Huang, G. Yang, H. Zhou, H. Dai, D. Yuan, and S. Yu. Vppfl: A verifiable privacy-preserving federated learning scheme against poisoning attacks. *Computers & Security*, 136:103562, 2024.

[16] Y. Jiang, Y. Li, Y. Zhou, and X. Zheng. Mitigating sybil attacks on differential privacy based federated learning. *arXiv preprint arXiv:2010.10572*, 2020.

[17] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

[18] J. Konecnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 8, 2016.

[19] Y. LeCun, C. Cortes, and C. J. Burges. Mnist handwritten digit database. `http://yann.lecun.com/exdb/mnist`, 2010.

[20] X. Li, X. Yang, Z. Zhou, and R. Lu. Efficiently achieving privacy preservation and poisoning attack resistance in federated learning. *IEEE Transactions on Information Forensics and Security*, 2024.

[21] C. Liu, S. Chakraborty, and D. Verma. Secure model fusion for distributed learning using partial homomorphic encryption. In *Policy-Based Autonomic Data Governance*, pages 154–179. Springer, 2019.

[22] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu. Privacy-enhanced federated learning against poisoning adversaries. *IEEE Transactions on Information Forensics and Security*, 16:4574–4588, 2021.

[23] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng. Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning. *IEEE Transactions on Information Forensics and Security*, 17:1639–1654, 2022.

[24] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[25] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, et al. {FLAME}: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1415–1432, 2022.

[26] K. Pillutla, S. M. Kakade, and Z. Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

[27] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE S&P'17*, pages 3–18. IEEE, 2017.

[28] J. Sun, A. Li, L. DiValentin, A. Hassanzadeh, Y. Chen, and H. Li. Fl-wbc: Enhancing robustness against model poisoning attacks in federated learning from a client perspective. *Advances in Neural Information Processing Systems*, 34:12613–12624, 2021.

[29] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[30] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou. A hybrid approach to privacy-preserving federated learning. In *ACM AISec'19*, pages 1–11, 2019.

[31] S. Wang, J. Hayase, G. Fanti, and S. Oh. Towards a defense against federated backdoor attacks under continuous training. *Transactions on Machine Learning Research*, 2022.

[32] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469, 2020.

[33] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[34] C. Xie, O. Koyejo, and I. Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence*, pages 261–270. PMLR, 2020.

[35] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, S. Kadhe, and H. Ludwig. Detrust-fl: Privacy-preserving federated learning in decentralized trust setting. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, pages 417–426. IEEE, 2022.

[36] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *ACM AISec'19*, pages 13–23, 2019.

[37] X. Xu and L. Lyu. A reputation mechanism is all you need: Collaborative fairness and adversarial robustness in federated learning. *arXiv preprint arXiv:2011.10464*, 2020.

[38] D. N. Yaldiz, T. Zhang, and S. Avestimehr. Secure federated learning against model poisoning attacks via client filtering. In *ICLR 2023 Workshop on Backdoor Attacks and Defenses in Machine Learning*, 2023.

[39] M. Yang, H. Cheng, F. Chen, X. Liu, M. Wang, and X. Li. Model poisoning attack in differential privacy-based federated learning. *Information Sciences*, 630:158–172, 2023.

[40] D. Yin, Y. Chen, R. Kannan, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. Pmlr, 2018.

[41] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *USENIX ATC'20)*, pages 493–506, 2020.

[42] K. Zhang, G. Tao, Q. Xu, S. Cheng, S. An, Y. Liu, S. Feng, G. Shen, P.-Y. Chen, S. Ma, et al. Flip: A provable defense framework for backdoor mitigation in federated learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[43] Z. Zhang, J. Li, S. Yu, and C. Makaya. Safelearning: Secure aggregation in federated learning with backdoor detectability. *IEEE Transactions on Information Forensics and Security*, 2023.

A Appendix

A.1 DDFed Algorithm

Algorithm 1: *DDFed* Training

```
Input: clients $\{\mathcal{C}_1, ..., \mathcal{C}_m\}$, each client $\mathcal{C}_i$ has its own dataset $D_i$; global training round $T$
Output: final global model $\boldsymbol{W}_G^{(T)}$.
Each client initializes with FHE key pair $(\mathrm{PK}, \mathrm{SK})$;
aggregation server $\mathcal{A}$ initializes the global model $\boldsymbol{W}_G^{(0)}$;
foreach training round $t \in \{1, ..., T\}$ do
    foreach client $\mathcal{C}_i \in \{\mathcal{C}_1, ..., \mathcal{C}_m\}$ do
        if not initial training round then
            $\mathcal{C}_i$ receives $[\![\boldsymbol{W}_G^{(t-1)}]\!]$ from $\mathcal{A}$ and acquires $\boldsymbol{W}_G^{(t-1)} \leftarrow \mathrm{FHE.D{\scriptsize EC}_{SK}}([\![\boldsymbol{W}_G^{(t-1)}]\!])$;
        if reaching final training round $T$ then
            return final model $\mathcal{M}_G^{(m)}$;
        if $\mathcal{C}_i$ is benign then $\mathcal{C}_i$ performs clipping on $\boldsymbol{W}_G^{(t-1)}$;
        $\mathcal{C}_i$ conducts local training $\boldsymbol{W}_i^{(t)} \leftarrow \mathrm{T{\scriptsize RAIN}}(\boldsymbol{W}_G^{(t-1)})$;
        $\mathcal{C}_i$ encrypts local model $[\![\boldsymbol{W}_i^{(t)}, \boldsymbol{W}_i^{L,(t)}]\!] \leftarrow \mathrm{FHE.E{\scriptsize NC}_{PK}}([\![\boldsymbol{W}_G^{(t-1)}]\!])$;
        $\mathcal{C}_i$ sends out $[\![\boldsymbol{W}_i^{(t)}, \boldsymbol{W}_i^{L,(t)}]\!]$ to $\mathcal{A}$;
    $\mathcal{A}$ waits and collects $\{[\![\boldsymbol{W}_i^{(t)}, \boldsymbol{W}_i^{L,(t)}]\!]\}_{i \in [1,...,m]}$;
    $\mathcal{A}$ retrieves $[\![\boldsymbol{W}_G^{L,(t-1)}]\!]$ from previous round and prepares perturbation $\Delta^{(t)}$;
    $\mathcal{A}$ performs $[\![\boldsymbol{s}'^{(t)}]\!] \leftarrow \{\langle [\![\boldsymbol{W}_i^{L,(t)}]\!] + \Delta^{(t)}, [\![\boldsymbol{W}_G^{L,(t-1)}]\!] \rangle\}_{i \in [1,...,m]}$ and sends $[\![\boldsymbol{s}'^{(t)}]\!]$ to $\{\mathcal{C}_1, ..., \mathcal{C}_m\}$;
    foreach client $\mathcal{C}_i \in \{\mathcal{C}_1, ..., \mathcal{C}_m\}$ do
        $\mathcal{C}_i$ decrypts $\boldsymbol{s}_i'^{(t)} \leftarrow \mathrm{FHE.D{\scriptsize EC}_{SK}}([\![\boldsymbol{s}'^{(t)}]\!])$, conducts threshold-based selection and sends back $\boldsymbol{s}_i^{(t)}$;
    $\mathcal{A}$ collects $\{\boldsymbol{s}_i^{(t)}\}_{i \int [1,...,m]}$ and selects $\boldsymbol{s}^{(t)}$ via majority of votes strategy;
    $\mathcal{A}$ generates fusion weights $\boldsymbol{f}_{\boldsymbol{s}^{(t)}}^{w}$ using $\boldsymbol{s}^{(t)}$;
    $\mathcal{A}$ conducts secure aggregation $[\![\boldsymbol{W}_G^{(t)}]\!] \leftarrow \langle [\![\boldsymbol{W}^{(t)}]\!], \boldsymbol{f}_{\boldsymbol{s}^{(t)}}^{\boldsymbol{W}^{(t)}} \rangle$ and sends $[\![\boldsymbol{W}_G^{(t)}]\!]$ to $\{\mathcal{C}_1, ..., \mathcal{C}_m\}$;
```

The *DDFed* algorithm is outlined in Algorithm 1. Assuming, without loss of generality, that at training round t, each client receives the aggregated and encrypted global model from the previous round. Upon decrypting this global model, benign clients clip it before conducting local training. They then encrypt their local model updates after applying a normalization preprocessing step to aid in detecting similarity-based poisoning attacks.

Once all encrypted model updates are collected from the clients, the aggregation server begins secure similarity computations using the abstracted last layer of these updates. It introduces differential privacy by adding perturbation noise and sends them back to each client for collaborative decryption and selection of benign clients.

Following this, based on majority votes, the aggregation server determines final aggregation groups and calculates fusion weights. Finally, it securely aggregates these with the fusion weights to produce an encrypted global model and concludes that round of federated learning (FL) training.

A.2 Additional Experimental Results

A.2.1 Impact of Epsilon with 100 Clients

Figure 6 presents the experimental findings on how different ε values affect perturbations during the secure similarity computation phase, with experiments focusing on an IPM attack scenario and involving 100 clients. These tests were carried out using the MNIST and Fashion-MNIST (FMNIST) datasets. For both datasets, we explored a range of epsilon values from 0.01 to 0.1, noting that lower epsilon values indicate enhanced privacy through increased noise addition. Initially, all configurations demonstrated high accuracy levels; however, performance fluctuations became evident following the attack. Specifically, the MNIST dataset exhibited a notable decrease in accuracy at certain epsilon settings, while the FMNIST dataset showed more moderate variations in performance. Ultimately, both datasets achieved relatively stable model accuracy. Determining the optimal ε setting is task-dependent and remains an area for future investigation.

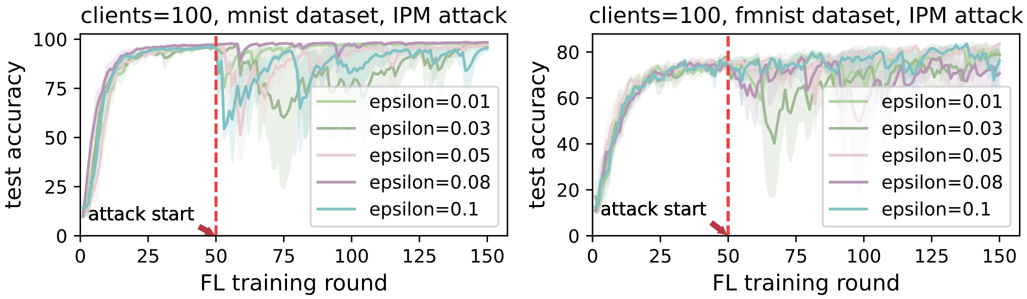

Figure 6: Impact of hyper-parameter ϵ of differential privacy based perturbation at secure similarity computation phase with client number 100, evaluated on MNIST (left) and FMNIST (right), under IPM attack.

Table 2: Time cost per training round of various defense approaches on MNIST and FMNIST datasets under SCALING attack

Approaches	MNIST, SCALING attack		FMNIST, SCALING attack	
	avg (s)	var (s)	avg (s)	var (s)
FedAvg	9.87	0.21	10.51	0.01
Krum	9.75	0.21	10.42	0.01
Median	9.81	0.15	10.29	0.02
Clipping Median	9.73	0.11	10.19	0.01
Trimmed Mean	9.76	0.22	10.24	0.01
Cos Defense	9.43	0.08	10.30	0.01
DDFed (Our Work)	12.15	0.03	12.25	0.18

A.2.2 Time Cost of Secure Aggregation on Scaling and ALIE attacks.

Table 2 and Table 3 report additional results on the time cost of each training round taken for various defense strategies against SCALING and ALIE attacks on the MNIST and FMNIST datasets, respectively. Consistent with the findings presented in Section 4.2, our *DDFed* approach adds only 2 seconds to the usual 10-second training round across multiple experiments, datasets, and attack scenarios, resulting in a 20% increase in time per round. Despite this slight increase, *DDFed* successfully defends against model poisoning attacks while ensuring robust privacy protection.

A.2.3 Performance of DDFed Against Cold-Start Model Poisoning Attacks.

The primary purpose that we initiated the attack at round 50 is to demonstrate the effectiveness of defense mechanisms and clearly show the comparative effects of different defense methods before and after an attack. This setup can also illustrate how various defensive measures impact training convergence and model quality, even without attacks.

DDFed is resilient to poisoning attacks from the beginning of training. Our design is not constrained by the attack's initiation round. Supplementary experimental results as reported in Table 4 on the FMNIST dataset with 100 clients in a non-iid setting support this claim.

A.2.4 Impact of Selected Layer Count on Poisoning Model Detection in DDFed.

In the main body of the paper, we use only the last layer for similarity computation because our primary goal is to integrate privacy-preserving functionality into existing poisoning defense strategies rather than optimizing these mechanisms. Our exploration shows that similarity-based methods and their variants provide comprehensive defense effectiveness, robust against various threat scenarios such as server reliance on validation data, types of model poisoning attacks, and the number of compromised clients. Therefore, we selected a typical similarity-based defense strategy (Cosine Defense) as a starting point to enhance privacy-preserving features. Our approach can easily extend to other similarity-based detection variants using full layers for secure similarity computation. As

Table 3: Time cost per training round of various defense approaches on MNIST and FMNIST datasets under ALE attack

Approaches	MNIST, ALE attack		FMNIST, ALE attack	
	avg (s)	var (s)	avg (s)	var (s)
FedAvg	10.31	0.11	10.38	0.16
Krum	10.19	0.10	10.06	0.08
Median	10.19	0.05	10.15	0.04
Clipping Median	9.98	0.09	10.05	0.12
Trimmed Mean	9.97	0.11	10.06	0.06
Cos Defense	9.78	0.17	10.14	0.09
DDFed (Our Work)	12.23	0.07	11.95	0.08

Table 4: Performance of DDFed Against Cold-Start attacks on FMNIST datasets.

Approaches	IPM Attacks	ALIE Attacks	SCALINE Attacks
FedAvg	0	10.1	0
Krum	69.05	73.69	69.95
Median	67.57	76.57	74.03
Clipping Median	61.1	73.8	75.49
Trimmed Mean	0	43.29	0
Cos Defense	81.87	82.97	81.11
DDFed (Our Work)	83.32	80.97	83.05

shown in Table 5, we conducted additional experiments with full-layer secure similarity computation on a larger dataset (CIFAR10) under various attacks.

Table 5: Comparison of Model Performance and Time Cost Across Different Layer Protection Settings on Evaluating the CIFAR10 Dataset with Setting of 60 Training Rounds.

Approaches	No attack		IPM attack		ALIE attack		SCALINE attack	
	Acc	Time(m)	Acc	Time(m)	Acc	Time(m)	Acc	Time(m)
FedAvg	70.16	46.23	0	46.62	10	46.89	0	46.48
DDFed (Last Layer)	-	-	70.3	50.66	64.62	51.3	69.61	51.63
DDFed (Full Layers)	-	-	69.84	58.95	69.73	58.78	68.89	59.01

A.3 Differential Privacy

A.3.1 Differential Privacy

Differential privacy is a mathematical framework designed to provide privacy guarantees for individuals in a dataset. The standard definition of differential privacy is as follows:

A randomized algorithm $\mathcal{M}$ is said to be (ε, δ)-differentially private if, for any two adjacent datasets D and D' (i.e., datasets differing by only one element), and for any subset of outputs $S \subseteq \text{Range}(\mathcal{M})$, the following inequality holds:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta \tag{5}$$

where ε is the privacy budget parameter, which controls the trade-off between privacy and utility. A smaller ε indicates stronger privacy. δ (delta) is a small probability that accounts for the possibility of the privacy guarantee being violated.

The Gaussian mechanism is a specific method to achieve differential privacy by adding Gaussian noise to the output of a function. The definition of the Gaussian mechanism is as follows:

Given a function f and any two adjacent datasets D and D', the sensitivity of f is defined as:

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\|_2 \tag{6}$$

The Gaussian mechanism adds noise drawn from a Gaussian distribution with mean 0 and standard deviation σ, where σ is determined by:

$$\sigma = \frac{\Delta f \sqrt{2 \ln(1.25/\delta)}}{\varepsilon} \tag{7}$$

Thus, the Gaussian mechanism is defined as:

$$\mathcal{M}(D) = f(D) + \mathcal{N}(0, \sigma^2) \tag{8}$$

where $\mathcal{N}(0, \sigma^2)$ denotes a Gaussian distribution with mean 0 and variance σ^2. By adding Gaussian noise in this manner, the Gaussian mechanism ensures that the output satisfies (ε, δ)-differential privacy.

A.3.2 Privacy Analysis of Differentially Private Similarity Computation in *DDFed*

The *DDFed* framework aims to enhance privacy protection and mitigate poisoning attacks within federated learning systems by integrating FHE and a similarity-based anomaly detection system. To further bolster privacy, *DDFed* incorporates DP during the similarity score computation process. This section provides a theoretical analysis of the differential privacy levels maintained by each participant in the *DDFed* framework, specifically focusing on clients during similarity score computation and feedback stages, as well as the aggregation server during model aggregation and similarity score processing.

In the similarity score computation phase, each client normalizes its local model updates before submitting them. To ensure DP, Gaussian noise is added to these normalized updates. By adding Gaussian noise, each client's similarity score computation adheres to (ε, δ)-differential privacy, ensuring that the privacy of the client's data is preserved even in the presence of adversaries.

During the feedback phase, clients decrypt the similarity scores and submit their results. Since these scores have already been DP due to the added Gaussian noise, the privacy level remains at (ε, δ)-differential privacy. This ensures that even when clients provide feedback, their privacy is not compromised.

In the model aggregation phase, the aggregation server receives encrypted model updates from clients. While FHE inherently provides a high level of security for these crucial parameters, the aggregation server further ensures privacy by applying DP during the similarity score calculation. The server aggregates the encrypted updates without accessing the plaintext data, thereby maintaining the privacy of the individual model updates.

For the similarity score processing phase, the aggregation server handles the scores submitted by clients, which have already been protected using differential privacy. Consequently, the server does not need to apply additional privacy mechanisms during this phase. The DP guarantees provided during the similarity score computation phase by clients are sufficient to protect the overall process.

Based on the analysis, the privacy levels for each client in *DDFed* framework can be summarized as follows. During the similarity score computation phase, clients achieve (ε, δ)-differential privacy by adding Gaussian noise to their normalized model updates. During the feedback phase, clients maintain (ε, δ)-differential privacy as the similarity scores they submit have already been differential private.

By thoughtfully designing and selecting parameters, the *DDFed* framework can provide robust privacy protection and maintain high model performance. The use of FHE for critical parameters and differential privacy for similarity scores ensures a balanced and comprehensive approach to privacy protection, addressing both security and utility needs effectively.

A.3.3 Impact of DP on FHE-based Similarity Computation in *DDFed*

Generally, the reader may concern about whether $[[x]] + \Delta$ equals $x + \Delta$, where x is under FHE protection. However, this depends on the precision of the employed FHE schemes. Proving such a statement theoretically may require delving into the specific construction algorithm of the FHE scheme, which is beyond the scope of machine learning-oriented venues.

Table 6: Impact of DP on FHE-based Similarity Detection in DDFed on evaluating CIFAR10 datasets.

Approaches	IPM Attacks	ALIE Attacks	SCALINE Attacks
DDFed (Simulated)	70.21	64.3	69.82
DDFed (Our Work)	70.31	64.62	69.6

This paper utilizes CKKS constructions, which natively support high-precision secure computation on floating-point numbers. As a result, adding DP noise to encrypted similarity results does not degrade performance. To validate this, we conducted supplementary experiments on CIFAR10 using a simulated DDFed setup where DP noise was added to non-encrypted parameters. The reported results in Table 6 support this claim.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims presented in the abstract and introduction are consistent with the contributions and scope detailed in the paper.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Sec 4.3

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

Justification: Sec 3.1 and Appendix A.3

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: Sec 4 and Appendix A.2

   Guidelines:

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

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

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

Answer: [Yes]

Justification: https://github.com/irxyzzz/DualDefense

Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: Sec 4 and Appendix A.2

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: Sec 4.2

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: Sec 4

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: This paper have state which version of the asset is used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

Answer: [NA]

Justification: Paper does not release new assets.

Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification:

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.