DENSE: Data-Free One-Shot Federated Learning

Jie Zhang[1*] **Chen Chen**[1*] **Bo Li**[2‡] **Lingjuan Lyu**[3‡]
Shuang Wu [2] **Shouhong Ding**[2] **Chunhua Shen**[1] **Chao Wu**[1‡]

[1]Zhejiang University [2]Youtu Lab, Tencent [3] Sony AI

zj_zhangjie@zju.edu.cn, libraboli@tencent.com, lingjuanlvsmile@gmail.com

Abstract

One-shot Federated Learning (FL) has recently emerged as a promising approach, which allows the central server to learn a model in a single communication round. Despite the low communication cost, existing one-shot FL methods are mostly impractical or face inherent limitations, a public dataset is required, clients' models are homogeneous, and additional data/model information need to be uploaded. To overcome these issues, we propose a novel two-stage **D**ata-fre**E** o**N**e-**S**hot federated l**E**arning (DENSE) framework, which trains the global model by a data generation stage and a model distillation stage. DENSE is a practical one-shot FL method that can be applied in reality due to the following advantages: (1) DENSE requires no additional information compared with other methods (except the model parameters) to be transferred between clients and the server; (2) DENSE does not require any auxiliary dataset for training; (3) DENSE considers model heterogeneity in FL, different clients can have different model architectures. Experiments on a variety of real-world datasets demonstrate the superiority of our method. For example, DENSE outperforms the best baseline method Fed-ADI by 5.08% on CIFAR10 dataset. Our code will soon be available.

1 Introduction

Deep neural networks (DNNs) have recently gained popularity as a powerful tool for advancing artificial intelligence in many fields [9, 8, 48, 49, 55, 5, 4, 15]. Federated learning (FL) [39] has emerged as a promising learning paradigm which allows multiple clients to collaboratively train a global model without exposing their private training data. In FL, each client trains a local model on its own data and is required to periodically share its high-dimensional model parameters with a central server. Recent years, FL has shown its potential to facilitate real-world applications in many fields, including medical image analysis [33, 6], recommender systems [31, 35], and natural language processing [59, 43].

The original FL framework requires participants to communicate frequently with the central server in order to exchange models. In real-world FL, such high communication cost [*] may be intolerable and impractical. Reducing the communication cost between clients and the server is desired both for system efficiency and to support the privacy goals of federated learning [37, 38]. Recent research proposed some common methods to reduce communication costs, utilize multiple local updates [17], employ compression techniques [45], and **one-shot FL** [10]. Among them, one-shot FL is a promising solution which only allows one communication round. The following are two advantages of one-shot

[*]Both authors contributed equally to this work. Work is completed during Jie Zhang's internship at Tencent Youtu Lab and Chen Chen's internship at Sony AI.

[‡]Corresponding author.

[*]Generally, the total communication cost is a product of the number of local iterations and the communication cost per iteration.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

FL: **1)** Note that multi-round training is not practical in some scenarios such as model markets [42]. In model market scenarios, users can only buy the pre-trained models from the market without any real data, which can not be applied in traditional FL that requires multi-round training [26]. **2)** Furthermore, frequent communication poses a high risk of being attacked. For instance, frequent communication can be easily intercepted by attackers, who can launch man-in-the-middle attacks [44] or even reconstruct the training data from gradients [51]. In this way, one-shot FL can reduce the probability of being intercepted by malicious attackers due to the one round property. Thus, in this paper, we mainly focus on one-shot FL.

However, existing one-shot FL studies [10, 26, 58, 7] are still hard to apply in real-world applications, due to impractical settings. For example, [10] and [26] involved a public dataset for training, which may be impractical in very sensitive scenarios such as the biomedical domains. [58] adopted dataset distillation [47] in one-shot FL, but they need to send distilled data to the central server, which causes additional communication cost and potential privacy leakage. [7] utilized cluster-based method in one-shot FL, which requires to upload the cluster means to the server, causing additional communication cost. Additionally, none of these methods consider model heterogeneity, different clients have different model architectures [28], which is very common in practical scenarios. For instance, in model market, models sold by different sellers are likely to be heterogeneous. Besides, when several medical institutions participate in federated learning, they may need to design their own model to meet distinct specifications. Therefore, developing a practical one-shot FL method is in urgent need.

In this work, we propose a novel two-stage **D**ata-fre**E** o**N**e-**S**hot federated l**E**arning (DENSE) framework, which trains the global model by a data generation stage and a model distillation stage. In the first stage, we utilize the ensemble models (ensemble of local models uploaded by clients) to train a generator, which can generate synthetic data for training in the second stage. In the second stage, we distill the knowledge of the ensemble models to the global model. In contrast to traditional FL methods based on FedAvg [39], our method does not require averaging of model parameters, thus it can support heterogeneous models, clients can have different model architectures. In summary, our main contributions are summarized as follows:

- We propose a novel data-free one-shot FL framework named DENSE, which consists of two stages. In the first stage, we train a generator that considers *similarity*, *stability*, and *transferability* at the same time. In the second stage, we use the ensemble models and the data generated by the generator to train a global model.
- The setting of DENSE is practical in the following aspects. First, DENSE requires no additional information (except the model parameters) to be transferred between clients and the server; Second, DENSE does not require any auxiliary dataset for training; Third, DENSE considers model heterogeneity, different clients can have different model architectures.
- DENSE is a compatible approach, which can be combined with any local training techniques to further improve the performance of the global model. For instance, we can adopt LDAM [1] to train the clients' local models, and improve the accuracy of the global model (refer to Section 2.3 and Section 3.2).
- Extensive experiments on various datasets verify the effectiveness of our proposed DENSE. For example, DENSE outperforms the best baseline method Fed-ADI [52] by 5.08% on CIFAR10 dataset.

2 Data-Free One-Shot Federated Learning

2.1 Framework Overview

To tackle the problems in recent one-shot FL methods, we propose a novel method named DENSE, which conducts one-shot FL without the need to share additional information or rely on any auxiliary dataset, while considering model heterogeneity. To simulate real-world applications, we consider a more challenging yet practical setting where the data on each client are not independent and identically distributed (non-IID).

The illustration of the learning procedure is demonstrated in Figure 1, and the whole training process of DENSE is shown in Algorithm 1. After clients upload their local models to the server, the server

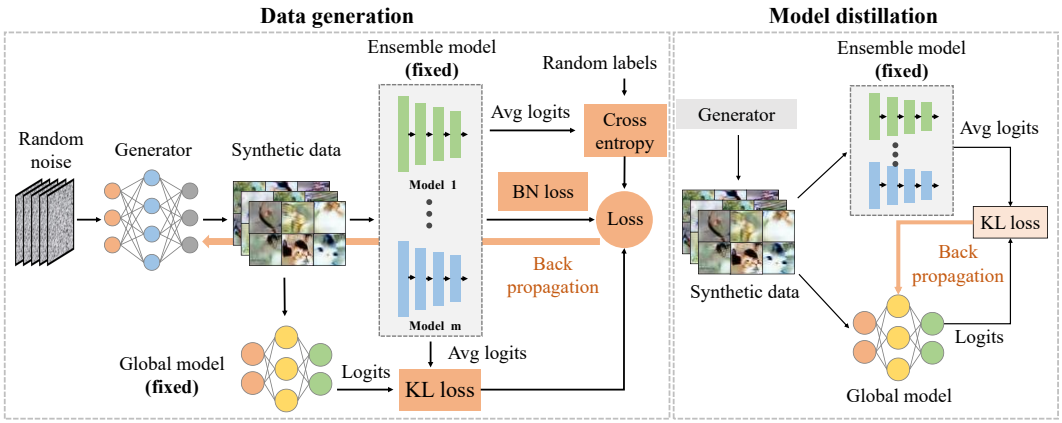

Figure 1: An illustration of training process of DENSE on the server, which consists of two stages: (1) In data generation stage, we train an auxiliary generator that considers similarity, stability, and transferability at the same time; (2) In model distillation stage, we distill the knowledge of the ensemble models and transfer to the global model. Note that the fixed global model is used as an additional discriminator in the divergence loss $\mathcal{L}_{div}$.

trains a global model with DENSE in two stages. In the data generation stage (first stage), we train an auxiliary generator that can generate synthetic data by the ensemble models, ensemble of local models uploaded by clients. In the model distillation stage (second stage), we use the ensemble models and the synthetic data (generated by the generator) to train the global model.

2.2 Data Generation

In the first stage, we aim to train a generator to generate synthetic data. Specifically, given the ensemble of well-trained models uploaded by clients, our goal is to train a generator that can generate data that have similar distribution to the training data of clients. In addition, we aim not to leak private information from our generated data, attackers are not able to predict any sensitive information of clients from the generated data. Recent work [32] generated data by utilizing a pre-trained generative adversarial network (GAN). However, such a method is unable to generate data as the pre-trained GAN is trained on public datasets, which is likely to have different data distribution from the training data of clients. Moreover, we need to consider model heterogeneity, which makes the problem more complicated.

To solve these issues, we propose to train a generator that considers *similarity*[†], *stability*, and *transferability*. The data generation process is shown in line 8 to 11 in Algorithm 1. In particular, given a random noise z (generated from a standard Gaussian distribution) and a random one-hot label y (generated from a uniform distribution), the generator $G(\cdot)$ aims to generate a synthetic data $\hat{x} = G(z)$ such that $\hat{x}$ is similar to the training data (with label y) of clients.

Similarity First, we need to consider the similarity between synthetic data $\hat{x}$ and the training data. Since we are unable to access the training data of clients, we cannot compute the similarity between the synthetic data and the training data directly. Instead, we first compute the average logits (outputs of the last fully connected layer) of $\hat{x}$ computed by the ensemble models.

$$D(\hat{x}; \{\theta^k\}_{k=1}^m) = \frac{1}{m} \sum_{k \in \mathcal{C}} f^k\left(\hat{x}; \theta^k\right), \tag{1}$$

where $m = |\mathcal{C}|$, and $D(\hat{x}; \{\theta^k\}_{k=1}^m)$ is the average logits of $\hat{x}$, θ^k is the parameter of the k-th client. And $f^k\left(\hat{\mathbf{x}}; \theta^k\right)$ is the prediction function of client k that outputs the logits of $\hat{x}$ given parameter θ^k. For simplicity, we use $D(\hat{x})$ to denote $D(\hat{x}; \{\theta^k\}_{k=1}^m)$ in the rest of the paper.

Then, we minimize the average logits and the random label y with the following cross-entropy (CE) loss.

$$\mathcal{L}_{CE}(\hat{x}, y; \theta_G) = CE(D(\hat{x}), y), \tag{2}$$

[†]Note that the ideal synthetic data should be visually distinct from the real data (to protect the privacy of data), but similar in distribution

It is expected that the synthetic images can be classified into one particular class with a high probability by the ensemble models. In fact, during the training phase, the loss between $D(\hat{x})$ and y can easily reduce to almost 0, which indicates the synthetic data matches the ensemble models perfectly. Moreover, we do not directly compute the similarity between the synthetic data and the training data, which can reduce the probability of leaking sensitive information of the clients.

However, by utilizing only the CE loss, we cannot achieve a high performance (please refer to Section 3.2 for detail). We conjecture this is because the ensemble models are trained on non-IID data, the generator may be unstable and trapped into sub-optimal local minima or overfit to the synthetic data [46, 29].

Stability Second, to improve the stability of the generator, we propose to add an additional regularization to stabilize the training. In particular, we utilize the Batch Normalization (BN) loss to make the synthetic data conform with the batch normalization statistics [52].

$$\mathcal{L}_{BN}(\hat{x};\theta_G) = \frac{1}{m}\sum_{k\in\mathcal{C}}\sum_{l}\left(\mu_l(\hat{x}) - \mu_{k,l} + \sigma_l^2(\hat{x}) - \sigma_{k,l}^2\right), \tag{3}$$

where $\mu_l(\hat{x})$ and $\sigma_l^2(\hat{x})$ are the batch-wise mean and variance estimates of feature maps corresponding to the l-th BN layer of the generator $G(\cdot)$[‡], $\mu_{k,l}$ and $\sigma_{k,l}^2$ are the mean and variance of the l-th BN layer [16] of $f^k(\cdot)$. The BN loss minimizes the distance between the feature map statistics of the synthetic data and the training data of clients. As a result, the synthetic data can have a similar distribution to the training data of clients, no matter if the data is non-IID or IID.

Transferability By utilizing the CE loss and BN loss, we can train a generator that can generate synthetic data, but we observed that the synthetic data are likely to be far away from the decision boundary (of the ensemble models), which makes the ensemble models (teachers) hard to transfer their knowledge to the global model (student). We illustrate the observation in the left panel of Figure 2. S and T are the decision boundaries of the global model (the detail of the global model is introduced in Section 2.3) and ensemble models respectively. The essence of knowledge distillation is transferring the information of decision boundary from the teacher model to the student model [12]. We aim to learn the decision boundary of global model and have a high classification accuracy on the real test data (blue diamonds). However, the generated synthetic data (red circles) are likely to be on the same side of the two decision boundaries and unhelpful to the transfer of knowledge [12]. To solve this problem, we argue to generate more synthetic data that fall between the decision boundaries of the ensemble models and the global model. We illustrate our idea in the right panel of Figure 2. Red circles are synthetic data on the same side of the decision boundary, which are less helpful in learning the global model. Black circles are synthetic data between the decision boundaries, the global model and the ensemble models have different predictions on these data. Black circles can help the global model better learn the decision boundary of the ensemble models.

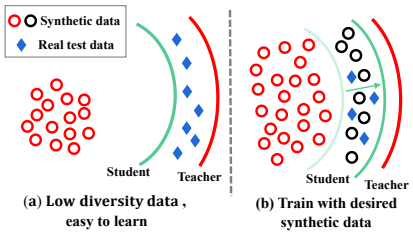

Figure 2: The illustration of generated data and decision boundary of ensemble models (teachers) and global model (student). **Left panel:** Synthetic data (red circles) are far away from the decision boundary, which is less helpful to the transfer of knowledge. **Right panel:** By utilizing our boundary support loss, we can generate more synthetic data near the decision boundaries (black circles), which helps the student better learn the decision boundary of the teacher.

Motivated by the above observations, we introduce a new boundary support loss, which urges the generator to generate more synthetic data between the decision boundaries of the ensemble models and the global model. We divide the synthetic data into 2 sets: (1) the global model and the ensemble models have the same predictions on data in the first set (${}_cD^{(c)}(\hat{x}) =_c f_S^{(c)}(\hat{x};\theta_S)$); (2) different predictions on data in the second set (${}_cD^{(c)}(\hat{x}) \neq_c f_S^{(c)}(\hat{x};\theta_S)$), where $D^{(c)}(\hat{x})$ and $f_S^{(c)}(\hat{x};\theta_S)$ are the logits for the c-th label of the ensemble models and the global model respectively. The data in

[‡]We assume the input is a batch of data.

the first set are on the same side of those two decision boundaries (red circles in Figure 2) while the data in the second set (black circles in Figure 2) are between the decision boundaries of the ensemble models and the global model. We maximize the differences of predictions of the global model and the ensemble models on data in the second set with Kullback-Leibler divergence loss as follows.

$$\mathcal{L}_{div}(\hat{x};\theta_G) = -\omega KL\left(D(\hat{x}), f_S(\hat{x};\theta_S)\right), \tag{4}$$

where $KL(\cdot,\cdot)$ denotes the Kullback-Leibler (KL) divergence loss, $\omega = 1({}_cD^{(c)}(\hat{x}) \neq_c f_S^{(c)}(\hat{x};\theta_S))$ outputs 0 for data in the first set and 1 for data in the second set, and $1(a)$ is the indicator function that outputs 1 if a is true and outputs 0 if a is false. By maximizing the KL divergence loss, the generator can generate more synthetic data that are more helpful to the model distillation stage (refer to Section 2.3 for detail) and further improve the transferability of the ensemble models.

By combining the above losses, we can obtain the generator loss as follows,

$$\mathcal{L}_{gen}(\hat{x}, y;\theta_G) = \mathcal{L}_{CE}(\hat{x}, y;\theta_G) + \lambda_1\mathcal{L}_{BN}(\hat{x};\theta_G) + \lambda_2\mathcal{L}_{div}(\hat{x};\theta_G), \tag{5}$$

where λ_1 and λ_2 are scaling factors for the losses.

Note that the generated synthetic data have similar features but different from the training data (of clients), which reduces the probability of leaking sensitive information of clients. More discussions of the privacy issues are in Section 3.3.3.

2.3 Model Distillation

In the second stage, we train the global model with the generator (discussed in the previous section) and the ensemble models. Previous research [56, 32] showed that model ensemble provides a general method for improving the accuracy and stability of learning models. Motivated by [56], we propose to use the ensemble models as a teacher to train a student (global) model. A straightforward method is to obtain the global model by aggregating the parameters of all client models (by FedAvg [39]). However, in real-world applications, clients are likely to have different model architectures [41], making FedAvg useless. Moreover, since the data in different clients are non-IID, FedAvg cannot deliver a good performance or even diverge [29].

To this end, we follow [32] to distill the knowledge of the ensemble models to the global model by minimizing the predictions between the ensemble models (teacher) and the global model (student) on the same synthetic data. The model distillation process is shown in line 13 to 14 in Algorithm 1. First, we compute the average logits of the synthetic data according to Eq. (1), $D(\hat{x}) = \frac{1}{m}\sum_{k\in\mathcal{C}} f^k\left(\hat{x};\theta^k\right)$. In contrast to traditional aggregation methods (FedAvg) that are unable to aggregate heterogeneous models, averaging logits can be easily applied to both heterogeneous and homogeneous FL systems.

Then, we use the average logits to distill the knowledge of the ensemble models by minimizing the following objective function.

$$\mathcal{L}_{dis}(\hat{x};\theta_S) = KL\left(D(\hat{x}), f_S(\hat{\mathbf{x}};\theta_S)\right). \tag{6}$$

By minimizing the KL loss, we can train a global model with the knowledge of the ensemble models and the synthetic data regardless of data and model heterogeneity.

Note that DENSE has no restriction on the clients' local models, clients can train models with arbitrary techniques. Thus, DENSE is a compatible approach, which can be combined with any local training techniques to further improve the performance of the global model. We further discuss the combination of local training techniques in Section 3.2.

Discussions on privacy-preserving Research [14] has shown that it is possible to launch an attack where a malicious user uses GANs to recreate samples of another participant's private datasets. Besides, in FL, exchange models between the server and clients can result in potential privacy leakage. Note that our method prohibits the generator from seeing the real data directly, and there is only one communication round, which reduces the risk of privacy leakage. In addition, we display our generated images in Section 3.3.3, which does not directly reveal the information of real data. Several existing privacy-preserving methods can be incorporated into our framework to protect clients from adversaries [34, 20]. We leave this as our future work.

Discussions on Knowledge Distillation in FL In traditional FL frameworks, all users have to agree on the specific architecture of the global model. To support model heterogeneity, [24] proposed a new federated learning framework that enables participants to independently design their models

by knowledge distillation [13]. With the use of a proxy dataset, knowledge distillation alleviates the model drift issue induced by non-IID data. However, the requirement of proxy data renders such a method impractical for many applications, since a carefully designed dataset is not always available on the server. Data-free knowledge distillation is a promising approach, which can transfer knowledge of a teacher model to a student model without any real data [2, 52]. [32] proposed data-free ensemble distillation for model fusion through synthetic data in each communication round, which requires high communication costs and computational costs. However, in this paper, we are more concerned with obtaining a good global model through only one round of communication in cases of heterogeneous models, which is more challenging and practical. [60] also proposed a data-free knowledge distillation approach for FL, which learns a generator derived from the prediction of local models. However, the learned generator is later broadcasted to all clients, and then clients need to send their generators to the server, which increases the communication burden. More seriously, the generator has direct access to the local data (the generator can easily remember the training samples [36]), which can cause privacy concerns. As the generator used in our method is always stored in the central server, it never sees any real local data.

3 Experiments

3.1 Experimental Setup

3.1.1 Datasets

Our experiments are conducted on the following 6 real-world datasets: MNIST [23], FMNIST [50], SVHN [40], CIFAR10 [21], CIFAR100 [21], and Tiny-ImageNet [22].MNIST dataset contains binary images of handwritten digits. There are 60,000 training images and 10,000 testing images in MNIST dataset. CIFAR10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images in CIFAR10 dataset. CIFAR100 dataset is similar to CIFAR10 dataset, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. Tiny-ImageNet contains 100000 images of 200 classes (500 for each class) downsized to 64×64 colored images. Each class has 500 training images, 50 validation images and 50 test images.

3.1.2 Data partition

To simulate real-world applications, we use Dirichlet distribution to generate non-IID data partition among clients [53, 25]. In particular, we sample $p_k \sim Dir(\alpha)$ and allocate a p_k^i proportion of the data of class k to client i. By varying the parameter α, we can change the degree of imbalance. A small α generates highly skewed data. We set $\alpha = 0.5$ as default.

3.1.3 Baselines

To ensure fair comparisons, we neglect the comparison with methods that require to download auxiliary models or datasets, such as FedBE [3] and FedGen [60]. Moreover, since there is only one communication round, aggregation methods that are based on regularization have no effect. Thus, we also omit the comparison with these regularization-based methods, FedProx [28], FedNova [46], and Scaffold [17]. Instead, we compare our proposed DENSE with FedAvg [39] and FedDF [32]. Furthermore, since DENSE is a data-free method, we derive some baselines from prevailing data-free knowledge distillation methods, including: 1) DAFL [2], a novel data-free learning framework based on generative adversarial networks; 2) ADI [52], an image synthesizing method that utilizes the image distribution to train a deep neural network without real data. We apply these methods to one-shot FL, and name these two baselines as Fed-DAFL and Fed-ADI.

3.1.4 Settings

For clients' local training, we use the SGD optimizer with momentum=0.9 and learning rate=0.01. We set the batch size $b = 128$, the number of local epochs $E = 200$, and the client number $m = 5$. Following the setting of [2], we train the auxiliary generator $G(\cdot)$ with a deep convolutional network. We use Adam optimizer with learning rate $\eta_G = 0.001$. We set the number of training rounds in each epoch as $T_G = 30$, and set the scaling factor $\lambda_1 = 1$ and $\lambda_2 = 0.5$. For the training of the server model $f_S()$, we use the SGD optimizer with learning rate $\eta_S = 0.01$ and momentum=0.9. The number of epochs for distillation $T = 200$. All baseline methods use the same setting as ours.

Table 1: Accuracy of different methods across $\alpha = \{0.1, 0.3, 0.5\}$ on different datasets.

Dataset	MNIST			FMNIST			CIFAR10			SVHN			CIFAR100			Tiny-ImageNet		
Method	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5
FedAvg	48.24	72.94	90.55	41.69	82.96	83.72	23.93	27.72	43.67	31.65	61.51	56.09	4.58	11.61	12.11	3.12	10.46	11.89
FedDF	60.15	74.01	92.18	43.58	80.67	84.67	40.58	46.78	53.56	49.13	73.34	73.98	28.17	30.28	36.35	15.34	18.22	27.43
Fed-DAFL	64.38	74.18	93.01	47.14	80.59	84.02	47.34	53.89	58.59	53.23	76.56	78.03	28.89	34.89	38.19	18.38	22.18	28.22
Fed-ADI	64.13	75.03	93.49	48.49	81.15	84.19	48.59	54.68	59.34	53.45	77.45	78.85	30.13	35.18	40.28	19.59	25.34	30.21
DENSE (ours)	**66.61**	**76.48**	**95.82**	**50.29**	**83.96**	**85.94**	**50.26**	**59.76**	**62.19**	**55.34**	**79.59**	**80.03**	**32.03**	**37.32**	**42.07**	**22.44**	**28.14**	**32.34**

Table 2: Accuracy comparisons across heterogeneous client models on CIFAR10. There are five clients in total, and each client has a personalized model.

Model	Client					Server (ResNet-18)			
	ResNet-18	CNN1	CNN2	WRN-16-1	WRN-40-1	FedDF	Fed-DAFL	Fed-ADI	**DENSE (ours)**
α=0.1	40.83	33.67	35.21	27.73	32.93	42.35	43.12	44.63	**49.76**
α=0.3	51.49	52.78	44.96	47.35	37.24	52.72	57.72	58.96	**63.25**
α=0.5	59.96	58.67	54.28	53.39	58.14	60.05	61.56	63.24	**67.42**

3.2 Results

3.2.1 Evaluation on real-world datasets

To evaluate the effectiveness of our method, we conduct experiments under different non-IID settings by varying $\alpha = \{0.1, 0.3, 0.5\}$ and report the performance on different datasets and different methods in Table 1. The results show that: (1) Our DENSE achieves the highest accuracy across all datasets. In particular, DENSE outperforms the best baseline method Fed-ADI [52] by 5.08% when $\alpha = 0.3$ on CIFAR10 dataset. (2) FedAvg has the worst performance, which implies that directly averaging the model parameters cannot achieve a good performance under non-IID setting in one-shot FL. (3) As α becomes smaller (data become more imbalanced), the performance of all methods decrease significantly, which shows that all methods suffer from highly skewed data. Even under highly skewed setting, DENSE still significantly outperforms other methods, which further demonstrates the superiority of our proposed method.

3.2.2 Impact of model distillation

We show the impact of model distillation by comparing with FedAvg. We first conduct one-shot FL and use FedAvg to aggregate the local models. We show the results of the global model and clients' local models across different local training epochs $E = \{20, 40, 60, \cdots, 400\}$ in the left panel of Figure 3. The global model achieves the best performance (test accuracy=34%) when $E = 40$, while a larger value of E can cause the model to degrade even collapse. This result can be attributed to the inconsistent optimization objectives with non-IID data [46], which leads to weight divergence [57]. Then, we show the results of one-shot FL when $E = 400$ and report the performance of FedAvg and DENSE in the right panel of Figure 3. We also plot the performance of clients' local models. DENSE outperforms each client's local model while FedAvg underperforms each client's local model. This validates that model distillation can enhance training while directly aggregating is harmful to the training under non-IID setting in one-shot FL.

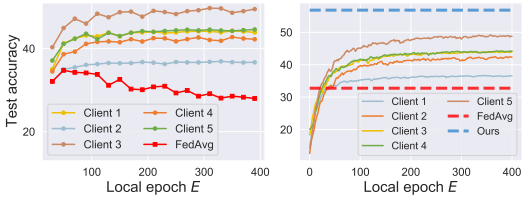

Figure 3: **Left panel:** Accuracy of FedAvg and clients' local models across different local training epochs $E = \{20, 40, 60, \cdots, 400\}$. **Right panel:** The accuracy curve for local training. The dotted lines represent the best results of two one-shot FL methods (FedAvg and DENSE). Our DENSE outperforms FedAvg and local models consistently.

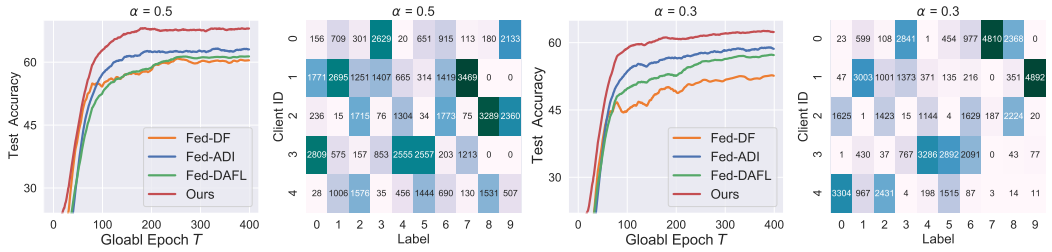

Figure 4: Visualization of the test accuracy and data distribution for CIFAR10 with $\alpha = \{0.3, 0.5\}$.

Table 3: Accuracy across different number of clients $m = \{5, 10, 20, 50, 100\}$ on CIFAR10 and SVHN datasets.

Dataset	CIFAR10					SVHN				
m	FedAvg	FedDF	Fed-DAFL	Fed-ADI	**DENSE (ours)**	FedAvg	FedDF	Fed-DAFL	Fed-ADI	**DENSE (ours)**
5	43.67	53.56	55.46	58.59	**62.19**	56.09	73.98	78.03	78.85	**80.03**
10	38.29	54.44	56.34	57.13	**61.42**	45.34	62.12	63.34	65.45	**67.57**
20	36.03	43.15	45.98	46.45	**52.71**	47.79	60.45	62.19	63.98	**66.42**
50	37.03	40.89	43.02	44.47	**48.47**	36.53	51.44	54.23	57.35	**59.27**
100	33.54	36.89	37.55	36.98	**43.28**	30.18	46.58	47.19	48.33	**52.48**

3.2.3 Results in heterogeneous FL

Note that our proposed DENSE can support heterogeneous models. We apply five different CNN models on CIFAR10 dataset with Dirichlet distribution $\alpha = \{0.1, 0.3, 0.5\}$. The heterogeneous models include: 1) one ResNet-18 [11], 2) two small CNNs: CNN1 and CNN2; 3) two Wide-ResNets (WRN) [54]: WRN-16-1 and WRN-40-1. For knowledge distillation, we use ResNet-18 as the server's global model. Detailed architecture information of the given deep networks can be found in Appendix. Table 2 evaluates all methods in heterogeneous one-shot FL under practical non-IID data settings. We omit the results for FedAvg as FedAvg does not support heterogeneous models. We remark that FL under both the non-IID data distribution and different model architecture setting is a quite challenging task. Even under this setting, our DENSE still significantly outperforms other baselines. In addition, we report the accuracy curve of global distillation. As shown in Figure 4, our method outperforms other baselines by a large margin.

3.3 Analysis of Our Method

Table 4: Performance analysis of DENSE+LDAM.

Dataset	CIFAR10			SVHN		
Method	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5
DENSE	50.26	59.76	62.19	55.34	79.59	80.03
DENSE+LDAM	**57.24**	**63.13**	**64.76**	**58.04**	**81.28**	**81.77**

3.3.1 Impact of the number of clients

Furthermore, we evaluate the performance of these methods on CIFAR10 and SVHN datasets by varying the number of clients $m = \{5, 10, 20, 50, 100\}$. According to [30], the server can become a bottleneck when the number of clients is very large, we are also concerned with the model performance when m increases. Table 3 shows the results of different methods across different m. The accuracy of all methods decreases as the number of clients m increases, which is consistent with observations in [30, 39]. Even though the number of clients can affect the performance of one-shot FL, our method still outperforms other baselines. The increasing number of clients poses new challenges for ensemble distillation, which we leave for future investigation.

3.3.2 Combination with imbalanced learning

The accuracy of federated learning reduces significantly with non-IID data, which has been broadly discussed in recent studies [25, 46]. Additionally, previous studies [1, 19] have demonstrated their superiority on imbalanced data. The combination of our method with these techniques to address imbalanced local data can lead to a more effective FL system. For example, by using LDAM [1] in clients' local training, we can mitigate the impact of data imbalance, and thereby build a more powerful ensemble model. We compare the performance of the original DENSE and DENSE combined with LDAM (DENSE+LDAM) across $\alpha = \{0.1, 0.3, 0.5\}$ on CIFAR10 and SVHN datasets.

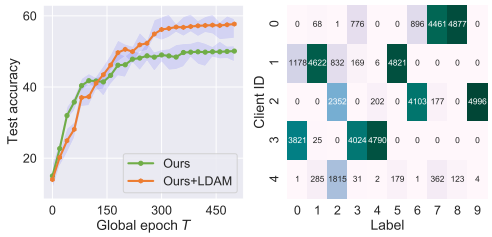

Figure 5: **Left panel:** Accuracy curves of DENSE and DENSE+LDAM. **Right panel:** Data distribution of different clients for CIFAR10 dataset (α=0.1).

As demonstrated in Table 4, DENSE+LDAM can significantly improve the performance, especially for highly skewed non-IID data (i.e. $\alpha = 0.1$). To help understand the performance gap and data skewness, in Figure 5, we visualize the accuracy curve and data distribution of CIFAR10 (α=0.1) in the left panel and right panel respectively. The number in the right panel stands for the number of examples associated with the corresponding label in one particular client. These figures imply that significant improvement can be achieved by combining DENSE with LDAM on highly skewed data.

3.3.3 Visualization of synthetic data

To compare the synthetic data with the training data, we visualize the synthetic data on CIFAR10 and SVHN datasets in Figure 6. As shown in the figure, the first / third row is the original data of CIFAR10 / SVHN dataset, and the second / last row is the synthetic data generated by the model trained on CIFAR10 / SVHN dataset. The synthetic data are not similar to the original data, which can effectively reduce the probability of leaking sensitive information of clients. Note that although the synthetic data look much different from the original data, our method still achieves a higher performance than other baseline methods by training with these synthetic data (as shown in Table 1). Note that the ideal synthetic data should be visually distinct from the real data (to protect the privacy of data).

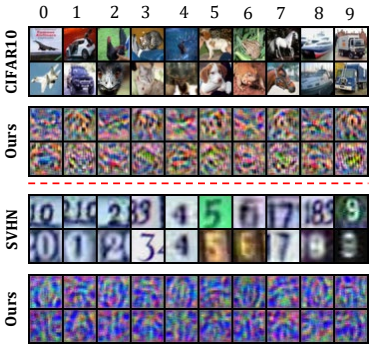

Figure 6: Visualization of synthetic data on CIFAR10 and SVHN datasets.

3.3.4 Extend to multiple rounds

To further show the effectiveness of DENSE, we extend DENSE to multi-round FL, there are multiple communication rounds between clients and server. Table 5 demonstrates the results of DENSE across different communication rounds $T_c = \{1, 2, 3, 4, 5\}$ on CIFAR10 and SVHN datasets. The local training epoch is fixed as $E = 10$. The performance of DENSE improves as T_c increases, and DENSE achieves the best performance when $T_c = 5$. This shows that DENSE can be extended

Table 5: Accuracy for multiple communication rounds.

Dataset	CIFAR10			SVHN		
Communication rounds	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5
$T_c = 1$	50.72	59.41	63.89	54.344	79.87	80.14
$T_c = 2$	63.08	65.90	71.16	56.13	79.75	85.18
$T_c = 3$	61.61	69.73	73.91	74.41	**86.42**	86.18
$T_c = 4$	66.26	69.40	74.39	78.67	86.36	86.43
$T_c = 5$	**67.65**	**71.42**	**76.01**	**80.28**	86.25	**86.55**

to multi-round FL and the performance can be further enhanced by increasing the communication rounds.

3.3.5 Contribution of $\mathcal{L}_{BN}$ and $\mathcal{L}_{div}$

We investigate the contributions of different loss functions used in data generation. We conduct leave-one-out testing and show the results by removing $\mathcal{L}_{div}$ (w/o $\mathcal{L}_{div}$), and removing $\mathcal{L}_{BN}$ (w/o $\mathcal{L}_{BN}$). Additionally, we report the result by removing both $\mathcal{L}_{div}$ and $\mathcal{L}_{BN}$, using only $\mathcal{L}_{CE}$ (w/ $\mathcal{L}_{CE}$). As illustrated in Table 6, using only $\mathcal{L}_{CE}$ to train the generator leads to poor performance. Besides, removing either the $\mathcal{L}_{BN}$ loss or $\mathcal{L}_{div}$ loss also affects the accuracy of the global model. A combination of these loss functions leads to a high performance of global model, which shows that each part of the loss function plays an important role in enhancing the generator.

Table 6: Impact of loss functions in data generation.

Dataset	CIFAR10	SVHN	CIFAR100
DENSE	**62.19**	**80.03**	**42.07**
w/ $\mathcal{L}_{CE}$	53.12	73.11	36.47
w/o $\mathcal{L}_{BN}$	61.05	78.36	39.89
w/o $\mathcal{L}_{div}$	59.18	77.59	39.14

4 Conclusion

In this paper, we propose an effective two-stage one-shot learning method named DENSE, which trains the global model by a data generation stage and a model distillation stage. Extensive experiments across various settings validate the efficacy of our method. Overall, DENSE is the first practical framework that can conduct data-free one-shot FL, while considers model heterogeneity. A promising future direction is to defend privacy attacks in one-shot FL. We leave further explorations of this aspect for future work.

5 Acknowledgement

This work was supported by Sony AI, the National Key Research and Development Project of China (2021ZD0110400 No. 2018AAA0101900), National Natural Science Foundation of China (U19B2042), Zhejiang Lab (2021KE0AC02), Academy Of Social Governance Zhejiang University, Fundamental Research Funds for the Central Universities.

References

[1] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss, 2019.

[2] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Dafl: Data-free learning of student networks. In *ICCV*, 2019.

[3] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning, 2021.

[4] Zhaoyu Chen, Bo Li, Shuang Wu, Jianghe Xu, Shouhong Ding, and Wenqiang Zhang. Shape matters: Deformable patch attack. In *European conference on computer vision*, 2022.

[5] Zhaoyu Chen, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Wenqiang Zhang. Towards practical certifiable patch defense with vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15148–15158, June 2022.

[6] Zhen Chen, Meilu Zhu, Chen Yang, and Yixuan Yuan. Personalized retrogress-resilient framework for real-world medical federated learning. In Marleen de Bruijne, Philippe C. Cattin, Stéphane Cotin, Nicolas Padoy, Stefanie Speidel, Yefeng Zheng, and Caroline Essert, editors, *Medical Image Computing and Computer Assisted Intervention - MICCAI 2021 - 24th International Conference, Strasbourg, France, September 27 - October 1, 2021, Proceedings, Part III*, volume 12903 of *Lecture Notes in Computer Science*, pages 347–356. Springer, 2021.

[7] Don Kurian Dennis, Tian Li, and Virginia Smith. Heterogeneity for the win: One-shot federated clustering, 2021.

[8] Jiahua Dong, Yang Cong, Gan Sun, Zhen Fang, and Zhengming Ding. Where and how to transfer: Knowledge aggregation-induced transferability perception for unsupervised domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):1–17, 2021.

[9] Jiahua Dong, Yang Cong, Gan Sun, Bineng Zhong, and Xiaowei Xu. What can be transferred: Unsupervised domain adaptation for endoscopic lesions segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4022–4031, June 2020.

[10] Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[12] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge distillation with adversarial samples supporting decision boundary. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3771–3778, 2019.

[13] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[14] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 603–618, 2017.

[15] Hao Huang, Yongtao Wang, Zhaoyu Chen, Yuze Zhang, Yuheng Li, Zhi Tang, Wei Chu, Jingdong Chen, Weisi Lin, and Kai-Kuang Ma. Cmua-watermark: A cross-model universal adversarial watermark for combating deepfakes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):989–997, Jun. 2022.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

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

[18] Anirudh Kasturi, Anish Reddy Ellore, and Chittaranjan Hota. Fusion learning: A one shot federated learning. In *International Conference on Computational Science*, pages 424–436. Springer, 2020.

[19] Jaehyung Kim, Jongheon Jeong, and Jinwoo Shin. M2m: Imbalanced classification via major-to-minor translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[20] Aashish Kolluri, Teodora Baluta, and Prateek Saxena. Private hierarchical clustering in federated networks. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 2342–2360. ACM, 2021.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[25] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study, 2021.

[26] Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. *arXiv preprint arXiv:2010.01017*, 2020.

[27] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *CoRR, arXiv:1908.07873*, 2019.

[28] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

[29] Xiaoxiao Li, Meirui JIANG, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In *International Conference on Learning Representations*, 2020.

[30] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent, 2017.

[31] Feng Liang, Weike Pan, and Zhong Ming. Fedrec++: Lossless federated recommendation with explicit feedback. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*.

[32] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[33] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 1013–1023. Computer Vision Foundation / IEEE, 2021.

[34] Ruixuan Liu, Yang Cao, Hong Chen, Ruoyang Guo, and Masatoshi Yoshikawa. FLAME: differentially private federated learning in the shuffle model. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 8688–8696. AAAI Press, 2021.

[35] Shuchang Liu, Shuyuan Xu, Wenhui Yu, Zuohui Fu, Yongfeng Zhang, and Amélie Marian. Fedct: Federated collaborative transfer for recommendation. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, editors, *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 716–725. ACM, 2021.

[36] Yi Liu, Jialiang Peng, JQ James, and Yi Wu. Ppgan: Privacy-preserving generative adversarial network. In *2019 IEEE 25Th international conference on parallel and distributed systems (ICPADS)*, pages 985–989. IEEE, 2019.

[37] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S Yu. Privacy and robustness in federated learning: Attacks and defenses. *arXiv preprint arXiv:2012.06337*, 2020.

[38] Lingjuan Lyu, Han Yu, Jun Zhao, and Qiang Yang. Threats to federated learning. In *Federated Learning*, pages 3–16. Springer, 2020.

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

[40] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[41] Lichao Sun and Lingjuan Lyu. Federated model distillation with noise-free differential privacy. In *IJCAI*, 2021.

[42] Manasi Vartak. MODELDB: A system for machine learning model management. In *8th Biennial Conference on Innovative Data Systems Research, CIDR 2017, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*. www.cidrdb.org, 2017.

[43] Chenghong Wang, Jieren Deng, Xianrui Meng, Yijue Wang, Ji Li, Fei Miao, Sanguthevar Rajasekaran, and Caiwen Ding. A secure and efficient federated learning framework for NLP. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*.

[44] Derui Wang, Chaoran Li, Sheng Wen, Surya Nepal, and Yang Xiang. Man-in-the-middle attacks against machine learning classifiers via malicious generative models. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[45] Hongyi Wang, Saurabh Agarwal, and Dimitris S. Papailiopoulos. Pufferfish: Communication-efficient models at no extra cost. *CoRR*, abs/2103.03936, 2021.

[46] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020.

[47] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2020.

[48] Kun Wei, Da Chen, Yuhong Li, Xu Yang, Cheng Deng, and Dacheng Tao. Incremental embedding learning with disentangled representation translation. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[49] Kun Wei, Cheng Deng, Xu Yang, and Maosen Li. Incremental embedding learning via zero-shot translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10254–10262, 2021.

[50] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[51] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16337–16346, 2021.

[52] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

[53] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks, 2019.

[54] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2017.

[55] Jie Zhang, Chen Chen, Jiahua Dong, Ruoxi Jia, and Lingjuan Lyu. Qekd: Query-efficient and data-free knowledge distillation from black-box models. *arXiv preprint arXiv:2205.11158*, 2022.

[56] Shaofeng Zhang, Meng Liu, and Junchi Yan. The diversified ensemble neural network. *Advances in Neural Information Processing Systems*, 33, 2020.

[57] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data, 2018.

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

[59] Xinghua Zhu, Jianzong Wang, Zhenhou Hong, and Jing Xiao. Empirical studies of institutional federated learning for natural language processing. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 625–634. Association for Computational Linguistics, 2020.

[60] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12878–12889. PMLR, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Algorithm 1 Training process of DENSE

Input: Number of client m, clients' local models $\{f^1(), \cdots, f^m()\}$, generator $G(\cdot)$ with parameter θ_G, learning rate of the generator η_G, number of training rounds T_G for generator in each epoch, global model $f_S()$ with parameter θ_S, learning rate of the global model η_S, global model training epochs T, and batch size b.

for each client $k \in \mathcal{C}$ **in parallel do**
  $\theta^k \leftarrow$ **LocalUpdate**(k)
end for
Initialize parameter θ_G and θ_S
for epoch=$1, \cdots, T$ **do**
  Sample a batch of noises and labels $\{z_i, y_i\}_{i=1}^b$
  // data generation stage
  for $j = 1, \cdots, T_G$ **do**
    Generate $\{\hat{x}_i\}_{i=1}^b$ with $\{z_i\}_{i=1}^b$ and $G(\cdot)$
    $\theta_G \leftarrow \theta_G - \eta_G \frac{1}{b} \sum_{i=1}^b \nabla_{\theta_G} \ell_{gen}(\hat{x}_i, y_i; \theta_G)$
  end for
  // model distillation stage
  Generate $\{\hat{x}_i\}_{i=1}^b$ with $\{z_i\}_{i=1}^b$ and $G(\cdot)$
  $\theta_S = \theta_S - \eta_S \frac{1}{b} \sum_{i=1}^b \nabla_{\theta_S} \ell_{dis}(\hat{x}_i; \theta_S)$
end for

6 Appendix

6.1 Detailed Algorithm

6.2 Preliminaries and Related Works

6.2.1 Federated Learning

Suppose there are m clients in a FL system, and each client k has its own dataset $\mathcal{D}_k = \{x_i, y_i\}_{i=1}^{n_k}$ with $n_k = \mathcal{D}_k$ being the size of local data. Each client k optimizes its local model by minimizing the following objective function,

$$\min_{\theta^k} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}(f^k(x_i), y_i), \tag{7}$$

where $f^k()$ and θ^k are the local model and model parameter, respectively. Then, client k uploads θ^k to the server for aggregation. After receiving the uploaded model parameters from the m clients, the server computes the aggregated global model parameter as follows: $\theta_S = \sum_{k=1}^m \frac{n_k}{n} \theta^k$, where $n = \sum_{k=1}^m n_k$ is the total amount of training data (of all clients) and θ_S is the server's global model parameter. Afterwards, the server distributes θ_S to all clients for training in the next round. However, such a training procedure needs frequent communication between the clients and the server, thus incurs a high communication cost, which may be intolerable in practice [26].

6.2.2 One-shot Federated Learning

A promising solution to reduce the communication cost in FL is one-shot FL, which is first introduced by [10]. In one-shot FL, each client only uploads its local model parameter to the server once. After obtaining the global model, the server does not need to distribute the global model to the clients for further training. There is only one unidirectional communication between clients and the server, thus it highly reduces the communication cost and makes it more practical in reality. Moreover, one-shot FL also reduces the risk of being attacked, since the communication happens only once. However, the main problem of one-shot FL is the difficult convergence of the global model and it is hard to achieve a satisfactory performance especially when the data on clients are not independent and identically distributed (non-IID).

[10] and [26] used ensemble distillation to improve the performance of one-shot FL. However, they introduced a public dataset to enhance training, which is not practical. In addition to model

distillation, dataset distillation [47] is also a prevailing approach. [58] and [18] proposed to apply dataset distillation to one-shot FL, where each client distills its private dataset and transmits distilled data to the server. However, data distillation methods fail to offer satisfactory performance compared with model distillation and sending distilled data can cause additional communication cost and potential privacy leakage. [7] utilized cluster-based method in one-shot FL, but they required to upload the cluster means to the server, which incurs additional communication cost.

Overall, none of the above methods can be practically applied. In addition, none of these studies consider model heterogeneity, which is a main challenge in FL [27]. This leads to a fundamental yet so far unresolved question: *"Is it possible to conduct one-shot FL without the need to share additional information or rely on any auxiliary dataset, while making it compatible with model heterogeneity?"*

6.2.3 Knowledge Distillation in FL

In traditional FL frameworks, all users have to agree on the specific architecture of the global model. To support model heterogeneity, [24] proposed a new federated learning framework that enables participants to independently design their models by knowledge distillation [13]. With the use of a proxy dataset, knowledge distillation alleviates the model drift issue induced by non-IID data. However, the requirement of proxy data renders such a method impractical for many applications, since a carefully designed dataset is not always available on the server.

Data-free knowledge distillation is a promising approach, which can transfer knowledge of a teacher model to a student model without any real data [2, 52]. [32] proposed data-free ensemble distillation for model fusion through synthetic data in each communication round, which requires high communication costs and computational costs. However, in this paper, we are more concerned with obtaining a good global model through only one round of communication in cases of heterogeneous models, which is more challenging and practical.

Zhu et.al. [60] also proposed a data-free knowledge distillation approach for FL, which learns a generator derived from the prediction of local models. However, the learned generator is later broadcasted to all clients, and then clients need to send their generators to the server, which increases the communication burden. More seriously, the generator has direct access to the local data (the generator can easily remember the training samples [36]), which can cause privacy concerns. As the generator used in our method is always stored in the central server, it never sees any real local data.