

## 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:

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

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:

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

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 defenses such as TrMean [47], Median [47], and Krum [6]. The adversary first calculates the mean of the benign updates,  $\nabla^b$ , and finds the *static* malicious direction  $w = -\text{sign}(\nabla^b)$ . It directs the benign average along the calculated direction and scales it with  $\gamma$  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,  $\nabla^b$ , and a data-dependent direction,  $w$ . The final poisoned update is calculated as  $\nabla' = \nabla^b + \gamma w$ , where the attack finds the largest  $\gamma$  that can bypass the AGR. They compare their attack with Stat-Opt and show that the dataset-tailored  $w$  and optimization-based scaling factor  $\gamma$  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  $5 \times 5$  convolutional layers, each followed by a  $2 \times 2$  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  $\gamma = 2$ . 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  $p_k \sim \text{Dir}_N(\alpha)$  and assigns  $p_{k,j}$  proportion of samples of class  $k$  to client  $j$ . A lower value of  $\alpha$  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  $\alpha$ , 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  $28 \times 28$ . 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  $32 \times 32$ . 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  $32 \times 32$ . 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	<i>no attack</i>	<i>attack</i>	<i>no attack</i>	<i>attack</i>	<i>no attack</i>	<i>attack</i>	<i>no attack</i>	<i>attack</i>	<i>no attack</i>	<i>attack</i>	<i>no attack</i>	<i>attack</i>
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  $\alpha = 0.3$  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  $\mu$ , i.e., higher diminishing factor, we achieve the best results. A lower  $\mu$  does give us better no-attack accuracy, but we lose a lot in the attack scenario.

Method	$\mu$	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.