Ensemble Distillation for Robust Model Fusion in Federated Learning

Tao Lin[*]**, Lingjing Kong**[*]**, Sebastian U. Stich, Martin Jaggi.**
MLO, EPFL, Switzerland
{tao.lin, lingjing.kong, sebastian.stich, martin.jaggi}@epfl.ch

Abstract

Federated Learning (FL) is a machine learning setting where many devices collaboratively train a machine learning model while keeping the training data decentralized. In most of the current training schemes the central model is refined by averaging the parameters of the server model and the updated parameters from the client side. However, directly averaging model parameters is only possible if all models have the same structure and size, which could be a restrictive constraint in many scenarios.

In this work we investigate more powerful and more flexible aggregation schemes for FL. Specifically, we propose ensemble distillation for model fusion, i.e. training the central classifier through unlabeled data on the outputs of the models from the clients. This knowledge distillation technique mitigates privacy risk and cost to the same extent as the baseline FL algorithms, but allows flexible aggregation over heterogeneous client models that can differ e.g. in size, numerical precision or structure. We show in extensive empirical experiments on various CV/NLP datasets (CIFAR-10/100, ImageNet, AG News, SST2) and settings (heterogeneous models/data) that the server model can be trained much faster, requiring fewer communication rounds than any existing FL technique so far.

1 Introduction

Federated Learning (FL) has emerged as an important machine learning paradigm in which a federation of clients participate in collaborative training of a centralized model [62, 51, 65, 8, 5, 42, 34]. The clients send their model parameters to the server but never their private training datasets, thereby ensuring a basic level of privacy. Among the key challenges in federated training are communication overheads and delays (one would like to train the central model with as few communication rounds as possible), and client heterogeneity: the training data (non-i.i.d.-ness), as well as hardware and computing resources, can change drastically among clients, for instance when training on commodity mobile devices.

Classic training algorithms in FL, such as federated averaging (FEDAVG) [51] and its recent adaptations [53, 44, 25, 35, 26, 58], are all based on directly averaging of the participating client's *parameters* and can hence only be applied if all client's models have the same size and structure. In contrast, ensemble learning methods [77, 15, 2, 14, 56, 47, 75] allow to combine multiple heterogeneous weak classifiers by averaging the *predictions* of the individual models instead. However, applying ensemble learning techniques directly in FL is infeasible in practice due to the large number of participating clients, as it requires keeping weights of all received models on the server and performing naive ensembling (logits averaging) for inference.

To enable federated learning in more realistic settings, we propose to use ensemble distillation [7, 22] for robust model fusion (FedDF). Our scheme leverages unlabeled data or artificially generated examples (e.g. by a GAN's generator [17]) to aggregate knowledge from all received (heterogeneous)
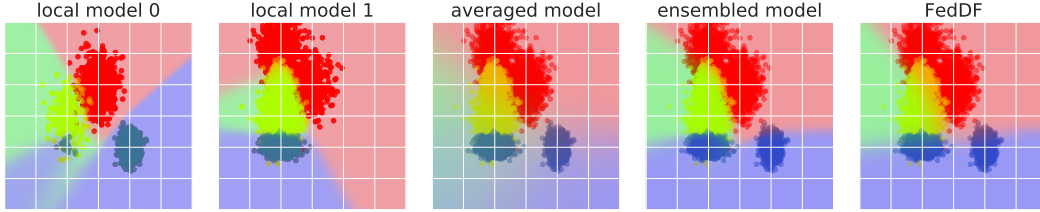
---

[*]Equal contribution.

Figure 1: **Limitations of FEDAVG.** We consider a toy example of a 3-class classification task with a 3-layer MLP, and display the decision boundaries (probabilities over RGB channels) on the input space. The left two figures show the individually trained local models. The right three figures evaluate aggregated models and the global data distribution; the averaged model results in much blurred decision boundaries. The used datasets are displayed in Figure 8 (Appendix C.1).

client models. We demonstrate with thorough empirical results that our ensemble distillation approach not only addresses the existing quality loss issue [24] of Batch Normalization (BN) [31] for networks in a homogeneous FL system, but can also break the knowledge barriers among heterogeneous client models. Our main contributions are:

- We propose a distillation framework for robust federated model fusion, which allows for heterogeneous client models and data, and is robust to the choices of neural architectures.
- We show in extensive numerical experiments on various CV/NLP datasets (CIFAR-10/100, ImageNet, AG News, SST2) and settings (heterogeneous models and/or data) that the server model can be trained much faster, requiring fewer communication rounds than any existing FL technique.

We further provide insights on when FedDF can outperform FEDAVG (see also Fig. 1 that highlights an intrinsic limitation of parameter averaging based approaches) and what factors influence FedDF.

2 Related Work

Federated learning. The classic algorithm in FL, FEDAVG [51], or local SGD [46] when all devices are participating, performs weighted parameter average over the client models after several local SGD updates with weights proportional to the size of each client's local data. Weighting schemes based on client loss are investigated in [53, 44]. To address the difficulty of directly averaging model parameters, [64, 74] propose to use optimal transport and other alignment schemes to first align or match individual neurons of the neural nets layer-wise before averaging the parameters. However, these layer-based alignment schemes necessitate client models with the same number of layers and structure, which is restrictive in heterogeneous systems in practice.

Another line of work aims to improve local client training, i.e., client-drift problem caused by the heterogeneity of local data [43, 35]. For example, FEDPROX [43] incorporates a proximal term for the local training. Other techniques like acceleration, recently appear in [25, 26, 58].

Knowledge distillation. Knowledge distillation for neural networks is first introduced in [7, 22]. By encouraging the student model to approximate the output logits of the teacher model, the student is able to imitate the teacher's behavior with marginal quality loss [59, 79, 36, 71, 37, 28, 1, 70]. Some work study the ensemble distillation, i.e., distilling the knowledge of an ensemble of teacher models to a student model. To this end, existing approaches either average the logits from the ensemble of teacher models [77, 15, 2, 14], or extract knowledge from the feature level [56, 47, 75].

Most of these schemes rely on using the original training data for the distillation process. In cases where real data is unavailable, some recent work [54, 52] demonstrate that distillation can be accomplished by crafting pseudo data either from the weights of the teacher model or through a generator adversarially trained with the student. FedDF can be combined with all of these approaches. In this work, we consider unlabeled datasets for ensemble distillation, which could be either collected from other domains or directly generated from a pre-trained generator.

Comparison with close FL work. Guha *et al*. [18] propose "one-shot fusion" through unlabeled data for SVM loss objective, whereas we consider multiple-round scenarios on diverse neural architectures and tasks. FD [33] utilizes distillation to reduce FL communication costs. To this end, FD synchronizes logits per label which are accumulated during the local training. The averaged logits per label (over local steps and clients) will then be used as a distillation regularizer for the next round's local training. Compared to FEDAVG, FD experiences roughly 15% quality drop on MNIST. In contrast, FedDF shows superior learning performance over FEDAVG and can significantly reduce the number of communication rounds to reach target accuracy on diverse challenging tasks.

FedMD [41] and the recently proposed Cronus [9] consider learning through averaged logits per sample on a public dataset. After the initial pre-training on the labeled public dataset, FedMD learns on the public and private dataset iteratively for personalization, whereas in Cronus, the public dataset (with soft labels) is used jointly with local private data for the local training. As FedMD trains client models simultaneously on both labeled public and private datasets, the model classifiers have to include all classes from both datasets. Cronus, in its collaborative training phase, mixes public and private data for local training. Thus for these methods, the public dataset construction requires careful deliberation and even prior knowledge on clients' private data. Moreover, how these modifications impact local training quality remains unclear. FedDF faces no such issues: we show that FedDF is robust to distillation dataset selection and the distillation is performed on the server side, leaving local training unaffected. We include a detailed discussion with FedMD, Cronus in Appendix A. When preparing this version, we also notice other contemporary work [68, 10, 81, 19] and we defer discussions to Appendix A.

3 Ensemble Distillation for Robust Model Fusion

Algorithm 1 Illustration of FedDF on K homogeneous clients (indexed by k) for T rounds, n_k denotes the number of data points per client and C the fraction of clients participating in each round. The server model is initialized as $\mathbf{x}_0$. While FEDAVG just uses the averaged models $\mathbf{x}_{t,0}$, we perform N iterations of server-side model fusion on top (line 7 – line 10).

1: **procedure** SERVER
2:     **for** each communication round $t = 1, \ldots, T$ **do**
3:         $S_t \leftarrow$ random subset (C fraction) of the K clients
4:         **for** each client $k \in S_t$ **in parallel do**
5:             $\hat{\mathbf{x}}_t^k \leftarrow$ Client-LocalUpdate$(k, \mathbf{x}_{t-1})$ ▷ detailed in Algorithm 2.
6:         initialize for model fusion $\mathbf{x}_{t,0} \leftarrow \sum_{k \in S_t} \frac{n_k}{\sum_{k \in S_t} n_k} \hat{\mathbf{x}}_t^k$
7:         **for** j in $\{1, \ldots, N\}$ **do**
8:             sample a mini-batch of samples $\mathbf{d}$, from e.g. (1) an unlabeled dataset, (2) a generator
9:             use ensemble of $\{\hat{\mathbf{x}}_t^k\}_{k \in S_t}$ to update server student $\mathbf{x}_{t,j-1}$ through AVGLOGITS
10:         $\mathbf{x}_t \leftarrow \mathbf{x}_{t,N}$
11:     **return** $\mathbf{x}_T$

In this section, we first introduce the core idea of the proposed Federated Distillation Fusion (FedDF). We then comment on its favorable characteristics and discuss possible extensions.

Ensemble distillation. We first discuss the key features of FedDF for the special case of homogeneous models, i.e. when all clients share the same network architecture (Algorithm 1). For model fusion, the server distills the ensemble of $|S_t|$ client teacher models to one single server student model. For the distillation, the teacher models are evaluated on mini-batches of unlabeled data on the server (forward pass) and their logit outputs (denoted by $f(\hat{\mathbf{x}}_t^k, \mathbf{d})$ for mini-batch $\mathbf{d}$) are used to train the student model on the server:

$$\mathbf{x}_{t,j} := \mathbf{x}_{t,j-1} - \eta \frac{\partial \text{KL} \left(\sigma \left(\frac{1}{|S_t|} \sum_{k \in S_t} f(\hat{\mathbf{x}}_t^k, \mathbf{d}) \right), \sigma \left(f(\mathbf{x}_{t,j-1}, \mathbf{d}) \right) \right)}{\partial \mathbf{x}_{t,j-1}} . \quad \text{(AVGLOGITS)}$$

Here KL stands for Kullback–Leibler divergence, σ is the softmax function, and η is the stepsize. FedDF can easily be extended to heterogeneous FL systems (Algorithm 3 and Figure 7 in Appendix B). We assume the system contains p distinct model prototype groups that potentially differ in neural architecture, structure and numerical precision. By ensemble distillation, each model architecture group acquires knowledge from logits averaged over *all* received models, thus mutual beneficial information can be shared across architectures; in the next round, each activated client receives the corresponding fused prototype model. Notably, as the fusion takes place on the server side, there is no additional burden and interference on clients.

Utilizing unlabeled/generated data for distillation. Unlike most existing ensemble distillation methods that rely on *labeled* data from the training domain, we demonstrate the feasibility of achieving model fusion by using *unlabeled* datasets from other domains for the sake of privacy-preserving FL. Our proposed method also allows the use of synthetic data from a pre-trained generator (e.g.

GAN[2]) as distillation data to alleviate potential limitations (e.g. acquisition, storage) of real unlabeled datasets.

Discussions on privacy-preserving extension. Our proposed model fusion framework in its simplest form—like most existing FL methods—requires to exchange models between the server and each client, resulting in potential privacy leakage due to e.g. memorization present in the models. Several existing protection mechanisms can be added to our framework to protect clients from adversaries. These include adding differential privacy [16] for client models, or performing hierarchical and decentralized model fusion through synchronizing locally inferred logits e.g. on *random* public data[3], as in the recent work [9]. We leave further explorations of this aspect for future work.

4 Experiments

4.1 Setup

Datasets and models. We evaluate the learning of different SOTA FL methods on both CV and NLP tasks, on architectures of ResNet [20], VGG [63], ShuffleNetV2 [48] and DistilBERT [60]. We consider federated learning CIFAR-10/100 [38] and ImageNet [39] (down-sampled to image resolution 32 for computational feasibility [11]) from scratch for CV tasks; while for NLP tasks, we perform federated fine-tuning on a 4-class news classification dataset (AG News [80]) and a 2-class classification task (Stanford Sentiment Treebank, SST2 [66]). The validation dataset is created for CIFAR-10/100, ImageNet, and SST2, by holding out 10%, 1% and 1% of the original training samples respectively; the remaining training samples are used as the training dataset (before partitioning client data) and the whole procedure is controlled by random seeds. We use validation/test datasets on the server and report the test accuracy over three different random seeds.

Heterogeneous distribution of client data. We use the Dirichlet distribution as in [78, 25] to create disjoint non-i.i.d. client training data. The value of α controls the degree of non-i.i.d.-ness: $\alpha = 100$ mimics identical local data distributions, and the smaller α is, the more likely the clients hold examples from only one class (randomly chosen). Figure 2 visualizes how samples are distributed among 20 clients for CIFAR-10 on different α values; more visualizations are shown in Appendix C.2.

Baselines. FedDF is designed for effective model fusion on the server, considering the accuracy of the global model on the test dataset. Thus we omit the comparisons to methods designed for personalization (e.g. FedMD [41]), security/robustness (e.g. Cronus [9]), and communication efficiency (e.g. [33], known for poorer performance than FEDAVG). We compare FedDF with SOTA FL methods, including 1) FEDAVG [51], 2) FEDPROX [43] (for better local training under heterogeneous systems), 3) accelerated FEDAVG a.k.a. FEDAVGM[4] [25, 26], and 4) FEDMA[5] [74] (for better model fusion). We elaborate on the reasons for omitted numerical comparisons in Appendix A.

The local training procedure. The FL algorithm randomly samples a fraction (C) of clients per communication round for local training. For the sake of simplicity, the local training in our experiments uses a constant learning rate (no decay), no Nesterov momentum acceleration, and no weight decay. The hyperparameter tuning procedure is deferred to Appendix C.2. Unless mentioned otherwise the learning rate is set to 0.1 for ResNet-like nets, 0.05 for VGG, and $1e-5$ for DistilBERT.

The model fusion procedure. We evaluate the performance of FedDF by utilizing either randomly sampled data from existing (unlabeled) datasets[6] or BigGAN's generator [6]. Unless mentioned otherwise we use CIFAR-100 and downsampled ImageNet (image size 32) as the distillation datasets for FedDF on CIFAR-10 and CIFAR-100 respectively. Adam with learning rate $1e-3$ (w/ cosine annealing) is used to distill knowledge from the ensemble of received local models. We employ early-stopping to stop distillation after the validation performance plateaus for $1e3$ steps (total $1e4$ update steps). The hyperparameter used for model fusion is kept constant over all tasks.

---

[2] GAN training is not involved in all stages of FL and cannot steal clients' data. Data generation is done by the (frozen) generator before the FL training by performing inference on random noise. Adversarially involving GAN's training during the FL training may cause the privacy issue, but it is beyond the scope of this paper.

[3] For instance, these data can be generated locally from identical generators with a controlled random state.

[4] The performance of FEDAVGM is coupled with local learning rate, local training epochs, and the number of communication rounds. The preprints [25, 26] consider small learning rate for at least 10k communication rounds; while we use much fewer communication rounds, which sometimes result in different observations.

[5] FEDMA does not support BN or residual connections, thus the comparison is only performed on VGG-9.

[6] Note the actual computation expense for distillation is determined by the product of the number of distillation steps and distillation mini-batch size (128 in all experiments), rather than the distillation dataset size.
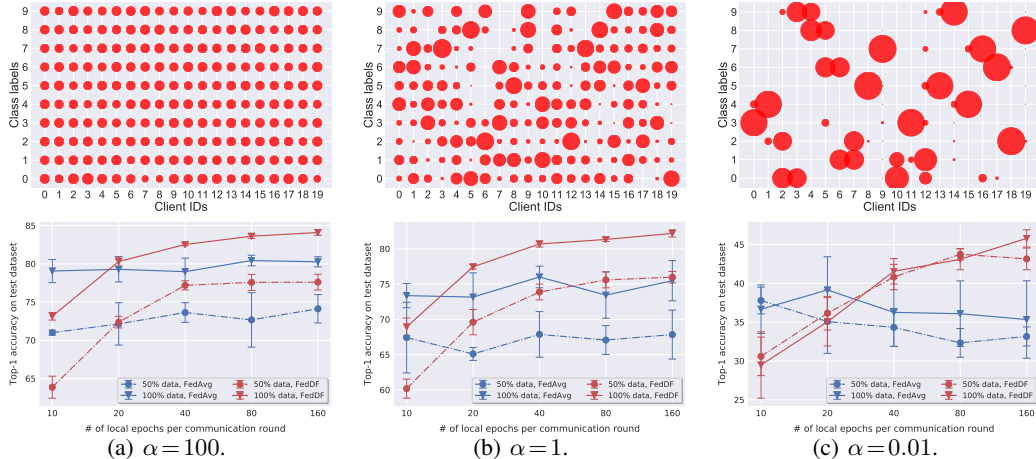
Figure 2: **Top: Illustration of # of samples per class allocated to each client** (indicated by dot sizes), for different Dirichlet distribution α values. **Bottom: Test performance** of **FedDF** and **FEDAVG** on **CIFAR-10** with **ResNet-8**, for different local training settings: non-i.i.d. degrees α, data fractions, and # of local epochs per communication round. We perform 100 communication rounds, and active clients are sampled with ratio $C = 0.4$ from a total of 20 clients. Detailed learning curves in these scenarios can be found in Appendix C.4.

4.2 Evaluation on the Common Federated Learning Settings

Performance overview for different FL scenarios. We can observe from Figure 2 that FedDF consistently outperforms FEDAVG for all client fractions and non-i.i.d. degrees when the local training is reasonably sufficient (e.g. over 40 epochs).

FedDF benefits from larger numbers of local training epochs. This is because the performance of the model ensemble is highly dependent on the diversity among its individual models [40, 67]. Thus longer local training leads to greater diversity and quality of the ensemble and hence a better distillation result for the fused model. This characteristic is desirable in practice as it helps reduce the communication overhead in FL systems. In contrast, the performance of FEDAVG saturates and even degrades with the increased number of local epochs, which is consistent with observations in [51, 8, 74]. As FedDF focuses on better model fusion on the server side, it is orthogonal to recent techniques (e.g. [61, 35, 12]) targeting the issue of non-i.i.d. local data. We believe combining FedDF with these techniques can lead to a more robust FL, which we leave as future work[7].

Ablation study of FedDF. We provide detailed ablation study for FedDF in Appendix C.4.1 to identify the source of the benefits. For example, Table 5 justifies the importance of using the uniformly averaged local models as a starting model (line 6 in Algorithm 1 and line 11 in Algorithm 3), for the quality of ensemble distillation in FedDF. We further investigate the effect of different optimizers (for on-server ensemble distillation) on the federated learning performance in Table 6 and Table 7.

Detailed comparison of FedDF with other SOTA federated learning methods for CV tasks. Table 1 summarizes the results for various degrees of non-i.i.d. data, local training epochs and client sampling fractions. In all scenarios, FedDF requires significantly fewer communication rounds than other SOTA methods to reach designated target accuracies. The benefits of FedDF can be further pronounced by taking more local training epochs as illustrated in Figure 2.

All competing methods have strong difficulties with increasing data heterogeneity (non-i.i.d. data, i.e. smaller α), while FedDF shows significantly improved robustness to data heterogeneity. In most scenarios in Table 1, the reduction of α from 1 to 0.1 almost triples the number of communication rounds for FEDAVG, FEDPROX and FEDAVGM to reach target accuracies, whereas less than twice the number of rounds are sufficient for FedDF.

Increasing the sampling ratio makes a more noticeable positive impact on FedDF compared to other methods. We attribute this to the fact that an ensemble tends to improve in robustness and quality, with a larger number of reasonable good participants, and hence results in better model fusion. Nevertheless, even in cases with a very low sampling fraction (i.e. $C = 0.2$), FedDF still maintains a considerable leading margin over the closest competitor.

---

[7] We include some preliminary results to illustrate the compatibility of FedDF in Table 8 (Appendix C.4.1).

Table 1: **Evaluating different FL methods in different scenarios** (i.e. different client sampling fractions, # of local epochs and target accuracies), in terms of **the number of communication rounds to reach target top-1 test accuracy**. We evaluate on ResNet-8 with CIFAR-10. For each communication round, a fraction C of the total 20 clients are randomly selected. T denotes the specified target top-1 test accuracy. Hyperparameters are fine-tuned for each method (FEDAVG, FEDPROX, and FEDAVGM); FedDF uses the optimal learning rate from FEDAVG. The performance upper bound of (tuned) centralized training is 86% (trained on all local data).

| | Local epochs | The number of communication rounds to reach target performance T | | | | | |
| | | $C=0.2$ | | $C=0.4$ | | $C=0.8$ | |
		$\alpha=1, T=80\%$	$\alpha=0.1, T=75\%$	$\alpha=1, T=80\%$	$\alpha=0.1, T=75\%$	$\alpha=1, T=80\%$	$\alpha=0.1, T=75\%$
FEDAVG	1	350 ± 31	546 ± 191	246 ± 41	445 ± 8	278 ± 83	361 ± 111
	20	144 ± 51	423 ± 105	97 ± 29	309 ± 88	103 ± 26	379 ± 151
	40	130 ± 13	312 ± 87	104 ± 52	325 ± 82	100 ± 76	312 ± 110
FEDPROX	20	99 ± 61	346 ± 12	91 ± 40	235 ± 41	92 ± 21	237 ± 93
	40	115 ± 17	270 ± 96	87 ± 49	229 ± 79	80 ± 44	284 ± 130
FEDAVGM	20	92 ± 15	299 ± 85	92 ± 46	221 ± 29	97 ± 37	235 ± 129
	40	135 ± 52	322 ± 99	78 ± 28	224 ± 38	83 ± 34	232 ± 11
FedDF (ours)	20	$\mathbf{61} \pm 24$	$\mathbf{102} \pm 42$	$\mathbf{28} \pm 10$	$\mathbf{51} \pm 4$	$\mathbf{22} \pm 1$	$\mathbf{33} \pm 18$
	40	$\mathbf{28} \pm 6$	$\mathbf{80} \pm 25$	$\mathbf{20} \pm 4$	$\mathbf{39} \pm 10$	$\mathbf{14} \pm 2$	$\mathbf{20} \pm 4$

Table 2: **The impact of normalization techniques** (i.e. BN, GN) for ResNet-8 on CIFAR (20 clients with $C=0.4$, 100 communication rounds, and 40 local epochs per round). We use a constant learning rate and tune other hyperparameters. The distillation dataset of FedDF for CIFAR-100 is ImageNet (with image size of 32).

| Datasets | | Top-1 test accuracy of different methods | | | | |
		FEDAVG, w/ BN	FEDAVG, w/ GN	FEDPROX, w/ GN	FEDAVGM, w/ GN	**FedDF**, w/ BN
CIFAR-10	$\alpha=1$	76.01 ± 1.53	78.57 ± 0.22	76.32 ± 1.98	77.79 ± 1.22	$\mathbf{80.69} \pm 0.43$
	$\alpha=0.1$	62.22 ± 3.88	68.37 ± 0.50	68.65 ± 0.77	68.63 ± 0.79	$\mathbf{71.36} \pm 1.07$
CIFAR-100	$\alpha=1$	35.56 ± 1.99	42.54 ± 0.51	42.94 ± 1.23	42.83 ± 0.36	$\mathbf{47.43} \pm 0.45$
	$\alpha=0.1$	29.14 ± 1.91	36.72 ± 1.50	35.74 ± 1.00	36.29 ± 1.98	$\mathbf{39.33} \pm 0.03$

Table 3: **Top-1 test accuracy of federated learning CIFAR-10 on VGG-9 (w/o BN)**, for 20 clients with $C=0.4$, $\alpha=1$ and 100 communication rounds (40 epochs per round). We by default drop dummy predictors.

| Methods | Top-1 test accuracy @ communication round | | | | |
	5	10	20	50	100
FEDAVG (w/o drop-worst)	45.72 ± 30.95	51.06 ± 35.56	53.22 ± 37.43	29.60 ± 40.66	7.52 ± 4.29
FEDMA (w/o drop-worst) [1]	23.41 ± 0.00	27.55 ± 0.10	41.56 ± 0.08	60.35 ± 0.03	65.0 ± 0.02
FEDAVG	64.77 ± 1.24	70.28 ± 1.02	75.80 ± 1.36	77.98 ± 1.81	78.34 ± 1.42
FEDPROX	63.86 ± 1.55	71.85 ± 0.75	75.57 ± 1.16	77.85 ± 1.96	78.60 ± 1.91
FedDF	$\mathbf{66.08} \pm 4.14$	$\mathbf{72.80} \pm 1.59$	$\mathbf{75.82} \pm 2.09$	$\mathbf{79.05} \pm 0.54$	$\mathbf{80.36} \pm 0.63$

[1] FEDMA does not support drop-worst operation due to its layer-wise communication/fusion scheme. The number of local training epochs per layer is 5 (45 epochs per model) thus results in stabilized training. More details can be found in Appendix C.2.

Comments on Batch Normalization. Batch Normalization (BN) [31] is the current workhorse in convolutional deep learning tasks and has been employed by default in most SOTA CNNs [20, 27, 48, 69]. However, it often fails on heterogeneous training data. Hsieh *et al*. [24] recently examined the non-i.i.d. data 'quagmire' for distributed learning and point out that replacing BN by Group Normalization (GN) [76] can alleviate some of the quality loss brought by BN due to the discrepancies between local data distributions.

As shown in Table 2, despite additional effort on architecture modification and hyperparameter tuning (i.e. the number of groups in GN), baseline methods with GN replacement still lag much behind FedDF. FedDF provides better model fusion which is robust to non-i.i.d. data, and is compatible with BN, thus avoids extra efforts for modifying the standard SOTA neural architectures. Figure 13 in Appendix C.3 shows the complete learning curves.

We additionally evaluate architectures originally designed without BN (i.e. VGG), to demonstrate the broad applicability of FedDF. Due to the lack of normalization layers, VGG is vulnerable to non-i.i.d. local distributions. We observe that received models on the server might output random prediction results on the validation/test dataset and hence give rise to uninformative results overwhelmed by large variance (as shown in Table 3). We address this issue by a simple treatment[8], "drop-worst", i.e., dropping learners with random predictions on the server validation dataset (e.g. 10% accuracy for CIFAR-10), in each round before applying model averaging and/or ensemble distillation. Table 3 examines the FL methods (FEDAVG, FEDPROX, FEDMA and FedDF) on VGG-9; FedDF consistently outperforms other methods by a large margin for different communication rounds.

---

[8] Techniques (e.g. Krum, Bulyan), can be adapted to further improve the robustness or defend against attacks.

| (a) AG News. | (b) SST2. |

Figure 3: **Federated fine-tuning DistilBERT** on (a) AG News and (b) SST-2. For simplicity, we consider 10 clients with $C = 100\%$ participation ratio and $\alpha = 1$; the number of local training epochs per communication round (10 rounds in total) is set to 10 and 1 respectively. The 50% of the original training dataset is used for the federated fine-tuning (for all methods) and the left 50% is used as the unlabeled distillation dataset for FedDF.

Table 4: **Federated learning with low-precision models (1-bit binarized ResNet-8) on CIFAR-10**. For each communication round (100 in total), 40% of the total 20 clients ($\alpha = 1$) are randomly selected.

Local Epochs	ResNet-8-BN (FEDAVG)	ResNet-8-GN (FEDAVG)	ResNet-8-BN (**FedDF**)
20	44.38 ± 1.21	$\mathbf{59.70} \pm 1.65$	59.49 ± 0.98
40	43.91 ± 3.26	64.25 ± 1.31	$\mathbf{65.49} \pm 0.74$
80	47.62 ± 1.84	65.99 ± 1.29	$\mathbf{70.27} \pm 1.22$

Extension to NLP tasks for federated fine-tuning of DistilBERT. Fine-tuning a pre-trained transformer language model like BERT [13] yields SOTA results on various NLP benchmarks [73, 72]. DistilBERT [60] is a lighter version of BERT with only marginal quality loss on downstream tasks. As a proof of concept, in Figure 3 we consider federated fine-tuning of DistilBERT on non-i.i.d. local data ($\alpha = 1$, depicted in Figure 11). For both AG News and SST2 datasets, FedDF achieves significantly faster convergence than FEDAVG and consistently outperforms the latter.

4.3 Case Studies

Federated learning for low-bit quantized models. FL for the Internet of Things (IoT) involves edge devices with diverse hardware, e.g. different computational capacities. Network quantization is hence of great interest to FL by representing the activations/weights in low precision, with benefits of significantly reduced local computational footprints and communication costs. Table 4 examines the model fusion performance for binarized ResNet-8 [57, 30]. FedDF can be on par with or outperform FEDAVG by a noticeable margin, without introducing extra GN tuning overheads.

Federated learning on heterogeneous systems. Apart from non-i.i.d. local distributions, another major source of heterogeneity in FL systems manifests in neural architectures [41]. Figure 4 visualizes the training dynamics of FedDF and FEDAVG[9] in a heterogeneous system with three distinct architectures, i.e., ResNet-20, ResNet-32, and ShuffleNetV2. On CIFAR-10/100 and ImageNet, FedDF dominates FEDAVG on test accuracy in each communication round with much less variance. Each fused model exhibits marginal quality loss compared to the ensemble performance, which suggests unlabeled datasets from other domains are sufficient for model fusion. Besides, the gap between the fused model and the ensemble one widens when the training dataset contains a much larger number of classes[10] than that of the distillation dataset. For instance, the performance gap is negligible on CIFAR-10, whereas on ImageNet, the gap increases to around 6%. In Section 5, we study this underlying interaction between training data and unlabeled distillation data in detail.

5 Understanding FedDF

FedDF consists of two chief components: ensembling and knowledge distillation via out-of-domain data. In this section, we first investigate what affects the ensemble performance on the global distribution (test domain) through a generalization bound. We then provide empirical understanding of how different attributes of the out-of-domain distillation dataset affect the student performance on the global distribution.

---

[9] Model averaging is only performed among models with identical structures.

[10] # of classes is a proxy measurement for distribution shift; labels are not used in our distillation procedure.

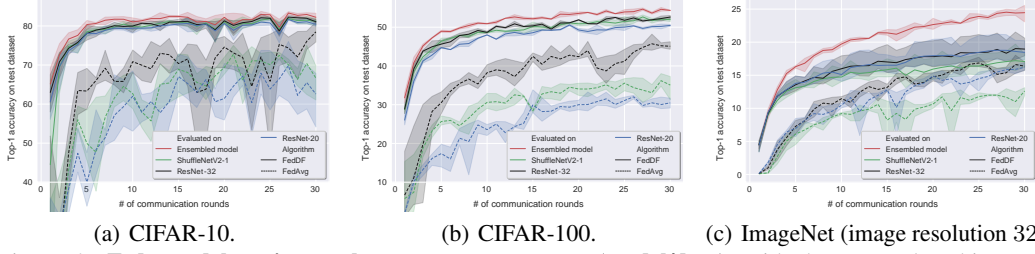| (a) CIFAR-10. | (b) CIFAR-100. | (c) ImageNet (image resolution 32). |

Figure 4: **Federated learning on heterogeneous systems (model/data)**, with three neural architectures (ResNet-20, ResNet-32, ShuffleNetV2) and non-i.i.d. local data distribution ($\alpha = 1$). We consider 21 clients for CIFAR (client sampling ratio $C = 0.4$) and 150 clients for ImageNet ($C = 0.1$); different neural architectures are evenly distributed among clients. We train 80 local training epochs per communication round (total 30 rounds). CIFAR-100, STL-10, and STL-10 are used as the distillation datasets for CIFAR-10/100 and ImageNet training respectively. The *solid* lines show the results of FedDF for a given communication round, while *dashed* lines correspond to that of FEDAVG; *colors* indicate model architectures.

| (a) CIFAR-10. | (b) CIFAR-10 (40 local epochs). | (c) CIFAR-100 (40 local epochs). |

Figure 5: **The performance of FedDF on different distillation datasets**: random uniformly sampled noises, randomly generated images (from the generator), CIFAR, downsampled ImageNet32, and downsampled STL-10. We evaluate ResNet-8 on CIFAR for 20 clients, with $C = 0.4$, $\alpha = 1$ and 100 communication rounds.

(a) The fusion performance of FedDF through unlabeled ImageNet, for different numbers of classes.

(b) The performance of FedDF via unlabeled ImageNet (100 classes), for different data fractions.

(c) The fusion performance of FedDF under different numbers of distillation steps.

Figure 6: **Understanding knowledge distillation behaviors of FedDF** on **# of classes** (6(a)), **sizes of the distillation dataset** (6(b)), and **# of distillation steps** (6(c)), for federated learning ResNet-8 on CIFAR-100, with $C = 0.4$, $\alpha = 1$ and 100 communication rounds (40 local epochs per round). ImageNet with image resolution 32 is considered as our base unlabeled dataset. For simplicity, only classes without overlap with CIFAR-100 classes are considered, in terms of the synonyms, hyponyms, or hypernyms of the class name.

Generalization bound.   Theorem 5.1 provides insights into ensemble performance on the global distribution. Detailed description and derivations are deferred to Appendix D.

Theorem 5.1 (informal)**.** *We denote the global distribution as $\mathcal{D}$, the k-th local distribution and its empirical distribution as $\mathcal{D}_k$ and $\hat{\mathcal{D}}_k$ respectively. The hypothesis $h \in \mathcal{H}$ learned on $\hat{\mathcal{D}}_k$ is denoted by $h_{\hat{\mathcal{D}}_k}$. The upper bound on the risk of the ensemble of K local models on $\mathcal{D}$ mainly consists of 1) the empirical risk of a model trained on the global empirical distribution $\hat{\mathcal{D}} = \frac{1}{K} \sum_k \hat{\mathcal{D}}_k$, and 2) terms dependent on the distribution discrepancy between $\mathcal{D}_k$ and $\mathcal{D}$, with the probability $1 - \delta$:*

$$L_{\mathcal{D}}\Big(\tfrac{1}{K} \textstyle\sum_k h_{\hat{\mathcal{D}}_k} \Big) \le L_{\hat{\mathcal{D}}}(h_{\hat{\mathcal{D}}}) + \frac{1}{K} \sum_k \left(\frac{1}{2} d_{\mathcal{H} \Delta \mathcal{H}}(\mathcal{D}_k, \mathcal{D}) + \lambda_k \right) + \sqrt{\frac{\log \frac{2K}{\delta}}{2m}} \,,$$

where $d_{\mathcal{H} \Delta \mathcal{H}}$ measures the distribution discrepancy between two distributions [3], m is the number of samples per local distribution, and λ_k is the minimum of the combined loss $\mathcal{L}_{\mathcal{D}}(h) + \mathcal{L}_{\mathcal{D}_k}(h)$, $\forall h \in \mathcal{H}$.

The ensemble of the local models sets the performance upper bound for the later distilled model on the global distribution as shown in Figure 4. Theorem 5.1 shows that compared to a model trained

on the global empirical distribution (ideal centralized case), the performance of the ensemble on the global distribution is associated with the discrepancy between local distributions $\mathcal{D}_k$'s and the global distribution $\mathcal{D}$. Besides, the shift between the distillation and the global distribution determines the knowledge transfer quality between these two distributions and hence the test performance of the fused model. In the following, we empirically examine how the choice of distillation data distributions and the number of distillation steps influence the quality of ensemble knowledge distillation.

Source, diversity and size of the distillation dataset. The fusion in FedDF demonstrates remarkable consistency across a wide range of realistic data sources as shown in Figure 5, although an abrupt performance declination is encountered when the distillation data are sampled from a dramatically different manifold (e.g. random noise). Notably, synthetic data from the generator of a pre-trained GAN does not incur noticeable quality loss, opening up numerous possibilities for effective and efficient model fusion. Figure 6(a) illustrates that in general the diversity of the distillation data does not significantly impact the performance of ensemble distillation, though the optimal performance is achieved when two domains have a similar number of classes. Figure 6(b) shows the FedDF is not demanding on the distillation dataset size: even 1% of data ($\sim 48\%$ of the local training dataset) can result in a reasonably good fusion performance.

Distillation steps. Figure 6(c) depicts the impact of distillation steps on fusion performance, where FedDF with a moderate number of the distillation steps is able to approach the optimal performance. For example, 100 distillation steps in Figure 6(c), which corresponds to 5 local epochs of CIFAR-100 (partitioned by 20 clients), suffice to yield satisfactory performance. Thus FedDF introduces minor time-wise expense.

Broader Impact

We believe that collaborative learning schemes such as federated learning are an important element towards enabling privacy-preserving training of ML models, as well as a better alignment of each individual's data ownership with the resulting utility from jointly trained machine learning models, especially in applications where data is user-provided and privacy sensitive [34, 55].
In addition to privacy, efficiency gains and lower resource requirements in distributed training reduce the environmental impact of training large machine learning models. The introduction of a practical and reliable distillation technique for heterogeneous models and for low-resource clients is a step towards more broadly enabling collaborative privacy-preserving and efficient decentralized learning.

Acknowledgements

References

[1] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9163–9171, 2019.

[2] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

[3] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[4] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

[5] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander. Towards federated learning at scale: System design, 2019.

[6] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[7] C. Buciluă, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[8] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[9] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

[10] H.-Y. Chen and W.-L. Chao. Feddistill: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

[11] P. Chrabaszcz, I. Loshchilov, and F. Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[12] Y. Deng, M. M. Kamani, and M. Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] N. Dvornik, C. Schmid, and J. Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[15] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

[16] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[18] N. Guha, A. Talwlkar, and V. Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

[19] C. He, S. Avestimehr, and M. Annavaram. Group knowledge transfer: Collaborative training of large cnns on the edge. In *Advances in Neural Information Processing Systems*, 2020.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] G. Hinton. Neural networks for machine learning, 2012.

[22] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[23] J. Hoffman, M. Mohri, and N. Zhang. Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems*, pages 8246–8256, 2018.

[24] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons. The non-iid data quagmire of decentralized machine learning. *arXiv preprint arXiv:1910.00189*, 2019.

[25] T.-M. H. Hsu, H. Qi, and M. Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[26] T.-M. H. Hsu, H. Qi, and M. Brown. Federated visual classification with real-world data distribution. In *European Conference on Computer Vision (ECCV)*, 2020.

[27] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[28] Z. Huang and N. Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

[29] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[30] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[31] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[32] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In *Appears at the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

[33] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[34] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning, 2019.

[35] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for on-device federated learning. *arXiv preprint arXiv:1910.06378*, 2019.

[36] J. Kim, S. Park, and N. Kwak. Paraphrasing complex network: Network compression via factor transfer. In *Advances in Neural Information Processing Systems*, pages 2760–2769, 2018.

[37] A. Koratana, D. Kang, P. Bailis, and M. Zaharia. LIT: Learned intermediate representation training for model compression. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3509–3518, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[38] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[40] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

[41] D. Li and J. Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[42] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

[43] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[44] T. Li, M. Sanjabi, A. Beirami, and V. Smith. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2020.

[45] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi. Dynamic model pruning with feedback. In *International Conference on Learning Representations*, 2020.

[46] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi. Don't use large mini-batches, use local SGD. In *ICLR - International Conference on Learning Representations*, 2020.

[47] I.-J. Liu, J. Peng, and A. G. Schwing. Knowledge flow: Improve upon your teachers. *arXiv preprint arXiv:1904.05878*, 2019.

[48] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[49] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13153–13164, 2019.

[50] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation with multiple sources. In *Advances in neural information processing systems*, pages 1041–1048, 2009.

[51] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[52] P. Micaelli and A. J. Storkey. Zero-shot knowledge transfer via adversarial belief matching. In *Advances in Neural Information Processing Systems*, pages 9547–9557, 2019.

[53] M. Mohri, G. Sivek, and A. T. Suresh. Agnostic federated learning. *arXiv preprint arXiv:1902.00146*, 2019.

[54] G. K. Nayak, K. R. Mopuri, V. Shaj, R. V. Babu, and A. Chakraborty. Zero-shot knowledge distillation in deep networks. *arXiv preprint arXiv:1905.08114*, 2019.

[55] A. Nedic. Distributed gradient methods for convex machine learning problems in networks: Distributed optimization. *IEEE Signal Processing Magazine*, 37(3):92–101, 2020.

[56] S. Park and N. Kwak. Feed: Feature-level ensemble for knowledge distillation. *arXiv preprint arXiv:1909.10754*, 2019.

[57] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[58] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[59] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015.

[60] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[61] N. Shoham, T. Avidor, A. Keren, N. Israel, D. Benditkis, L. Mor-Yosef, and I. Zeitak. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

[62] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[63] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[64] S. P. Singh and M. Jaggi. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*, 2020.

[65] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[66] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics.

[67] P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. In *Advances in neural information processing systems*, pages 190–196, 1996.

[68] L. Sun and L. Lyu. Federated model distillation with noise-free differential privacy. *arXiv preprint arXiv:2009.05537*, 2020.

[69] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[70] Y. Tian, D. Krishnan, and P. Isola. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*, 2019.

[71] F. Tung and G. Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1365–1374, 2019.

[72] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3261–3275, 2019.

[73] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics.

[74] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020.

[75] A. Wu, W. Zheng, X. Guo, and J. Lai. Distilled person re-identification: Towards a more scalable system. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[76] Y. Wu and K. He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[77] S. You, C. Xu, C. Xu, and D. Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1285–1294, New York, NY, USA, 2017. Association for Computing Machinery.

[78] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, T. N. Hoang, and Y. Khazaeni. Bayesian nonparametric federated learning of neural networks. *arXiv preprint arXiv:1905.12022*, 2019.

[79] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[80] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[81] Y. Zhou, G. Pu, X. Ma, X. Li, and D. Wu. Distilled one-shot federated learning. *2009.07999*, 2020.

A   Detailed Related Work Discussion

Prior work.   We first comment on the two close approaches (FedMD and Cronus), in order to address 1) Distinctions between FedDF and prior work, 2) Privacy/Communication traffic concerns, 3) Omitted experiments on FedMD and Cronus.

- Distinctions between FedDF and prior work. As discussed in the related work, most SOTA FL methods directly manipulate received model parameters (e.g. FedAvg/FedAvgM/FedMA). To our best knowledge, FedMD and Cronus are the only two that utilize logits information (of neural nets) for FL. The distinctions from them are made below.
- Different objectives and evaluation metrics. Cronus is designed for robust FL under poisoning attack, whereas FedMD is for personalized FL. In contrast, FedDF is intended for on-server model aggregation (evaluation on the aggregated model), whereas neither FedMD nor Cronus aggregates the model on the server.
- Different Operations.
  1. FedDF, like FedAvg, *only* exchanges models between the server and clients, without transmitting input data. In contrast, FedMD and Cornus rely on exchanging public data logits. As FedAvg, FedDF can include privacy/security extensions and has the same communication cost per round.
  2. FedDF performs ensemble distillation with unlabeled data *on the server*. In contrast, FedMD/Cronus use averaged logits received from the server for *local client training*.
- Omitted experiments with FedMD/Cronus.
  1. FedMD requires to locally pre-train on the *labeled* public data, thus the model classifier necessitates an output dimension of # of public classes *plus* # of private classes (c.f. the output dimension of # of private classes in other FL methods). We cannot compare FedMD with FedDF with the same architecture (classifier) to ensure fairness.
  2. Cronus is shown to be consistently worse than FedAvg in normal FL (i.e. no attack case) in their Tab. IV & VI.
  3. Different objectives/metrics argued above. We thoroughly evaluated SOTA baselines with the same objective/metric.

Contemporaneous work.   We then detail some contemporaneous work, e.g. [68, 10, 81, 19]. [68] slightly extends FedMD by adding differential privacy. In [81], the server aggregates the synthetic data distilled from clients' private dataset, which in turn uses for one-shot on-server learning. He *et al* [19] improve FL for resource-constrained edge devices by combing FL with Split Learning (SL) and knowledge distillation: edge devices train compact feature extractor through local SGD and then synchronize extracted features and logits with the server, while the server (asynchronously) uses the latest received features and logits to train a much larger server-side CNN. The knowledge distillation is used on both the server and clients to improve the optimization quality.

FedDistill [10] is very similar to us, where it resorts to stochastic weight average-Gaussian (SWAG) [49] and the ensemble distillation is achieved via cyclical learning rate schedule with SWA [32]. In Table 7 below, we empirically compare our FedDF with this contemporaneous work (i.e. FedDistill).

B   Algorithmic Description

Algorithm 2 below details a general training procedure on local clients. The local update step of FEDPROX corresponds to adding a proximal term (i.e. $\eta \frac{\partial \frac{\mu}{2} \left\| \mathbf{x}_t^k - \mathbf{x}_{t-1}^k \right\|_2^2}{\partial \mathbf{x}_t^k}$) to line 5.

Algorithm 3 illustrates the model fusion of FedDF for the FL system with heterogeneous model prototypes. The schematic diagram is presented in Figure 7. To perform model fusion in such heterogeneous scenarios, FedDF constructs several prototypical models on the server. Each prototype represents all clients with identical architecture/size/precision etc.

Algorithm 2 Illustration of local client update in FEDAVG. The K clients are indexed by k; $\mathcal{P}_k$ indicates the set of indexes of data points on client k, and $n_k = |\mathcal{P}_k|$. E is the number of local epochs, and η is the learning rate. ℓ evaluates the loss on model weights for a mini-batch of an arbitrary size.

1: **procedure** CLIENT-LOCALUPDATE($k, \mathbf{x}_{t-1}^k$)
2:     Client k receives $\mathbf{x}_{t-1}^k$ from server and copies it as $\mathbf{x}_t^k$
3:     **for** each local epoch i from 1 to E **do**
4:        **for** mini-batch $b \subset \mathcal{P}_k$ **do**
5:           $\mathbf{x}_t^k \leftarrow \mathbf{x}_t^k - \eta \frac{\partial \ell(\mathbf{x}_t^k; b)}{\partial \mathbf{x}_t^k}$ $\triangleright$ can be arbitary optimizers (e.g. Adam)
6:     **return** $\mathbf{x}_t^k$ to server

Algorithm 3 Illustration of FedDF for heterogeneous FL systems. The K clients are indexed by k, and n_k indicates the number of data points for the k-th client. The number of communication rounds is T, and C controls the client participation ratio per communication round. The number of total iterations used for model fusion is denoted as N. The distinct model prototype set $\mathcal{P}$ has p model prototypes, with each initialized as $\mathbf{x}_0^P$.

1: **procedure** SERVER
2:     initialize HashMap $\mathcal{M}$: map each model prototype P to its weights $\mathbf{x}_0^P$.
3:     initialize HashMap $\mathcal{C}$: map each client to its model prototype.
4:     initialize HashMap $\tilde{\mathcal{C}}$: map each model prototype to the associated clients.
5:     **for** each communication round $t = 1, \ldots, T$ **do**
6:        $\mathcal{S}_t \leftarrow$ a random subset (C fraction) of the K clients
7:        **for** each client $k \in \mathcal{S}_t$ **in parallel do**
8:           $\hat{\mathbf{x}}_t^k \leftarrow$ Client-LocalUpdate($k, \mathcal{M}[\mathcal{C}[k]]$) $\triangleright$ detailed in Algorithm 2.
9:        **for** each prototype $P \in \mathcal{P}$ **in parallel do**
10:           initialize the client set $\mathcal{S}_t^P$ with model prototype P, where $\mathcal{S}_t^P \leftarrow \tilde{\mathcal{C}}[P] \cap \mathcal{S}_t$
11:           initialize for model fusion $\mathbf{x}_{t,0}^P \leftarrow \sum_{k \in \mathcal{S}_t^P} \frac{n_k}{\sum_{k \in \mathcal{S}_t^P} n_k} \hat{\mathbf{x}}_t^k$
12:           **for** j in $\{1, \ldots, N\}$ **do**
13:              sample $\mathbf{d}$, from e.g. (1) an unlabeled dataset, (2) a generator
14:              use ensemble of $\{\hat{\mathbf{x}}_t^k\}_{k \in \mathcal{S}_t}$ to update server student $\mathbf{x}_{t,j}^P$ through AVGLOGITS
15:           $\mathcal{M}[P] \leftarrow \mathbf{x}_{t,N}^P$
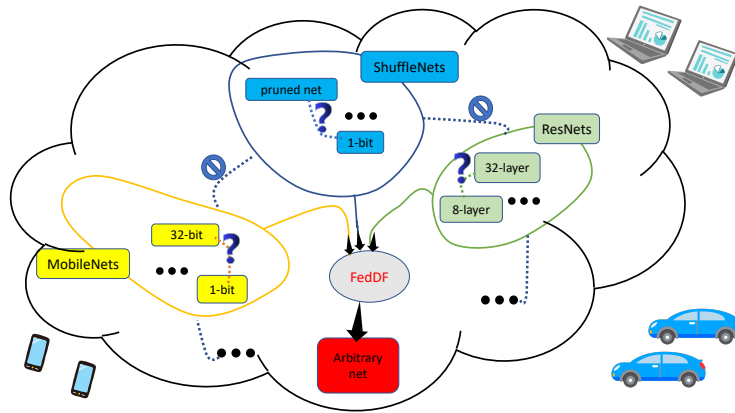16:   **return** $\mathcal{M}$

Figure 7: **The schematic diagram for heterogeneous model fusion.** We use dotted lines to indicate model parameter averaging FL methods such as FEDAVG. We could notice the architectural/precision discrepancy invalidates these methods in heterogeneous FL systems. However, FedDF could aggregate knowledge from all available models without hindrance.

C   Additional Experimental Setup and Evaluations

C.1   Detailed Description for Toy Example (Figure 1)

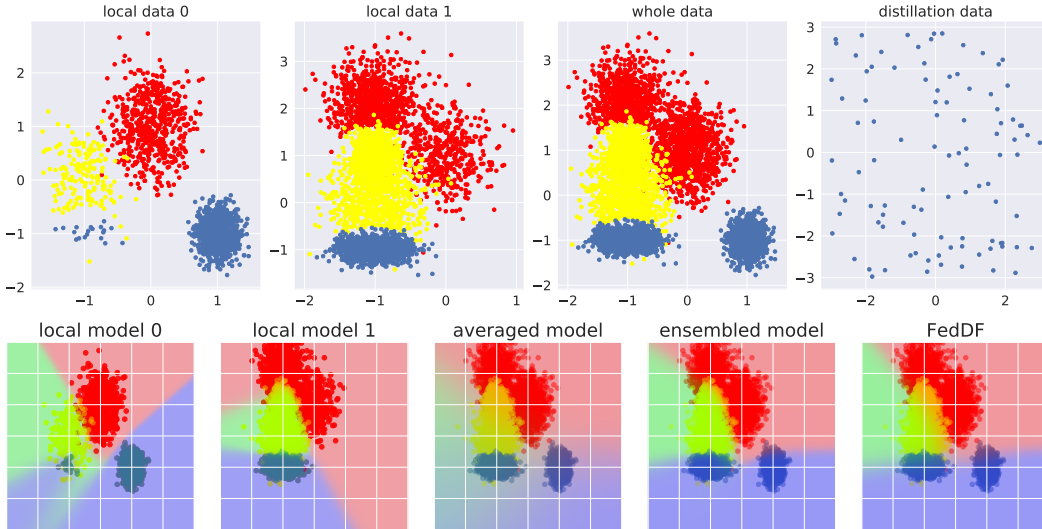Figure 8 provides a detailed illustration of the limitation in FEDAVG.

Figure 8: **The limitation of FEDAVG.** We consider a toy example of a 3-class classification task with a 3-layer MLP, and display the decision boundaries (probabilities over RGB channels) on the input space. We illustrate the used datasets in the **top** row; the distillation dataset consists of 60 data points, with each uniformly sampled from the range of $(-3, 3)$. In the **bottom** row, the left two figures consider the individually trained local models. The right three figures evaluate aggregated models and the global data distribution; the averaged model (FEDAVG) results in much blurred decision boundaries.

C.2   Detailed Experiment Setup

The detailed hyperparameter tuning procedure.   The tuning procedure of hyperparameters ensures that the best hyperparameter lies in the middle of our search grids; otherwise, we extend our search grid. The initial search grid of learning rate is $\{1.5, 1, 0.5, 0.1, 0.05, 0.01\}$. The initial search grid of proximal factor in FEDPROX is $\{0.001, 0.01, 0.1, 1\}$. The initial search grid of momentum factor β in FEDAVGM is $\{0.1, 0.2, 0.3, 0.4\}$; the update scheme of FEDAVGM follows $\Delta \mathbf{v} := \beta \mathbf{v} + \Delta \mathbf{x} ; \mathbf{x} := \mathbf{x} - \Delta \mathbf{v}$, where $\Delta \mathbf{x}$ is the model difference between the updated local model and the sent global model, for previous communication round.

Unless mentioned (i.e. Table 1), otherwise the learning rate is set to 0.1 for ResNet like architectures (e.g. ResNet-8, ResNet-20, ResNet-32, ShuffleNetV2), 0.05 for VGG and $1e-5$ for DistilBERT. When comparing with other methods, e.g. FEDPROX, FEDAVGM, we always tune their corresponding hyperparameters (e.g. proximal factor in FEDPROX and momentum factor in FEDAVGM).

Experiment details of FEDMA.   We detail our attempts of reproducing FEDMA experiments on VGG-9 with CIFAR-10 in this section. We clone their codebase from GitHub and add functionality to sample clients after synchronizing the whole model.

Different from other methods evaluated in the paper, FEDMA uses a layer-wise local training scheme. For each round of the local training, the involved clients only update the model parameters from one specific layer onwards, while the already matched layers are frozen. The fusion (matching) is only performed on the chosen layer. Such a layer is gradually chosen from the bottom layer to the top layer, following a bottom-up fashion [74]. One complete model update cycle of FEDMA requires more frequent (but slightly cheaper) communication, which is equivalent to the number of layers in the neural network.

In our experiments of FEDMA, the number of local training epochs is 5 epochs per layer (45 epochs per model update), which is slightly larger than 40 epochs used by other methods. We ensure a similar[11] number of model updates in terms of the whole model. We consider global-wise learning

---

[11] The other methods use 40 local training epochs per whole model update. Given the fact of layer-wise training scheme in FEDMA, as well as the used 9-layer VGG (same as the one used in [74] and we are unable to

rate, different from the layer-wise one in Wang et al. [74]. We also turn off the momentum and weight decay during the local training for a consistent evaluation. The implementation of VGG-9 follows `https://github.com/kuangliu/pytorch-cifar/`.

The detailed experimental setup for FedDF (low-bit quantized models). FedDF increases the feasibility of robust model fusion in FL for binarized ResNet-8. As stated in Table 4 (Section 4.3), we employ the "Straight-through estimator" [4, 21, 29, 30] or the "error-feedback" [45] to simulate the on-device local training of the binarized ResNet-8. For each communication round, the server of the FL system will receive locally trained and binarized ResNet-8 from activated clients. The server will then distill the knowledge of these low-precision models to a full-precision one[12] and broadcast to newly activated clients for the next communication round. For the sake of simplicity, the case study demonstrated in the paper only considers reducing the communication cost (from clients to the server), and the local computational cost; a thorough investigation on how to perform a communication-efficient and memory-efficient FL is left as future work.

The synthetic formulation of non-i.i.d. client data. Assume every client training example is drawn independently with class labels following a categorical distribution over M classes parameterized by a vector $\mathbf{q}$ ($q_i \geq 0, i \in [1, M]$ and $\|\mathbf{q}\|_1 = 1$). Following the partition scheme introduced and used in [78, 25][13], to synthesize client non-i.i.d. local data distributions, we draw $\alpha \sim \text{Dir}(\alpha\mathbf{p})$ from a Dirichlet distribution, where $\mathbf{p}$ characterizes a prior class distribution over M classes, and $\alpha > 0$ is a concentration parameter controlling the identicalness among clients. With $\alpha \to \infty$, all clients have identical distributions to the prior; with $\alpha \to 0$, each client holds examples from only one random class.

To better understand the local data distribution for the datasets we considered in the experiments, we visualize the partition results of CIFAR-10 and CIFAR-100 on $\alpha = \{0.01, 0.1, 0.5, 1, 100\}$ for 20 clients, in Figure 9 and Figure 10, respectively.

In Figure 11 we visualize the partitioned local data on 10 clients with $\alpha = 1$, for AG News and SST-2.

(a) $\alpha = 100$ (b) $\alpha = 1$ (c) $\alpha = 0.5$
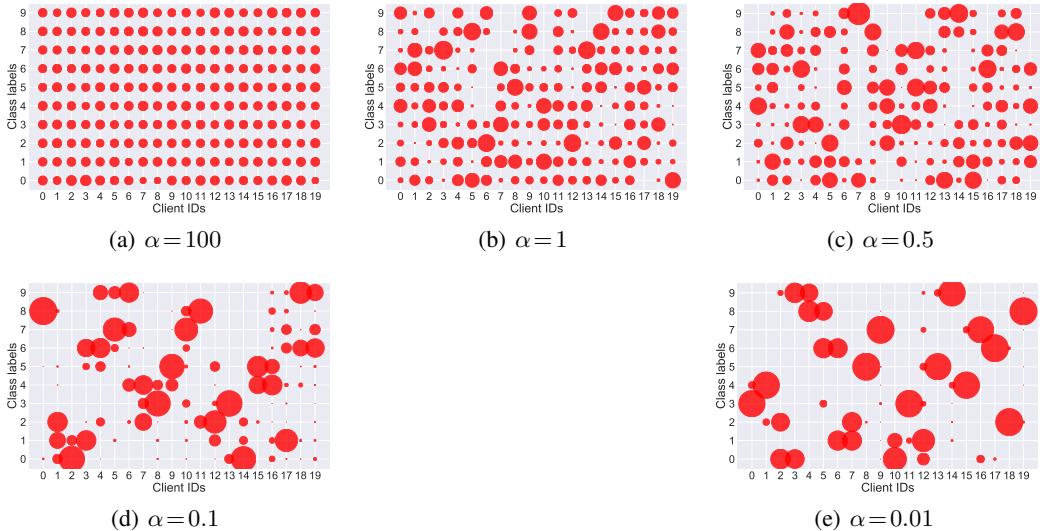
(d) $\alpha = 0.1$ (e) $\alpha = 0.01$
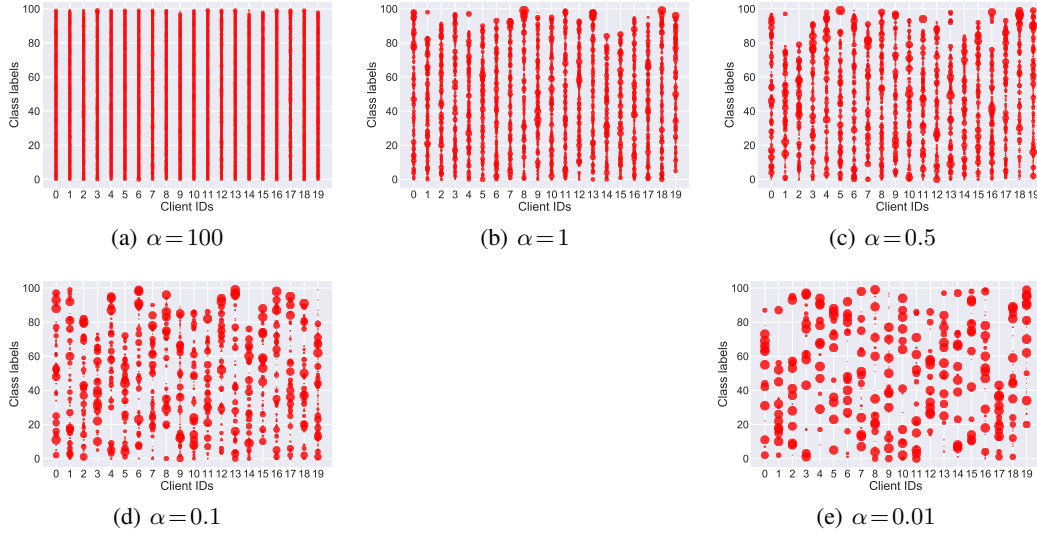
Figure 9: Classes allocated to each client at different Dirichlet distribution alpha values, for CIFAR-10 with 20 clients. The size of each dot reflects the magnitude of the samples number.

C.3 Some Empirical Understanding of FEDAVG

Figure 12 reviews the general behaviors of FEDAVG under different non-iid degrees of local data, different local data sizes, different numbers of local epochs per communication round, as well as the learning rate schedule during the local training. Since we cannot observe the benefits of decaying the

---

adapt their code to other architectures due to their hard-coded architecture manipulations), we decide to slightly increase the number of local epochs per layer for FEDMA.

[12] The training of the binarized network requires to maintain a full-precision model [29, 30, 45] for model update (quantized/pruned model is used during the backward pass).

[13] We heavily borrowed the partition description of [25] for the completeness of the paper.

Figure 10: Classes allocated to each client at different Dirichlet distribution alpha values, for CIFAR-100 with 20 clients. The size of each dot reflects the magnitude of the samples number.
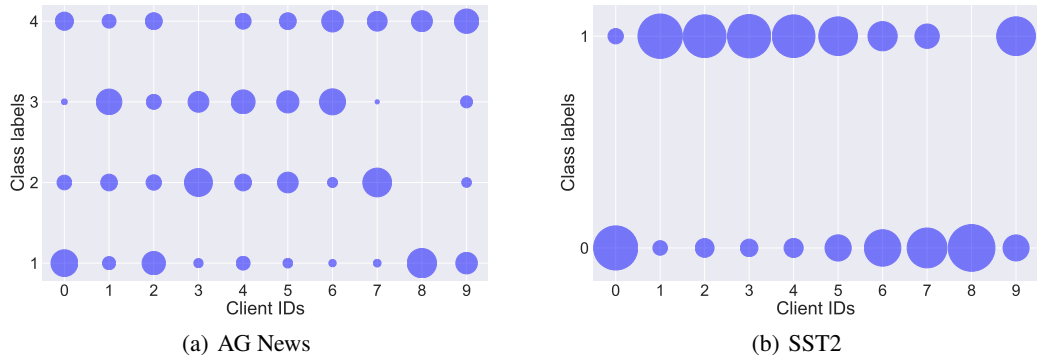

Figure 11: Classes allocated to each client at Dirichlet distribution $\alpha = 1$, for AG News and SST2 datasets with 10 clients. The size of each dot reflects the magnitude of the samples number.

learning rate during the local training phase, we turn off the learning rate decay for the experiments in the main text.

In Figure 13, we visualize the learning curves of training ResNet-8 on CIFAR-10 with different normalization techniques. The numerical results correspond to Table 2 in the main text.

Figure 12: **The ablation study of FEDAVG for different # of local epochs and learning rate schedules**, for standard federated learning on CIFAR-10 with ResNet-8. For each communication round (100 in total), 40% of the total 20 clients are randomly selected. We use α to synthetically control the non-iid degree of the local data, as in [78, 25]. The smaller α, the larger discrepancy between local data distributions ($\alpha = 100$ mimics identical local data distributions). We report the top-1 accuracy (on three different seeds) on the test dataset.

Figure 13: The impact of different normalization techniques, i.e., Batch Normalization (BN), Group Normalization (GN), for federated learning on CIFAR-10 with ResNet-8 with $\alpha = 1$. For each communication round (100 in total), 40% of the total 20 clients are randomly selected for 40 local epochs.

C.4 The Advantages of FedDF

C.4.1 Ablation Study

The Importance of the Model Initialization in FedDF. We empirically study the importance of the initialization (before performing ensemble distillation) in FedDF. Table 5 demonstrates the performance difference of FedDF for two different model initialization schemes: 1) "from average", where the uniformly averaged model from this communication round is used as the initial model (i.e. the default design choice of FedDF as illustrated in Algorithm 1 and Algorithm 3); and 2) "from previous", where we initialize the model for ensemble distillation by utilizing the fusion result of FedDF from the previous communication round. The noticeable performance differences illustrated in Table 5 identify the importance of using the uniformly averaged model[14] (from the current communication round) as a starting model for better ensemble distillation.

Table 5: **Understanding the importance of model initialization in FedDF**, on CIFAR-10 with ResNet-8. For each communication round (100 in total), 40% of the total 20 clients are randomly selected. The scheme "from average" indicates initializing the model for ensemble distillation from the uniformly averaged model of this communication round; while the scheme "from previous" instead uses the fused model from the previous communication round as the starting point. We report the top-1 accuracy (on three different seeds) on the test dataset.

local training epochs	$\alpha = 1$		$\alpha = 0.1$	
	from average	from previous	from average	from previous
40	80.43 ± 0.37	74.13 ± 0.91	71.84 ± 0.86	62.94 ± 1.12
80	81.17 ± 0.53	76.37 ± 0.60	74.73 ± 0.65	67.88 ± 0.90

The performance gain in FedDF. To distinguish the benefits of FedDF from the small learning rate (during the local training) or Adam optimizer (used for ensemble distillation in FedDF), we report the results of using Adam (lr=1e-3) for both local training and model fusion (over three seeds), on CIFAR-10 with ResNet-8, in Table 6. Improving the local training through Adam might help Federated Learning but the benefit vanishes with higher data heterogeneity (e.g. $\alpha = 0.1$). Performance gain from FedDF is robust to data heterogeneity and also orthogonal to effects of learning rates and Adam.

Table 7 examines the effect of different optimization schemes on the quality of ensemble distillation. We can witness that with two extra hyper-parameters (sampling scale for SWAG and the number of models to be sampled), SWAG can slightly improve the distillation performance. In contrast, we use Adam with default hyper-parameters as our design choice in FedDF: it demonstrates similar performance (compared to the choice of SWAG) with trivial tuning overhead.

The compatibility of FedDF with other methods. Table 8 justifies the compatibility of FedDF. Our empirical results demonstrate a significant performance gain of FedDF over the FEDAVG, even

---

[14] The related preprints [41, 9] are closer to the second initialization scheme. They do not or cannot introduce the uniformly averaged model (on the server) into the federated learning pipeline; instead, they only utilize the averaged logits (on the same data) for each client's local training.

Table 6: **Understanding the impact of local training quality**, on CIFAR-10 with ResNet-8. For each communication round (100 in total), 40% of the total 20 clients are randomly selected for 40 local epochs. We report the top-1 accuracy (on three different seeds) on the test dataset.

	$\alpha=1$		$\alpha=0.1$	
local client training scheme	FedDF	FEDAVG	FedDF	FEDAVG
SGD	80.27	72.73	71.52	62.44
Adam	83.32	78.13	72.58	62.53

Table 7: **On the impact of using different optimizers for ensemble distillation** in FedDF, on CIFAR-10 with ResNet-8. For each communication round (100 in total), 40% of the total 20 clients are randomly selected for 40 local epochs. We report the top-1 accuracy (on three different seeds) on the test dataset. "SGD" uses the same learning rate scheduler as our "Adam" choice (i.e. cosine annealing), and with fine-tuned initial learning rate. "SWAG" refers to the mechanism to form an approximated posterior distribution [49] where more models can be sampled from, and [10] further propose to use SWAG on the received client models for better ensemble distillation; our default design resorts to directly averaged logits from received local clients with Adam optimizer. To ensure a fair comparison, we use the same distillation dataset as in FedDF (i.e., CIFAR-100) for "SWAG" [10]. We fine-tune other hyper-parameters in "SWAG": we use all received client models and 10 sampled models from Gaussian distribution (as suggested in [10]) for the ensemble distillation.

	$\alpha=1$		$\alpha=0.1$	
optimizer used on the server	FedDF	FEDAVG	FedDF	FEDAVG
SGD	76.68	72.73	57.33	62.44
Adam (our default design)	80.27	72.73	71.52	62.44
SWAG [49, 10]	80.84	72.73	72.40	62.44

in the case of using local proximal regularizer to avoid catastrophically over-fitting the heterogeneous local data, which reduces the diversity of local models that FedDF benefits from.

Table 8: **The compatibility of FedDF with other training schemes**, on CIFAR-10 with ResNet-8. For each communication round (100 in total), 40% of the total 20 clients are randomly selected for 40 local epochs. We consider the fine-tuned proximal penalty from FedDF. We report the top-1 accuracy (on three different seeds) on the test dataset.

	$\alpha=1$		$\alpha=0.1$	
local client training scheme	FedDF	FEDAVG	FedDF	FEDAVG
SGD	80.27	72.73	71.52	62.44
SGD + proximal penalty	80.56	76.11	71.64	62.53

C.4.2 Comparison with FEDAVG

Figure 14 complements Figure 2 in the main text and presents a thorough comparison between FEDAVG and FedDF, for a variety of different local training epochs, data fractions, non-i.i.d. degrees. The detailed learning curves of the cases in this figure are visualized in Figure 15, Figure 16, and Figure 17.
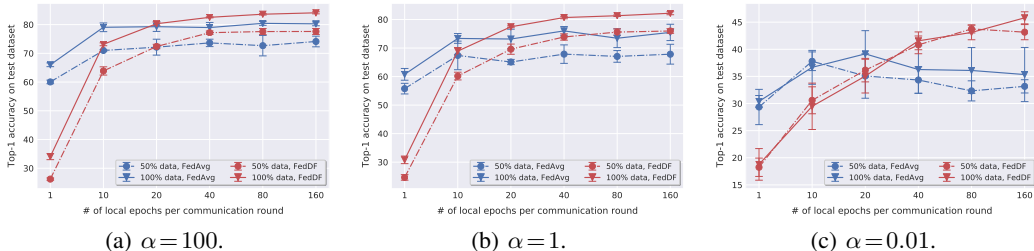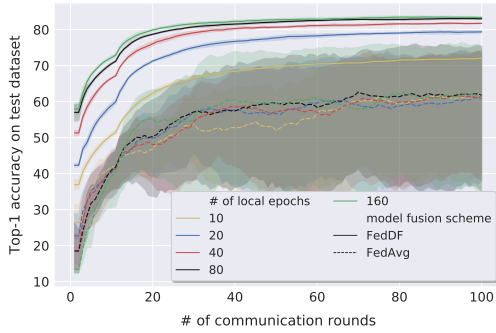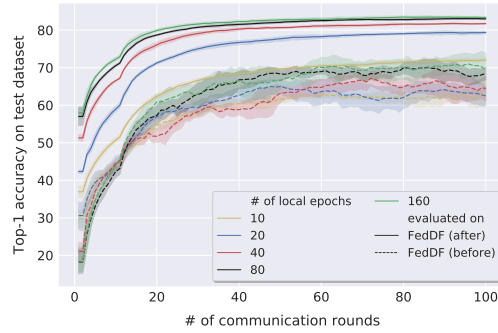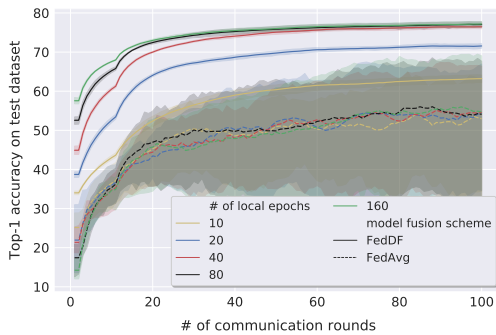

Figure 14: The **test performance** of **FedDF** and **FEDAVG** on **CIFAR-10** with **ResNet-8**, for different local data non-iid degrees α, data fractions, and # of local epochs per communication round. For each communication round (100 in total), 40% of the total 20 clients are randomly selected. We report the top-1 accuracy (on three different seeds) on the test dataset. This Figure complements Figure 2.
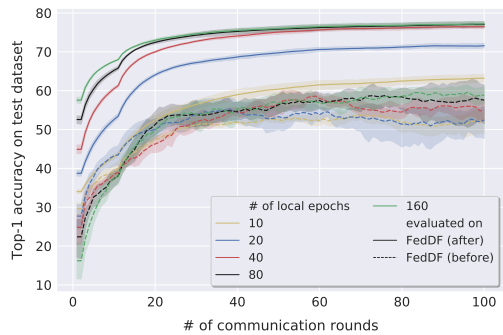
(a) The learning behaviors of FedDF and FEDAVG. We evaluate different # of local epochs on 100% local data.

(b) The fused model performance before (i.e. line 6 in Algorithm 1) and after FedDF (i.e. line 10 in Algorithm 1). We evaluate different # of local epochs on 100% local data.
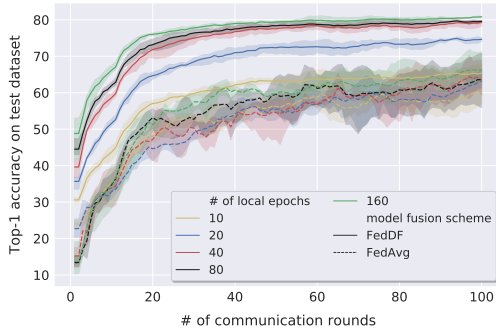
(c) The learning behaviors of FedDF and FEDAVG. We evaluate different # of local epochs on 50% local data.
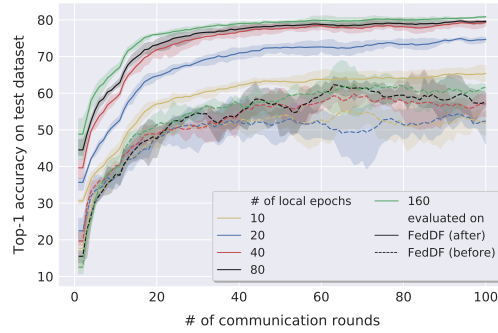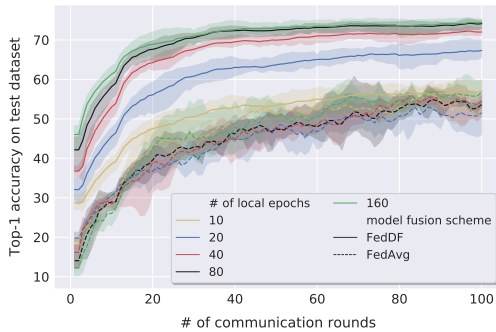
(d) The fused model performance before (i.e. line 6 in Algorithm 1) and after FedDF (i.e. line 10 in Algorithm 1). We evaluate different # of local epochs on 50% local data.

Figure 15: **Understanding the learning behaviors of FedDF** on CIFAR-10 with ResNet-8 for $\alpha = 100$. For each communication round (100 in total), 40% of the total 20 clients are randomly selected. We report the top-1 accuracy (on three different seeds) on the test dataset.
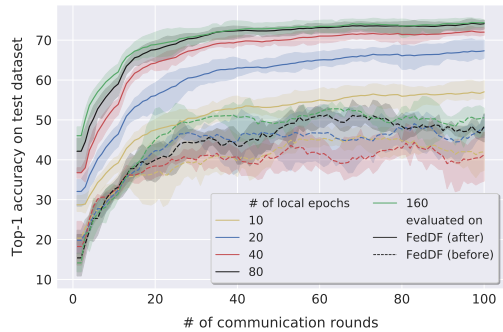
(a) The learning behaviors of FedDF and FEDAVG. We evaluate different # of local epochs on 100% local data.

(b) The fused model performance before (i.e. line 6 in Algorithm 1) and after FedDF (i.e. line 10 in Algorithm 1). We evaluate different # of local epochs on 100% local data.
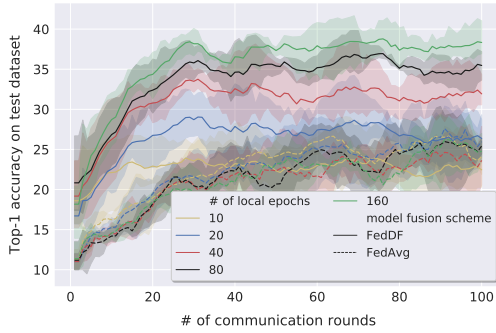
(c) The learning behaviors of FedDF and FEDAVG. We evaluate different # of local epochs on 50% local data.
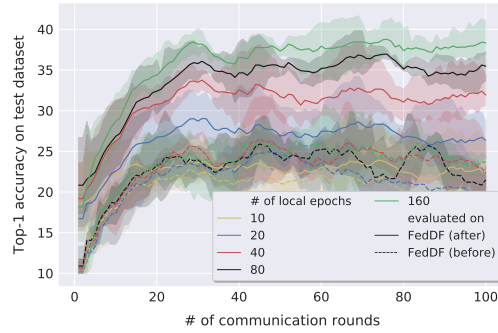
(d) The fused model performance before (i.e. line 6 in Algorithm 1) and after FedDF (i.e. line 10 in Algorithm 1). We evaluate different # of local epochs on 50% local data.
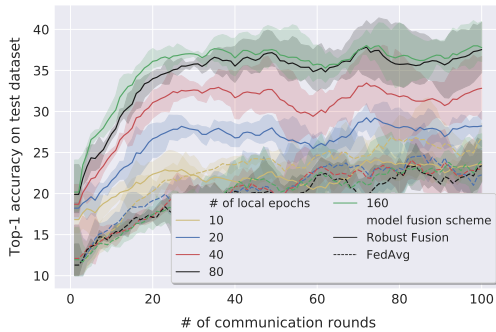
Figure 16: **Understanding the learning behaviors of FedDF** on CIFAR-10 with ResNet-8 for $\alpha = 1$. For each communication round (100 in total), 40% of the total 20 clients are randomly selected. We report the top-1 accuracy (on three different seeds) on the test dataset.
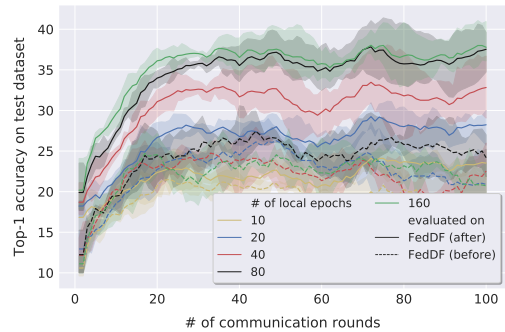
(a) The learning behaviors of FedDF and FEDAVG. We evaluate different # of local epochs on 100% local data.

(b) The fused model performance before (i.e. line 6 in Algorithm 1) and after FedDF (i.e. line 10 in Algorithm 1). We evaluate different # of local epochs on 100% local data.

(c) The learning behaviors of FedDF and FEDAVG. We evaluate different # of local epochs on 50% local data.

(d) The fused model performance before (i.e. line 6 in Algorithm 1) and after FedDF (i.e. line 10 in Algorithm 1). We evaluate different # of local epochs on 50% local data.

Figure 17: **Understanding the learning behaviors of FedDF** on CIFAR-10 with ResNet-8 for $\alpha = 0.01$. For each communication round (100 in total), 40% of the total 20 clients are randomly selected. We report the top-1 accuracy (on three different seeds) on the test dataset.

D Details on Generalization Bounds

The derivation of the generalization bound starts from the following notations. In FL, each client has access to its own data distribution $\mathcal{D}_i$ over domain $\Xi := \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \in \mathbb{R}^d$ is the input space and $\mathcal{Y}$ is the output space. The global distribution on the server is denoted as $\mathcal{D}$. For the empirical distribution by the given dataset, we assume that each local model has access to an equal amount (m) of local data. Thus, each local empirical distribution has equal contribution to the global empirical distribution: $\hat{\mathcal{D}} = \frac{1}{K} \sum_{k=1}^{K} \hat{\mathcal{D}}_k$, where $\hat{\mathcal{D}}_k$ denotes the empirical distribution from client k.

For our analysis we assume a binary classification task, with hypothesis h as a function $h : \mathcal{X} \to \{0, 1\}$. The loss function of the task is defined as $\ell(h(\mathbf{x}), y) = |\hat{y} - y|$, where $\hat{y} := h(\mathbf{x})$. Note that $\ell(\hat{y}, y)$ is convex with respect to $\hat{y}$. We denote $\arg\min_{h \in \mathcal{H}} L_{\hat{\mathcal{D}}}(h)$ by $h_{\hat{\mathcal{D}}}$.

The theorem below leverages the domain measurement tools developed in multi-domain learning theory [3] and provides insights for the generalization bound of the ensemble[15] of local models (trained on local empirical distribution $\hat{\mathcal{D}}_i$).

Theorem D.1. *The difference between* $L_{\mathcal{D}}(\frac{1}{K} \sum_k h_{\hat{\mathcal{D}}_k})$ *and* $L_{\hat{\mathcal{D}}}(h_{\hat{\mathcal{D}}})$, *i.e., the distance between the risk of our "ensembled" model in FedDF and the empirical risk of the "virtual ERM" with access to all local data, can be bounded with probability at least* $1 - \delta$:

$$
L_{\mathcal{D}}\left(\frac{1}{K} \sum_k h_{\hat{\mathcal{D}}_k}\right) \leq L_{\hat{\mathcal{D}}}(h_{\hat{\mathcal{D}}}) + \sqrt{\frac{\log \frac{2K}{\delta}}{2m}} + \frac{1}{K} \sum_k \left(\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_k, \mathcal{D}) + \lambda_k\right),
$$

where $\hat{\mathcal{D}} = \frac{1}{K} \sum_k \hat{\mathcal{D}}_k$, $d_{\mathcal{H}\Delta\mathcal{H}}$ *measures the domain discrepancy between two distributions [3], and* $\lambda_k = \inf_{h \in \mathcal{H}} (\mathcal{L}_{\mathcal{D}}(h) + \mathcal{L}_{\mathcal{D}_k}(h))$.

Remark D.2. *Theorem D.1 shows that, the upper bound on the risk of the ensemble of K local models on* $\mathcal{D}$ *mainly consists of 1) the empirical risk of a model trained on the global empirical distribution* $\hat{\mathcal{D}} = \frac{1}{K} \sum_k \hat{\mathcal{D}}_k$, *and 2) terms dependent on the distribution discrepancy between* $\mathcal{D}_k$ *and* $\mathcal{D}$.

The ensemble of the local models sets the performance upper bound for the later distilled model on the test domain as shown in Figure 4. Theorem 5.1 shows that compared to a model trained on aggregated local data (ideal case), the performance of an ensemble model on the test distribution is affected by the domain discrepancy between local distributions $\mathcal{D}_k$'s and the test distribution $\mathcal{D}$. The shift between the distillation and the test distribution determines the knowledge transfer quality between these two distributions and hence the test performance of the fused model. Through the lens of the domain adaptation theory [3], we can better spot the potential influence/limiting factors on our ensemble distillation procedure.

Remark D.3. *In the area of multiple-source adaptation, [50, 23] point out that the standard convex combinations of the source hypotheses may perform poorly on the test distribution. They propose combinations with weights derived from source distributions. However, FL scenarios require the server only access local models without any further local information. Thus we choose to uniformly average over local hypotheses as our global hypothesis. A privacy-preserved local distribution estimation is left for future work.*

D.1 Proof for Generalization Bounds

Theorem D.4 (Domain adaptation [3]). *Considering the distributions* $\mathcal{D}_S$ *and* $\mathcal{D}_T$, *for every* $h \in \mathcal{H}$ *and any* $\delta \in (0, 1)$, *with probability at least* $1 - \delta$ *(over the choice of the samples), there exists:*

$$
L_{\mathcal{D}_T}(h) \leq L_{\mathcal{D}_S}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda, \tag{1}
$$

where $\lambda = L_{\mathcal{D}_S}(h^\star) + L_{\mathcal{D}_T}(h^\star)$. $h^\star := \arg\min_{h \in \mathcal{H}} L_{\mathcal{D}_S}(h) + L_{\mathcal{D}_T}(h)$ *corresponds to* ideal joint hypothesis *that minimizes the combined error.*

Proof of Theorem D.1. We start from the risk of our "ensembled" model $L_{\mathcal{D}}(\frac{1}{K} \sum_k h_{\hat{\mathcal{D}}_k})$ and derive a series of upper bounds.

---

[15] The uniformly weighted hypothesis average in multi-source adaptation is equivalent to the ensemble of a list of models, by considering the output of each hypothesis/model.

Considering the distance between $L_\mathcal{D}(\frac{1}{K}\sum_k h_{\hat{\mathcal{D}}_k})$ **and** $L_{\hat{\mathcal{D}}}(h_{\hat{\mathcal{D}}})$**.** By convexity of ℓ and Jensen inequality, we have

$$L_\mathcal{D}(\frac{1}{K}\sum_k h_{\hat{\mathcal{D}}_k}) \leq \frac{1}{K}\sum_k L_\mathcal{D}(h_{\hat{\mathcal{D}}_k}) \,. \tag{2}$$

Using the domain adaptation theory in Theorem D.4, we transfer from domain $\mathcal{D}$ to $\mathcal{D}_k$,

$$L_\mathcal{D}(h_{\hat{\mathcal{D}}_k}) \leq L_{\mathcal{D}_k}(h_{\hat{\mathcal{D}}_k}) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_k, \mathcal{D}) + \lambda_k \,, \tag{3}$$

where $\lambda_k := \mathcal{L}_\mathcal{D}(h^\star) + \mathcal{L}_{\mathcal{D}_k}(h^\star)$ and $h^\star := \arg\min_{h\in\mathcal{H}} \mathcal{L}_\mathcal{D}(h) + \mathcal{L}_{\mathcal{D}_k}(h)$.
We can bound the risk with its empirical counterpart through Hoeffding inequality. A simple application of the Hoeffding's inequality gives

$$\Pr\left[\left|L_{\mathcal{D}_k}(h_{\hat{\mathcal{D}}_k}) - L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}_k})\right| \geq \epsilon\right] \leq 2\exp\frac{-2m^2\epsilon^2}{\sum_{j=1}^m (b-a)^2} \,,$$

where $[a,b]$ is the range of loss function. In our case, the loss function is bounded in $[0,1]$ so $(b-a)^2 \leq 1$, thereby, with probability at least $1 - \frac{\delta}{K}$, over the draw of m i.i.d. samples S_k from $\mathcal{D}_k$,

$$L_{\mathcal{D}_k}(h_{\hat{\mathcal{D}}_k}) \leq L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}_k}) + \sqrt{\frac{\log\frac{2}{\frac{\delta}{K}}}{2m}} \,, \tag{4}$$

Thus for K sources, we have

$$\Pr_{S_1\sim\mathcal{D}_1^m,\ldots,S_K\sim\mathcal{D}_K^m}\left[\bigcap_{k=1}^K\left\{L_{\mathcal{D}_k}(h_{\hat{\mathcal{D}}_k}) \leq L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}_k}) + \sqrt{\frac{\log\frac{2}{\frac{\delta}{K}}}{2m}}\right\}\right]$$

$$= 1 - \Pr_{S_1\sim\mathcal{D}_1^m,\ldots,S_K\sim\mathcal{D}_K^m}\left[\bigcup_{k=1}^K\left\{L_{\mathcal{D}_k}(h_{\hat{\mathcal{D}}_k}) \geq L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}_k}) + \sqrt{\frac{\log\frac{2}{\frac{\delta}{K}}}{2m}}\right\}\right] \tag{5}$$

$$\geq 1 - \sum_{k=1}^K \Pr_{S_1\sim\mathcal{D}_1^m,\ldots,S_K\sim\mathcal{D}_K^m}\left[\left\{L_{\mathcal{D}_k}(h_{\hat{\mathcal{D}}_k}) \geq L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}_k}) + \sqrt{\frac{\log\frac{2K}{\delta}}{2m}}\right\}\right]$$

$$\geq 1 - \delta \,.$$

Based on the definition of ERM, we have $L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}_k}) \leq L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}})$, where $h_{\hat{\mathcal{D}}}$ corresponds to the classifier trained with data from all workers. By using the definition of $\hat{\mathcal{D}}$ ($\hat{\mathcal{D}} = \frac{1}{K}\sum_k \hat{\mathcal{D}}_k$) and the linearity of expectation, we have

$$\frac{1}{K}\sum_k L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}_k}) \leq \frac{1}{K}\sum_k L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}}) = L_{\hat{\mathcal{D}}}(h_{\hat{\mathcal{D}}}) \,. \tag{6}$$

Putting these equations together, we have with probability of at least $1 - \delta$ over $S_1 \sim \mathcal{D}_1^m,\ldots,S_K\sim\mathcal{D}_K^m$ that

$$L_\mathcal{D}(\frac{1}{K}\sum_k h_{\hat{\mathcal{D}}_k}) \leq \frac{1}{K}\sum_k L_\mathcal{D}(h_{\hat{\mathcal{D}}_k})$$

$$\leq \frac{1}{K}\sum_k\left(L_{\mathcal{D}_k}(h_{\hat{\mathcal{D}}_k}) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_k, \mathcal{D}) + \lambda_k\right)$$

$$\leq \frac{1}{K}\sum_k\left(L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}_k}) + \sqrt{\frac{\log\frac{2K}{\delta}}{2m}} + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_k, \mathcal{D}) + \lambda_k\right)$$

$$\leq \frac{1}{K}\sum_k L_{\hat{\mathcal{D}}_k}(h_{\hat{\mathcal{D}}_k}) + \sqrt{\frac{\log\frac{2K}{\delta}}{2m}} + \frac{1}{K}\sum_k\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_k, \mathcal{D}) + \lambda_k\right)$$

$$\leq L_{\hat{\mathcal{D}}}(h_{\hat{\mathcal{D}}}) + \sqrt{\frac{\log\frac{2K}{\delta}}{2m}} + \frac{1}{K}\sum_k\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_k, \mathcal{D}) + \lambda_k\right) \,,$$

where $\lambda_k = \inf_{h\in\mathcal{H}}\left(\mathcal{L}_\mathcal{D}(h) + \mathcal{L}_{\mathcal{D}_k}(h)\right)$.

$\square$