# Addressing Heterogeneity in Federated Learning via Distributional Transformation

Haolin Yuan[1*], Bo Hui[1*], Yuchen Yang[1*], Philippe Burlina[1,2], Neil Zhenqiang Gong[3], and Yinzhi Cao[1]

[1] Department of Computer Science, Johns Hopkins University
{hyuan4, bo.hui, yc.yang, yinzhi.cao}@jhu.edu
[2] Johns Hopkins University Applied Physics Laboratory (JHU/APL)
Philippe.Burlina@jhuapl.edu
[3] Duke University
neil.gong@duke.edu

**Abstract.** Federated learning (FL) allows multiple clients to collaboratively train a deep learning model. One major challenge of FL is when data distribution is heterogeneous, i.e., differs from one client to another. Existing personalized FL algorithms are only applicable to narrow cases, e.g., one or two data classes per client, and therefore they do not satisfactorily address FL under varying levels of data heterogeneity. In this paper, we propose a novel framework, called DisTrans, to improve FL performance (i.e., model accuracy) via train and test-time distributional transformations along with a double-input-channel model structure. DisTrans works by optimizing distributional offsets and models for each FL client to shift their data distribution, and aggregates these offsets at the FL server to further improve performance in case of distributional heterogeneity. Our evaluation on multiple benchmark datasets shows that DisTrans outperforms state-of-the-art FL methods and data augmentation methods under various settings and different degrees of client distributional heterogeneity (e.g., for CelebA and 100% heterogeneity DisTrans has accuracy of 80.4% vs. 72.1% or lower for other SOTA approaches).

## 1 Introduction

Federated learning [35,30,18,48] (FL) is an emerging distributed machine learning (ML) framework that enables clients to learn models together with the help of a central server. In FL, each client learns a local model that is sent to the FL server for aggregation, and subsequently the FL server returns the aggregated model to the client. The process is repeated until convergence. One emerging and unsolved FL challenge is that the data distribution at each client can be heterogeneous. For example, for FL based skin diagnostics, the skin disease distribution for each hospital / client can vary significantly. In another use case of smartphone face verification, data distributions collected at each mobile device can vary from one

[*] The first three authors have equal contributions to the paper.

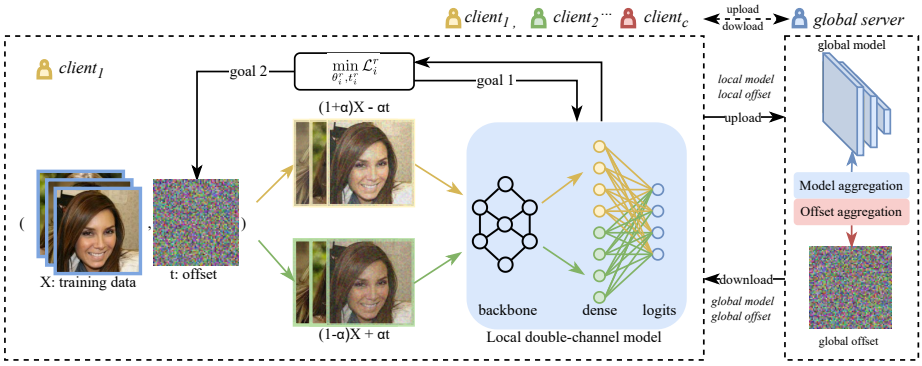



Fig. 1: The pipelines of DisTrans. Each client jointly optimizes the offset and model in local training phase, then uploads both to the central server for aggregation. The aggregated model and offset are sent back to clients for next-round.

client to another. Such distributional heterogeneity often leads to suboptimal accuracy of the final FL model.

There are two types of approaches to learn FL models under data heterogeneity: (i) improving FL's training process and (ii) improving clients' local data. Unfortunately, neither improves FL under varied levels of data heterogeneity. On one hand, existing FL methods [35,2,14], especially personalized FLs [28,26], learn a model (or even multiple models) using customized loss functions or model architectures based on heterogeneity level. However, existing personalized FL algorithms are designed for highly heterogeneous distribution. FedAwS [50] can only train FL models when local client's data has one positive label. The performance of pfedMe [43] and pfedHN [39] degrades to even 5% to 18% lower accuracy than FedAvg [35], when the data distribution is between heterogeneity and homogeneity.

On the other hand, traditional centralized machine learning also rely on data transformations, i.e., data augmentation, [7,8,46,52,27,9,53,31] to improve model's performance. Such transformations could be used for a pre-processing of all the training data or an addition to the existing training set. Until very recently, data transformations are also used during test time [38,40,21,15,42] to improve learning models, e.g., adversarial robustness [38]. However, it remains unclear whether and how data transformation can improve FL particularly under different client heterogeneity. The major challenge is how to tailor transformations for each client with different data distributions.

In this paper, we propose the *first* FL distributional transformation framework, called DisTrans, to address this heterogeneity challenge by altering local data distributions via a client-specific data shift applied both on train and test/inference data. Our distributional transformation alters each client's data distribution so that such distribution becomes less heterogeneous and thus the local models can be better aggregated at the server. Specifically, DisTrans performs a so-called *joint optimization*, at each client, to train the local model

and generate an offset that is added to the local data. That is, an DisTrans's client alternately performs two steps in each round: 1) optimizing the personalized *offset* to transform the local data via distribution shifts and 2) optimizing a local model to fit its offsetted local data. After client-side optimization, the FL server aggregates both the personalized offsets and the local models from all the clients and sends the aggregated global model and offset back to each client. During testing, each client adds its personalized offset to each testing input before using the global model to predict its label.

DisTrans is designed with a special network architecture, called a double-input-channel model, to accommodate client-side offsets. This double-input-channel model has a backbone network shared by both channels, a dense layer accepting outputs from two channels in parallel, and a logits layer that merges channel-related outputs from the dense layer. This double architecture allows the offset to be added to an (training or testing) input in one channel but subtracted from the input in the other. Such addition and subtraction better preserves the information in the original training and testing data because the original data can be recovered from the data with offset in the two channels.

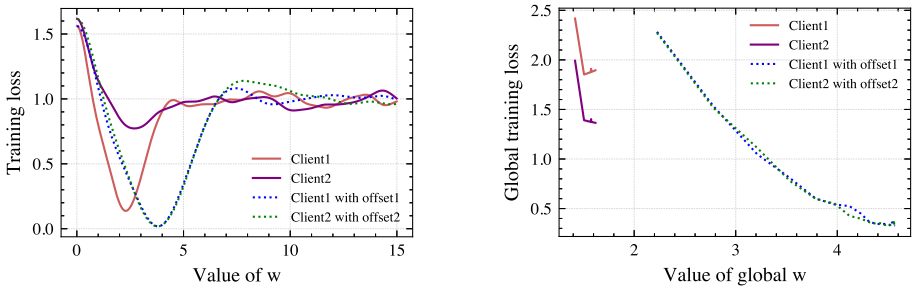
We perform extensive evaluation of DisTrans using five different image datasets and compare it against state-of-the-art (SOTA) methods. Our evaluation shows that DisTrans outperforms SOTA FL methods across various distributional settings of the clients' local data by 1%–10% with respect to testing accuracy. Moreover, our evaluation shows that DisTrans achieves 1%–7% higher testing accuracy than other data transformation / augmentation approaches, i.e., mixup [51] and AdvProp [46]. The code for DisTrans is made available under (https://github.com/hyhmia/DisTrans).

## 2 Related Work

Existing federated learning (FL) studies focus on improving accuracy [35,50,43,39], convergence [12,6,37,17,45,32], communication cost [24,41,22,3,23,13,34,49], security and privacy [36,10,5,4], or others [16,20,11,47]. Our work focuses on FL accuracy.

**Personalized Federated Learning.** Prior studies [43,50,39] have attempted to address personalization, i.e., to make a model better fit a client's local training data. For instance, FedAwS [50] investigates FL problems where each local model only has access to the positive data associated with only a single class and imposes a geometric regularizer at the server after each round to encourage classes to spread out in the embedding space. pFedMe [43] formulates a new bi-level optimization problem and uses Moreau envelopes to regularize each client loss function and to decouple personalized model optimization from the global model learning. pFedHN [39] utilizes a hypernetwork model as the global model to generate weights for each local model. MOON [29] uses contrastive learning to maximize the agreement between local and global model.

**Data Transformation.** Data transformation applies label-preserving transformations to images and is a standard technique to improve model accu-




(a) $y = cos(wx)$ on local clients. (b) $y = wx$ on FL clients.

Fig. 2: Training loss with respect to optimal weight $w$ on two clients' local training data with and w/o offset. We observe that offsets can make the training loss against weight more consistent on local clients and help FL model converge.

racy in centralized learning. Most of the recent data transformation methods [7,8,46,52,27,9,53,31] focus on transforming datasets during the training phase. For instance, mixup [51] transforms the training data by mixing up the features and their corresponding labels; and AdvProp [46] transforms the training data by adding adversarial examples. Additionally, transforming data at testing time [38,40,21,15,42] has received increased attention. The basic test-time transformations use multiple data augmentations [15,42] at test time to classify one image and get the averaged results. Pérez et.al [38] aims to enhance adversarial robustness via test-time transformation. As a comparison, DisTrans is the first to utilize test-time transformation to improve federated learning accuracy under data heterogeneity.

## 3 Motivation

DisTrans's intuition is to transform each client's training and testing data with offsets to improve FL under heterogeneous data. That is, DisTrans transforms the client-side data distribution so that the learned local models are less heterogeneous and can be better aggregated. To better illustrate this intuition, we describe two simple learning problems as motivating examples. Specifically, we show that well-optimized and selected offsets can (i) align two learning problems at different FL clients and (ii) help the aggregated model converge.

**Local Non-convex Learning Problems.** We consider a non-convex learning problem, i.e., $f(x) = cos(wx)$ where $w \in \mathbb{R}$, at two local clients with heterogeneous data. The local data is generated via $x, y \in \mathbb{R}$ with $y = cos(w^{true}_{clientk} x) + \epsilon_{clientk}$, where $x$ is drawn i.i.d from Gaussian distribution and $\epsilon_{clientk}$ is Gaussian noise with mean value as 0. The offsets are $px+q$ where $p$ is a fixed value at both clients and $q$ is chosen via brute force search. Figure 2a shows the squared training loss with and without offsets. The difference between the training losses of two learning models are reduced, thus making two clients consistent.

**Linear Regression Problems with An Aggregation Server.** We train two local linear models, i.e., $f(x) = wx$ with the model parameter $w \in \mathbb{R}^2$, aggregate

**Algorithm 1** Pseudo-code of DisTrans

**Input:** Number of clients $C$, local training dataset $D_i$ for client $i$, number of rounds $R$, batch size $B$, number of epochs $E$, and learning rates $\eta$ and $\eta_p$ for model and offset $t$, respectively
**Output:** Offset $t_i$ for client $i$ and global model $\theta$

Server initializes global model $\theta^0$ and offset $t_i^0$ for each client $i$
**for** $r = 0$ to $R-1$ **do**
  Server sends $\theta^r$ and $t_i^r$ to client $i$
  **for** $i = 0$ to $C-1$ **do**
    $\theta_i^r \leftarrow \theta^r$ // Initialize local model $\theta_i^r$ for client $i$
    **for** $e = 0$ to $E-1$ **do**
      **for** each mini-batch $D_m$ from $D_i$ **do**
        $t_i^r \leftarrow SGD(\nabla_{t_i^r} \mathcal{L}_i^r, t_i^r, \eta_t)$ // Update offset $t_i^r$
        $x_t \leftarrow ((1-\alpha)x + \alpha t_i^r,\ (1+\alpha)x - \alpha t_i^r)$ // Combine $t_i^r$ with each $x \in D_m$
        $\theta_i^r \leftarrow SGD(\nabla_{\theta_i^r} \mathcal{L}_i^r, \theta_i^r, \eta)$ // Update local model
      **end for**
    **end for**
    Client $i$ sends $\theta_i^r$ and $t_i^r$ to server
  **end for**
  Server updates global model: $\theta^{r+1} \leftarrow \frac{1}{C} \sum_{i \in [C]} \theta_i^r$
  Server updates offset $t_i^{r+1}$ for each client $i$ via Offset Aggregation
**end for**

the parameters at a server following FL, and then repeat the two steps following FL until convergence. The local training data is heterogeneous and generated as $y = w_{clientk}^{true} x + \epsilon_{clientk}$, where each of the two dimensions of $x$ is drawn i.i.d from normal distribution and $\epsilon_{clientk}$ is a Gaussian noise. The offset is the same as the non-convex learning problem. We fix $p$ and optimize $q$ and $w$ via SGD at each client to minimize learning loss respectively. Figure 2b shows the squared training loss with respect to the optimal $w$ (sum value of two dimensions) with and without the offsets. Clearly, when offsets are not present, the aggregated model does not converge, resulting in a set of sub-optimal weights. Instead, the aggregated model converges with a small training loss with the presence of offsets, confirming our intuition.

## 4 Method

In this section, we present our proposed method in detail. DisTrans aims to learn a single shared global model for the clients. Algorithm 1 shows the pseudo-code of DisTrans. In each round, each client learns a local model and an offset, which are sent to the central server. The server aggregates the clients' local models and local offsets, and sends them back to the clients. Based on the intuition presented above, we propose a joint optimization method to learn a local model and offset for a client in each round. Figure 1 illustrates our joint optimization that each client performs.

**Notations.** We assume $C$ clients and denote by $D_i$ the local training dataset for client $i$, where $i = 1, 2, \cdots, C$ and $|D_i| = n_i$. We consider $z = (x, y)$ a training sample, where $x \in \mathbb{R}^m$ denotes the training input and $y$ the label of the training input. We also denote by $D_{ti}$ the offsetted local training dataset for client $i$, $x_t$ an offsetted training input, and $z_t = (x_t, y)$ a training sample offsetted with offset. We denote by $\theta$ the global model.

### 4.1 Double-Input-Channel Model Architecture

DisTrans uses a double-input-channel neural network architecture (see Figure 1) for a local/global model. Our architecture has a shared backbone network, a dense layer concatenating two channels' outputs, and a logits layer merging outputs from the dense layer. Specifically, these two channels shift the local data distribution in two different ways using the same offset $t$. Formally, Eq. 1 shows our two linear shifts:

$$x_t = ((1-\alpha)x + \alpha t,\ (1+\alpha)x - \alpha t)), \tag{1}$$

where the first channel adds the offset $t$ to the input $x$ with a coefficient $\alpha$ (i.e., $(1-\alpha)x + \alpha t$ is the input for the first channel) and the second subtracts $t$ from $x$ with the same $\alpha$ (i.e., $(1+\alpha)x - \alpha t$ is the input to the second channel). Unless otherwise mentioned, our default setting for $\alpha$ is 0.3 in our experiments.

### 4.2 Joint Optimization

In our joint optimization, each client aims to achieve the following two goals:

- *Goal 1.* Optimizing *offset* to shift local data distribution to better fit with local model.
- *Goal 2.* Optimizing local model to fit with offsetted local data distribution.

We formulate the two goals as an optimization problem. Specifically, client $i$ aims to solve the following optimization problem in round $r$:

$$\min_{\theta_i^r, t_i^r} \mathcal{L}_i^r = \frac{1}{n} \sum_{z_t \in D_{ti}} l(\theta_i^r, z_t), \tag{2}$$

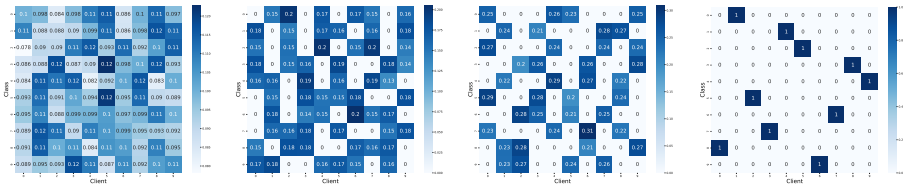
where $\theta_i^r$ is the local model of client $i$, $t_i^r$ is the offset of client $i$, and $\mathcal{L}_i^r$ is the loss function of client $i$ in round $r$. We choose cross entropy as loss term in our implementation. Solving $t_i^r$ in Eq. 2 while fixing $\theta_i^r$ achieves Goal 1; and solving $\theta_i^r$ in Eq. 2 while fixing $t_i^r$ achieves Goal 2. Therefore, we initialize $\theta_i^r$ as the global model $\theta^r$ and alternately optimize $t_i^r$ and $\theta_i^r$ for each mini-batch. Algorithm 1 illustrates our pseudo-code.

### 4.3 Model and Offset Aggregation

The server aggregates both the local models and the offsets from the clients. The model aggregation follows the traditional FL, e.g., the server computes the mean of the clients' local models as the global model like FedAvg [35]. Our offset aggregation leverages the class distribution at each client. Next, we first introduce a metric to measure *distributional heterogeneity* and then our offset aggregation method based on the metric.

**Distributional Heterogeneity.** We define distributional heterogeneity to characterize the class heterogeneity among the clients. Formally, we denote distributional heterogeneity as $DH$ and define it as follows:

$$DH = 1 - \frac{\sum_{j \in [1,N]} c_j}{N \times C}, \tag{3}$$



(a) $DH = 0\%$ (b) $DH = 40\%$ (c) $DH = 60\%$ (d) $DH = 100\%$

Fig. 3: Different distributional heterogeneity levels on CIFAR-10.

where $N$ is the total number of classes, $C$ is the total number of clients, and $c_j$ is defined as follows:

$$c_j = \begin{cases} 0, & \text{if only one client has data from class j,} \\ k, & \text{if } k > 1 \text{ clients have data from class j.} \end{cases} \tag{4}$$

Our defined $DH$ has a value between 0 and 100%. In particular, $DH = 0\%$ means that each client has data from the $C$ classes, e.g., the clients' local data are i.i.d., while $DH = 100\%$ means that each class of data belongs to only one client, i.e., an extreme non-i.i.d. setting. Figure 3 shows examples of different levels of distributional heterogeneity visualized by heatmaps for clients' local data in our experiments on CIFAR-10. We list clients on the x-axis and classes on the y-axis; and each cell is the fraction of the data from the corresponding class that are on the corresponding client.

### 4.4 Offset Aggregation Methods

DisTrans aggregates clients' offsets based on distributional heterogeneity. Intuitively, when the distributional heterogeneity is very large, the offset of one client may not be informative for the offset of another client, as their data distributions are substantially different. Therefore, we aggregate clients' offsets only if the distributional heterogeneity is smaller than a threshold (we set the threshold to be 50% in experiments).

Suppose the distributional heterogeneity of an FL system is smaller than the threshold. One naive way to aggregate the clients' offsets is to compute their average as a global offset, which is sent back to all clients. However, such naive aggregation method uses the same global offset for all clients, which achieves suboptimal accuracy as shown in our experiments. Therefore, we propose a neural network based aggregation method, which produces different aggregated offsets for the clients. Specifically, the server maintains a neural network, which takes a client-specific embedding vector $e \in \mathbb{R}^{1\times N}$ and a client's offset as input and outputs an aggregated offset for the client, where an entry $e_i$ of the embedding vector is the fraction of the training data in class $i$ that are on the client.

The server learns the offset aggregation network by treating it as a regression problem during the FL training process. Specifically, in each round of DisTrans,

the server collects a set of pairs $(t_i, t'_i)$, where $t_i$ is the offset from client $i$ in the current round, $t'_i$ is the aggregated offset the server outputs for client $i$ in the previous round, and $i = 1, 2, \cdots, C$. The server learns the offset aggregation network by minimizing the $\ell_2$ distance between $t_i$ and $t'_i$, i.e., $\min \sum_{i=1}^{C} ||t_i - t'_i||_2$, using Stochastic Gradient Descent (SGD).

## 5 Experiments

**Hyperparameters.** Our model's architecture is the double-input-channel model as shown in Figure 1. Our default $\alpha$ value is 0.3, number of epochs $E = 1$, and the learning rates for the model and offset optimization are 5e-3 and 1e-3 respectively. Our neural network based offset aggregator's architecture is a single-input-channel generator with four convolutional layers.

**Datasets and Model Architectures.** We use six different datasets in the experiment to show the generality of DisTrans. (i) The BioID [1] dataset contains 1521 gray level images with the frontal view of 23 people's face and eye positions. We keep 20 people's images in a descending order and central-crop the images into 256×256. (ii) The CelebA [33] dataset contains 202,599 face images of 10,177 unique, unnamed celebrities. Due to computation resource limit, we choose images of 50 identities in descending order, central-crop them to 178×178, and then resize to 128×128. (iii) The CH-MNIST [19] dataset contains eight classes of 5,000 histology tiles images (64x64) from patients with colorectal cancer, (iv) The CIFAR-10 [25] dataset contains 60,000 32×32 color images in 10 different classes, we resize them to 64×64, (v)The CIFAR-100 [25] dataset contains 60,000 32×32 color images in 100 different classes, and (vi) Caltech-UCSD Birds-200-2011 [44] (referred as Bird-200. The Bird-200 dataset contains 11,788 image from 200 bird species. Due to computation resource limit, we resize them to 128×128.

Here are the model architectures for each dataset. We use LeNet as the backbone for BioID, AlexNet for CelebA, CH-MNIST and CIFAR-100, ResNet18 and ResNet50 for CIFAR-100, and ResNet18 for Bird200.

**Local Data Distribution.** Our local data distribution ranges from entirely i.i.d. to extreme non-i.i.d., i.e., with distributional heterogeneity value ranging from 0% to 100%. Our data splitting method follows SOTA approach [39]. Specifically, we first assign a specific number of classes $u$ out of total classes $N$ for each client. Then, we sample $s_{i,c} \in (0.4, 0.6)$ for each client $i$ and a selected class $c$, and then assign the client with $\frac{s_{i,c}}{\sum_n s_{n,c}}$ of the samples for the class $c$. We repeat the same process for each client.

### 5.1 Results under Different Data Distributions

We evaluate DisTrans's accuracy with different data distributions and compare with SOTA personalized FL works.

**Extreme non-i.i.d.** The extreme non-i.i.d. setting, following prior work [50], is a setup where each client only has one class (called positive labels), thus being disjointed from each other. The distributional heterogeneity value is thus 100%.

Table 1: DisTrans vs. SOTA under different data distribution. — means that the approach is not applicable under that setting, and DH means distributional heterogeneity (0%: i.i.d. and 100%: extreme non-i.i.d.). We did not evaluate the datasets of BioID and CelebA under other distributional settings due to the relative small number of images per class.

| Dataset | # clients | DH | DisTrans (ours) | FedAvg | pFedMe | pFedHN | MOON | FedAwS |
|---|---|---|---|---|---|---|---|---|
| CH-MNIST | 8 | 0% (i.i.d.) | **0.908** | 0.891 | 0.778 | 0.702 | 0.887 | — |
| | | 50% | **0.907** | 0.892 | 0.834 | 0.871 | 0.894 | — |
| | | 100% | **0.946** | 0.908 | 0.908 | 0.641 | 0.910 | 0.942 |
| CIFAR-10 | 10 | 0% (i.i.d.) | **0.829** | 0.809 | 0.520 | 0.652 | 0.789 | — |
| | | 40% | **0.819** | 0.782 | 0.523 | 0.721 | 0.809 | — |
| | | 60% | **0.846** | 0.751 | 0.673 | 0.785 | 0.798 | — |
| | | 80% | **0.891** | 0.702 | 0.736 | 0.869 | 0.794 | — |
| | | 100% | **0.860** | 0.726 | 0.751 | 0.629 | 0.813 | 0.829 |
| CIFAR-100 | 10 | 0% (i.i.d.) | **0.533** | 0.531 | 0.020 | 0.354 | 0.532 | — |
| | | 40% | **0.586** | 0.538 | 0.018 | 0.492 | 0.564 | — |
| | | 60% | **0.646** | 0.523 | 0.017 | 0.604 | 0.628 | — |
| | | 80% | **0.734** | 0.461 | 0.013 | 0.669 | 0.709 | — |
| | | 100% | **0.834** | 0.524 | 0.015 | 0.469 | 0.820 | — |
| Bird-200 | 10 | 0% (i.i.d.) | **0.556** | 0.518 | 0.018 | 0.053 | 0.523 | — |
| | | 40% | **0.548** | 0.521 | 0.015 | 0.064 | 0.528 | — |
| | | 60% | **0.542** | 0.528 | 0.012 | 0.086 | 0.532 | — |
| | | 80% | **0.565** | 0.524 | 0.010 | 0.125 | 0.550 | — |
| | | 100% | **0.641** | 0.549 | 0.014 | 0.309 | 0.621 | — |
| BioID | 20 | 100% | **0.988** | 0.911 | 0.902 | 0.932 | 0.961 | 0.983 |
| CelebA | 50 | 100% | **0.804** | 0.639 | 0.527 | 0.545 | 0.497 | 0.721 |

We single out this setting, because the evaluation metrics are different from other settings given that each client only has positive images. That is, the same amount of negative images (i.e., randomly-selected images from other classes) are introduced in the testing dataset just like prior work [50].

The rows with 100% distributional heterogeneity values in Table 1 show the model's accuracy of DisTrans and the comparison with SOTA works. As shown in those results, DisTrans outperforms all prior works with five different datasets with an improvement ranging from 0.4% to 7.7%. FedAwS is clearly the SOTA, which always performs next to DisTrans, because it is designed for this extreme setting. Due to the negative test images, pFedHN performs poor since the server assigns each client model weights that are trained on only positive images according to its mechanism. FedAvg performs better than we expect because the features of negative examples are aggregated from other clients. We did not evaluate CIFAR-100 or Bird-200 under positive labels scenario (FedAws), since the number of classes per client does not satisfy positive labels setting when # clients equals to 10 for them. Instead, each client is assigned 10 or 20 disjoint classes as the extreme non-i.i.d. case.

Table 2: Comparison with data transformation for CH-MNIST dataset.

| Method | Distributional heterogeneity | | | | |
|---|---|---|---|---|---|
| | 0% | 25% | 50% | 75% | 100% |
| FedAvg | 0.891 | 0.893 | 0.892 | 0.847 | 0.908 |
| DisTrans | **0.908** | **0.904** | **0.907** | **0.905** | **0.946** |
| mixup | 0.896 | 0.895 | 0.882 | 0.839 | 0.901 |
| AdvProp | 0.879 | 0.880 | 0.877 | 0.859 | 0.919 |

**Other Distributional Settings.** Other settings include distributional heterogeneity values ranging from 0% (i.i.d.) to 80% . The evaluation also follows prior FL works [35,43], i.e., each client evaluates testing data with the same classes as its training data. Table 1 also shows the accuracy of DisTrans and four other SOTA works (FedAvg, pFedMe, MOON, and pFedHN). DisTrans outperforms STOA works in every data distribution for all datasets. Note that we do not evaluate FedAwS in these settings because its design is only applicable to the extreme non-i.i.d. setting.

### 5.2 Comparing with Data Transformation

We compare DisTrans with two state-of-the-art, popular data transformation (augmentation) methods, mixup [51] and AdvProp [46]. The former, i.e., mixup, augments training data with virtual training data based on existing data samples and one hot encoding of the label. The latter, i.e., AdvProp, augments training data with its adversarial counterpart. We add both data transformation methods for local training data at each client of FedAvg.

The comparison results are shown in Table 2. DisTrans appears to outperform both mixup and AdvProp in different data distributions from i.i.d. to non-i.i.d. There are two major reasons. First, DisTrans shifts local training and testing data distribution to fit the global model, but existing data transformation only improves training data. Second, DisTrans aggregates the offset based on data distributions, but neither data transformation approaches did so. Another thing worth noting is that mixup improves FedAvg under an i.i.d. setting, but AdvProp improves FedAvg under a non-i.i.d. setting. On one hand, that is likely because virtual examples under a non-i.i.d. setting may introduce further distributional discrepancies, while adversarial examples may help each local model better know the boundary. On the other hand, the distribution is the same under an i.i.d. setting and so does the virtual examples, but different adversarial examples may explore different boundaries at different clients.

### 5.3 Ablation Studies

**Single vs. Double-Input Channel.** We compare the performance of single vs. double-input-channel models to demonstrate the necessity in using the double-input-channel model. Table 3 shows the model's accuracy on three datasets with different distributional heterogeneity values. As shown, the double-input-channel

Table 3: Ablation study on model structures. We adopt different distributional heterogeneity values according to the number of classes in the dataset, i.e., 0%, 25%, 50%, 75%, and 100% for CH-MNIST (8 classes) and 0%, 40%, 60%, 80%, and 100% for CIFAR-10 (10 classes) and Bird-200 (200 classes).

| Dataset | Structure | Distributional heterogeneity | | | | |
|---|---|---|---|---|---|---|
| | | 0% | 40%/25% | 60%/50% | 80%/75% | 100% |
| CH-MNIST | single | 0.874 | 0.871 | 0.872 | 0.874 | 0.889 |
| | double | **0.908** | **0.904** | **0.907** | **0.905** | **0.946** |
| CIFAR-10 | single | 0.775 | 0.802 | 0.785 | 0.796 | 0.811 |
| | double | **0.829** | **0.819** | **0.846** | **0.891** | **0.860** |
| Bird-200 | single | 0.569 | 0.512 | 0.497 | 0.501 | 0.505 |
| | double | **0.556** | **0.548** | **0.542** | **0.565** | **0.641** |

Table 4: Ablation study on aggregation methods. We adopt different distributional heterogeneity values according to the number of classes in the dataset, i.e., 0%, 25%, 50%, 75%, and 100% for CH-MNIST (8 classes) and 0%, 40%, 60%, 80%, and 100% for CIFAR-10 (10 classes) and Bird-200 (200 classes).

| Dataset | Aggregation | Distributional heterogeneity | | | | |
|---|---|---|---|---|---|---|
| | | 0% | 40%/25% | 60%/50% | 80%/75% | 100% |
| CH-MNIST | no agg | 0.868 | 0.887 | 0.907 | 0.905 | 0.946 |
| | avg agg | 0.903 | 0.902 | 0.907 | 0.865 | 0.899 |
| | nn agg | 0.908 | 0.904 | 0.905 | 0.887 | 0.921 |
| | nn+no (default) | **0.908** | **0.904** | **0.907** | **0.905** | **0.946** |
| CIFAR-10 | no agg | 0.767 | 0.789 | 0.846 | 0.891 | 0.860 |
| | avg agg | 0.811 | 0.814 | 0.813 | 0.702 | 0.798 |
| | nn agg | 0.829 | 0.819 | 0.799 | 0.743 | 0.839 |
| | nn+no (default) | **0.829** | **0.819** | **0.846** | **0.891** | **0.860** |
| Bird-200 | no agg | 0.501 | 0.522 | 0.542 | 0.565 | 0.641 |
| | avg agg | 0.526 | 0.529 | 0.525 | 0.515 | 0.489 |
| | nn agg | 0.556 | 0.548 | 0.551 | 0.532 | 0.513 |
| | nn+no (default) | **0.556** | **0.548** | **0.551** | **0.565** | **0.641** |

model always outperforms the single-input-channel with around 3%–9% accuracy improvement on three different datasets.

**Different Offset Aggregations.** We compare different offset aggregation methods, i.e., no aggregation, average aggregation and neural network (NN) based aggregation, on three datasets with various distributional heterogeneity values. Table 4 shows the comparison results. No aggregation performs best when the distributional heterogeneity is greater than 50%, and NN aggregation performs the best when the distributional heterogeneity is smaller than 50%. Average aggregation always performs worse than the other two. This motivates the design of DisTrans in adopting no aggregation for greater than 50% distributional heterogeneity and NN aggregation for less than 50% distributional heterogeneity, which is the "nn+no (default)" row in Table 4.

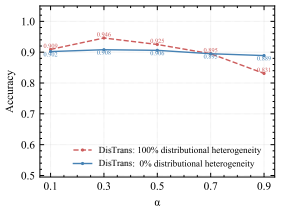



Fig. 4: Accuracy vs. $\alpha$ for CH-MNIST.

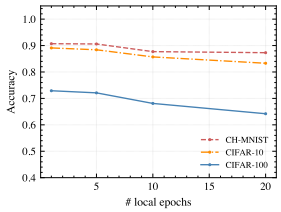



Fig. 5: Accuracy vs. # of local epochs.

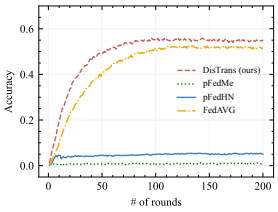



Fig. 6: Accuracy vs. # of rounds for Birds-200.

**Different $\alpha$ Values.** We evaluate top-1 accuracy of DisTrans with different $\alpha$ values, and 0% and 100% distributional heterogeneity to justify why we choose 0.3 as $\alpha$. Figure 4 shows the results. The accuracy with 100% distributional heterogeneity is more sensitive to $\alpha$ than that with 0%. In both data distributions, the accuracy is the highest when $\alpha$ equals 0.3. The reason is as follows. When $\alpha$ is small, the offset is too weak to shift the distribution. When the $\alpha$ is large, the offset is too strong in overriding the original data distribution.

**Different Local Epochs.** We study different local training epochs for each round. Figure 5 shows the accuracy for CH-MNIST, CIFAR-100, and Bird-200 with epochs from 1, 5, 10, to 20. The accuracy is the highest with the local epoch as 1, and decreases when the epoch increases. The reason is too much local training makes offsets become overfitted to local data.

**Convergence Rate.** We study the convergence rate of three SOTA works and DisTrans in terms of communication rounds between the server and local clients. Figure 6 shows the number of communication rounds as the x-axis and the model's accuracy as the y-axis for the Birds-200 dataset under the i.i.d. setting (i.e., 0% distributional heterogeneity). There are two things worth noting. First, as shown, the convergence rate of DisTrans is similar to that of FedAvg, which needs approximately 100 rounds. Second, the accuracy of DisTrans is constantly better than FedAvg for each communication between client and server.

### 5.4 Scalability

We study the scalability of DisTrans using two datasets CH-MNIST and CIFAR-100 as the number of FL clients increases. First, we test the number of clients from 8, 16, 24, to 40 using 50% distributional heterogeneity for CH-MNIST. The third column in Table 5 shows the accuracy of four different works including DisTrans as the number of clients increases. The fourth to eighth columns in Table 5 show the total number of rounds to reach certain accuracy. Generally, the convergence needs more rounds with more clients, which aligns with the previous work [35]. Second, we show the testing accuracy in Table 6 for CIFAR-100 (with ResNet18) of 50, 100, and 500 clients. Each client has 10 classes and the sample rate is 0.2 [29]. DisTrans outperforms SOTA by 8.8%–30.7%. Note that the accuracy of DisTrans drops by 8.4% while SOTA drops by 10.8% to 27.6% for 500 clients.

Table 5: Best accuracy and the number of rounds to achieve it vs. different number of clients using the CH-MNIST dataset when reaching listed accuracy, e.g., DisTrans needs 5 rounds to achieve a 0.800 accuracy with 8 clients. (—: the approach cannot reach the accuracy under that setting.)

| | # clients | Best accuracy | # of rounds to achieve | | | | |
|---|---|---|---|---|---|---|---|
| | | | >0.700 | >0.800 | >0.850 | >0.870 | >0.890 |
| DisTrans (ours) | 8 | **0.907** | **2** | **5** | **10** | 36 | **63** |
| | 16 | **0.898** | 8 | 15 | 25 | 51 | **123** |
| | 24 | **0.897** | 12 | 26 | **37** | **68** | **154** |
| | 40 | **0.895** | 14 | 29 | **40** | **87** | **192** |
| FedAVG | 8 | 0.892 | 3 | **5** | 15 | **24** | 78 |
| | 16 | 0.883 | 7 | **12** | **23** | **48** | — |
| | 24 | 0.880 | 10 | **21** | 39 | 74 | — |
| | 40 | 0.878 | 19 | 34 | 44 | 100 | — |
| pFedMe | 8 | 0.834 | 690 | 779 | — | — | — |
| | 16 | 0.805 | 844 | 1,225 | — | — | — |
| | 24 | 0.725 | 1,859 | — | — | — | — |
| | 40 | 0.719 | 3,071 | — | — | — | — |
| pFedHN | 8 | 0.871 | 3 | 8 | 23 | 117 | — |
| | 16 | 0.817 | **6** | 29 | — | — | — |
| | 24 | 0.816 | **4** | 28 | — | — | — |
| | 40 | 0.832 | **5** | **27** | — | — | — |

Table 6: Best accuracy vs. number of clients on CIFAR-100.

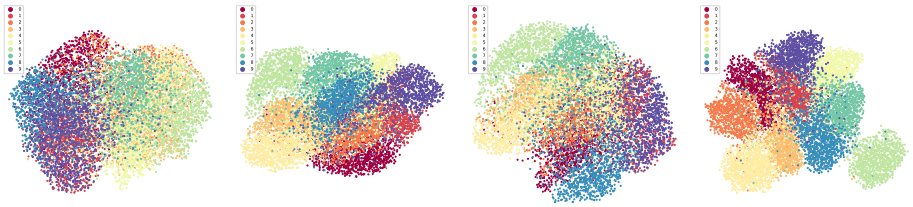
| | DisTrans | | | pFedHN | | | pFedHN-pc | | | MOON | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| #Client | 50 | 100 | 500 | 50 | 100 | 500 | 50 | 100 | 500 | 50 | 100 | 500 |
| Accuracy | **0.729** | **0.681** | **0.645** | 0.614 | 0.538 | 0.338 | 0.623 | 0.541 | 0.372 | 0.615 | 0.593 | 0.507 |

Table 7: Communication overhead of weights and offset for 64x64 RGB images.

| | Baseline (in bytes) | DisTrans (in bytes) | | $\Delta$ overhead |
|---|---|---|---|---|
| | single-input-channel weight | double-input-channel weight | offset | |
| LeNet | 4,346,447 | 4,350,415 | 49,280 | 1.225% |
| AlexNet | 244,449,263 | 244,489,263 | 49,280 | 0.036% |
| ResNet18 | 44,805,709 | 44,826,189 | 49,280 | 0.155% |
| ResNet50 | 94,326,992 | 94,408,912 | 49,280 | 0.139% |

### 5.5 Communication Overhead

We study the communication overhead by calculating the $\Delta$ bytes brought by our double-input-channel weights and offset in each communication round. The comparison baseline used is FedAvg with conventional single-input-channel model. Table 7 shows results for different backbone neural network architectures: The overhead is between 0.036% to 1.225% with an average value of 0.389% for different network architectures because the double-input-channel model only introduces one additional layer and the offsets are small.



(a) FedAvg (b) FedAwS (c) pFedMe (d) DisTrans (ours)

Fig. 7: UMAP visualization of embedded feature representations in the global model for test images in CIFAR-10. DisTrans learns better feature representations than FedAvg, FedAwS, and pFedMe.

### 5.6 Prediction Visualization

We perform an experiment on CIFAR-10 using ten FL clients where each client has data for only one class, and visualize hidden feature representations using Uniform Manifold Approximation and Projection (UMAP) in Figure 7. The model trained using FedAvg learns poor features, which are mixed and indistinguishable. The feature representations of FedAwS and pFedMe also highly overlap. By contrast, the feature representations of DisTrans are well separated in Figure 7d as a result of shifting the local data distributions via personalized offsets.

## 6 Conclusion

FL often needs to contend with client-side local training data with different distributions with high heterogeneity. This paper advances a novel approach, DisTrans, based on distributional transformation, that jointly optimizes local model and data with a personalized offset and then aggregates both at a central server. We perform an empirical evaluation of DisTrans using five different datasets, which shows that DisTrans outperforms SOTA FL and data augmentation methods, under different degrees of data distributional heterogeneity ranging from extreme non-i.i.d. to i.i.d.

## Acknowledgements

This work was supported in part by Johns Hopkins University Institute for Assured Autonomy (IAA) with grants 80052272 and 80052273, and National Science Foundation (NSF) under grants CNS-21-31859, CNS-21-12562, and CNS-18-54001. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or JHU-IAA.

## References


1. Bioid face dataset. https://www.bioid.com/facedb/ 8
2. Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, H.B., et al.: Towards federated learning at scale: System design. arXiv preprint arXiv:1902.01046 (2019) 2
3. Caldas, S., Konečny, J., McMahan, H.B., Talwalkar, A.: Expanding the reach of federated learning by reducing client resource requirements. arXiv preprint arXiv:1812.07210 (2018) 3
4. Cao, X., Fang, M., Liu, J., Gong, N.Z.: Fltrust: Byzantine-robust federated learning via trust bootstrapping. arXiv preprint arXiv:2012.13995 (2020) 3
5. Cao, X., Jia, J., Gong, N.Z.: Provably secure federated learning against malicious clients. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 6885–6893 (2021) 3
6. Chen, M., Poor, H.V., Saad, W., Cui, S.: Convergence time optimization for federated learning over wireless networks. IEEE Transactions on Wireless Communications **20**(4), 2457–2471 (2021). https://doi.org/10.1109/TWC.2020.3042530 3
7. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 2, 4
8. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.: Randaugment: Practical automated data augmentation with a reduced search space. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 18613–18624. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf 2, 4
9. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.: Randaugment: Practical automated data augmentation with a reduced search space. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 18613–18624. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf 2, 4
10. Fang, M., Cao, X., Jia, J., Gong, N.: Local model poisoning attacks to byzantine-robust federated learning. In: 29th USENIX Security Symposium (USENIX Security 20). pp. 1605–1622 (2020) 3
11. Guo, P., Wang, P., Zhou, J., Jiang, S., Patel, V.M.: Multi-institutional collaborations for improving deep learning-based magnetic resonance image reconstruction using federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2423–2432 (June 2021) 3
12. Haddadpour, F., Mahdavi, M.: On the convergence of local descent methods in federated learning. arXiv preprint arXiv:1910.14425 (2019) 3
13. Hamer, J., Mohri, M., Suresh, A.T.: Fedboost: A communication-efficient algorithm for federated learning. In: International Conference on Machine Learning. pp. 3973–3983. PMLR (2020) 3
14. Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., Ramage, D.: Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604 (2018) 2
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016) 2, 4

16. Hsu, T.M.H., Qi, H., Brown, M.: Federated visual classification with real-world data distribution (2020) 3
17. Jin, Y., Jiao, L., Qian, Z., Zhang, S., Lu, S., Wang, X.: Resource-efficient and convergence-preserving online participant selection in federated learning. In: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS). pp. 606–616 (2020). https://doi.org/10.1109/ICDCS47774.2020.00049 3
18. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019) 1
19. Kather, J.N., Weis, C.A., Bianconi, F., Melchers, S.M., Schad, L.R., Gaiser, T., Marx, A., Zöllner, F.G.: Multi-class texture analysis in colorectal cancer histology. Scientific reports **6**(1), 1–11 (2016) 8
20. Kim, H., Park, J., Bennis, M., Kim, S.L.: Blockchained on-device federated learning. IEEE Communications Letters **24**(6), 1279–1283 (2020). https://doi.org/10.1109/LCOMM.2019.2921755 3
21. Kim, I., Kim, Y., Kim, S.: Learning loss for test-time augmentation. In: Proceedings of Advances in Neural Information Processing Systems (2020) 2, 4
22. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016) 3
23. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016) 3
24. Konečný, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence (2016) 3
25. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep. (2009) 8
26. Laguel, Y., Pillutla, K., Malick, J., Harchaoui, Z.: Device heterogeneity in federated learning: A superquantile approach. arXiv preprint arXiv:2002.11223 (2020) 2
27. Lemley, J., Bazrafkan, S., Corcoran, P.: Smart augmentation learning an optimal data augmentation strategy. IEEE Access **5**, 5858–5869 (2017). https://doi.org/10.1109/ACCESS.2017.2696121 2, 4
28. Li, D., Wang, J.: Fedmd: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581 (2019) 2
29. Li, Q., He, B., Song, D.: Model-contrastive federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021) 3, 12
30. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine **37**(3), 50–60 (2020). https://doi.org/10.1109/MSP.2020.2975749 1
31. Li, Y., Yu, Q., Tan, M., Mei, J., Tang, P., Shen, W., Yuille, A.L., Xie, C.: Shape-texture debiased neural network training. CoRR **abs/2010.05981** (2020), https://arxiv.org/abs/2010.05981 2, 4
32. Liu, W., Chen, L., Chen, Y., Zhang, W.: Accelerating federated learning via momentum gradient descent. IEEE Transactions on Parallel and Distributed Systems **31**(8), 1754–1766 (2020). https://doi.org/10.1109/TPDS.2020.2975189 3
33. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV) (December 2015) 8

34. Luo, B., Li, X., Wang, S., Huang, J., Tassiulas, L.: Cost-effective federated learning design. In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications. pp. 1–10. IEEE (2021) 3
35. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017) 1, 2, 3, 6, 10, 12
36. Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 739–753. IEEE (2019) 3
37. Nguyen, H.T., Sehwag, V., Hosseinalipour, S., Brinton, C.G., Chiang, M., Vincent Poor, H.: Fast-convergent federated learning. IEEE Journal on Selected Areas in Communications **39**(1), 201–218 (2021). https://doi.org/10.1109/JSAC.2020.3036952 3
38. Pérez, J.C., Alfarra, M., Jeanneret, G., Rueda, L., Thabet, A., Ghanem, B., Arbeláez, P.: Enhancing adversarial robustness via test-time transformation ensembling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021) 2, 4
39. Shamsian, A., Navon, A., Fetaya, E., Chechik, G.: Personalized federated learning using hypernetworks. In: Proceedings of the 38th International Conference on Machine Learning (ICML), PMLR 139 (2021) 2, 3, 8
40. Shanmugam, D., Blalock, D.W., Balakrishnan, G., Guttag, J.V.: Better aggregation in test-time augmentation. In: Proceedings of International Conference on Computer Vision (ICCV) (2021) 2, 4
41. Suresh, A.T., Felix, X.Y., Kumar, S., McMahan, H.B.: Distributed mean estimation with limited communication. In: International Conference on Machine Learning. pp. 3329–3337. PMLR (2017) 3
42. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015) 2, 4
43. T Dinh, C., Tran, N., Nguyen, T.D.: Personalized federated learning with moreau envelopes. Advances in Neural Information Processing Systems **33** (2020) 2, 3, 10
44. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011) 8
45. Wang, J., Xu, Z., Garrett, Z., Charles, Z., Liu, L., Joshi, G.: Local adaptivity in federated learning: Convergence and consistency (2021) 3
46. Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A.L., Le, Q.V.: Adversarial examples improve image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 2, 3, 4, 10
47. Xu, J., Glicksberg, B.S., Su, C., Walker, P., Bian, J., Wang, F.: Federated learning for healthcare informatics. Journal of Healthcare Informatics Research **5**(1), 1–19 (2021) 3
48. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019) 1
49. Yao, X., Huang, T., Wu, C., Zhang, R., Sun, L.: Towards faster and better federated learning: A feature fusion approach. In: 2019 IEEE International Conference on Image Processing (ICIP). pp. 175–179 (2019). https://doi.org/10.1109/ICIP.2019.8803001 3

50. Yu, F., Rawat, A.S., Menon, A., Kumar, S.: Federated learning with only positive labels. In: International Conference on Machine Learning. pp. 10946–10956. PMLR (2020) 2, 3, 8, 9
51. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (2018) 3, 4, 10
52. Zhang, H., Moustapha Cisse, Yann N. Dauphin, D.L.P.: mixup: Beyond empirical risk minimization. International Conference on Learning Representations (ICLR) (2018), https://openreview.net/forum?id=r1Ddp1-Rb 2, 4
53. Zhang, X., Wang, Q., Zhang, J., Zhong, Z.: Adversarial autoaugment (2019) 2, 4