

488 Appendix

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

- 491 • Limitations and Future Work (Appendix [A](#))
- 492 • Terminology/Techniques (Appendix [B](#))
- 493 • Adversarial Settings (Appendix [C](#))
- 494 • Experimental Setup (Appendix [D](#))
- 495 • Additional Results (Appendix [E](#))

496 A Limitations and Future Work

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

512 B Terminology/Techniques

513 B.1 FedNTD

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

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

519 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
520 server model’s not-true logits, respectively.

521 B.2 MOON

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

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

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

527 B.3 Shallow Layer and Shallow Distillation

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

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

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

543 C Adversarial Settings

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

546 C.1 Threat Model

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

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

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

566 C.2 Attacks we use in our evaluation

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

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

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

582 C.3 Defense we use in our evaluation

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

587 D Experimental Setup

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

595 **FL Settings:** For FedNTD, we use 100 clients with a sampling ratio of 0.1, i.e., 10 clients are
 596 selected every round. We use momentum SGD with an initial learning rate of 0.1, weight decay of
 597 $1 \times e^{-5}$, batch size of 50, and momentum of 0.9. Each run consists of 200 rounds with 5 local epochs.
 598 For MOON, we use 10 clients with a sampling ratio of 1. We use SGD with an initial learning rate
 599 of 0.01, weight decay of $1 \times e^{-5}$, batch size of 64, and momentum of 0.9. Each run consists of 30
 600 rounds with 10 local epochs, sufficient for convergence.

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

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

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

620 E Additional Results

621 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>										
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

622 E.1 FedNTD

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

627 E.2 MOON

628 We also ran ablation with MNIST for different shallow layers and diminishing coefficients. We show
 629 the results in Table 4, where we can see that at a lower μ , i.e., higher diminishing factor, we achieve
 630 the best results. A lower μ does give us better no-attack accuracy, but we lose a lot in the attack
 631 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.