HYDRA-FL: Hybrid Knowledge Distillation for Robust and Accurate Federated Learning

Momin Ahmad Khan
Umass Amherst
makhan@umass.edu

Yasra Chandio
Umass Amherst
ychandio@umass.edu

Fatima Muhammad Anwar
Umass Amherst
fanwar@umass.edu

Abstract

Data heterogeneity among Federated Learning (FL) users poses a significant challenge, resulting in reduced global model performance. The community has designed various techniques to tackle this issue, among which Knowledge Distillation (KD)-based techniques are common. While these techniques effectively improve performance under high heterogeneity, they inadvertently cause higher accuracy degradation under model poisoning attacks (known as *attack amplification*). This paper presents a case study to reveal this critical vulnerability in KD-based FL systems. We show why KD causes this issue through empirical evidence and use it as motivation to design a hybrid distillation technique. We introduce a novel algorithm, *Hybrid Knowledge Distillation for Robust and Accurate FL (HYDRA-FL)*, [1], which reduces the impact of attacks in attack scenarios by offloading some of the KD loss to a shallow layer via an auxiliary classifier. We model HYDRA-FL as a generic framework and adapt it to two KD-based FL algorithms, FedNTD and MOON. Using these two as case studies, we demonstrate that our technique outperforms baselines in attack settings while maintaining comparable performance in benign settings.

1 Introduction

Federated Learning (FL) [32] is an emerging machine learning paradigm enabling multiple users' collaborative model training without data sharing. Each user, termed a *client*, only shares their local model with a *server*, which aggregates all local models into a single global model and redistributes it to the clients. Due to its decentralized, privacy-preserving, and highly-scalable nature, FL has been adopted by Google's Gboard [2] for next-word prediction, Apple's Siri [1] for automatic speech recognition, and WeBank [43] for credit risk prediction.

Despite its benefits, FL faces challenges with data heterogeneity [28, 51, 13, 24]. FL performs well when client data is independent and identically distributed (IID) and achieves similar convergence as a single model trained on all the clients' data but struggles when clients have diverse data (non-IID). In this case, the client's local data is not a good representation of the overall data distribution (unlike an ideal IID case), causing local models to *drift* away from each other. This drift results in a global model with significant accuracy degradation compared to the IID scenario. Numerous solutions [25, 20, 22, 53, 23, 46, 15, 27] address data heterogeneity, including Knowledge Distillation (KD) [12] to reduce the drift between local models.

[1] We will release the open source code with the final version of this paper.

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

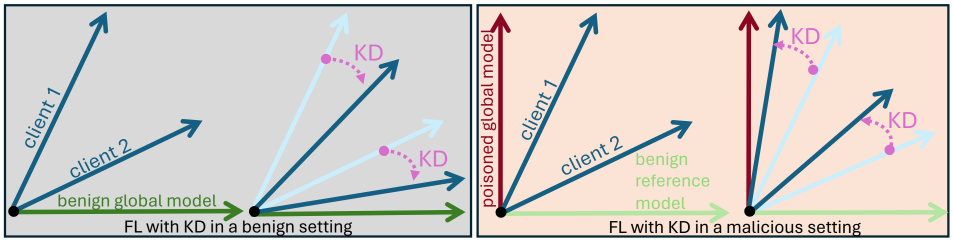


Figure 1: Overview of attack amplification through knowledge distillation. **a)** In the benign setting, KD reduces drift and brings benign local models closer to the benign global model. **b)** In the malicious setting, KD *unknowingly* reduces drift between benign local models and the poisoned global model.

Besides data heterogeneity, FL also faces the issue of Byzantine robustness [14], where *untrusted clients* can inject *poisoned* models into the aggregator by altering client data (data poisoning [35]) or client models (model poisoning [11, 4, 33, 45, 5, 3, 41]). Research by [40] shows that model poisoning attacks are more potent as they directly manipulate local models. To counteract poisoning in FL, various defenses have been developed [6, 47, 50, 7, 26, 9, 8].

In this work, we identify a critical vulnerability in KD-based FL techniques under model poisoning attacks. These techniques unknowingly align benign client models with a poisoned server model (Figure 1). We study two such classes of KD-based solutions: FedNTD [20], which reduces the loss between not-true logits of the server and client models, and MOON [25], which reduces the contrastive loss between the representation vector of the server and client models. While these techniques improve global model accuracy in benign settings compared to FedAvg [32] (standard FL aggregator), they *reduce performance below FedAvg under attack*, a phenomenon we term *attack amplification*, especially noticeable at higher heterogeneity levels.

Motivated by our findings, we propose a Hybrid Knowledge Distillation for Robust and Accurate FL (HYDRA-FL) framework for KD-based techniques that restricts attack amplification under poisoning attacks while retaining performance in the benign setting. Unlike traditional KD methods that apply KD-loss only at the final layer, HYDRA-FL introduces KD-loss at a *shallow layer* via an auxiliary classifier and reduces the KD-loss impact at the final layer. This approach draws inspiration from Self-Distillation (SD) [49] and Skeptical Students (SS) [18], but with a distinct focus on enhancing robustness against heterogeneity and model poisoning attacks in FL. SD improves model accuracy by self-distillation, while SS distills from "nasty teachers" [30] to shallow layers. In contrast, our approach uses auxiliary classifiers to enhance FL client robustness against heterogeneity and model poisoning attacks. We design a generic loss function adaptable to specific KD-based algorithms. Extensive experiments show that HYDRA-FL significantly boosts accuracy over FedNTD and MOON in attack settings while maintaining performance in benign settings.

Contributions. This work addresses the critical issue of attack amplification in KD-based FL techniques to counter data heterogeneity. In doing so we make the following contributions:

- **Proving KD amplifies model poisoning:** our motivational case study (§3) on two KD-based techniques, FedNTD and MOON, shows that KD improves accuracy in benign settings but helps the malicious clients propagate poisoning through the KD-loss in adversarial settings. We empirically and theoretically show that this attack amplification issue is inherent to any technique aligning client outputs/representations with the server.

- **Designing HYDRA-FL:** Using our observations as a guideline, we design HYDRA-FL (§4) to prevent attack amplification while retaining performance in the benign setting. HYDRA-FL is formulated as a general loss function adaptable to any FL algorithm to use as its local model training objective.

- **Implementation and Evaluation:** we adapt HYDRA-FL to FedNTD and MOON and modify their local training objectives (§5). Our qualitative and quantitative analysis (§6) shows HYDRA-FL achieves higher accuracy in attack settings and maintains accuracy in benign settings.

2 Background and Related Work

2.1 Federated Learning (FL)

In FL [14, 32], a service provider, called *server*, trains a *global model*, θ^g, on the private data from multiple collaborating clients, all without directly collecting their data. The server selects n out of total N clients in every FL round and shares the most recent global model (θ_g^t) with them, where t is the round number. Then, a client k uses their local data D_k to compute an update ∇_k^t and shares it with the server. The server aggregates these updates using some *aggregation rule*, like FedAvg [32] algorithm. In *FedAvg*, a client k *fine-tunes* θ_g^t on their local data using stochastic gradient descent (SGD) for a fixed number of local epochs E, resulting in an updated local model θ_k^t. The client then computes their update as the difference $\nabla_k^t = \theta_k^t - \theta_g^t$ and shares ∇_k^t with the server. Next, the server computes an aggregate of client updates, f_{agg} using mean, i.e.,

$$\nabla_{\mathsf{agg}}^t = f_{\mathsf{mean}}(\nabla_{\{k \in [n]\}}^t). \tag{1}$$

The server then updates the global model of the $(t+1)^{th}$ round using SGD and server learning η as:

$$\theta_g^{t+1} \leftarrow \theta_g^t + \eta \nabla_{\mathsf{agg}}^t \tag{2}$$

2.1.1 Data Heterogeneity in FL

Data heterogeneity is a well-explored problem [28, 51, 13, 24] in FL. Each client in FL generates its data, leading to local data distributions that vary across clients and do not accurately represent the global data distribution. By extension, a global model learned by aggregating local models using FedAvg may not be the best representation of all the client's local data. Studies have shown that this data heterogeneity degrades performance and have proposed various methods to address this issue [25, 20, 22, 53, 23, 46, 15, 27]. This degradation is more prominent in the presence of poisoning attacks. Research on poisoning attacks in FL has demonstrated that such attacks become more successful under high heterogeneity [11, 41]. This increased risk is because the malicious clients can more easily hide between drifted benign client models, making it difficult for the server to differentiate between heterogeneous benign clients and malicious ones. [16] highlights that overlooking this heterogeneity is a critical oversight in FL defense evaluations.

2.1.2 Poisoning in FL

FL is vulnerable to poisoning attacks [6, 4, 5, 3, 33, 11, 31, 45, 35, 41], where malicious clients aim to compromise the training process by degrading the global model's performance. These attacks come in various forms: In *data poisoning* [3], malicious clients poison their local data to introduce a backdoor in the local model. This backdoor then propagates to the global model upon aggregation. In *model poisoning* [11, 4, 33, 45, 5, 3, 41], malicious clients perturb their local models so that, when aggregated, the global model is poisoned. Poisoning attacks can be further classified based on their targets: If the performance degradation is on specific inputs, the attack is termed as *targeted poisoning* [5, 3], and if it is on all inputs, then it is termed as *untargeted poisoning* [11, 4, 33, 31, 45]. We explain the attacks used in this paper in §C.2.

2.2 Knowledge Distillation (KD)

Knowledge Distillation (KD) [12] transfers knowledge from a large, complex model (*teacher*) to a smaller, more computationally efficient model (*student*). This process involves distilling the teacher's rich and intricate information into the student by aligning their predictions. Formally, if the teacher and student models produce the output probabilities y_t^i and y_s^i respectively for the i^{th} input (x^i, y^i), KD aims to match these probabilities by applying the Kullback-Leibler (KL) divergence between them. The KL-divergence between their softened probabilities is given by: $KL(softmax(y_t^i/\tau)||softmax(y_s^i/\tau)$, where τ is the temperature parameter that softens the probabilities. The overall KD loss function combines this KL-divergence with the usual loss function such as cross-entropy (CE) loss with β (balances the importance of the KL-divergence and CE loss) as:

$$\mathcal{L} = (1-\beta) \cdot \mathcal{L}_{CE}(y_s^i, y^i) + \beta \cdot \mathcal{L}_{KL}(softmax(y_s^i/\tau)||softmax(y_t^i/\tau)) \tag{3}$$

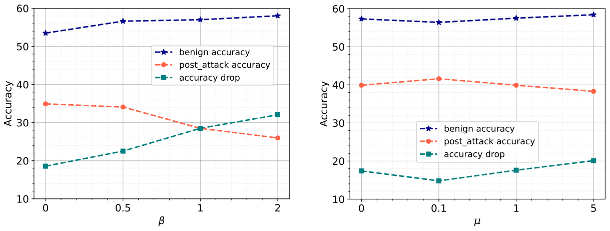
KD in FL is becoming essential as it addresses critical challenges such as non-IID data distributions, enhances model performance, accelerates convergence, reduces communication overhead, and improves robustness by making the global model learn from an ensemble of local models [10, 22, 29, 52]. In FL, data is often non-IID across clients, leading to significant discrepancies in local models. KD mitigates these discrepancies by aligning the local models with the global model, ensuring that the global model captures a more generalized representation of the data. The general approach is to reduce the local model drift by improving the aggregation through distillation using unlabeled auxiliary data. However, the auxiliary data may not always be available, and methods have also been developed to enable KD without such data [48, 53].

3 Attack Amplification through Knowledge Distillation

Hypothesis. KD-based techniques in FL improve accuracy in non-adversarial settings but result in more significant accuracy degradation under model poisoning attacks compared to the baseline techniques such as FedAvg.

Motivational case study. In this case study, we compare FedAvg against two distinct KD-based solutions addressing the local model drift from non-IID. MOON [25] uses model-contrastive learning to align local and global model *representations*, while FedNTD [20] uses KL-divergence to align *not-true logits* of client models with those of the server. FedNTD penalizes prediction divergence measured through distillation loss, improving knowledge transfer and stability, while MOON penalizes *representation divergence* measured through contrastive loss, enhancing robustness and generalization. This comparison will help us understand the trade-offs of using KD in FL, especially under adversarial conditions. Throughout this paper, benign conditions mean that no attacks are present, while adversarial conditions mean that model poisoning attacks are present. We implement the same settings and hyperparameters for FedAvg as for MOON and FedNTD to ensure a fair comparison, so FedAvg results may vary between these techniques. This is not an inconsistency. *We do not directly compare FedNTD to MOON unless stated otherwise*, as the original FedNTD work already did so. Our goal is to test how adversarial settings affect these two fundamentally different techniques similarly, demonstrating that *attack amplification is inherent to KD and not specific to a particular technique.*

Adversarial conditions. We simulate untargeted model poisoning attacks using techniques from [41, 11]. To observe their effects on accuracy in both benign and adversarial settings, we vary key hyperparameters — LD-divergence loss coefficient β for FedNTD and contrastive loss coefficient μ for MOON. The baseline for comparison is FedAvg with $\beta = 0$ and $\mu = 0$. To ensure high heterogeneity in both settings, the Dirichlet distribution [34] parameter α is fixed at 0.1.



(a) FedNTD, $\beta = 0$ is FedAvg (b) MOON, $\mu = 0$ is FedAvg

Figure 2: Impact of increasing KL-divergence loss for FedNTD and contrastive loss for MOON on accuracy.

Findings. In Figures 2(a) and 2(b), we present three key results: benign accuracy (blue), post-attack accuracy (orange), and the accuracy drop (green). We make the following observations from increasing β and μ are as follows: (1) the global model accuracy improves in benign settings; (2) post-attack accuracy decreases; and (3) accuracy drop increases. Our analysis shows a significant trade-off: *the very mechanisms that improve performance in benign conditions (increasing β and μ) also make the models more vulnerable to adversarial attacks.*

What causes attack amplification? The fundamental nature of KD-based FL methods aims to align local models with the global model. In benign scenarios, these methods significantly outperform FedAvg [25, 20]. However, in the presence of model poisoning attacks, *this model alignment process inadvertently forces local models to align its representation/predictions to the poisoned global model, amplifying the attack's impact*. This is illustrated in Figure 1, where clients *unknowingly distill knowledge* from a poisioned server model.

Formally: Consider a set of n clients $c_1, c_2, \ldots, c_n$ with m being malicious. Using an aggregation rule such as FedAvg, the server aggregates updates from both benign ($\nabla_{i\in[m+1,n]}$) and malicious ($\nabla^m_{i\in[m]}$) clients:

$$\nabla_g = f_{\text{agr}}(\nabla^m_{i\in[m]} \cup \nabla_{i\in[m+1,n]}) \tag{4}$$

When $m = 0$, the server model ∇^b_g is benign. For $m \neq 0$, the server model $\nabla^{'}_g$ is poisoned, deviating from the ideal unpoisoned global model due to the nature of these attacks [41, 11, 40]. Aligning local models with a poisoned global model reduces gradient diversity, making local models more similar to the poisoned global model [20] through KL-divergence or contrastive loss. We rewrite Equation 3 to formalize the loss function for an FL client, using KD, where the client is the student with output $\hat{y}_c$, and the server is the teacher with output y_s:

$$\mathcal{L} = \mathcal{L}_{CE}(\hat{y}_c, y) + \beta\mathcal{L}_{KL}(\hat{y}_c, y_s) \tag{5}$$

Note that for the sake of derivation here, we are using $\hat{y}_c$, which represents the generic client model output. In the case of FedNTD, it can be replaced by $\tilde{y}_c$ that represents the not-true logits of the client model, and in the case of MOON, it can be replaced by z_c that represents the client model's high dimensional representation.

In benign scenarios, this loss function ($\mathcal{L} = \mathcal{F}(\beta)$) decreases monotonically with β because KD brings local models closer to an unpoisoned global model. Conversely, in adversarial scenarios, it increases with β because KD brings local models to the poisoned global model. We can write the relation of this loss function with β as:

$$\mathcal{L}(\beta) \text{ is } \begin{cases} \text{monotonically decreasing}, & m = 0 \\ \text{monotonically increasing}, & m \neq 0 \end{cases} \tag{6}$$

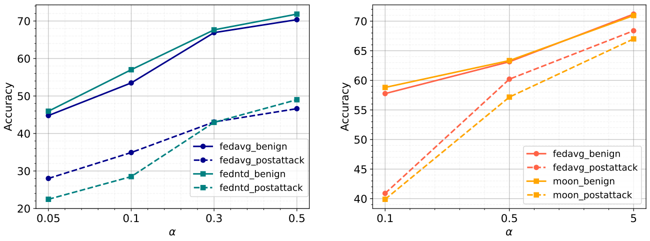
Then, the derivative of the loss function is:

$$\frac{d\mathcal{L}}{d\beta} = \begin{cases} < 0, & m = 0 \\ > 0, & m \neq 0 \end{cases} \tag{7}$$

Our derivation shows that while the distillation process decreases loss in the absence of malicious clients, it increases loss in their presence, thereby leading to reduced global model accuracy. This formal analysis highlights the need for a solution that mitigates the accuracy degradation under adversarial conditions while retaining the benefits of KD under benign conditions.

Impact of Heterogeneity.

Now, we explore the effect of heterogeneity on the performance of FedNTD, MOON, and FedAvg in both benign and adversarial conditions to gain deeper insights into the role of heterogeneity in the KD performance gain vs. vulnerability tradeoff. As shown in Figure 3(a), several interesting observations emerge. First, both FedNTD and FedAvg achieve higher accuracy at lower heterogeneity levels (indicated by higher α). In benign settings, FedNTD consistently outperforms FedAvg. However, *the trend reverses in adversarial settings:* FedAvg achieves higher accuracy than FedNTD, except at $\alpha = 0.5$. A similar pattern is observed with MOON in Figure 3(b), where FedAvg outperforms MOON across all heterogeneity levels in adversarial settings. In the benign setting, as expected, MOON slightly outperforms FedAvg at high heterogeneity. This comparison highlights again how the alignment mechanisms in FedNTD and MOON with higher heterogeneity exacerbate the vulnerability of KD methods to attacks.



(a) FedNTD vs FedAvg

(b) MOON vs FedAvg

Figure 3: Impact of the heterogeneity parameter, α in benign and adversarial settings. We use the Dirichlet distribution where a higher α means lower heterogeneity.

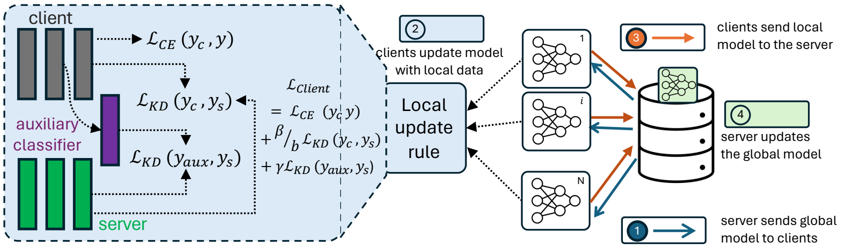


Figure 4: HYDRA-FL framework: we refine client model training by reducing the final layer's KD-loss and incorporating shallow KD-loss at an earlier shallow layer via an auxiliary classifier.

4 HYDRA-FL: Hybrid Knowledge Distillation for Robust and Accurate FL

4.1 Generic Formulation

In this section, we propose Hybrid Knowledge Distillation for Robust and Accurate FL (HYDRA-FL), a technique to mitigate the *attack amplification* caused by KD in FL. We take a hybrid distillation approach, applying KD-loss at both the final and a shallow layer of the client model (Figure 4). This method incorporates shallow distillation, which applies KD-loss at an intermediate layer and helps reduce the impact of poisoning by preventing over-reliance on final layer alignment. Shallow distillation previously used to handle *nasty teachers* trained adversarially [18], to reduce the impact of poisoning. In summary, shallow layers capture basic features, and shallow distillation ensures these features are robustly learned, protecting the model from adversarial influences that could corrupt deeper layers and final outputs. We first formulate the generic loss function of an FL client using KD in HYDRA-FL as:

$$\mathcal{L} = \mathcal{L}_{CE}(y_c, y) + \frac{\beta}{b}\mathcal{L}_{KD}(y_c, y_s) + \gamma\mathcal{L}_{KD}(y_{aux}, y_s) \tag{8}$$

This loss function has three key components:

1. **Cross-entropy loss** ($\mathcal{L}_{CE}(y_c, y)$) is the loss between the client's prediction y_c and the target y, drives the client model to learn from its own data, ensuring it captures *in-distribution knowledge* such as features and patterns specific to its data.
2. **Diminished KD loss** ($\frac{\beta}{b}\mathcal{L}_{KD}(y_c, y_s)$) is the loss between the client's output/representation y_c and the server's output/representation y_s[2]. It is a strategic reduction of the KD loss to ensure that the local model benefits from the global model's knowledge while remaining robust against adversarial attacks. This approach helps balance the trade-offs between learning efficiency and model integrity. In practice, this is achieved by introducing a *diminishing factor* b to the KD loss at the client model's output layer to diminish the poisoning effect. The KD loss coefficient β is divided by b, effectively reducing its weight in the total loss calculation, thus reducing its influence on the local model's training. This diminishing factor is essential, as shown later in §6.2 and Figure 7, where reducing the β yields better results.
3. **Shallow distillation loss** ($\gamma\mathcal{L}_{KD}(y_{aux}, y_s)$) is applied at a shallow layer of the local model, enhancing robustness without heavily relying on the final layer alignment. This loss, between the auxiliary classifier's output/representation y_{aux} at the client model's shallow layer and the server's output/representation y_s, is scaled by γ to control the amount of distillation. This approach reduces the impact of poisoning on the client model. Simply reducing the KD-loss in FedNTD or MOON improves post-attack accuracy but reduces benign setting accuracy, as shown in Figure 2. Our shallow distillation loss helps maintain the balance between accuracy in benign settings and lowering the impact of poisoning on the client model in adversarial settings.

Differences with previous works. The key difference between our work and [18] lies in our approach to shallow distillation. [18] aims to distill from models that are designed to be undistillable, a.k.a

[2] y_c and y_s in generic $\mathcal{L}_{KD}$ loss can be either outputs or representations, because the method can involve either type of comparison (e.g., MOON uses representation-based loss while FedNTD has output-based loss.)

nasty teachers [30]. While both use hybrid shallow distillation, [18] completely removes the KD-loss from the model's output layer and uses self-distillation to compensate for performance loss due to shallow distillation. In contrast, we retain a scaled-down KD-loss at the output layer. We found that completely removing the KD-loss at the output layer may cause a more negative impact than keeping it in a reduced form. Additionally, the untargeted poisoning is different from the poisoning in the "nasty teacher" paper [30]. The "nasty teacher" performs near-perfect under normal conditions unless a malicious model distills from it. In untargeted FL poisoning, the global model is poisoned and performs poorly regardless of its use for distillation.

In HYDRA-FL, we use both final layer and shallow layer distillation to enhance robustness. *Final layer distillation* aligns client outputs with server outputs for consistent predictions, whereas the *shallow layer distillation* aligns intermediate representations to improve robustness against attacks. This dual approach reduces vulnerability to poisoning attacks, enhances learning by leveraging knowledge transfer from multiple layers, and maintains high accuracy in benign settings while being resilient under attack conditions.

4.2 Adapting HYDRA-FL to State-of-Art Techniques

In this section, we will adapt our generic HYDRA-FL to two state-of-the-art KD techniques for FL.

FedNTD with shallow distillation and auxiliary classifiers. We modify the FedNTD base model by introducing auxiliary classifiers. The base model includes two convolutional layers, a linear layer, and a classification layer. Auxiliary classifiers, each consisting of a linear layer (hidden dimension 512) followed by a classification layer, are added after each convolutional layer. We update the loss function to include a shallow-distillation term, representing the KL-divergence loss between the not-true logits of an auxiliary classifier and the global model. The final loss function is a weighted sum of the standard cross-entropy loss, KL-divergence loss between the not-true logits of the global model and the client model, and the KL-divergence loss between the not-true logits of the global model and the auxiliary classifier. The revised loss function in Equation 8 for FedNTD is:

$$\mathcal{L} = \mathcal{L}_{CE}(y_c, y) + \frac{\beta}{b}\mathcal{L}_{KL}(\tilde{y}_c, \tilde{y}_s) + \gamma\mathcal{L}_{KL}(\tilde{y}_{aux}, \tilde{y}_s) \tag{9}$$

Here y is the target label, y_c is the client model's output, $\tilde{y}_s$, $\tilde{y}_c$, and $\tilde{y}_{aux}$ are the client model's, server model's, and auxiliary classifier's not-true logits respectively.

MOON with shallow distillation and auxiliary classifiers. MOON base model has two convolution layers, two linear layers, and an output classification layer. We insert auxiliary classifiers after each convolution layer. Each auxiliary classifier has two linear layers, with a hidden dimension of 256 and an output dimension of 10. We adapt Equation 8 to MOON to compute the contrastive loss at the hidden representation layer of the auxiliary classifier as:

$$\mathcal{L} = \mathcal{L}_{CE}(y_c, y) + \frac{\mu}{b}\mathcal{L}_{con}(z_c, z_s) + \gamma\mathcal{L}_{con}(z_{aux}, z_s) \tag{10}$$

Here y is the target label, y_c is the client's output, z_c is the representation from the client's final layer, z_s is the representation from the server's final layer, z_{aux} is the representation from the client's auxiliary classifier, and y_s is the server model's output. For simplicity, we do not write the previous round's representation in the loss function here.

5 Experimental Results

5.1 Experimental Settings

Datasets and Models: We conduct our experiments over three popular datasets: MNIST, CIFAR10, and CIFAR100. To ensure a fair comparison with previous works, MOON and FedNTD, we utilized the same models and hyperparameters they used. Specifically, we incorporated our algorithm as a simple modification into their publicly available codes [21, 37] (more details in Appendix D).

5.2 Shallow Not-True Distillation

Our hybrid shallow not-true distillation technique significantly improves post-attack accuracy over the baseline FedNTD. As shown in Table 1, we achieve higher post-attack accuracy across all

Table 1: Test accuracy for three techniques on three datasets. In the no-attack setting, (↑ ↓) shows comparison to FedAvg. In the attack setting, we use bold if our technique outperforms FedNTD.

Dataset	MNIST		CIFAR10						CIFAR100	
			$\alpha = 0.05$		$\alpha = 0.1$		$\alpha = 0.5$			
Techniques	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*
Fedavg	92.12	74.48	44.69	31.27	54.67	35.67	70.57	48.27	26.17	12.92
FedNTD	93.03↑	58.09	46.94↑	21.72	56.95↑	32.61	71.79↑	52.51	29.1↑	13.92
HYDRA-FL(Ours)	92.69↑	**76.67**	46.92↑	**25.15**	57.12↑	**34.25**	71.22↑	**52.57**	28.9↑	**14.33**

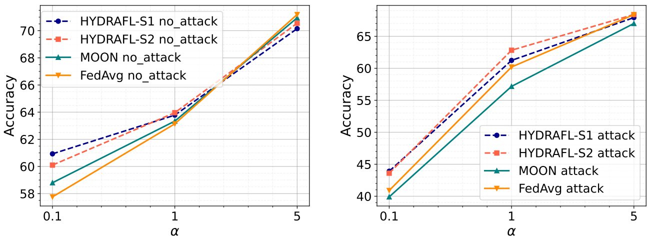
Table 2: Test accuracy for three techniques on three datasets. In the no-attack setting, (↑ ↓) shows comparison to FedAvg. In the attack setting, we use bold if our technique outperforms MOON.

Dataset	MNIST		CIFAR10						CIFAR100	
			$\alpha = 0.1$		$\alpha = 0.5$		$\alpha = 5$			
Methods	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*
Fedavg	88.02	77.55	57.76	40.9	63.14	60.2	71.19	68.38	28.36	24.21
MOON	91.13↑	72.32	58.8↑	39.9	63.34↑	57.17	70.95↓	67	29.34↑	23.81
HYDRA-FL(Ours)	92.04↑	**76.65**	60.1↑	**43.6**	63.32↑	**59.93**	70.55↓	**68.4**	29.48↑	**25.18**

heterogeneity levels. By retaining a diminished NTD loss at the output layer, we maintain similar accuracy to FedNTD in no-attack scenarios and, in some cases, even achieve slightly higher accuracy. We also compare no-attack and post-attack accuracies for FedAvg, the foundational algorithm for many FL aggregation methods.

5.3 Shallow MOON

Our shallow-distillation design effectively prevents attack amplification in MOON while maintaining nearly the same no-attack accuracy. Table 2 shows that we achieve higher post-attack accuracy across all heterogeneity levels. Our technique also outperforms FedAvg, except in a few scenarios. Techniques like MOON are designed to enhance accuracy under high heterogeneity ($\alpha = 0.1$). HYDRA-FL achieves a no-attack [attack] accuracy of $60.1[43.6]$, surpassing both MOON ($58.8[39.9]$) and FedAvg ($57.76[40.9]$).



(a) No Attack

(b) Attack

Figure 5: HYDRA-FL vs. MOON and FedAvg when auxiliary classifiers are placed at different shallow layers.

6 Analysis

In this section, we provide an in-depth analysis of HYDRA-FL. We begin with a qualitative analysis using t-distributed stochastic neighbor embedding (t-SNE [42]) plots to visualize the representations of the models. Then, we explore the impact of different design choices through ablation studies, focusing on the choice of the shallow layer for auxiliary classifiers and the distillation coefficients.

6.1 Qualitative Analysis

We show the t-SNE plots of the representations (Figure 6) generated by the client model for FedAvg, MOON, and HYDRA-FL for both attack and no-attack scenarios. The t-SNE plots show the classes as clusters. In the MOON attack scenario, the deviation from the no-attack scenario is much higher than the deviation between HYDRA-FL with and without attack, as evident from the spread of the class clusters, especially along the x-axis.

6.2 Ablation Study

Impact of choice of the shallow layer. Figure 5 illustrates the impact of the choice of the layer at which we insert our auxiliary classifier. We represent these choices by *HYDRAFL-S1* and *HYDRAFL-S2*, where the auxiliary classifier is inserted after the first and second convolutional layers, respectively.

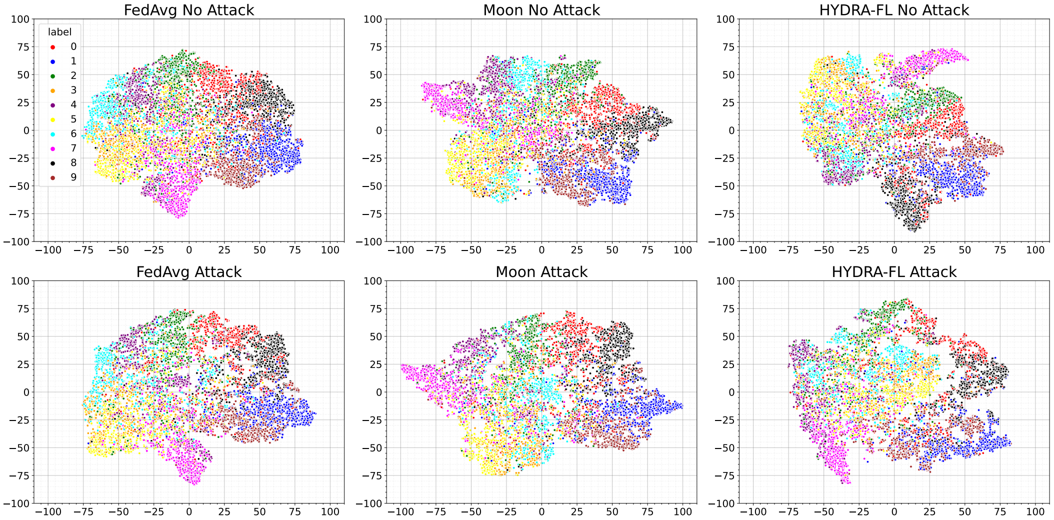


Figure 6: T-SNE visualizations of CIFAR10 on local model's hidden representations ($\alpha = 0.5$) on FedAvg, MOON, and HYDRA-FL (ours). The attack vs. no-attack plot shows the deviation of the attack clusters from the no-attack clusters. Visually we can see MOON-attack has the greatest deviation, particularly along the x-axis, compared to FedAvg and HYDRA-FL.

We compare them in both attack and no-attack settings with simple MOON and FedAvg. In Figure 5(a), both HYDRAFL-S1 and HYDRAFL-S2 outperform other techniques at low heterogeneity in the absence of an attack but slightly underperform in low heterogeneity when $\beta = 5$. Figure 5(b) shows that both HYDRAFL-S1 and HYDRAFL-S2 achieve higher post-attack accuracy at all heterogeneity levels, with HYDRAFL-S2 giving a slightly higher accuracy than HYDRAFL-S1. The benefit from the contrastive loss reduces as we go shallower, so an optimal balance is necessary.

Impact of distillation coefficients. We examine the impact of distillation coefficients on the performance of FedNTD and HYDRA-FL. Figure 7 shows the post-attack accuracies with two different values of the *diminishing factor* $b = 1, 4$, resulting in output-layer NTD-loss coefficients of $\beta = 1$ and $\beta = 0.25$. Diminishing the coefficient β leads to improved performance, with a significant increase in post-attack accuracy for $\beta = 0.25$ at high heterogeneity ($\alpha = 0.05, 0.1$). As demonstrated in §3, β contributes to attack amplification in FedNTD. Reducing it while performing distillation at the auxiliary classifier yields the best performance. For example, at $\alpha = 0.05$, HYDRA-FL achieves 25.15% accuracy at $\beta = 1$, but a much higher accuracy of 28.81% at $\beta = 0.25$. Similar improvements are observed at other heterogeneity levels.

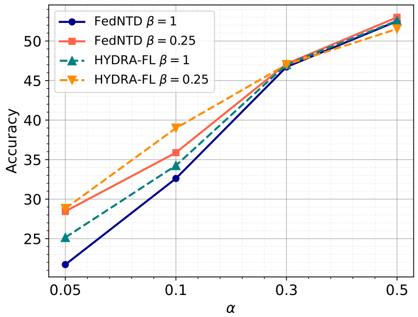


Figure 7: Comparison of performance of FedNTD-S with different values of β

7 Conclusion

In this paper, we first identified a critical issue in KD-based FL techniques that aim to tackle data heterogeneity: in the presence of model poisoning attacks, these techniques help the attacker amplify its effect, leading to reduced global model performance. We presented empirical evidence and theoretical reasoning to back this claim. This motivated us to propose HYDRA-FL: a hybrid knowledge distillation technique for robust and accurate FL technique that aims to tackle both data heterogeneity and model poisoning, two of the biggest problems in FL. Through extensive evaluation across three datasets and comparing with baseline techniques, FedNTD and MOON, we showed that HYDRA-FL achieves superior results.

References

[1] How Apple personalizes Siri without hoovering up your data. https://www.technologyreview.com/2019/12/11/131629/apple-ai-personalizes-siri-federated-learning/.

[2] Federated learning: Collaborative machine learning without centralized training data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *AISTATS*, 2020.

[4] Moran Baruch, Baruch Gilad, and Yoav Goldberg. A Little Is Enough: Circumventing Defenses For Distributed Learning. In *NeurIPS*, 2019.

[5] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *ICML*, 2019.

[6] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*, 2017.

[7] X. Cao, J. Jia, Z. Zhang, and N. Gong. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 326–343, Los Alamitos, CA, USA, may 2023. IEEE Computer Society.

[8] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Provably Secure Federated Learning against Malicious Clients. In *AAAI*, 2021.

[9] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer. *arXiv:1912.11279*, 2019.

[10] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

[11] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *USENIX*, 2020.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[13] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[14] Peter Kairouz, H Brendan McMahan, Brendan Avent, et al. Advances and open problems in federated learning. *arXiv:1912.04977*, 2019.

[15] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[16] Momin Ahmad Khan, Virat Shejwalkar, Amir Houmansadr, and Fatima M Anwar. On the pitfalls of security evaluation of robust federated learning. In *2023 IEEE Security and Privacy Workshops (SPW)*, pages 57–68. IEEE, 2023.

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

[18] Souvik Kundu, Qirui Sun, Yao Fu, Massoud Pedram, and Peter Beerel. Analyzing the confidentiality of undistillable teachers in knowledge distillation. *Advances in Neural Information Processing Systems*, 34:9181–9192, 2021.

[19] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[20] Gihun Lee, Minchan Jeong, Yongjin Shin, Sangmin Bae, and Se-Young Yun. Preservation of the global knowledge by not-true distillation in federated learning. *Advances in Neural Information Processing Systems*, 35:38461–38474, 2022.

[21] Lee-Gihun. Fedntd. `https://github.com/Lee-Gihun/FedNTD/tree/master`.

[22] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[23] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *AAAI*, 2019.

[24] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pages 965–978. IEEE, 2022.

[25] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.

[26] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *ICML*, 2021.

[27] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[28] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[29] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *NeurIPS*, 2020.

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

[31] Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. Universal multi-party poisoning attacks. In *ICML*, 2019.

[32] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[33] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The Hidden Vulnerability of Distributed Learning in Byzantium. In *ICML*, 2018.

[34] Thomas Minka. Estimating a Dirichlet distribution, 2000.

[35] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *AISec*, 2017.

[36] PyTorch Documentation. `https://pytorch.org/`, 2019.

[37] QinbinLi. Moon. `https://github.com/QinbinLi/MOON/tree/main`.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[39] Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive Federated Optimization. In *ICLR*, 2020.

[40] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 1117–1134, Los Alamitos, CA, USA, may 2022. IEEE Computer Society.

[41] Virat Shejwalkar and Amir Houmansadr. Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning. In *NDSS*, 2021.

[42] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[43] Utilization of FATE in Risk Management of Credit in Small and Micro Enterprises. `https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-enterprises/`, 2019.

[44] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized byzantine-tolerant sgd. *arXiv:1802.10116*, 2018.

[45] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation. *arXiv:1903.03936*, 2019.

[46] Yueqi Xie, Weizhong Zhang, Renjie Pi, Fangzhao Wu, Qifeng Chen, Xing Xie, and Sunghun Kim. Robust federated learning against both data heterogeneity and poisoning attack via aggregation optimization. *arXiv preprint*, 2022.

[47] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2018.

[48] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10174–10183, 2022.

[49] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3713–3722, 2019.

[50] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2545–2555, 2022.

[51] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[52] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.

[53] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, pages 12878–12889. PMLR, 2021.

Appendix

We provide additional information for our paper, HYDRA-FL: Hybrid Knowledge Distillation for Robust and Accurate Federated Learning, in the following order:

- Limitations and Future Work (Appendix A)
- Terminology/Techniques (Appendix B
- Adversarial Settings (Appendix C)
- Experimental Setup (Appendix D)
- Additional Results (Appendix E

A Limitations and Future Work

Federated Learning can have very diverse setups, especially FL in an adversarial setting. We can have many setup combinations as we can choose between different aggregation rules, attacks, defenses, datasets, data modalities, data distribution types, data heterogeneity levels, number of clients, etc. Therefore, evaluating against all combinations of these settings is well beyond the scope of one paper. Hence, for this paper, we chose only a few combinations of FL settings and tried our best to show that the problem we identified using two representative FL techniques will also exist in similar techniques. Similarly, we laid out our solution as a general framework to achieve good performance under high heterogeneity and model poisoning simultaneously. To show generalizability, we tailored it to our two representative techniques, but it would be interesting to see how our solution adapts to and performs with other FL techniques in future works. Also, we have only used unimodal, i.e., image datasets for our evaluations. This was done to stay consistent with the implementations of the techniques chosen for our case study, FedNTD and MOON. However, the language modality is becoming popular now, and multimodal models such as CLIP [38] are being widely used as they achieve superior performance by combining both image and language modalities. We hope to incorporate language and multimodal models in our future works.

B Terminology/Techniques

B.1 FedNTD

FedNTD [20] is a KD-based technique that tackles the problem of data heterogeneity in FL. They first demonstrate that Data Heterogeneity causes local models to forget out-distribution knowledge, i.e., the data samples not part of the client's local data. Therefore, to preserve the out-distribution knowledge, they introduce not-true distillation, which basically modifies the loss function for the client model's local objective. FedNTD's loss function is given by:

$$\mathcal{L} = \mathcal{L}_{CE}(y_c, y) + \frac{\beta}{b}\mathcal{L}_{KL}(\tilde{y}_c, \tilde{y}_s) \tag{11}$$

Here y is the target label, y_c is the client model's output, $\tilde{y}_s$ and $\tilde{y_c}$ are the client model's and the server model's not-true logits, respectively.

B.2 MOON

MOON [25] also aims to solve the problem of data heterogeneity in FL. They do so by reducing the distance between the representation learned by the local model with that of the global model. MOON's loss function is given by:

$$\mathcal{L} = \mathcal{L}_{CE}(y_c, y) + \frac{\mu}{b}\mathcal{L}_{con}(z_c, z_s) \tag{12}$$

Here y is the target label, y_c is the client's output, z_c is the representation from the client's final layer, z_s is the representation from the server's final layer, and y_s is the server model's output.

B.3 Shallow Layer and Shallow Distillation

Shallow layer. in a neural network refers to one of the early layers close to the input, as opposed to deeper layers that are closer to the output. In the context of a deep learning model, shallow layers generally capture low-level features, such as edges in images or simple patterns in data, while deeper layers capture more complex, abstract representations.

Shallow distillation. is a technique used in KD where the knowledge transfer happens at a shallow layer of the neural network rather than at the final output layer. In traditional KD, the student model tries to mimic the teacher model's output at the final layer. In shallow distillation, an additional distillation loss is applied at one of the shallow layers of the student model. This helps the student model learn intermediate representations from the teacher, providing a more comprehensive learning experience. By aligning these intermediate representations, the student model gains a more robust understanding of the data, leading to better *generalization*.

Robustness against poisoning. Shallow layers are less affected by adversarial attacks that target the final output of the model. Applying distillation at a shallow layer reduces the impact of a poisoned global model because the knowledge transferred is more fundamental and less influenced by the adversarial manipulations that typically affect the deeper layers.

C Adversarial Settings

Here we present the details of the adversarial settings of our experiments. We explain our threat model, which attacks we are using and why we are using them, and the defense we are using.

C.1 Threat Model

Goal: Our untargeted poisoning adversary controls m out of N clients to manipulate the global model to misclassify all the inputs it can during testing. Unless stated otherwise, we assume 20% malicious clients. Most defense works assume high percentages of malicious clients to demonstrate that their defenses work even in highly adversarial settings. Hence, although unreasonable in practical FL settings [40], we follow prior defense works and use 20% malicious clients.

Knowledge: Following most of the defense works, we assume that the adversary knows the robust AGR that the server uses. As assumed by most works, the adversary knows the server's AGR. To test the efficacy of our technique with a strong adversary, we consider the case where the adversary has access to not only the malicious clients' data but also the benign clients' data. This enables us to determine the upper bound of the efficacy of our technique.

Capabilities: Our adversary is strong enough to directly manipulate model updates of the malicious clients it controls. While poisoning attacks come in various types and flavors, we restrict ourselves to only model poisoning attacks. This is because model poisoning attacks are much stronger. It has been shown in [40] that model poisoning attacks are much stronger because they directly perturb the local model parameters. In contrast, data poisoning attacks perturb the data, subsequently perturbing the local and global models upon aggregation. Poisoning attacks can also be classified based on their error specificity. If the goal is to misclassify certain classes only, then it is a *targeted attack* and is often achieved by inserting a backdoor in the model that activates only for certain inputs. On the other hand, an *untargeted attack* indiscriminately lowers the accuracy for all inputs.

C.2 Attacks we use in our evaluation

We use two model poisoning attacks for our evaluations. By testing which attack worked well, we chose the Stat-Opt attack for MOON and the Dyn-Opt attack for FedNTD. Below, we briefly explain how they work:

- **Stat-Opt [11]:** gives an untargeted model poisoning framework and tailors it to specific defenes such as TrMean [47], Median [47], and Krum [6]. The adversary first calculates the mean of the benign updates, ∇^b, and finds the *static* malicious direction $w = -sign(\nabla^b)$. It directs the benign average along the calculated direction and scales it with γ to obtain the final poisoned update, $-\gamma w$.

- **Dyn-Opt [41]:** also gives an untargeted model poisoning framework and tailors it to specific defenses, similar to Stat-Opt but differs in the *dynamic* and *data-dependent* nature of the perturbation. The attack first computes the mean of benign updates, ∇^b, and a data-dependent direction, w. The final poisoned update is calculated as $\nabla^{`} = \nabla^b + \gamma w$, where the attack finds the largest γ that can bypass the AGR. They compare their attack with Stat-Opt and show that the dataset-tailored w and optimization-based scaling factor γ make their attack much stronger.

C.3 Defense we use in our evaluation

We use the Trimmed Mean defense in our evaluations. Trimmed Mean [47, 44] is a foundational defense used in advanced AGRs [7, 50, 41]. The server receives model updates from each client, sorts each input dimension j, discards the m largest and smallest values (where m indicates malicious clients), and averages the rest.

D Experimental Setup

Models: For MOON, we use a base encoder with two 5×5 convolutional layers, each followed by a 2×2 max pooling layer and two fully connected layers with ReLU activation. The base encoder is followed by a projection head with an output dimension of 256. For FedNTD, we use a model (similar to the one in [32]) having two convolutional layers followed by a linear layer and a classification layer. For FedNTD, we test with different values and settle upon a diminishing factor $b = 1$ and γ=2. For MOON, we set $\beta = 0$ and set $\gamma = 1$. We used PyTorch [36] for our implementation on an 8GB NVIDIA RTX 3060 Ti GPU. Each run of FedNTD and MOON took about 2-3 hours on our machine.

FL Settings: For FedNTD, we use 100 clients with a sampling ratio of 0.1, i.e., 10 clients are selected every round. We use momentum SGD with an initial learning rate of 0.1, weight decay of $1 \times e^{-5}$, batch size of 50, and momentum of 0.9. Each run consists of 200 rounds with 5 local epochs. For MOON, we use 10 clients with a sampling ratio of 1. We use SGD with an initial learning rate of 0.01, weight decay of $1 \times e^{-5}$, batch size of 64, and momentum of 0.9. Each run consists of 30 rounds with 10 local epochs, sufficient for convergence.

Data Partitioning: We use the widely used Dirichlet [34] distribution to generate the non-IID partitioning of data between clients. Dirichlet distribution works by sampling $p_k \sim Dir_N(\alpha)$ and assigns $p_{k,j}$ proportion of samples of class k to client j. A lower value of α corresponds to a higher level of heterogeneity since it means that most of the samples of a certain class belong to one client. Conversely, at a higher value of α, the class samples are more evenly distributed between the clients. Also, a characteristic of the Dirichlet distribution is that both local dataset size and local per-class distribution vary across clients.

Datasets: The three datasets we use in our experiments are:

- **MNIST [19]:** MNIST is a 10-class digit image classification dataset, which contains 70,000 grayscale images of size 28×28. We divide all data among FL clients (100 for FedNTD and 10 for MOON) using the Dirichlet [39] distribution.
- **CIFAR10 [17]:** CIFAR10 is a 10-class classification task with 60,000 total RGB images, each of size 32×32. Each class has 6000 training images and 1000 testing images. We divide all the data among 100 clients using the Dirichlet distribution, a popular synthetic strategy to generate FL datasets.
- **CIFAR100 [17]:** CIFAR100 is similar to CIFAR10, except that it is a 100-class classification task where each class has 600 images of size 32×32. There are 500 training images and 100 test images per class. Like other datasets, we also partition this dataset using the Dirichlet distribution.

E Additional Results

In this section, we present some of the additional results we have obtained.

Table 3: FedNTD

Dataset	**MNIST**		**CIFAR10**								**CIFAR100**	
			0.05		**0.1**		**0.3**		**0.5**			
Techniques	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*	*no attack*	*attack*
Fedavg	92.12	74.48	44.69	31.27	54.67	35.67	66.34	42.53	70.57	48.27	26.17	12.92
MOON	93.03	58.09	46.94	21.72	56.95	32.61	68	46.72	71.79	52.51	29.1	13.92
Ours	92.69	76.67	46.92	25.15	57.12	34.25	68.1	47.03	71.22	52.57	28.9	14.33

E.1 FedNTD

For visual symmetry, we did not include the full table in §5, but we had also run our FedNTD experiments at $\alpha = 0.3$. We show the full FedNTD results in Table 3. Here, we can see that at at $\alpha = 0.3$ too, we achieve superior results FedAvg and FedNTD in both benign and adversarial conditions.

E.2 MOON

We also ran ablation with MNIST for different shallow layers and diminishing coefficients. We show the results in Table 4, where we can see that at a lower μ, i.e., higher diminishing factor, we achieve the best results. A lower μ does give us better no-attack accuracy, but we lose a lot in the attack scenario.

Method	μ	**no-attack**	**attack**
HYDRA-FL s1	1	94.41	68.68
HYDRA-FL s2	1	91.78	68.13
HYDRA-FL s1	0.3	92.03	72.35
HYDRA-FL s2	0.3	92.92	73.55
HYDRA-FL s1	0.1	92.04	76.65
HYDRA-FL s2	0.1	93.93	72.54

Table 4: Comparison of HYDRA-FL for MOON with different distillation coefficients.