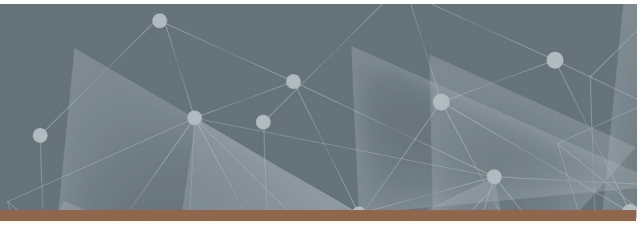
CHALMERS
UNIVERSITY OF TECHNOLOGY



Weighted Ensemble Distillation in Federated Learning with Non-IID Data

Leveraging auxiliary data to efficiently aggregate models in heterogeneous settings

Master's thesis in Complex Adaptive Systems

OSCAR ERIKSSON

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Weighted Ensemble Distillation in Federated Learning with Non-IID Data

Leveraging auxiliary data to efficiently aggregate models in heterogeneous settings

OSCAR ERIKSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2022

Weighted Ensemble Distillation in Federated Learning with Non-IID Data
Leveraging auxiliary data to efficiently aggregate models in heterogeneous settings
OSCAR ERIKSSON

Supervisors at company:
Mattias Åkesson, Data Scientist, Scaleout Systems
Fredrik Wrede, Data Scientist, Scaleout Systems

Supervisor and examiner:
Tobias Gebäck, Associate Professor, Department of Mathematical Sciences

Master's Thesis 2022
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Weighted Ensemble Distillation in Federated Learning with Non-IID Data
Leveraging auxiliary data to efficiently aggregate models in heterogeneous settings
OSCAR ERIKSSON
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Federated distillation (FD) is a novel algorithmic idea for federated learning (FL) that allows clients to use heterogeneous model architectures. This is achieved by distilling aggregated local model predictions on an unlabeled auxiliary dataset into a global model. While standard FL algorithms are often based on averaging local parameter updates over multiple communication rounds, FD can be performed with only one communication round, giving favorable communication properties when local models are large and the auxiliary dataset is small. However, both FD and standard FL algorithms experience a significant performance loss when training data is not independently and identically distributed (non-IID) over the clients. This thesis investigates weighting schemes to improve the performance with FD in non-IID scenarios. In particular, the sample-wise weighting scheme FedED-w2 is proposed, where client predictions on auxiliary data are weighted based on the similarity with local data. Data similarity is measured with the reconstruction loss on auxiliary samples when passed through an autoencoder (AE) model that is trained on local data. Image classification experiments with convolutional neural networks performed in this study show that FedED-w2 exceeds the test accuracy of FL baseline algorithms with up to 15 % on the MNIST and EMNIST datasets for varying degrees of non-IID data over 10 clients. The performance of FedED-w2 is lower than FL baselines on the CIFAR-10 dataset, where the experiments display up to 5 % lower test accuracy.

Keywords: federated learning, federated distillation, knowledge distillation, weighted ensembles, artificial intelligence, privacy, image classification.

Acknowledgements

I would like to thank everyone at Scaleout Systems for giving me the opportunity to explore federated learning and its challenges. A special thanks to Mattias Åkesson and Fredrik Wrede for your feedback and help to initiate the project. I would also like to thank AI Sweden for providing computational hardware and a welcoming community for us master thesis students. Finally, I want to thank Tobias Gebäck, my supervisor and examiner at Chalmers University of Technology, for the feedback and guidance.

Oscar Eriksson, Enköping, May 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order.

AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
CE	Cross Entropy
CNN	Convolutional Neural Network
FD	Federated Distillation
FL	Federated Learning
GDPR	General Data Protection Regulation
IID	Independently and Identically Distributed
IoT	Internet of Things
ML	Machine Learning
MSE	Mean Squared Error
ReLU	Rectified Linear Unit

Nomenclature

Below is the nomenclature of indices, sets, functions, parameters and variables that have been used throughout this thesis.

Indices

k	Index for client
i,j	Indices for data samples
c	Index for class
l	Index for image pixel
t	Index for communication round

Sets

$\mathcal{D}$	Set of traning data
$\mathcal{D}_k$	Set of traning data for client k
$\mathcal{D}_0$	Set of auxiliary data
$\mathcal{I}_k$	Set of indices for data samples at client k
$\mathcal{K}$	Set of clients
$\mathcal{B}$	Set of batches from local data

Functions

f_k	Function representing the local model at client k
f_{S}	Function representing the student model
L_k	Local objective function for client k
$\mathcal{L}$	Global objective function
$\mathcal{L}_{\mathrm{S}}$	Student loss function
$\mathcal{L}_{\mathrm{KL}}$	Kullback-Leibler divergence loss function

$\mathcal{L}_{\text{MSE}}$	Mean squared error loss function
$\mathcal{L}_{\text{REC}}$	Reconstruction loss function
σ	Activation function

Parameters

N	Number of data samples
N	Number of data samples at client k
N_k^c	Number of data samples at client k in class c
K	Number of clients
C	Number of classes
E	Number of epochs
B	Batch size
η	Learning rate
α	Dirichlet concentration parameter
μ	Regularization parameter
T	Number of communication rounds
γ	Fraction of participating clients
W	Input width of feature map
O	Output width of feature map
F	Kernel size
P	Padding size
S	Stride size
τ	Temperature parameter

Variables

$\boldsymbol{\theta}_0$	Initial global model weights
$\boldsymbol{\theta}_{t+1}$	Global weights at round $t+1$
$\boldsymbol{\theta}_{t+1}^k$	Local weights for client k at round $t+1$
$\boldsymbol{\theta}_{\text{S}}$	Student model weights
$\boldsymbol{x}_0$	Batch of auxiliary data
$\boldsymbol{z}_k$	Model output from client k
w_k	Ensemble weight for client k
w_k^j	Ensemble weight for client k at auxiliary sample with index j

$\boldsymbol{x}^{(i)}$	Data sample i
$y^{(i)}$	True label for data sample i
$\boldsymbol{z}_{\mathrm{T}}$	Aggregated predictions from clients (teacher predictions)
$\boldsymbol{z}_{\mathrm{S}}$	Student predictions
$\boldsymbol{z}_k$	Predictions from client k
$\bar{\boldsymbol{y}}$	One-hot encoded true label
b	Bias term in neural network model
$\boldsymbol{q}, \boldsymbol{v}$	Temperature weighted class probabilities
b	Communication cost
$\boldsymbol{h}$	Latent variable

Contents

List of Figures

List of Tables

1

Introduction

In the last decade, significant progress has been made in the field of machine learning and AI. Increased computing capabilities, improved algorithms and datasets have made it possible to perform tasks such as image classification, object detection and speech recognition with models that learn from data. Models used for such tasks often thrive on big datasets, which might need to be built by collecting data at different locations and user devices. In the standard case of machine learning, the *centralized* case, this distributed data is sent to a central server where the machine learning model is trained. However, with increasing privacy concerns among the public, it has become undesirable and in some cases even illicit to move users' private data to a central location. A recent study show that 9 out of 10 Americans are concerned about their online privacy, which clearly demonstrates the hesitation toward sharing private information [1]. As integrity becomes a more important question regarding private data, new regulations have been introduced to protect the users' privacy. An example of such a privacy act is GDPR [2], which regulates how companies and organizations are allowed to manage user data.

A technique proposed to overcome this privacy challenge in machine learning is federated learning (FL) [3]. Since the data cannot be stored centrally, FL moves the model training to the client where the private data is present. A global model is then formed by combining information from locally trained models, without accessing the private data. Aggregating the local models into a global model is a crucial step for the end performance, and one of the many open challenges in FL [4]. The standard algorithm for this is FedAvg [3], which iteratively averages over the local model parameters into a global model. This is a simple approach to aggregate the local models, but it has some drawbacks. Previous work has shown that FedAvg converges to the centrally trained model when sampled client data is independently and identically distributed (IID) over the clients [3, 5]. Unfortunately, the IID assumption does not always hold in practice, and when client data is non-IID, the performance of FedAvg decreases drastically [6]. Furthermore, FedAvg requires all clients to train on identical model architectures, making it less flexible for system heterogeneity.

The main constraint in FL is that client data needs to be kept private. However, in many real-world applications, there are publicly available auxiliary data with distributions similar to client data. For instance, many FL solutions for computer vision and natural language processing could obtain auxiliary data from the public

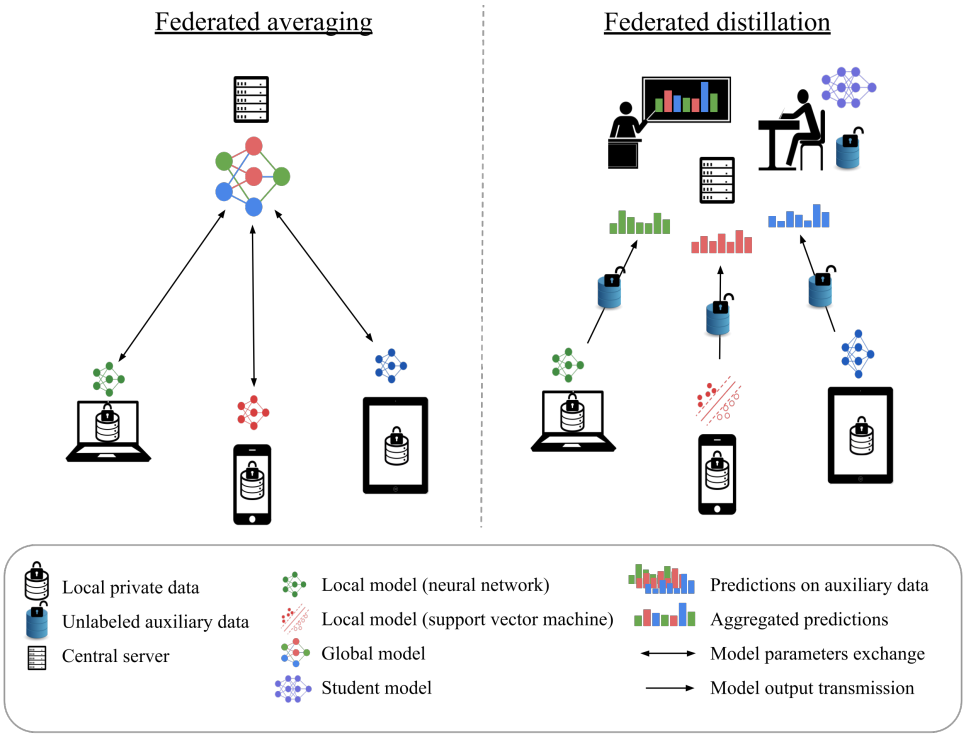


Figure 1.1: A schematic illustration of FD compared to the standard FedAvg algorithm. FedAvg exchange model updates based on local training on private data, whereas FD transmits local model predictions on unlabeled auxiliary data.

databases ImageNet [7] or WikiText [8]. These databases contain millions of data samples but might lack the necessary label information for the specific FL problem. In recent years, a novel algorithmic method called federated ensemble distillation, or federated distillation (FD), has been proposed for FL problems where such auxiliary data is available. In contrast to the parameter averaging algorithm FedAvg and its recently modified versions [9–12], FD allows for clients to train heterogeneous model architectures since clients only share predictions on the auxiliary data, which are used to train a separate model on the central server (Figure 1.1). This flexibility is beneficial for cases when clients are running heterogeneous systems, making it possible to choose a model that suits the clients specific system or computational capabilities. FD also has the potential to reduce the overall communication in FL, since communication scale with the size of the auxiliary dataset rather than the model size. Lastly, since it has been shown that local data can be partially reconstructed based on parameter or gradient updates [13], FD could be a more privacy-preserving alternative.

1.1 Problem formulation

FD-based algorithms have potential advantages in regards to system heterogeneity, communication and privacy properties compared to algorithms based on parameter averaging. However, as with FedAvg, an FD-based algorithm is not immune to performance drops in the non-IID scenario. One way to approach this problem is

to use a weighting scheme so the impact of each local model is weighted differently towards the global model. The problem is the following: Consider a set of clients $\mathcal{K}$, where each client $k \in \mathcal{K}$ holds a private dataset $\mathcal{D}_k$, a shared auxiliary dataset $\mathcal{D}_0$ and a machine learning model $f_k(\boldsymbol{\theta}_k)$ with parameters $\boldsymbol{\theta}_k$. Each private dataset consist of feature-label pairs $(\boldsymbol{x}^{(i)}, y^{(i)})$ and the local parameters have been adjusted to minimize the local objective function L_k on this data. Let $\boldsymbol{z}_k = f_k(\boldsymbol{x}_0; \boldsymbol{\theta}_k)$ be the output from client k on a batch of samples $\boldsymbol{x}_0 \in \mathcal{D}_0$. To perform distillation, the aggregated predictions,

$$\boldsymbol{z}_{\mathrm{T}} = \sum_{k \in \mathcal{K}} w_k \boldsymbol{z}_k, \tag{1.1}$$

referred to as *teacher* predictions, are used to train a *student* model f_{S} on the data pairs $(\boldsymbol{x}_0, \boldsymbol{z}_{\mathrm{T}})$ with the goal of capturing the collective behavior of the local models. The weights w_k allow to control the impact of each client and is typically chosen to $1/|\mathcal{K}|$ when data is assumed IID. In the non-IID scenario, w_k would need to be chosen differently due to the varying performance of local models. This leads to the main goal of this project, which is to investigate different weighting schemes w_k to improve performance in non-IID scenarios with FD.

1.2 Purpose and research questions

This study intends to implement a test framework to compare FD against standard FL algorithms based on parameter averaging in non-IID settings. Both FD and parameter averaging algorithms suffer significant performance loss when client data is non-IID, but it is not established whether one method outperforms the other in this scenario. The first research question for this project is therefore:

> *How does the performance of FD compare to standard FL algorithms in non-IID settings?*

Using weighting schemes is a way to directly control the impact of each client, which motivates the investigation of suitable weighting schemes to improve FD in non-IID scenarios. The second research question is hence:

> *What is a suitable weighting scheme to improve FD in non-IID settings?*

Implementing FD also introduces the problem of choosing the student model and having enough auxiliary data available. This leads to the final research question:

> *How do the student model and quantity of auxiliary data affect the performance of FD?*

1.3 Delimitations

The primary focus of this study is to investigate the performance of FD compared to standard FL algorithms on different distributions of client data. It is therefore the performance relative to these chosen baselines that are of interest and not the

absolute performance. For this reason, local models have relatively low complexity for most experiments and hyperparameters have only been tuned to some extent.

In the real case FL scenario, local models are trained in parallel at client devices. Since the purpose of this project is to evaluate the algorithmic performances, implementations have been simplified to execute model training sequentially on a single device where the data is pseudo-distributed. This means that system heterogeneity based on different bandwidths and computational power among the clients is not considered.

The algorithms covered in this study are only designed and tested for supervised image classification. It is therefore not certain that the result presented in this study would translate to other machine learning tasks.

1.4 Contributions

This project contributes with extended knowledge on the limits of FD and its relative performance to standard FL algorithms in non-IID settings. Insights on how to improve FD in the presence of data heterogeneity is given by introducing a sample-wise weighting scheme that achieves good performance compared to FL baselines.

2

Theory and related work

This chapter will explain the theory and current research related to the problem addressed in this thesis. The first section will cover the essentials of image classification with supervised machine learning and what kind of models will be used in this project. This is then followed by an overview of FL where the main application areas and challenges are discussed. The algorithmic paradigm FD is then introduced, which sets the foundation for the algorithm that is investigated in this project. Lastly, a literature review is given on related FD studies.

2.1 Supervised image classification

Machine learning (ML) is a branch of artificial intelligence (AI) that covers the use of data and algorithms to enable computers to perform certain tasks without being explicitly programmed. This project will consider the task of image classification, using supervised learning. In this case, data is available in the form

$$(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(N)}, y^{(N)}), \tag{2.1}$$

where $y^{(i)}$ is the label (or class) for image $\boldsymbol{x}^{(i)}$. The goal is to learn a function $f(\boldsymbol{x}; \boldsymbol{\theta}) \approx y$ with parameters $\boldsymbol{\theta}$ that predicts a label y based on input image $\boldsymbol{x}$ as accurately as possible. This is typically done by tuning the parameters $\boldsymbol{\theta}$ for a chosen function $f(\boldsymbol{x}; \boldsymbol{\theta})$, using the optimization problem

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \quad \text{with} \quad L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} l\left(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}\right), \tag{2.2}$$

where $l\left(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}\right)$ is the loss function evaluated on $(\boldsymbol{x}^{(i)}, y^{(i)})$. The loss function should be defined such that correct predictions by $f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ results in a low value, and a high value for incorrect predictions. The predictions from $f(\boldsymbol{x}; \boldsymbol{\theta})$ are usually given as an output vector $\boldsymbol{z}$, where each element z_c corresponds to the probability of sample $\boldsymbol{x}$ belonging to class $c \in \{1, \ldots, C\}$. Similarly, the true labels y are represented with one-hot encoding $\bar{\boldsymbol{y}}$, where all elements are zero except element y, which is one. This label representation is used in the cross-entropy (CE) loss

$$l\left(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),y^{(i)}\right)=\sum_{c=1}^{C}-\bar{y}_c^{(i)}\log z_c^{(i)}, \tag{2.3}$$

which is the default choice of loss function for classification problems in supervised ML, and will be used throughout this thesis.

2.1.1 Gradient descent

With a loss function defined, the next question is how to minimize the loss function in (2.2). In the ML context, the method chosen to minimize the loss function is called an optimizer. A common technique to find the minimum or maximum of a function is to find where the derivative of the function is zero. In the gradient descent optimizer, a minimum is searched for by taking steps in the negative direction of the derivative ∇L with respect to $\boldsymbol{\theta}$. This is done by calculating ∇L for a number of samples, and then updating the parameters according to

$$\boldsymbol{\theta}:=\boldsymbol{\theta}-\eta\nabla L \quad \text{where} \quad \nabla L=\left(\frac{\partial L}{\partial\theta_1},\ldots,\frac{\nabla L}{\partial\theta_{|\boldsymbol{\theta}|}}\right) \tag{2.4}$$

Here, η is the chosen step size, known as the learning rate in ML. With large datasets it can be computationally infeasible to perform gradient descent on the complete dataset in one iteration. This can be solved by doing gradient descent on one batch of data at a time. This is referred to as batch gradient descent, or mini-batch gradient descent when the batch size is fairly small (roughly between 10 and 10^3). This is typically done by shuffling the data samples before splitting it into batches and proceeding with gradient descent. Many optimizers have been developed to further accelerate the training with gradient descent. A popular optimizer is Adam [14], which is one of many improved optimizers that introduce an adaptive learning rate.

2.2 Convolutional neural networks

What remains to be defined in the optimization objective (2.2) is the model $f(\boldsymbol{x};\boldsymbol{\theta})$. A popular choice of model for image classification is a convolutional neural network (CNN), which will be explained in this section.

2.2.1 Fully connected neural network

CNN's are special types of artificial neural networks (ANN's) that have been developed in particular for data with spatial dimensions. The main building block in an ANN is the neuron, which applies a non-linear activation function $\sigma(\cdot)$ on the dot product between input $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$, plus a bias term b, as

$$f(\boldsymbol{x};\boldsymbol{\theta})=\sigma(\boldsymbol{x}\cdot\boldsymbol{\theta}+b). \tag{2.5}$$

The neuron computation is illustrated in Figure 2.1.

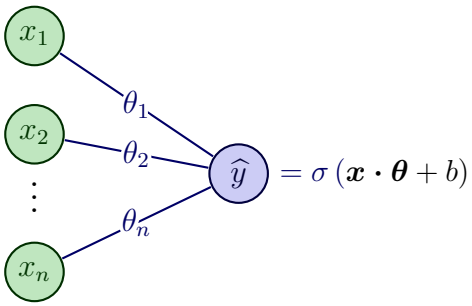


Figure 2.1: Schematic view of a neuron with n input variables.

A fully connected ANN can be formed by stacking neurons in multiple layers, where each neuron has its specific weights connected to each neuron from the previous layer. With more neurons and layers, the complexity of the network increases, which gives the model a higher ability to learn non-linear patterns. However, this is only possible if non-linear activation functions are used in the network. A popular choice of activation function for the intermediate layers is the rectified linear unit (ReLU), $\sigma(x) = \max(0, x)$. For the final layer, a softmax function $\sigma(\boldsymbol{x})_c = e^{x_c} / \sum_{j=1}^{C} e^{x_j}$ for $c = 1, \ldots, C$ is the default choice for classification problems. Intermediate layers in an ANN are usually called hidden layers, as indicated in Figure 2.2 where an example of a fully connected ANN is presented.

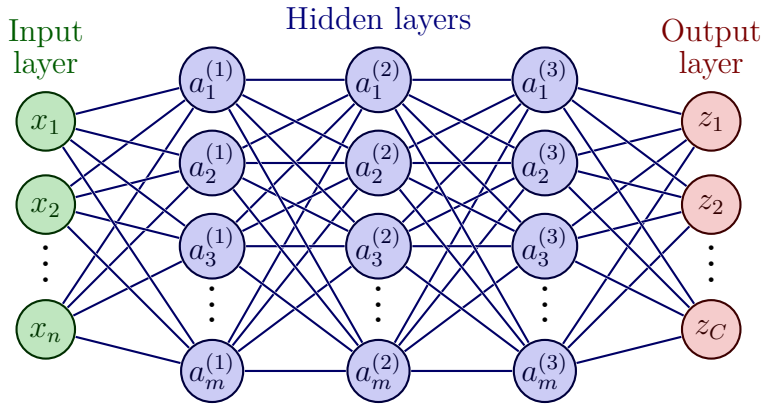


Figure 2.2: Schematic view of an ANN with 3 hidden layers.

2.2.2 2D convolutional layer

When using a fully connected ANN with images as input data, each pixel in an image would have a weight connected to each neuron in the next layer. This can result in lots of parameters and lead to inefficient training of the network. This is one motivation for the introduction of CNN's where the number of parameters are reduced by using a convolutional kernel operation. This operation can be done for various dimensions but this project only considers the 2-dimensional case. The kernel holds a number of weights, determined by the kernel size K, and iteratively do

the neuron computation (2.5) on a $K \times K$ window of the input data. Since the kernel weights are shared over all input data, the number of parameters are significantly reduced. The complexity of a 2D convolutional layer can then be increased by having multiple kernels with separate weights.

The output from a convolutional layer is usually called a feature map. The spatial width O of the feature map is determined by the width W of input data, kernel size F, padding P and stride S, according to $O = (W - F + 2P)/S + 1$. This formula assumes that all sizes spans equally in both dimensions. An example of a convolutional kernel computation is illustrated in Figure 2.3.

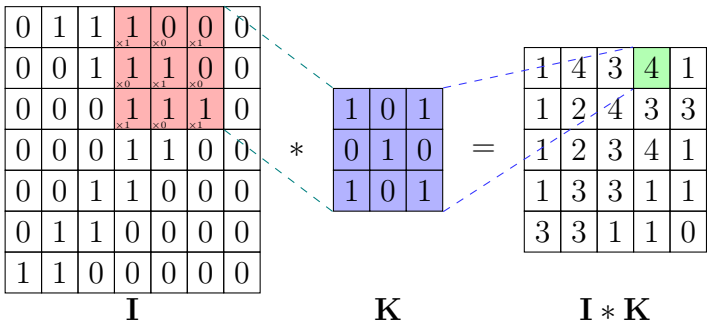


Figure 2.3: Output values from a 2D-convolution of input **I** and kernel **K** with $W = 5, F = 3, P = 0$ and $S = 1$.

2.2.3 Pooling layer

In CNN's, convolutional layers are usually followed by a pooling layer. A pooling layer is used to reduce the spatial dimension of the feature map. This means that the computational cost of training the network is reduced since fewer parameters are required in subsequent layers. A pooling layer also allows the following convolutional layer to extract features at a different resolution than previous layers. A commonly used pooling layer is max pooling, where the output is maximum values for patches of the preceding feature map, see Figure 2.4. Pooling layers do not contain any parameters and are therefore not trainable layers.

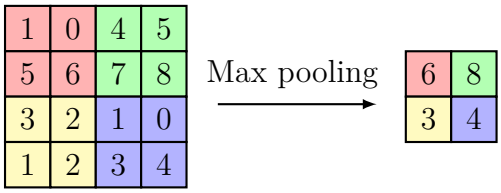


Figure 2.4: Output values from a max pooling operation with $W = 4, F = 2$ and $S = 2$ and $P = 0$.

2.3 Federated learning

Due to the growing computational power on our devices, such as mobile phones, wearable technology and autonomous vehicles, it has become more attractive to move computations previously done on a central server to the edge device. In the case of training ML models, the rising public concern for privacy makes it undesirable to move client data to the central server, which also motivates the interest in moving the model training to the devices. These two motives have together led to a growing interest in a decentralized and collaborative approach to machine learning, called *federated learning*. The term federated learning was coined in 2016 by McMahan et al. [3] and given a broader definition by Kairouz et al. [4] as:

> **Federated learning** is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective.

FL was initially introduced with a focus on applications for mobile phones and edge devices, referred to as the cross-device case. Since then, interest has also been shown for applications with a smaller number of clients involved, for example, a collaboration between multiple companies or organizations, referred to as the cross-silo case. Both of these cases have many possible applications, some of which are discussed below.

Mobile phones. Machine learning models can be trained for next-word prediction, face detection and voice recognition by utilizing data from several mobile phones [15–17]. Due to privacy concerns, users may not want to share their data. FL has the potential to transform the development of these personalized and assistive features on mobile phones to be more privacy-preserving, which has already been applied in consumer digital products. One example of this is Google's keyboard for mobile phones (Gboard mobile keyboard) [15, 16], which uses FL to build models for next word suggestions. The same technique is also integrated with Android Messages to give better and more personalized smart features, such as Smart Reply and other assistive suggestions [18]. Apple has also introduced cross-device FL in iOS 13 when training models for personalized suggestions in the QuickType keyboard and voice classification in the application "Hey Siri" [17].

Organizations. A client, or device, can also be an organization or institution in the FL context. This case has gained great interest in healthcare, since hospitals have big collections of patient data that could aid ML models for predictive healthcare. Hospitals have strict privacy rules that require patient data to remain inside the hospital. FL has the potential to solve this problem for predictive healthcare, as it would allow hospitals to collaboratively train ML models, without exposing sensitive patient data. This is an ongoing field of research in FL, and applications have been proposed for medical data segmentation [19, 20] to predicting clinical outcomes in patients with COVID-19 [21].

Internet of things. Autonomous vehicles, smart homes and wearable devices are all examples of modern IoT systems. These systems can have multiple sensors that allow them to adjust their behavior based on incoming data. This adjustment could be an updated model of traffic or construction for a fleet of autonomous vehicles, which is needed for the vehicles to continue operating safely. Due to data privacy and limited connectivity of devices, it might be a problem to safely aggregate models in these scenarios. FL could be used in this scenario to train models that adjusts to system changes and keeps user data private [22]. Another possible advantage of using FL for autonomous vehicles is that the heavy communication of transmitting the big data from the vehicles to the central server can be avoided.

2.3.1 Challenges in federated learning

FL faces many practical challenges, where the main ones are how to secure the privacy of client data, how to address heterogeneities among clients and how to handle expensive communication. These challenges are discussed shortly below. See Kairouz et al. [4] and Li et al. [23] for a more in-depth discussion.

Privacy. FL has partially emerged by the motivation to solve the privacy problem in training ML models, where the solution is to only communicate information from locally trained models. In many FL algorithms, the information shared is model parameters or gradients. However, these algorithms alone cannot guarantee complete protection of client data, since its been shown that input data can be partially reconstructed based on the observed model updates [13]. To this extent, tools like multiparty computation [24] and differential privacy [25] have emerged to enhance the privacy of federated learning. Unfortunately, these methods often come at the cost of reduced model performance.

Data heterogeneity. Each client connected to a federated network has a dataset that is generated by a specific user or organization. In both of these cases, clients may observe or generate different data points based on , e.g., their location, time zone or user preferences. These differences could be present between local label or feature distributions, as well as differences in the quantity of data [4, Sec. 3.1]. All of these scenarios are referenced as cases with non-IID data in FL. It is also possible for the set of active clients to change over time, which introduce another dimension of non-IID. The original goal in FL to train a single global model on the union of client data becomes harder in the non-IID scenario. This is a widely researched topic in FL, and proposed solutions ranges from techniques to balance the local data [26], weight client contributions [27] or to modify existing algorithms, e.g. using probabilistic ML or ensemble methods [28]. However, in some cases training a single global model might not be the right goal. Clients could benefit by having a personalized model for their specific data, e.g. for next word prediction in mobile phones. In this case, non-IID data could be more of a feature than a problem.

Systems heterogeneity. The devices participating in a federated network could use different hardware and have different network connectivity or power available, leading to varying capabilities in regards to storage, computation and communication. Devices can also be unreliable and drop out from the FL process due to bad

connection or low energy. This problem is most relevant for the cross-device scenario, and algorithms developed for this case must therefore be robust to a low amount of participation, heterogeneous hardware and drop-out clients. In the cross-silo case, system heterogeneity might be less of a problem, since a participating device at a company or organization will probably not face the same limited resources.

Communication. Federated networks can potentially consist of millions of devices, e.g. a network of smartphones, and communication in the network might be much slower than the local computations. It is, therefore, necessary to develop communication-efficient algorithms to enable training a model to fit the data generated by the devices. This means that the messages or model updates from the local training process, and the total number of communication rounds, should be minimized.

2.3.2 Federated averaging

The problem in FL is to learn a global model based on data that is partitioned over multiple clients. This global model is learned under the constraint that client data cannot be transmitted to a central server. Instead, local model parameters are communicated. What has become the standard algorithm for this problem is federated averaging (FedAvg) [3], where McMahan et al. rewrites the ML objective (2.2) as

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \frac{N_k}{N} L_k(\boldsymbol{\theta}) \quad \text{with} \quad L_k(\boldsymbol{\theta}) = \frac{1}{N_k} \sum_{i \in \mathcal{I}_k} l_i(\boldsymbol{\theta}), \tag{2.6}$$

where K is the number of clients, $\mathcal{I}_k$ is the set of indices of data examples on client k and $N_k = |\mathcal{I}_k|$. Here, $l_i(\boldsymbol{\theta})$ is an abbreviation for $l(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$. In FedAvg, a selected fraction γ of K clients starts with the same initial model and performs E epochs of gradient descent (2.4) on its private data. The central server then takes a weighted average over the resulting local model parameters, which is then sent to the clients once again for another training round. The details of FedAvg are outlined in Algorithm 1.

If the local data distributions $\mathcal{D}_k$ were formed by distributing data examples uniformly at random over the clients, it would hold that $\mathbb{E}_{\mathcal{D}_k}[L_k(\boldsymbol{\theta})] = \mathcal{L}(\boldsymbol{\theta})$, meaning that minimizing any local objective L_k would be the same as minimizing the global objective $\mathcal{L}$. This IID assumption is typically a criterion for algorithms in distributed optimization, where the partitioned data is initially contained in a central dataset. In contrast, FL scenarios gives no control over the distribution of client data, which could make the IID assumption invalid and complicate the minimization of the global objective (2.6).

Previous studies [6, 29] have shown that local weights tend to diverge in the non-IID setting with FedAvg, which makes the global objective hard to reach when aggregating the local models. One idea to counteract the weight divergence for each client is to add a regularization term towards the latest global weights. This is done in FedProx [11], where the local minimization objective is modified to

Algorithm 1: The FedAvg [3] algorithm. γ is the fraction of participating clients, K is the total number of clients in the federated network, η is the learning rate and $\mathcal{D}_k$ is the local dataset at client k with $N_k = |\mathcal{D}_k|$ and $N = \sum_{k \in \mathcal{K}_t} N_k$.

Server executes:
initialize $\boldsymbol{\theta}_0$
  for each round $t = 1, \ldots, T$ **do**
    $\mathcal{K}_t \leftarrow$ random subset (γ fraction) from K local models
    for each client $k \in \mathcal{K}_t$ **do**
      $\boldsymbol{\theta}_{t+1}^k \leftarrow \text{ClientUpdate}(k, \boldsymbol{\theta}_t)$
    end
    $\boldsymbol{\theta}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{N_k}{N} \boldsymbol{\theta}_{t+1}^k$
  end

ClientUpdate$(k, \boldsymbol{\theta})$**:**
  $\mathcal{B} \leftarrow$ (split $\mathcal{D}_k$ into batches of size B)
  for each local epoch $i = 1, \ldots, E$ **do**
    for each batch $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{B}$ **do**
      $\widehat{\boldsymbol{y}} = f_k(\boldsymbol{x}; \boldsymbol{\theta}_k)$
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla L_k(\widehat{\boldsymbol{y}}, \boldsymbol{y})$
    end
  end
  return $\boldsymbol{\theta}$ to server

$$\min_{\boldsymbol{\theta}} H_k(\boldsymbol{\theta}, \boldsymbol{\theta}_t) = L_k(\boldsymbol{\theta}) + \frac{\mu}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2, \tag{2.7}$$

which only changes the weight update to $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla (L_k(\boldsymbol{\theta}) + \frac{\mu}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}^t\|^2)$ compared to FedAvg (Algorithm 1). Here, μ is hyperparameter that determines the impact of the regularization term. FedProx has been shown to improve the convergence behavior of FL in non-IID settings.

2.4 Federated ensemble distillation

In recent years, a novel algorithmic paradigm for FL problems has been proposed for dealing with many of the challenges described in Section 2.3.1. This method is called federated ensemble distillation, or federated distillation (FD), which applies the knowledge distillation (KD) procedure [30, 31] in a federated setting. KD is a technique that uses an auxiliary dataset to transfer the knowledge of one or multiple ML classifiers into one single classifier. In the centralized setting, this can be done before deployment to reduce model size and complexity. This process is often explained in terms of a teacher-student analogy, where the goal is to transfer knowledge from the teacher model to a student model.

2.4.1 The algorithm

There exist many variations on FD algorithms in the literature. In this project, the simplest approach is considered where only one communication round of local predictions is needed, referred to as FEDED, which is then used to perform KD at the central server. This procedure starts by letting each client train a local model $f_k(\boldsymbol{\theta}_k)$ on its local data $\mathcal{D}_k$. The clients then generate predictions $\boldsymbol{z}_k$ on the unlabeled auxiliary dataset $\mathcal{D}_0$. These predictions are then aggregated at the central server to form the teacher predictions $\boldsymbol{z}_\mathrm{T}$, which are used as targets to train a student model $f_\mathrm{S}(\boldsymbol{\theta}_\mathrm{S})$ with a chosen loss function $\mathcal{L}_\mathrm{S}$ for T distillation steps. The details of FEDED are outlined in Algorithm 2.

Algorithm 2: The FEDED algorithm.

Local Training: Train each local model $f_k(\boldsymbol{\theta}_k)$ with $\mathcal{D}_k$.
for each distillation step $t = 1, \ldots, T$ **do**
  $\mathcal{K}_t \leftarrow$ random subset (γ fraction) from K local models
  $\boldsymbol{x}_0 \leftarrow$ a batch of public data from $\mathcal{D}_0$ with size B
  for each local $k \in \mathcal{K}_t$ **do**
    $\boldsymbol{z}_k \leftarrow f_k(\boldsymbol{x}_0; \boldsymbol{\theta}_k)$
  end
  $\boldsymbol{z}_\mathrm{T} \leftarrow$ aggregate $\{\boldsymbol{z}_k | k \in \mathcal{K}_t\}$
  $\boldsymbol{z}_\mathrm{S} \leftarrow f_\mathrm{S}(\boldsymbol{x}_0; \boldsymbol{\theta}_\mathrm{S})$
  Update: $\boldsymbol{\theta}_\mathrm{S} \leftarrow \boldsymbol{\theta}_\mathrm{S} - \eta \nabla \mathcal{L}_\mathrm{S}(\boldsymbol{z}_\mathrm{S}, \boldsymbol{z}_\mathrm{T})$
end

When choosing the loss function $\mathcal{L}_\mathrm{S}$, it is common to use a measure based on the softmax output, i.e. class probabilities, between teacher and student model, since valuable information can be retrieved from the ratios of these probabilities. In the standard distillation procedure, Hinton et al. [31] softens the class probabilities as

$$\boldsymbol{q}_\mathrm{T}^c = \frac{e^{\boldsymbol{z}_\mathrm{T}^c/\tau}}{\sum_c e^{\boldsymbol{z}_\mathrm{T}^c/\tau}}, \tag{2.8}$$

with a temperature parameter τ, before applying the Kullback-Leibler divergence

$$\mathcal{L}_\mathrm{KL}(\boldsymbol{q}_\mathrm{T}, \boldsymbol{q}_\mathrm{S}) = \sum_c \boldsymbol{q}_\mathrm{T}^c \log \frac{\boldsymbol{q}_\mathrm{T}^c}{\boldsymbol{q}_\mathrm{S}^c}, \tag{2.9}$$

as loss function for training the student model. Here, $\boldsymbol{q}_\mathrm{S}$ is the temperature weighted class probabilities from the student model. When using a high temperature τ, minimizing (2.9) simplifies to minimizing the mean squared error (MSE) [31]

$$\mathcal{L}_\mathrm{MSE}(\boldsymbol{z}_\mathrm{S}, \boldsymbol{z}_\mathrm{T}) = \frac{1}{C} \sum_c (\boldsymbol{z}_\mathrm{S}^c - \boldsymbol{z}_\mathrm{T}^c)^2, \tag{2.10}$$

which can also be used in combination with $\mathcal{L}_\mathrm{KL}$ [32] by summing the losses.

The simplest approach to aggregate the teacher predictions is by averaging:

$$\boldsymbol{z}_{\mathrm{T}} = \frac{1}{|\mathcal{K}_t|} \sum_{k \in \mathcal{K}_t} \boldsymbol{z}_k, \tag{2.11}$$

where $\mathcal{K}_t$ is the set of participating clients for distillation round t. However, this approach deteriorates when local models have varying performance, for example due to data heterogeneity. Ideas on how to improve this aggregation in non-IID scenarios are presented in Section 3.1.

2.4.2 Distillation vs. averaging

Using FD has three major benefits compared to parameter sharing algorithms. These benefits are summarized below.

Model agnostic. In contrast to FedAVG, the FD approach is independent of the model architecture used at each client and therefore allows clients to train models of different type and size. This gives an additional flexibility which could be particularly beneficial when clients are running on heterogeneous systems.

Low communication cost. The distillation process does not require the raw model parameters to be communicated with a central server. Instead, the only information leaving the clients is the softmax output on the auxiliary dataset. This changes the communication cost, which now scales with the size of the auxiliary dataset $|\mathcal{D}_0|$ and the number of classes C as

$$\mathsf{b} \in \mathcal{O}(|\mathcal{D}_0|C), \tag{2.12}$$

instead of the shared model parameters

$$\mathsf{b} \in \mathcal{O}(T|\boldsymbol{\theta}|) \tag{2.13}$$

as in FedAVG. This can give significantly reduced communication when the auxiliary dataset is smaller than the local models, especially since FD can be executed with only one communication round, whereas parameter sharing algorithms often need multiple rounds.

Robustness and privacy. By sharing model parameters, FL algorithms based on parameter sharing leaves the models completely exposed to adversarial or malicious clients that could influence the training. This is a severe privacy vulnerability as the model parameters contain all information a model has about the data its been trained on [33, 34]. FD is potentially a better choice in this aspect, since it is protected against attacks directly on the global model parameters. Furthermore, it would not be possible to reconstruct local data from model predictions, as is possible to some extent when sharing intermediate model updates [13].

2.5 Related work

The most common differences among currently published FD algorithms are the assumptions on auxiliary data, the communication procedure and what information is being sent from the clients to the central server. This project will consider a *one-shot* FD approach where only one round of communication is used between clients and the central server. It is assumed that an unlabeled auxiliary dataset is available, on which clients share its predictions after local training is finished. This approach is also considered by Guha et al. [35] and Gong et al. [36]. Guha et al. compare one-shot FD against different ensemble methods and show that performance of FD approach the ensemble performance as the size of the auxiliary data is increased. However, there is no comparison against FedAvg and no experiments are presented for non-IID settings.

Different approaches have been proposed for dealing with non-IID data using FD algorithms. This project will specifically consider the approach of using a weighting scheme to control each clients' impact on the aggregated predictions. While weighted ensembles have been widely studied for centralized ML [37–39], the weights are often based on each models' performance on a shared validation set, which consequently need to be labeled. Among the published FD algorithms that have been found during this literature review, only two studies considers the use of a weighting scheme to handle non-IID data. In the first one, FedAD by Gong et al. [36], a class-wise weighting scheme is proposed, such that the predictions from client k in class c is weighted by $N_k^c / \sum_k N_k^c$. Their method is shown to outperform FedAvg in non-IID settings, but it is uncertain if this is due to the weighting scheme or the additional sharing of attention maps. This weighting scheme also cause additional privacy concerns, since the label distribution for all clients would need to be stored at each client or the central server.

The second study that considers a weighting scheme in FD is by Sattler et al. [40], who propose the algorithm FedAux where weighting is performed sample-wise by using certainty scores. The certainty scores are based on the loss from a logistic regression classifier that is trained to separate the local data from a subset of the auxiliary data. Their method is shown to outperform FedAvg in the considered experiments and to maintain stable performance when the level of non-IID is increased. In contrast to Algorithm 2 considered in this study, the communication procedure in FedAux is customized to run for multiple rounds. However, FedAux was also shown to outperform the considered baselines by a large margin when only using one round of communication.

As mentioned before, many variations on FD algorithms have been published. FedMD, proposed by Li and Wang [41], assumes a labeled auxiliary dataset and start each round by letting the clients train on both auxiliary and private data and then transmits predictions on the auxiliary data to the central server for aggregation. In the subsequent rounds, the aggregated predictions are used as targets on the auxiliary data during training. Lin et al. [42] propose FedDF, where ensemble distillation is performed on top of FedAvg to refine the global model. This is shown to result in fewer communications rounds compared to federated averaging methods, but it

is still based on the same communication protocol as FedAvg. A probabilistic approach is considered in FedBE by Chen and Chao [28], where Bayesian tools are used to select a subset of all local models as teacher models. Distillation is then performed with unlabeled auxiliary data to transfer the Bayesian teachers' knowledge into a global student model, which then acts as an initial model for the next round of local training. This approach is also more related to FedAvg than FD since model parameters are communicated between clients and central server, making it impossible to have heterogeneous models.

3

Method

This chapter will first introduce and motivate the weighting schemes integrated with FD in this project. The following sections will then cover the datasets used and how non-IID distributions of data have been generated. Lastly, some details are given on the experimental setup and hyper parameters used for training the ML models.

3.1 Weighting schemes

In this study, three different weighting schemes have been considered for improving the one-shot FD algorithm (Algorithm 2) in non-IID settings. The first one considers the case where clients are weighted based on their local data sizes

$$\boldsymbol{z}_{\mathrm{T}} = \sum_k w_k \boldsymbol{z}_k, \quad w_k = \frac{N_k}{N}, \tag{3.1}$$

which is the same weighting that is used in the original FedAvg algorithm. Since clients share their softmax output on a batch of auxiliary data, FD also allows to weight client contributions class-wise, as

$$\boldsymbol{z}^c_{\mathrm{T}} = \sum_k w^c_k \boldsymbol{z}^c_k, \quad w^c_k = \frac{N^c_k}{\sum_k N^c_k}, \tag{3.2}$$

where N^c_k is the number of samples client k has in class c. This weighting scheme has earlier been tested by Gong et al. [36], but its properties have not been fully investigated in different non-IID scenarios. This weighting scheme has therefore also been included in the experiments of this study.

Another possibility is to weight client contributions sample-wise, as previously investigated by Sattler et al. [40], where a client prediction on a sample is weighted based on the similarity of that sample with the local dataset. This means that each client k would have a weight w^j_k for each sample $\boldsymbol{x}_j \in \mathcal{D}_0$. With this approach, local data sizes or class distributions do not have to be shared directly, as is the case with weighting schemes (3.1) and (3.2).

To measure the data similarity, the naive approach would be to model the local data distributions $\mathcal{P}_k(\boldsymbol{x})$ and use some metric (i.e distance to mean of $\mathcal{P}_k(\boldsymbol{x})$) to determine how much a sample $\boldsymbol{x}_j \in \mathcal{D}_0$ deviates from $\mathcal{P}_k(\boldsymbol{x})$. However, when using high

dimensional data such as images, defining a distribution that models the complex correlations between all pixels is a challenging task. From the studies in this thesis, it is proposed to simplify this problem by introducing a latent variable $\boldsymbol{h}$, and instead define a conditional distribution $\mathcal{P}_k(\boldsymbol{x}|\boldsymbol{h})$ for the local data $\mathcal{D}_k$. The latent variable is a low-level representation of the high-dimensional input data and can be used to provide an abstract view of the data distribution.

This thesis has specifically investigated the possibility of modeling the conditional distribution $\mathcal{P}_k(\boldsymbol{x}|\boldsymbol{h})$ using ML with an ANN architecture called autoencoder (AE). This network has a bottleneck structure and is trained to minimize the reconstruction loss $\mathcal{L}_{\text{REC}}$ when samples are compressed and then expanded, see Figure 3.1. A common choice of reconstruction loss is the mean squared error

$$\mathcal{L}_{\text{REC}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{n}\sum_{l=1}^{n}(x_l - x_l')^2, \tag{3.3}$$

where x_l' is the l'th pixel of the autoencoder output $\boldsymbol{x}'$.

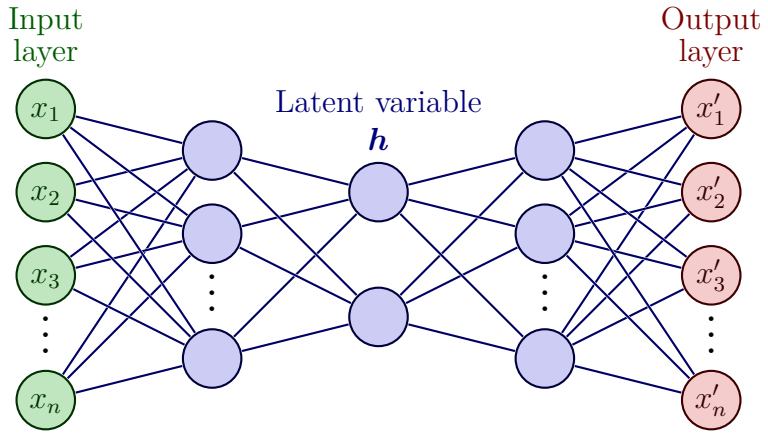
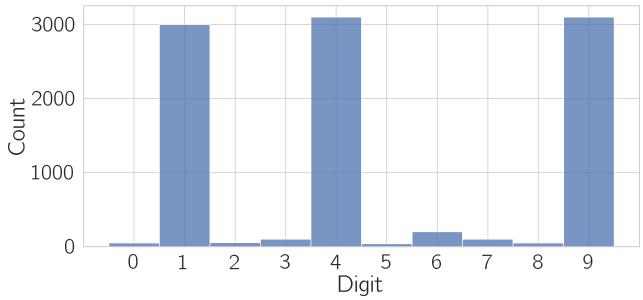


Figure 3.1: The AE network architecture.

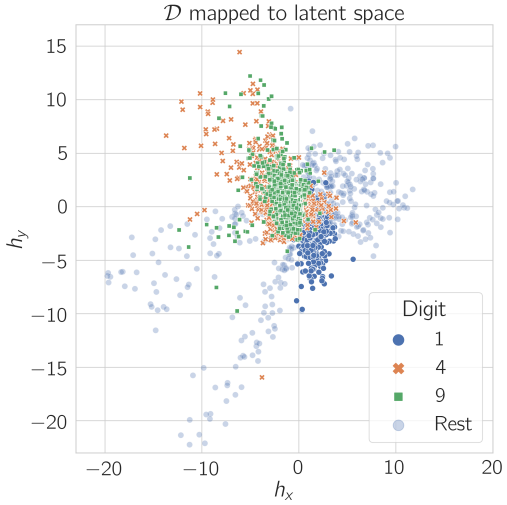
When the AE has learned a conditional distribution $\mathcal{P}_k(\boldsymbol{x}|\boldsymbol{h})$ by minimizing the MSE (3.3) on data $\mathcal{D}_k$, the loss for samples $\boldsymbol{x}_0 \in \mathcal{D}_0$ will be low if $\boldsymbol{x}_0$ is similar to samples in $\mathcal{D}_k$. For the AE, this means that these samples map to the same location in latent space. This is illustrated with an example in Figure 3.2. To apply this in a federated setting, each client k would train an AE model on local data $\mathcal{D}_k$ and calculate the losses $l_k(\boldsymbol{x}_j) := \mathcal{L}_{\text{REC}}(\boldsymbol{x}_j, \boldsymbol{x}_j')$ for all auxiliary samples $\boldsymbol{x}_j \in \mathcal{D}_0$. Since low reconstruction loss corresponds to a similarity between sample $\boldsymbol{x}_j$ and the local data $\mathcal{D}_k$, the weight w_k^j should be high when $l_k(\boldsymbol{x}_j)$ is low. This can be achieved with the relation $w_k^j = 1/l_k(\boldsymbol{x}_j)^\beta$, where $\beta \in \mathbb{R}^+$. This concludes the third weighting scheme implemented in this project, defined as

$$\boldsymbol{z}_{\text{T}}^j = \sum_k w_k^j \boldsymbol{z}_k^j, \quad w_k^j = \frac{1}{l_k(\boldsymbol{x}_j)^6}, \tag{3.4}$$

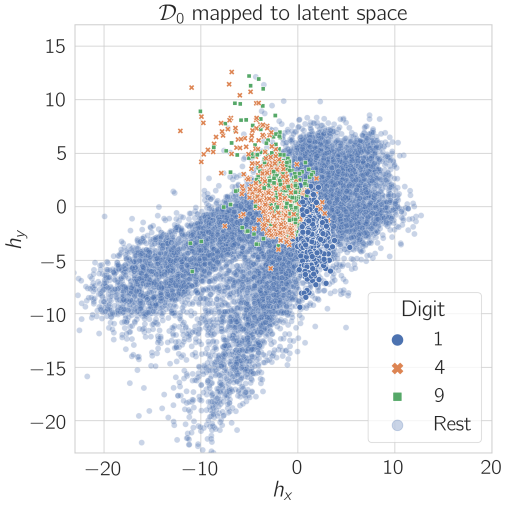
where $\beta = 6$ was chosen by means of experimentation. This was done by tuning among the integers 1-7.



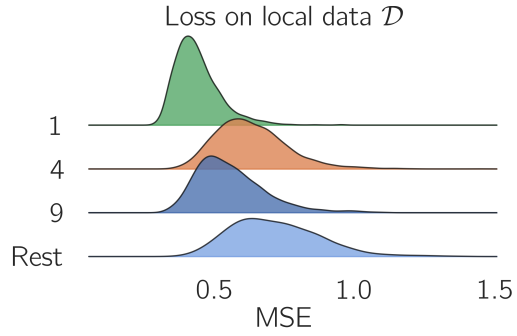
(a) Example distribution of client data $\mathcal{D}$.



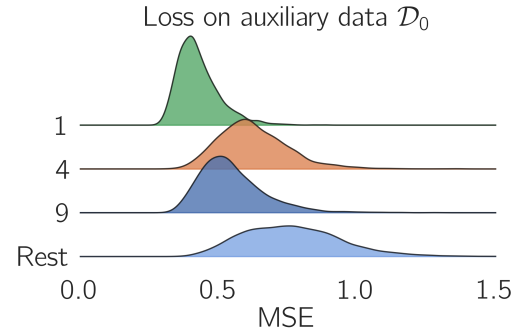
(b) The AE maps training data $\mathcal{D}$ into clusters in latent space.



(c) Similar samples from $\mathcal{D}_0$ are mapped to the same clusters.



(d) The local reconstruction loss is low for the most frequent digits.



(e) Similar samples from $\mathcal{D}_0$ will also have a low reconstruction loss.

Figure 3.2: An example showing how data similarity can be measured with the reconstruction loss from an AE, using the MNIST dataset. The auxiliary data is here taken as a subset of the complete dataset.

3.2 Datasets

This project considers the task of image classification. For this purpose, three common datasets for computer vision problems have been used in the experiments. These datasets are summarized below.

MNIST. The MNIST [43] dataset is a collection of gray scale images of handwritten digits that is commonly used for training and testing models for computer vision tasks. Since its introduction in 2010, it has become a standard benchmark for image classification models. Each image has one color channel and a size of 28x28 pixels. The dataset is divided into a training set and test set of 60000 and 10000 examples, respectively, both of which are balanced over the 10 classes.

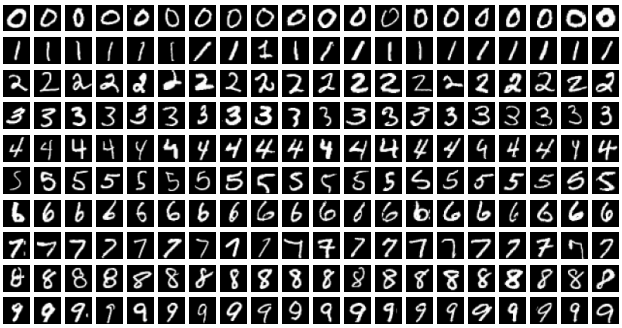


Figure 3.3: Sample images from the MNIST dataset, corresponding to handwritten digits 0-9.

EMNIST. EMNIST is an extension of the MNIST dataset, which contains handwritten upper and lower case letters in addition to handwritten digits. The images have the same format as MNIST and there are 6 different splits available for the dataset. This project uses the letters split of the dataset, which consists of 125600 training examples and 20000 test examples. The training and test set are both balanced over the 26 available classes.

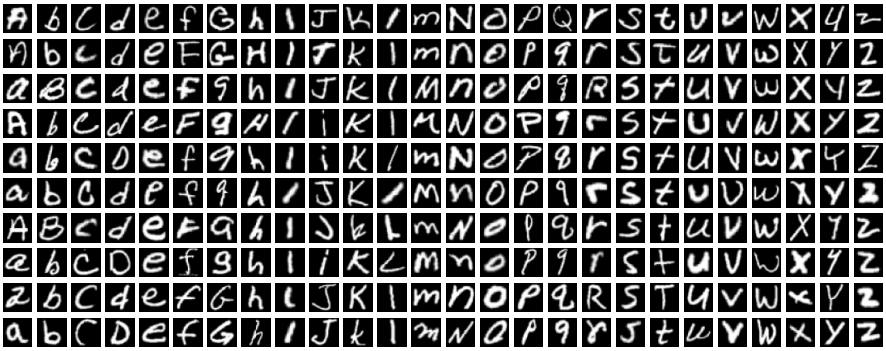


Figure 3.4: Sample images from the EMNIST letters dataset, corresponding to handwritten letters a-z and A-Z.

CIFAR-10. A more advanced dataset that is also very common as benchmark for image classification is the CIFAR-10 [44] dataset. This dataset consists of 32x32 pixel images of objects in the categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The data is divided into a training set of 50000 examples and a test set of 10000 examples, both of which are balanced over the classes.

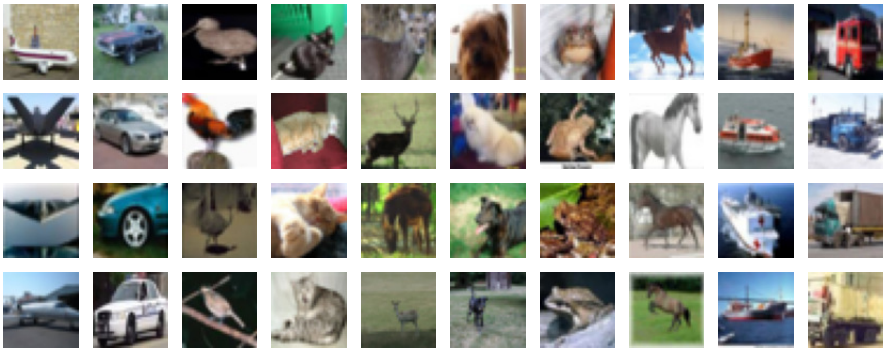


Figure 3.5: Sample images from the CIFAR-10 dataset, corresponding to objects in the categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck

3.3 Generation of non-IID data

Local training data has been generated with the commonly used Dirichlet sampling procedure proposed by Hsu et al. [29]. This strategy involves a concentration parameter α that allows to easily adjust the level of heterogeneity among clients. The data is generated by sampling a matrix

$$\boldsymbol{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_C] \in \mathbb{R}^{K \times C}, \quad \boldsymbol{p}_1, \ldots, \boldsymbol{p}_C \sim \mathrm{Dir}(\alpha) \tag{3.5}$$

from the symmetric K-categorical Dirichlet distribution, where K is the number of clients and C the number of classes. It then holds that $\|\boldsymbol{p}_i\|_1 = 1$ for $i = 1, \ldots, C$. Each client i is then assigned $P_{ij}N^j$ non-overlapping samples, where N^j is the total number of training samples in class j. The data splitting procedure is illustrated in Figure 3.6 with $K = 10$ and $C = 10$.

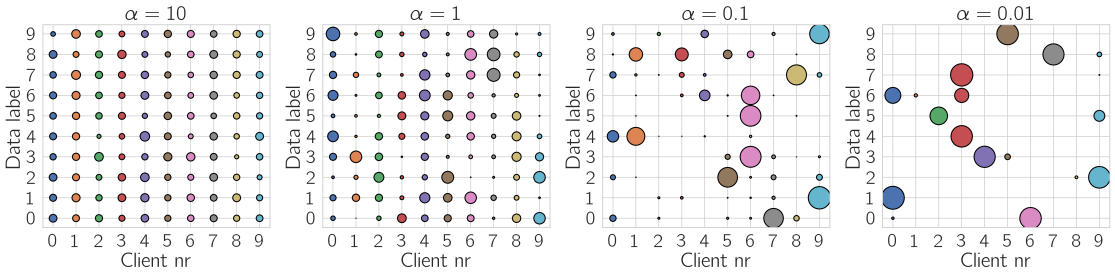


Figure 3.6: Illustration of the Dirichlet splitting procedure used in this project. The dot size is relative to the total number of samples over the clients. The data splits are more non-IID for lower values of α.

3.4 Experimental setup

This section explains the details of the setup used in the performed experiments. Code implementation can be found at Github[1].

Datasets and models. The algorithm FedED (Algorithm 2) have been evaluated on MNIST, EMNIST and CIFAR-10 with weighting schemes (3.1), (3.2) and (3.4), referred to as FedED-w0, FedED-w1 and FedED-w2. For each dataset, one half is distributed as local data, and the other half is used as auxiliary data. For MNIST and EMNIST, three different CNNs have been tested in the experiments. These models are referred to as CNN1, CNN2 and CNN3. The local model is the CNN1 model for all MNIST and EMNIST experiments and the student model has been tested for all three architectures. More details on these models are presented in Appendix A. For CIFAR-10, a Resnet-18 [45] architecture was used for both local models and student model. Autoencoder architectures used for FedED-w2 are presented in A.2.

Table 3.1: Sizes of used models, assuming a float is 24 bytes.

Name	CNN1		CNN2		CNN3		Resnet18
Dataset	MNIST	EMNIST	MNIST	EMNIST	MNIST	EMNIST	CIFAR-10
Size (MB)	0.03	0.06	0.10	0.50	0.20	0.57	28.36

[1] https://github.com/oscarerikssson/FedML-master-thesis

Federated environment. Each experiment were conducted with $K = 10$ clients and client data generated with α values $10, 1, 0.1$ and 0.01, according to the Dirichlet sampling procedure explained in Section 3.3. The fraction γ of participating clients each round was set to 1.

Distillation settings. The number of local epochs was set to $E = 20$ for all distillation experiments. The weighting schemes have been tested for five different sizes $|\mathcal{D}_0|$ of auxiliary data for each dataset. The student loss function has been tested as both MSE (2.10) and CE (2.3).

Optimization. Local models were trained with SGD (stochastic gradient descent) optimizer with learning rate 1e−3 and momentum 0.9. Student models were trained with the Adam optimizer with learning rate 1e−3 for FedED-w0 and FedED-w1. For FedED-w2, performance was improved with a lower learning rate, and was therefore set to 1e−5. The AE models in FedED-w2 were also trained with the Adam optimizer with learning rate 1e−3.

Baselines. The performance is compared against to the state-of-the-art FL algorithms FedAvg and FedProx. The auxiliary dataset is assumed unlabeled in this project and is therefore not included as training data for FedAvg and FedProx. For these algorithms, the local epochs was set to $E = 1$ and communication rounds to $T = 100$. These settings was observed to be enough for converged performance and therefore seen as sufficient for a fare comparison. The proximal parameter μ was chosen to 0.1 for all datasets and distributions after tuning among the values $[0.01, 0.1, 0.5, 1]$. Included in the comparison is also the ensemble performance, referred to as Ensemble-w0, Ensemble-w1 and Ensemble-w2, which is the aggregated predictions from all local models.

Hardware. Experiments have been executed on a virtual machine provided by AI Sweden, using Nvidia RTX5000 with 4 cores, 32 GB ram and 1 TB disk space.

4

Results

The results are divided into three parts, based on the different datasets explored. All algorithms are evaluated and compared based on the accuracy on the test dataset for different values of the concentration parameter α. Each weighting scheme is tested for different auxiliary data sizes $|\mathcal{D}_0|$. MNIST and EMNIST results are presented for different student models with varying complexity. For further details, see Section 3.4. The loss function used when training the student model is MSE, unless stated otherwise. See model details in Appendix A. The results have been generated and averaged over 10 different seeds, such that each algorithm has been tested for the same distributions of local data for a given α. For more detailed results on training progresses and global test performances, the reader is referred to Appendix B, where training curves are presented for all considered datasets and algorithms.

4.1 MNIST

Table 4.1 shows how the implemented FD weighting schemes compares to the chosen baselines for different values of α. The auxiliary data size is here set to $|\mathcal{D}_0| = 30000$ samples and the student model is CNN3, which is the largest model considered in the experiments, but still with a lower model size than all clients combined. For a low degree of data heterogeneity ($\alpha = 10$ and $\alpha = 1$), FD displays similar performance as the baselines. However, algorithm FedED-w2 seems to perform slightly worse for these α values. For increased data heterogeneity ($\alpha = 0.1$ and $\alpha = 0.01$), the performance drop is greatest for FedED-w0 and FedED-w1, while FedED-w2 maintains high accuracy and performs better than baselines. Algorithm FedED-w2 with CE loss is the only case where test accuracy seems to increase with higher degree of non-IID data.

Figure 4.1 shows how the auxiliary data size $|\mathcal{D}_0|$ affects the performance of the considered weighting schemes. In general, FedED-w2 needs more auxiliary samples to reach high accuracy, while the performance of FedED-w0 and FedED-w1 do not change that much when $|\mathcal{D}_0|$ is increased. As presented in Table 4.1, FedED-w2 achieves significantly higher accuracy than the other weighting schemes for $\alpha = 0.1$ and $\alpha = 0.01$.

Table 4.1: Mean test accuracy and standard deviation on MNIST for different α with $|\mathcal{D}_0| = 30000$ and the CNN3 student model.

Algorithm	$\alpha = 10$	$\alpha = 1$	$\alpha = 0.1$	$\alpha = 0.01$
FedAvg	94.14±0.75	93.96±0.92	91.83±1.16	88.37±1.39
FedProx	94.11±0.74	93.99±0.91	91.93±1.14	88.30±1.47
FedED-w0	94.08±0.65	93.78±0.62	76.43±6.53	42.23±6.65
FedED-w1	**94.27±0.73**	**94.19±0.62**	80.94±9.60	37.12±3.58
FedED-w2	93.05±0.58	93.02±0.46	90.98±1.88	90.36±1.94
FedED-w2 (CE)	92.14±1.20	92.73±0.86	**93.28±1.10**	**93.87±0.92**
Ensemble-w0	90.70±1.47	91.11±1.14	75.52±6.79	39.58±5.89
Ensemble-w1	90.78±1.45	91.66±1.09	88.46±2.86	38.06±13.27
Ensemble-w2	90.87±1.43	92.02±0.96	93.08±1.16	92.73±1.48

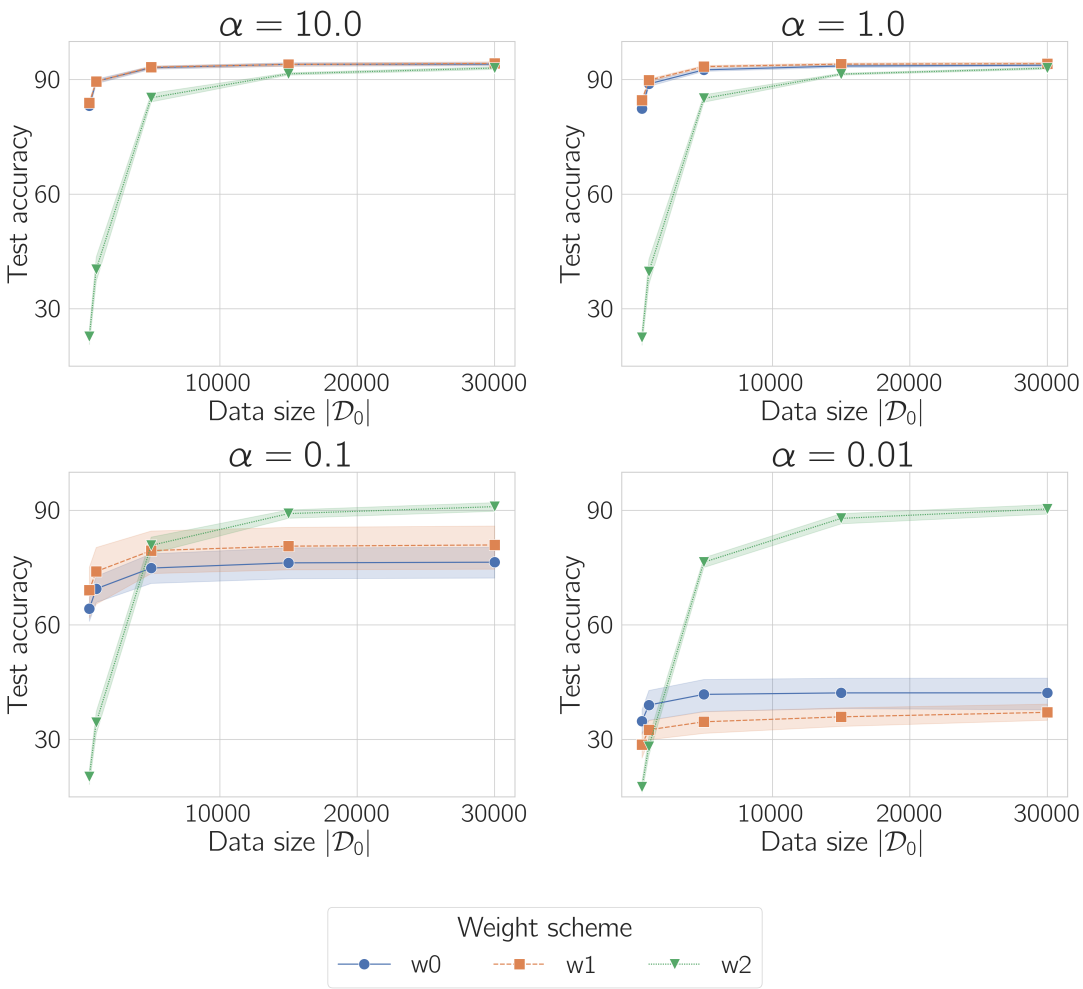


Figure 4.1: Comparison of test accuracy on MNIST for the implemented weighting schemes with varying data size $|\mathcal{D}_0|$. The student model is CNN3 and loss function is MSE.

Figure 4.2 compares the test accuracy of the weighting schemes with different student models and $\alpha = 10$. The choice of student model mostly affects the performance of FedED-w2. As mentioned in Section 3.4, the local model for each client is CNN1. In this case, CNN1 can also be used as student model and achieve good performance. However, the performance increases with model complexity, as can be seen from the results with CNN2 and CNN3.

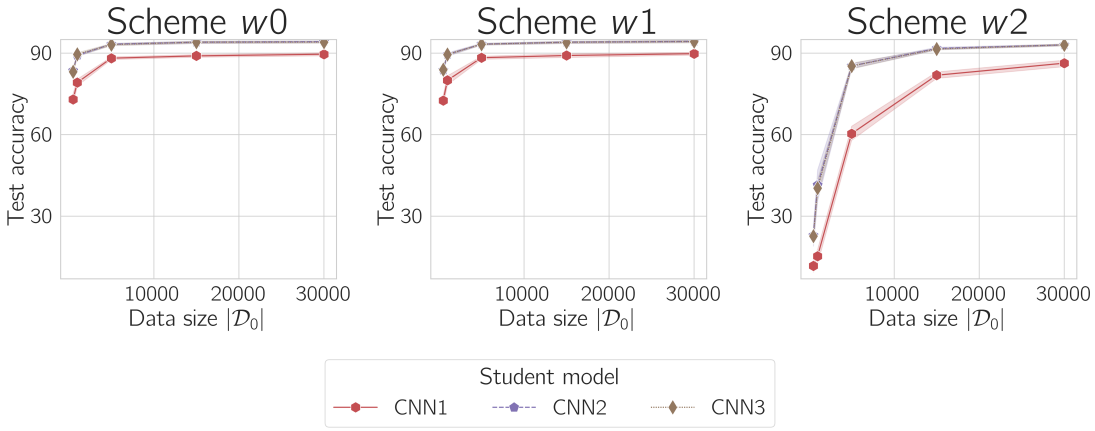


Figure 4.2: Comparison of test accuracy on MNIST for different student models with $\alpha = 10$.

4.2 EMNIST

As with MNIST, Table 4.2 shows that when using enough auxiliary data and student model complexity, FD is comparable with the baselines for lower levels of non-IID data. Overall, FedED-w2 with CE loss is the best performing algorithm on EMNIST, whose performance increases with lower α.

Table 4.2: Mean test accuracy and standard deviation on EMNIST for different α with $|\mathcal{D}_0| = 60000$ and the CNN3 student model.

Algorithm	$\alpha = 10$	$\alpha = 1$	$\alpha = 0.1$	$\alpha = 0.01$
FedAvg	**78.85±1.14**	78.29±1.11	72.36±1.51	65.95±1.92
FedProx	78.79±1.15	78.30±1.13	72.39±1.45	66.00±1.90
FedED-w0	77.11±0.64	76.11±0.65	62.64±3.78	37.91±4.23
FedED-w1	77.29±0.48	77.81±0.26	63.26±8.92	31.22±6.78
FedED-w2	75.10±0.74	74.98±0.49	71.80±1.33	70.37±1.82
FedED-w2 (CE)	78.47±1.25	**78.92±1.06**	**78.58±1.32**	**80.12±1.20**
Ensemble-w0	78.09±1.05	78.23±1.07	64.79±3.52	34.93±4.36
Ensemble-w1	78.18±1.08	79.08±0.98	70.59±4.53	38.34±14.52
Ensemble-w2	78.26±1.10	79.74±0.89	80.48±1.12	81.18±1.06

Similar to MNIST, Figure 4.3 shows that FedED-w2 needs more auxiliary samples

to reach high accuracy also on EMNIST, compared to FedED-w0 and FedED-w1. For $\alpha = 0.1$ and $\alpha = 0.01$, test accuracy is significantly higher with FedED-w2.

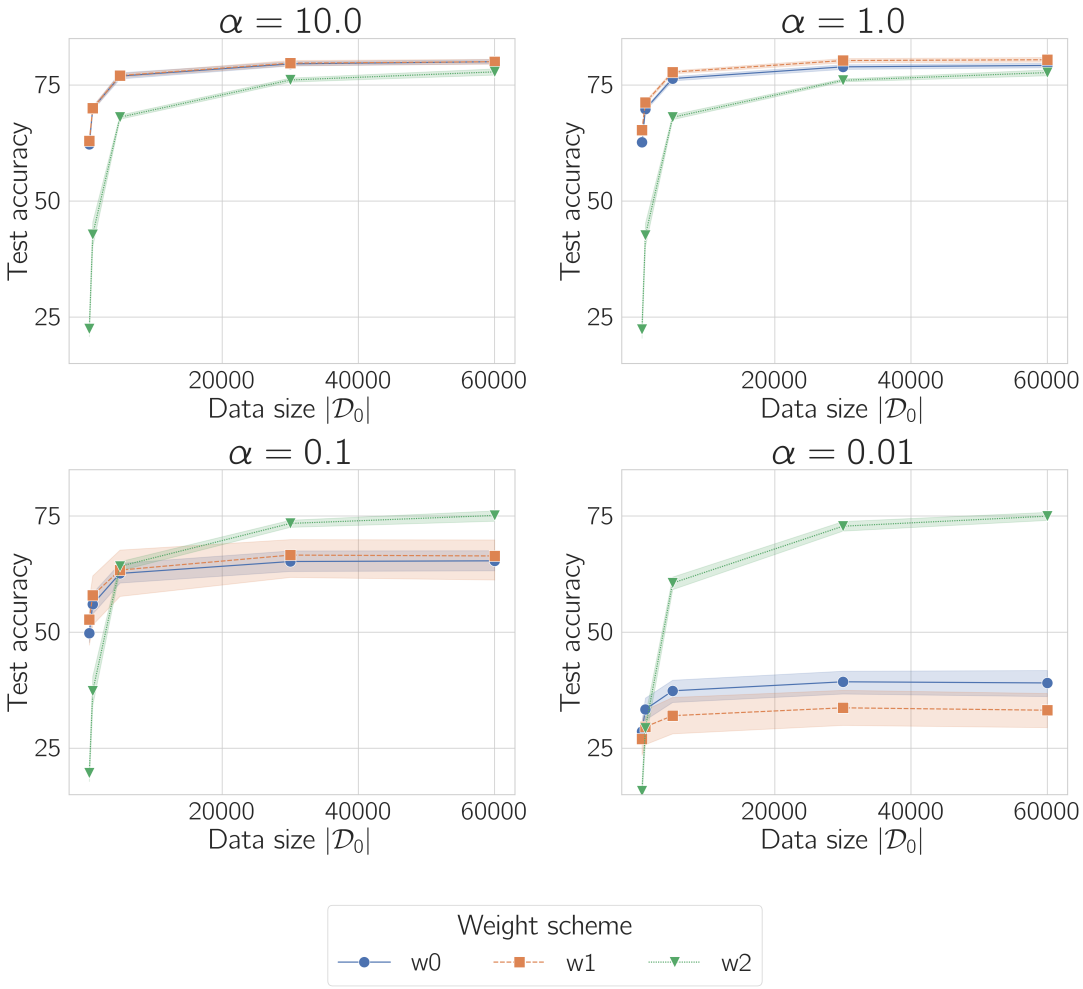


Figure 4.3: Comparison of test accuracy on EMNIST for the implemented weighting schemes with varying data size $|\mathcal{D}_0|$. The student model is CNN3 and loss function is MSE.

In Figure 4.4, the results show that the complexity of the student model affect the performance considerably for all weighting schemes. In this case, the local model architecture CNN1 does not achieve comparable performance to baselines when used as student model. The larger models, CNN2 and CNN3, are needed to achieve comparable performance.

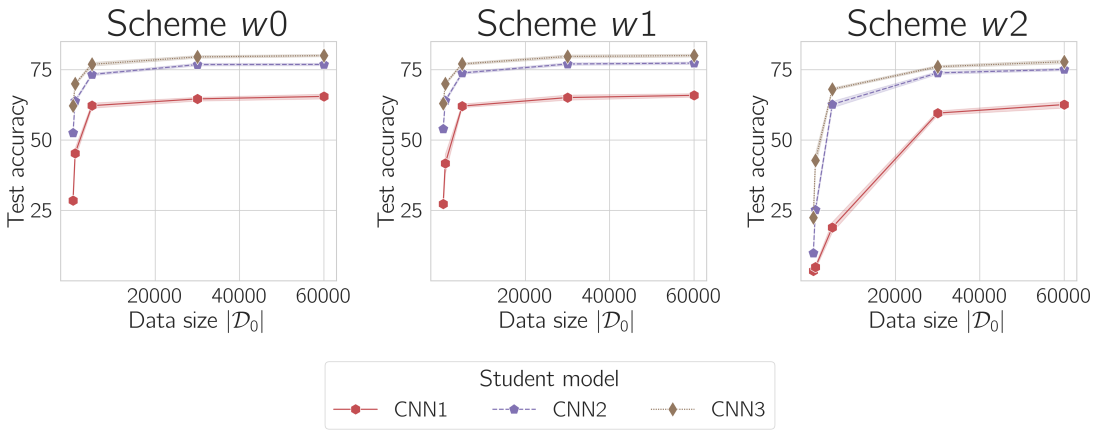


Figure 4.4: Comparison of test accuracy on EMNIST for different student models with $\alpha = 10$.

4.3 CIFAR-10

CIFAR-10 is a harder image classification task than previously considered MNIST and EMNIST, which can be seen from the overall lower accuracy in Table 4.3. For this dataset, the FD algorithms do not achieve higher test accuracy than baselines for any α with the considered student model and auxiliary data. However, the ensemble model Ensemble-w2 performance is close to baseline for $\alpha = 0.1$ and $\alpha = 0.01$.

Table 4.3: Test accuracy for different α with $|\mathcal{D}_0| = 25000$ and the Resnet-18 student model.

Algorithm	$\alpha = 10$	$\alpha = 1$	$\alpha = 0.1$	$\alpha = 0.01$
FedAvg	51.36±0.46	**50.17±0.51**	**40.42±2.33**	**29.99±2.62**
FedProx	**51.58±0.59**	49.83±0.75	40.27±2.96	29.91±2.69
FedED-w0	48.36±1.29	47.77±0.63	37.41±2.51	24.16±5.56
FedED-w1	48.27±0.99	46.88±1.98	33.81±4.69	15.87±3.93
FedED-w2	47.37±0.73	46.31±0.87	35.38±1.45	24.17±7.47
FedED-w2 (CE)	45.05±0.82	44.03±0.73	35.32±1.53	24.82±6.68
Ensemble-w0	50.65±0.59	49.98±0.58	38.04±3.34	24.52±5.84
Ensemble-w1	50.61±0.58	49.96±0.49	38.14±5.07	18.55±5.75
Ensemble-w2	50.53±0.51	49.79±0.68	40.14±1.73	28.43±7.89

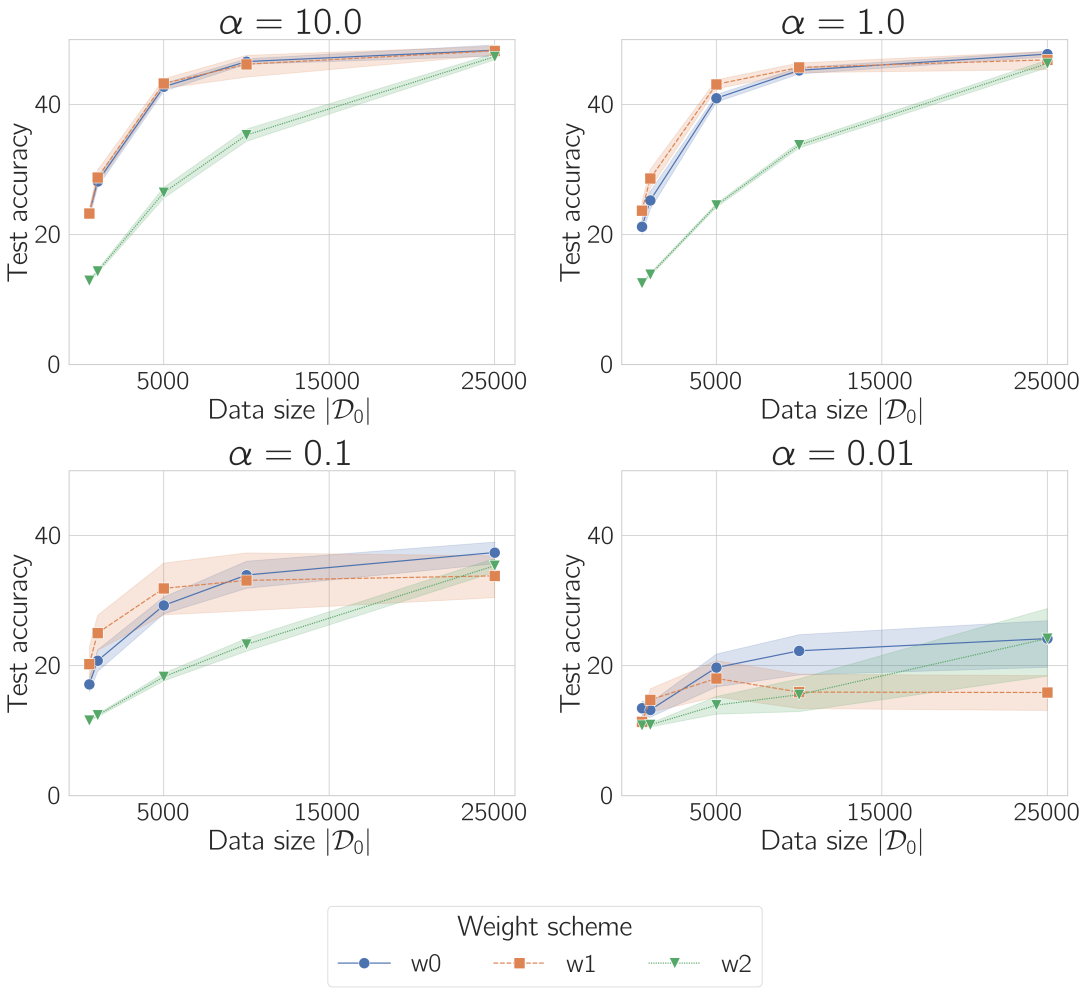


Figure 4.5: Comparison of test accuracy for the implemented weighting schemes with varying data size $|\mathcal{D}_0|$. The student model is Resnet-18 and loss function is MSE.

4.4 Communication costs

The communication costs presented in Table 4.4 shows that FedED transmits less data than FedAvg and FedProx in all of the considered experiments. The difference in communication is especially large on the CIFAR-10 dataset, which has the smallest size of auxiliary data and the largest model size among the experiments.

Table 4.4: Communication costs for the experiments conducted, measured in MB transferred between clients and central server and assuming a float is 24 bytes. The auxiliary data size $|\mathcal{D}_0|$ is here set to the largest size used for each dataset.

	Cost formula	MNIST	EMNIST	CIFAR-10
FedED	$\|\mathcal{D}_0\|KC$	7.2	37.4	6.0
FedAvg/FedProx	$2\|\boldsymbol{\theta}\|KT$	25.0	63.0	28359.4

5

Discussion

This chapter provides an analysis of the results obtained in this study. A short discussion is also given on the practical use of FD.

5.1 Performance analysis of weighting schemes

The results presented in the previous section indicate that the one-shot FD algorithm can outperform standard FL algorithms on MNIST and EMNIST by using the sample-wise weighting scheme FedED-w2. Weighting schemes FedED-w0 and FedED-w1 achieve baseline performance when data is close to IID ($\alpha = 10$). However, the performance is much lower than baseline for lower values of α. A possible reason for this is the loss of information in class probabilities when clients do not have enough data in all classes. In this case, local models will only generate class probabilities in classes they have been trained on, independent of the input sample from the auxiliary dataset. This would make the aggregated class probabilities less useful for training a student model since they will more likely represent the global class distribution, rather than provide a valid prediction on an auxiliary sample.

As an example, assume a client only have data in classes 0 and 1 out of 10 total classes. The local model then only learns to distinguish between class 0 and 1, and hence only generates class probabilities in these classes when predicting on auxiliary data. In this case, FedED-w0 and FedED-w1 can only reduce these class probabilities if the local data size is small relative to other clients. If the local data size is large relative to other clients, these class probabilities will be amplified and the aggregated class probabilities will be heavily weighted towards classes 0 and 1, which makes them less informative for training the student model.

This thesis introduces the novel weighting scheme FedED-w2, which is shown to obtain high performance with non-IID distributions of MNIST and EMNIST. The best performance is achieved when using CE as loss function when training the student model, where the results show an increase in test accuracy when data heterogeneity is increased. This is a reasonable outcome since it is easier for the local AE to achieve low reconstruction loss if training samples have few and similar features. This is the case when the level of non-IID is high since this means that clients only have data in a few classes, resulting in fewer features that the AE needs to reconstruct with low loss.

The performance of FedED-w2 is improved when using CE loss compared to MSE. A possible reason for this improvement is that when the reconstruction loss is low for a certain sample, the local model have likely seen many similar samples. This in turn means that the model probably has a confident prediction for this certain sample, which receives a large weight when forming the aggregated predictions $\boldsymbol{z}_{\mathrm{T}}$ due to the low reconstruction loss. In this case, the MSE loss will diminish faster compared to CE loss when student predictions $\boldsymbol{z}_{\mathrm{S}}$ are improved, and therefore halts the training of the student model at an earlier point. This can be understood by observing the MSE loss $(z_{\mathrm{S}}^c - z_{\mathrm{T}}^c)^2$ converges to 0 faster than the CE loss $-\log z_{\mathrm{S}}^c$ when z_{S}^c approaches 1, assuming that the aggregated class probabilities produce a confident prediction $z_{\mathrm{T}}^c \approx 1$.

On CIFAR-10, the overall test accuracy is lower with FD compared to the considered baselines. However, Table 4.3 shows that the ensemble performance is close to baselines in some cases, e.g. for Ensemble-w2 with $\alpha = 0.1$. Since the results on MNIST and EMNIST show that the student model can achieve the ensemble test accuracy for many cases, given enough complexity and auxiliary data, some performance gain could be obtained on CIFAR-10 by improving the distillation procedure. This could be done by changing the student model architecture, use more auxiliary data, or try different distillation techniques, such as softening the class probabilities as explained in section 2.4.1.

5.2 Using federated distillation in practice

FD introduces some additional challenges compared to standard FL algorithms that need to be addressed. First, an unlabeled auxiliary dataset need to be available. This is no problem for general use cases where public datasets from the same domain could already be available, but it might be a problem for more specific tasks. An advantage with the FD algorithm considered in this study is that the auxiliary data can be unlabeled. This algorithm is thus suitable for cases where auxiliary data is unlimited but labeling is expensive. One example is image segmentation or object detection in video material. Video material can in some cases be an unlimited resource, but the labeling might be time consuming or require an expert.

Second, the student model architecture and distillation procedure need to be selected. The results from this study indicate that the choice of student model can have a large impact on the final performance, as well as the choice of loss function for training the student model. However, this problem can be addressed separately from local training when only using one communication round in FD, meaning that tuning the student model training would not impose additional communication costs.

6

Conclusion

The experiments in this study indicate that the considered FD algorithm can achieve similar performance to standard parameter averaging algorithms when data is close to IID on less complex image classification tasks. When local data is non-IID, the experiments show that FD experience a greater performance loss than parameter averaging algorithms, which is not improved with weighting schemes based on the local data size or label distribution. This study has shown that the sample-wise weighting scheme FedED-w2 can withstand this performance loss and outperforms the considered baselines for two of the used datasets in the experiments. The FD approach introduces many challenging problems, where this study considers the problem of aggregating local model predictions on auxiliary data when local training data is non-IID. An important aspect of this method is to ensure the quantity and quality of auxiliary data. This study has shown that the size of auxiliary data is important to achieve high performance, especially with FedED-w2, which also demands a certain similarity between the local and auxiliary dataset. Therefore, future work includes investigating FD algorithms that are less dependent on auxiliary data.

Bibliography

[1] Forbes. (2019). "Rethinking privacy for the ai era," [Online]. Available: https://www.forbes.com/sites/insights-intelai/2019/03/27/rethinking-privacy-for-the-ai-era/?sh=479bcdb67f0a (visited on 05/13/2022).

[2] B. Wolford. (2022). "What is gdpr, the eu's new data protection law?" [Online]. Available: https://gdpr.eu/what-is-gdpr/ (visited on 05/13/2022).

[3] H. B. McMahan, E. Moore, D. Ramage, *et al.*, "Communication-efficient learning of deep networks from decentralized data," 2017. arXiv: 1602.05629 [cs.LG].

[4] P. Kairouz, H. B. McMahan, B. Avent, *et al.*, "Advances and open problems in federated learning," 2021. arXiv: 1912.04977 [cs.LG].

[5] F. Zhou and G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018.

[6] Y. Zhao, M. Li, L. Lai, *et al.*, "Federated learning with non-iid data," 2018. arXiv: 1806.00582 [cs.LG].

[7] J. Deng, W. Dong, R. Socher, *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[8] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *CoRR*, vol. abs/1609.07843, 2016. arXiv: 1609.07843.

[9] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," 2019. arXiv: 1902.00146 [cs.LG].

[10] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," 2020. arXiv: 1905.10497 [cs.LG].

[11] T. Li, A. K. Sahu, M. Zaheer, *et al.*, "Federated optimization in heterogeneous networks," 2020. arXiv: 1812.06127 [cs.LG].

[12] S. P. Karimireddy, S. Kale, M. Mohri, *et al.*, "Scaffold: Stochastic controlled averaging for federated learning," 2021. arXiv: 1910.06378 [cs.LG].

[13] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" *CoRR*, vol. abs/2003.14053, 2020. arXiv: 2003.14053.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[15] A. Hard, K. Rao, R. Mathews, *et al.*, "Federated learning for mobile keyboard prediction," 2018.

[16] T. Yang, G. Andrew, H. Eichner, *et al.*, "Applied federated learning: Improving google keyboard query suggestions," 2018.

[17] Apple. (2019). "Designing for privacy (video and slides)," [Online]. Available: https://developer.apple.com/videos/play/wwdc2019/708 (visited on 05/13/2022).

[18] support.google. (2019). "Your chats stay private while messages improves suggestions," [Online]. Available: https://support.google.com/messages/answer/9327902 (visited on 05/13/2022).

[19] I. PR. (2020). "Intel works with university of pennsylvania in using privacy-preserving ai to identify brain tumors," [Online]. Available: https://newsroom.intel.com/news/intel-works-university-pennsylvania-using-privacy-preserving-ai-identify-brain-tumors/#gs.vkzoux (visited on 05/13/2022).

[20] P. Courtiol, C. Maussion, M. Moarii, *et al.*, "Deep learning-based classification of mesothelioma improves prediction of patient outcome," *NATURE MEDICINE*, vol. 25, pp. 1519–+, 2019.

[21] I. Daya, H. Roth, and A. e. a. Zhong, "Federated learning for predicting clinical outcomes in patients with covid-19," *NATURE MEDICINE*, vol. 27, pp. 1735–1743, 2021.

[22] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency V2V communications," *CoRR*, vol. abs/1805.09253, 2018. arXiv: 1805.09253.

[23] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[24] B. Jayaraman, L. Wang, D. Evans, and Q. Gu, "Distributed learning without distress: Privacy-preserving empirical risk minimization," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, *et al.*, Eds., vol. 31, Curran Associates, Inc., 2018.

[25] Y. Li, T.-H. Chang, and C.-Y. Chi, "Secure federated averaging algorithm with differential privacy," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2020, pp. 1–6.

[26] W. Hao, M. El-Khamy, J. Lee, *et al.*, "Towards fair federated learning with zero-shot data augmentation," *CoRR*, vol. abs/2104.13417, 2021. arXiv: 2104.13417.

[27] J. Reyes, L. Di-Jorio, C. Low-Kam, and M. Kersten-Oertel, "Precision-weighted federated learning," *CoRR*, vol. abs/2107.09627, 2021. arXiv: 2107.09627.

[28] H.-Y. Chen and W.-L. Chao, "Fedbe: Making bayesian model ensemble applicable to federated learning," 2021. arXiv: 2009.01974 [cs.LG].

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

[30] C. Buciluundefined, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: Association for Computing Machinery, 2006, pp. 535–541.

[31] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015. arXiv: 1503.02531 [stat.ML].

[32] U. Asif, J. Tang, and S. Harrer, "Ensemble knowledge distillation for learning improved and efficient networks," *CoRR*, vol. abs/1909.08097, 2019. arXiv: 1909.08097.

[33] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," 2018.

[34] M. Nasr, R. Shokri, and A. Houmansadr, "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)*, IEEE, 2019.

[35] N. Guha, A. Talwalkar, and V. Smith, "One-shot federated learning," 2019.

[36] X. Gong, A. Sharma, S. Karanam, *et al.*, "Ensemble attention distillation for privacy-preserving federated learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 15 076–15 086.

[37] D. Jimenez, "Dynamically weighted ensemble neural networks for classification," in *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, vol. 1, 1998, pp. 753–756.

[38] P. Sollich and A. Krogh, "Learning with ensembles: How over-fitting can be useful," in *Proceedings of the 8th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA: MIT Press, 1995, pp. 190–196.

[39] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," Chapman and Hall, 1993, pp. 126–142.

[40] F. Sattler, T. Korjakow, R. Rischke, and W. Samek, "Fedaux: Leveraging unlabeled auxiliary data in federated learning," 2021.

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

[42] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," 2021. arXiv: 2006.07242 [cs.LG].

[43] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[44] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385.

A

Model architectures

A.1 CNN architectures

Table A.1: Architecture used for CNN1, CNN2 and CNN3 in the MNIST experiments. CNN1: $N_{\text{Ch}} = 2$, 1042 trainable parameters. CNN2: $N_{\text{Ch}} = 8$, 4138 trainable parameters. CNN3: $N_{\text{Ch}} = 16$, 8266 trainable parameters.

Layer	#Channels	F	S	P	Activation	Output size
Input layer	-	-	-	-	-	1x28x28
Conv	N_{Ch}	5	1	2	ReLU	N_{Ch}x28x28
Max Pooling	-	4	2	-	-	N_{Ch}x7x7
Fully Connected	-	-	-	-	Linear	10

Table A.2: Architecture used for CNN1 and CNN2 in the EMNIST experiments. CNN1: $N_{\text{Ch}} = 2$, 2626 trainable parameters. CNN2: $N_{\text{Ch}} = 16$, 20826 trainable parameters.

Layer	#Channels	F	S	P	Activation	Output size
Input layer	-	-	-	-	-	1x28x28
Conv	N_{Ch}	5	1	2	ReLU	N_{Ch}x28x28
Max Pooling	-	4	2	-	-	N_{Ch}x7x7
Fully Connected	-	-	-	-	Linear	26

Table A.3: Architecture used for CNN3 in the EMNIST experiments. 23834 trainable parameters.

Layer	#Channels	F	S	P	Activation	Output size
Input layer	-	-	-	-	-	1x28x28
Conv	8	5	1	2	ReLU	8x28x28
Max Pooling	-	2	2	-	-	8x14x14
Conv	16	5	1	2	ReLU	16x14x14
Max Pooling	-	2	2	-	-	16x7x7
Fully Connected	-	-	-	-	Linear	26

A.2 Autoencoder architectures

Table A.4: Autoencoder architecture used for MNIST and EMNIST.

Layer	#Channels	F	S	P	Activation	Output size
Input layer	-	-	-	-	-	1x28x28
Conv	8	3	2	1	ReLU	8x14x14
Batch norm	-	-	-	-	-	-
Conv	16	3	2	1	ReLU	16x7x7
Batch norm	-	-	-	-	-	-
Conv	32	3	2	0	ReLU	32x3x3
Flatten	-	-	-	-	-	288
Fully Connected	-	-	-	-	ReLU	128
Fully Connected	-	-	-	-	Linear	4
Fully Connected	-	-	-	-	ReLU	128
Fully Connected	-	-	-	-	ReLU	288
Unflatten	-	-	-	-	-	32x3x3
ConvTranspose	16	3	2	0	ReLU	16x7x7
Batch norm	-	-	-	-	-	-
ConvTranspose	8	3	2	1	ReLU	8x14x14
Batch norm	-	-	-	-	-	-
ConvTranspose	1	3	2	1	ReLU	1x28x28

Table A.5: Autoencoder architecture used for CIFAR-10.

Layer	#Channels	F	S	P	Activation	Output size
Input layer	-	-	-	-	-	3x32x32
Conv	32	3	2	1	ReLU	32x16x16
Batch norm	-	-	-	-	-	-
Conv	64	3	2	1	ReLU	64x8x8
Batch norm	-	-	-	-	-	-
Conv	128	3	2	0	ReLU	128x3x3
Flatten	-	-	-	-	-	1152
Fully Connected	-	-	-	-	ReLU	128
Fully Connected	-	-	-	-	Linear	64
Fully Connected	-	-	-	-	ReLU	128
Fully Connected	-	-	-	-	ReLU	1152
Unflatten	-	-	-	-	-	128x3x3
ConvTranspose	64	3	2	0	ReLU	64x8x8
Batch norm	-	-	-	-	-	-
ConvTranspose	32	3	2	0	ReLU	32x16x16
Batch norm	-	-	-	-	-	-
ConvTranspose	3	3	2	0	ReLU	3x32x32

B

Training curves

B.1 MNIST

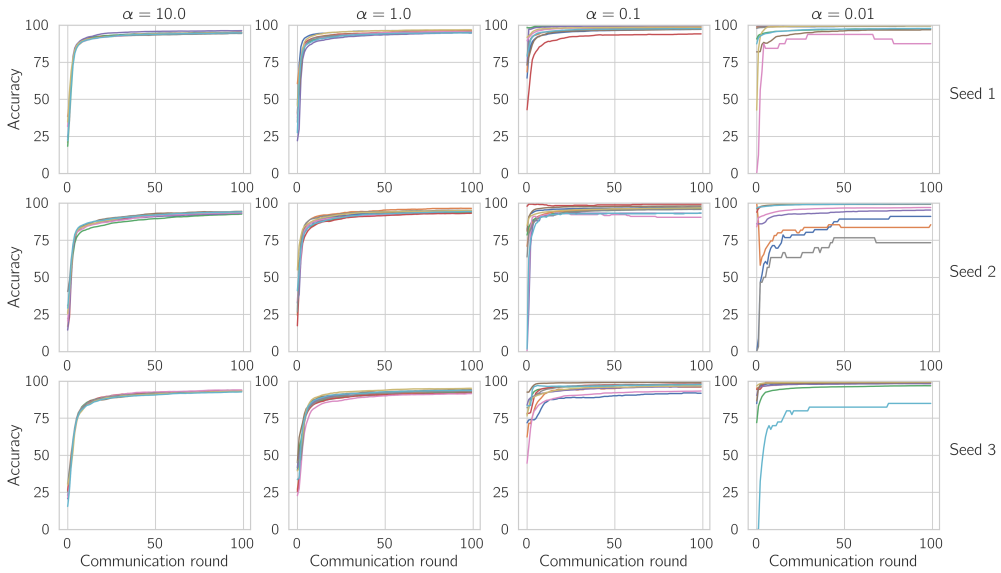


Figure B.1: MNIST: FedAvg local training accuracies with the CNN1 model for 3 different seeds.

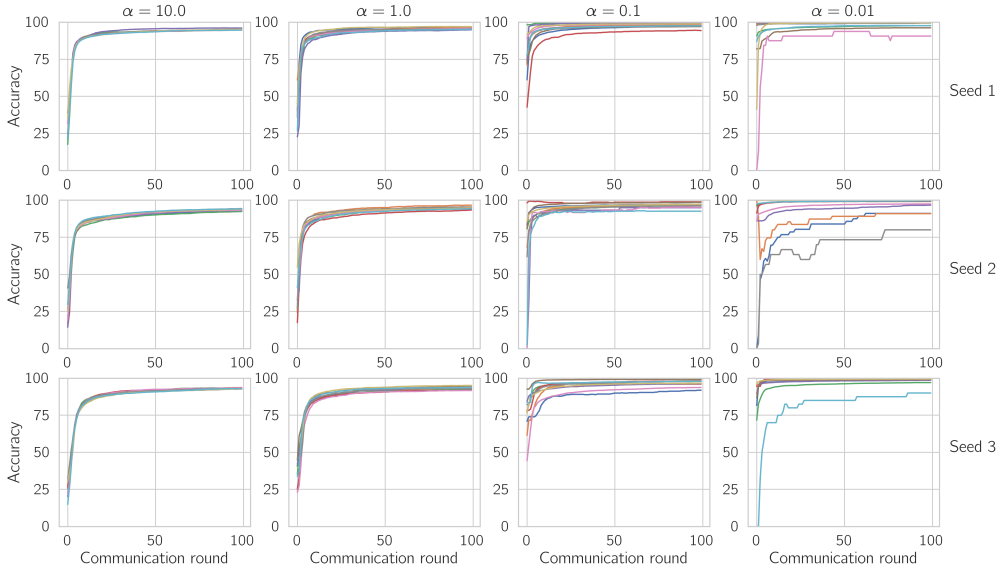


Figure B.2: MNIST: FedProx local training accuracies with the CNN1 model for 3 different seeds.

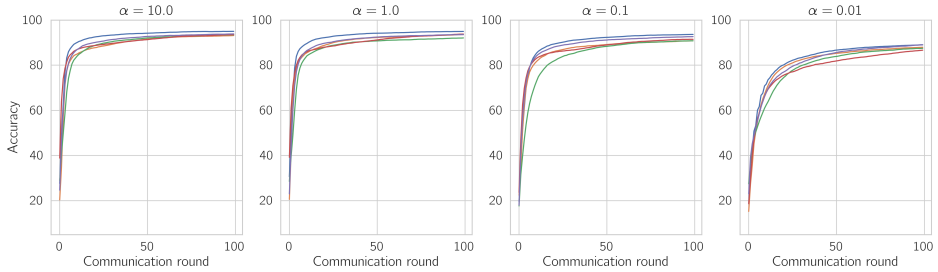


Figure B.3: MNIST: FedAvg global test accuracy with the CNN1 model for 5 different seeds.

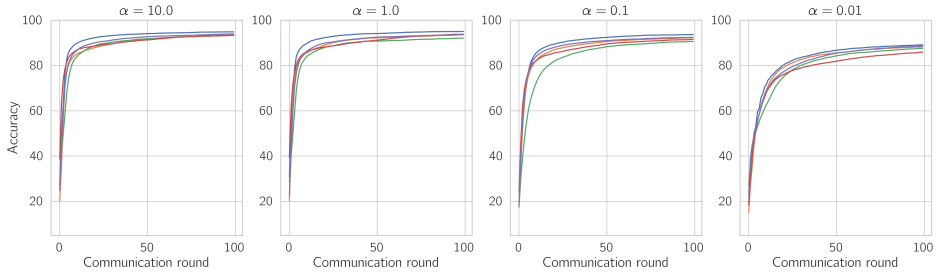


Figure B.4: MNIST: FedProx global test accuracy with the CNN1 model for 5 different seeds.

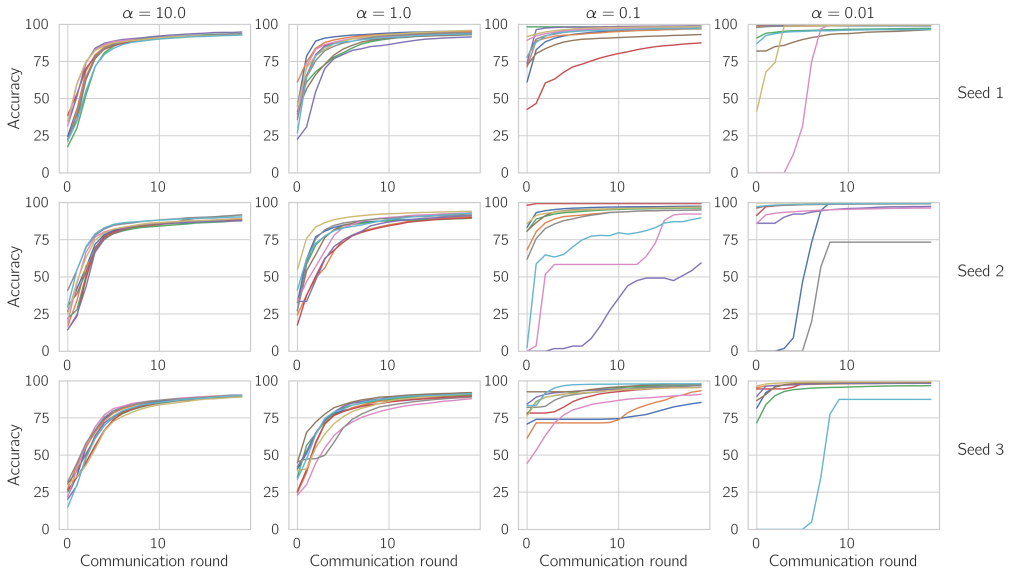


Figure B.5: MNIST: FedED local training accuracies with the CNN1 model for 3 different seeds.

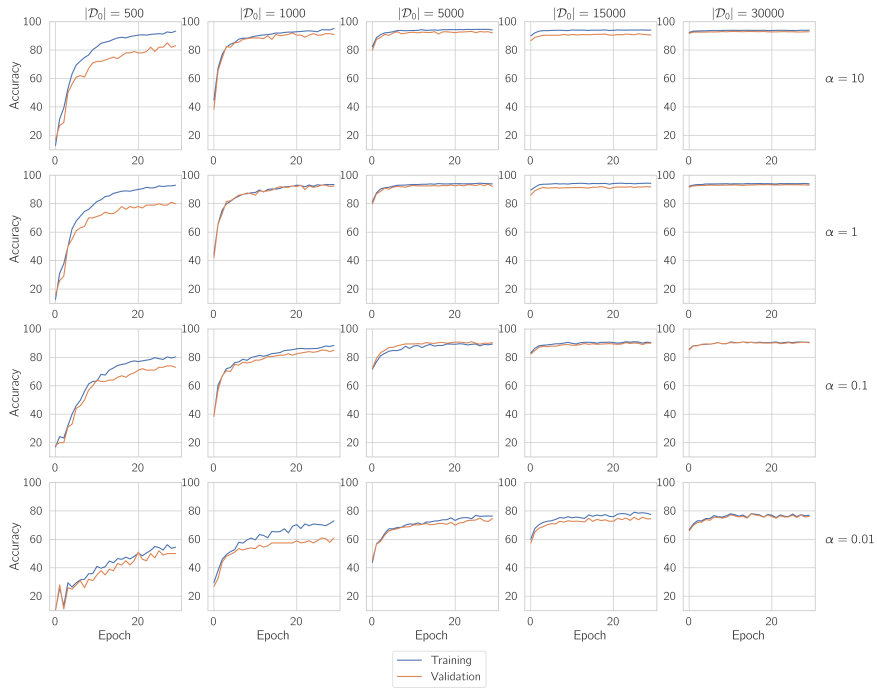


Figure B.6: MNIST: FedED-w0 student training accuracy with the CNN3 model and MSE loss for 1 seed.

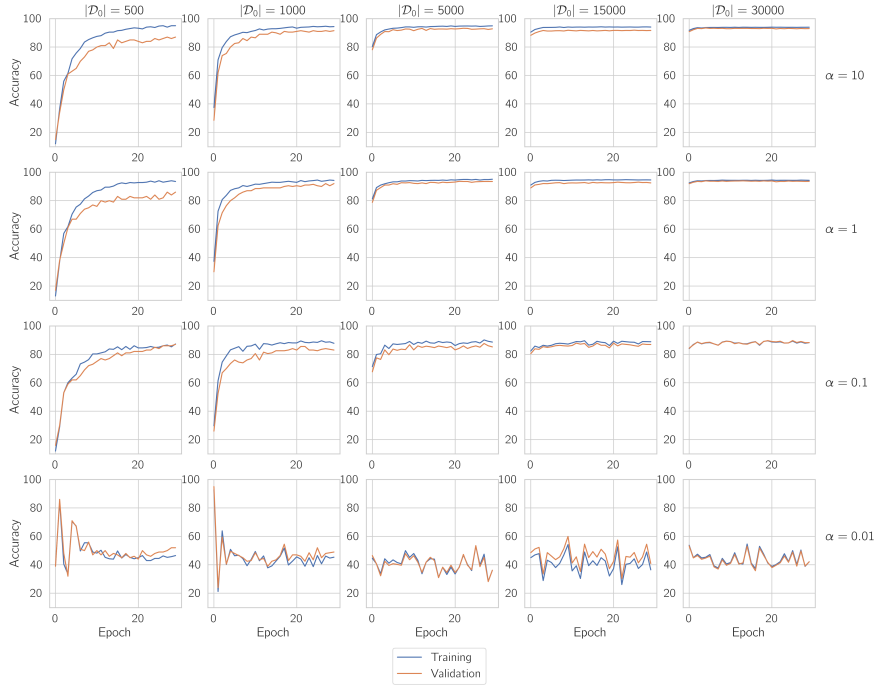


Figure B.7: MNIST: FedED-w1 student training accuracy with the CNN3 model and MSE loss for 1 seed.

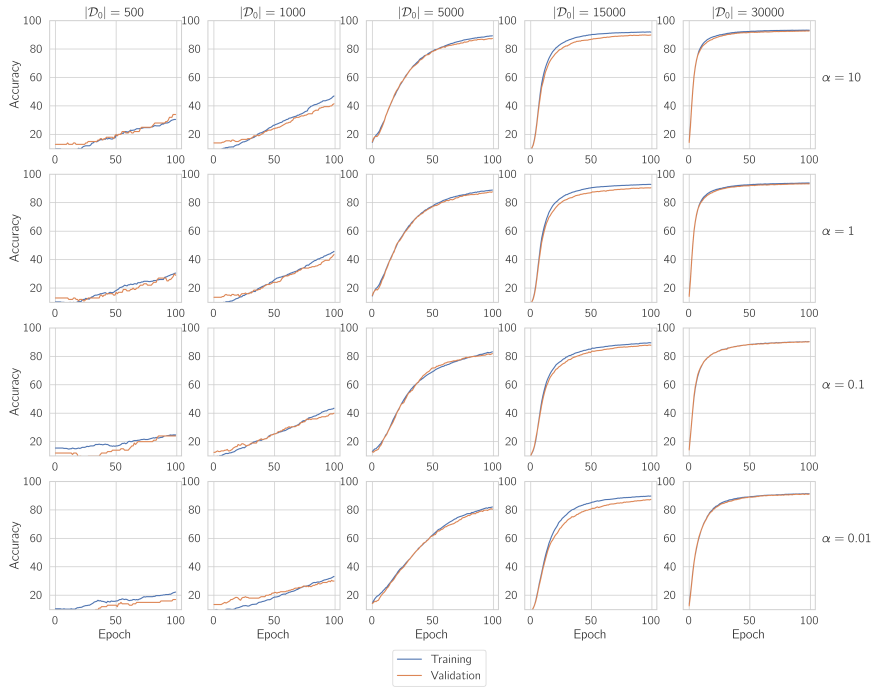


Figure B.8: MNIST: FedED-w2 student training accuracy with the CNN3 model and MSE loss for 1 seed.

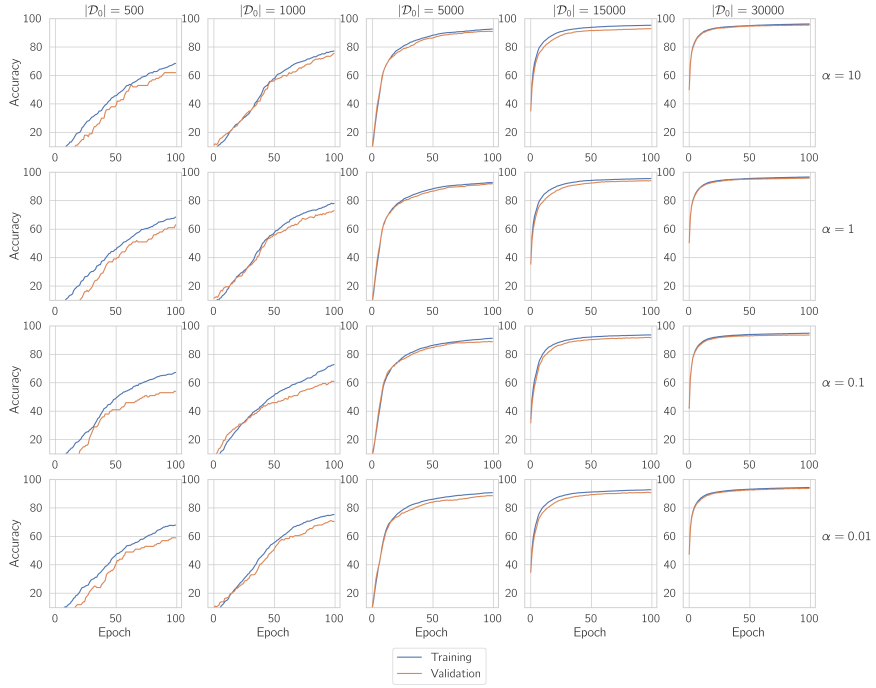


Figure B.9: MNIST: FedED-w0 student training accuracy with the CNN3 model and CE loss for 1 seed.

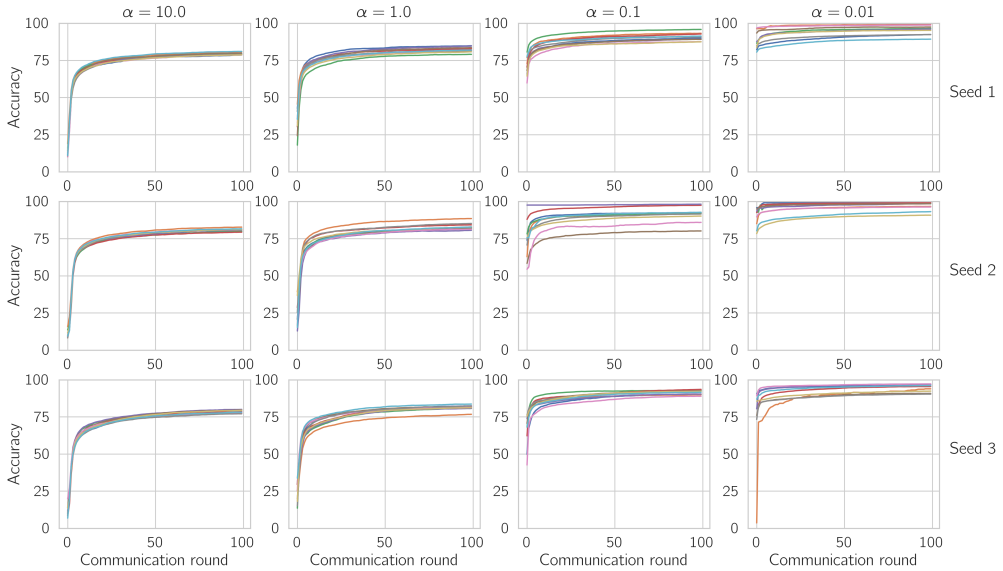


Figure B.11: EMNIST: FedProx local training accuracies with the CNN1 model for 3 different seeds.

B.2 EMNIST

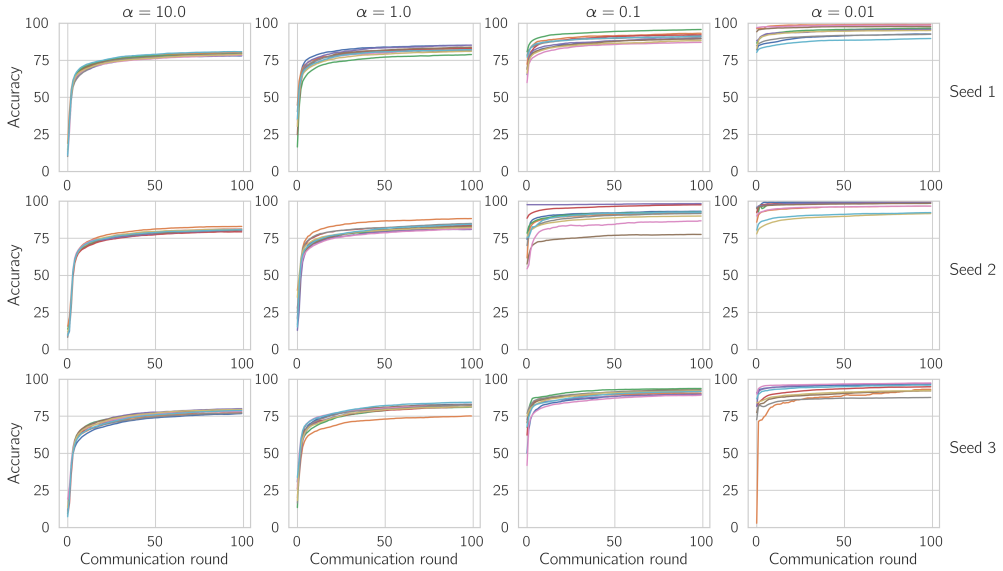


Figure B.10: EMNIST: FedAvg local training accuracies with the CNN1 model for 3 different seeds.

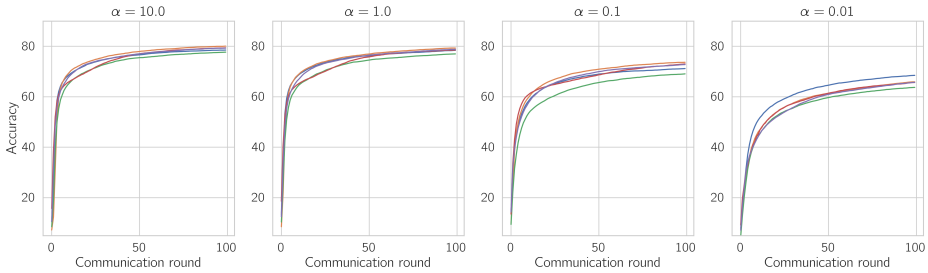


Figure B.12: EMNIST: FedAvg global test accuracy with the CNN1 model for 5 different seeds.

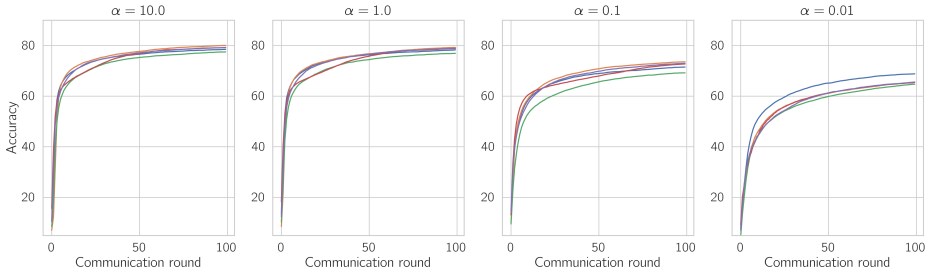


Figure B.13: EMNIST: FedProx global test accuracy with the CNN1 model for 5 different seeds.

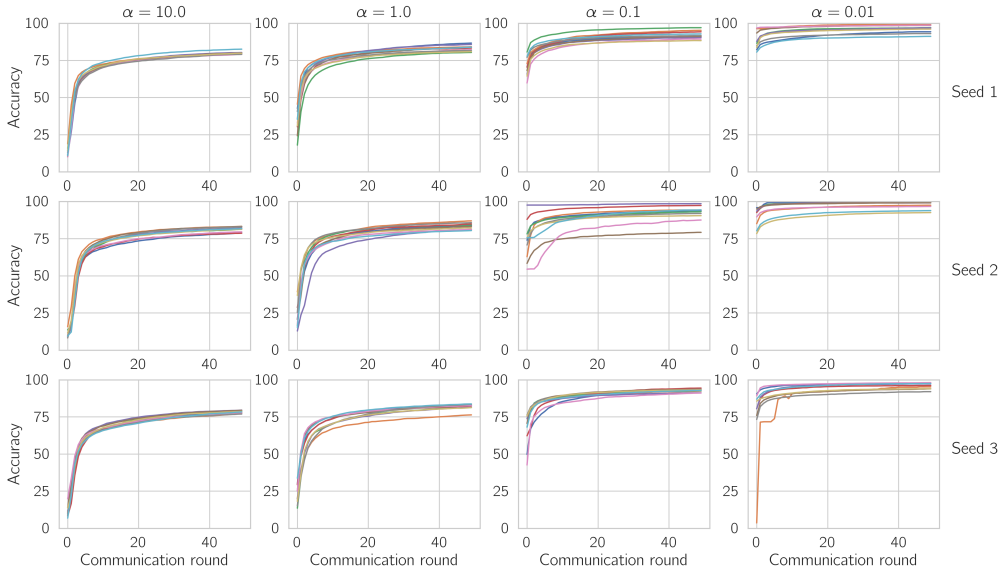


Figure B.14: EMNIST: FedED local training accuracies with the CNN1 model for 3 different seeds.

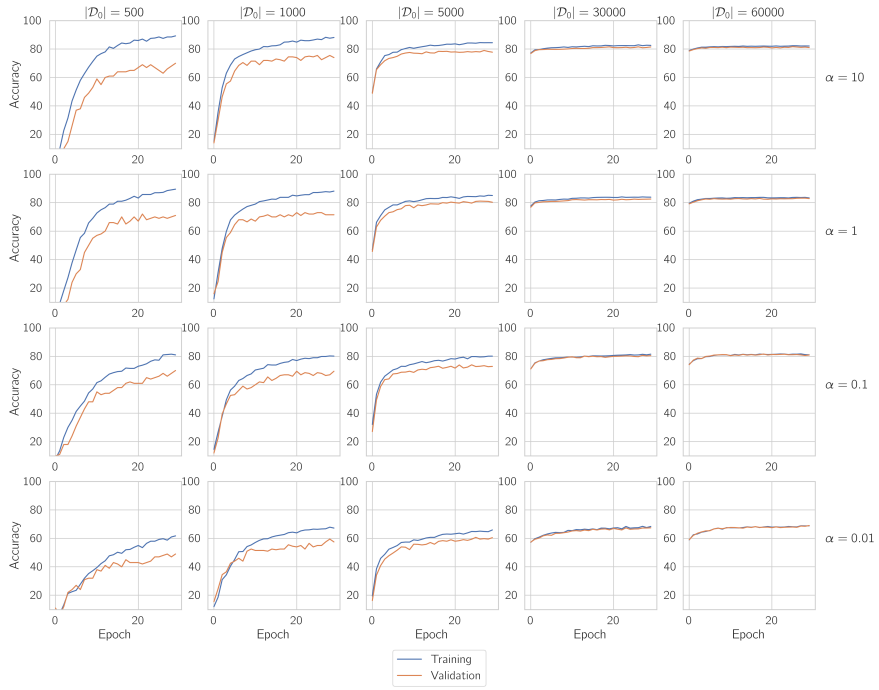


Figure B.15: EMNIST: FedED-w0 student training accuracy with the CNN3 model and MSE loss for 1 seed.

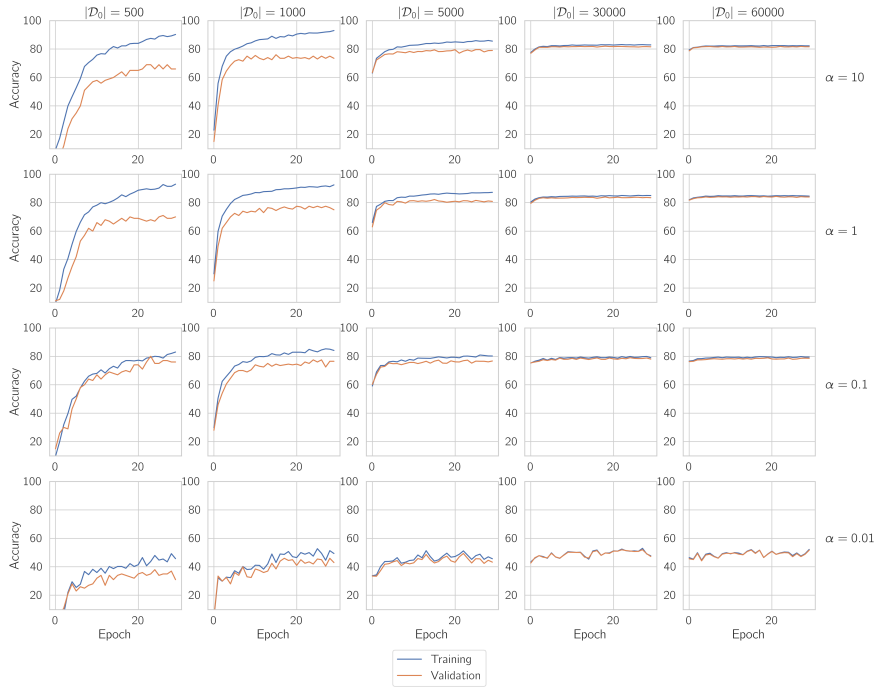


Figure B.16: EMNIST: FedED-w1 student training accuracy with the CNN3 model and MSE loss for 1 seed.

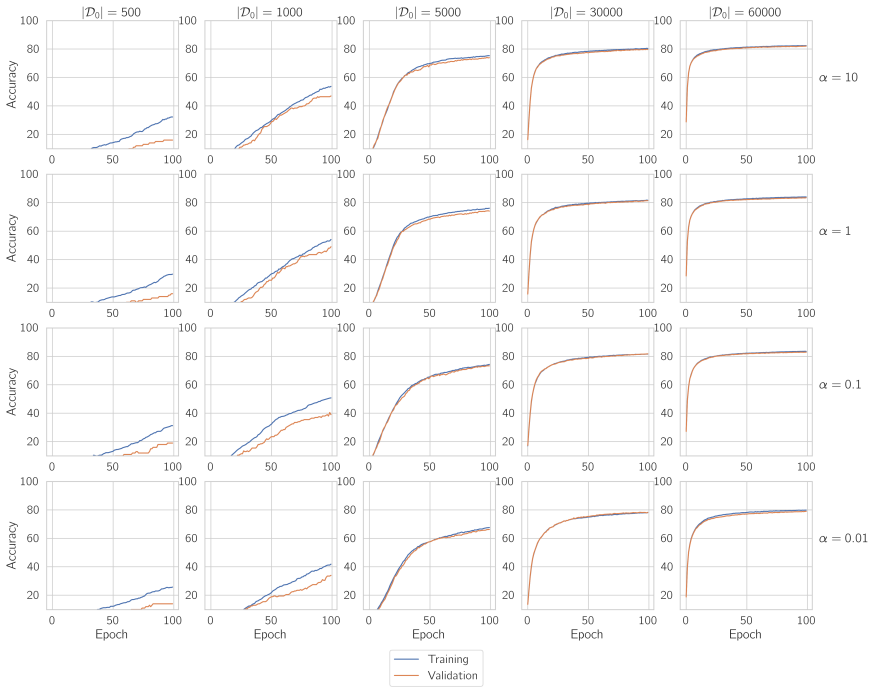


Figure B.17: EMNIST: FedED-w2 student training accuracy with the CNN3 model and MSE loss for 1 seed.

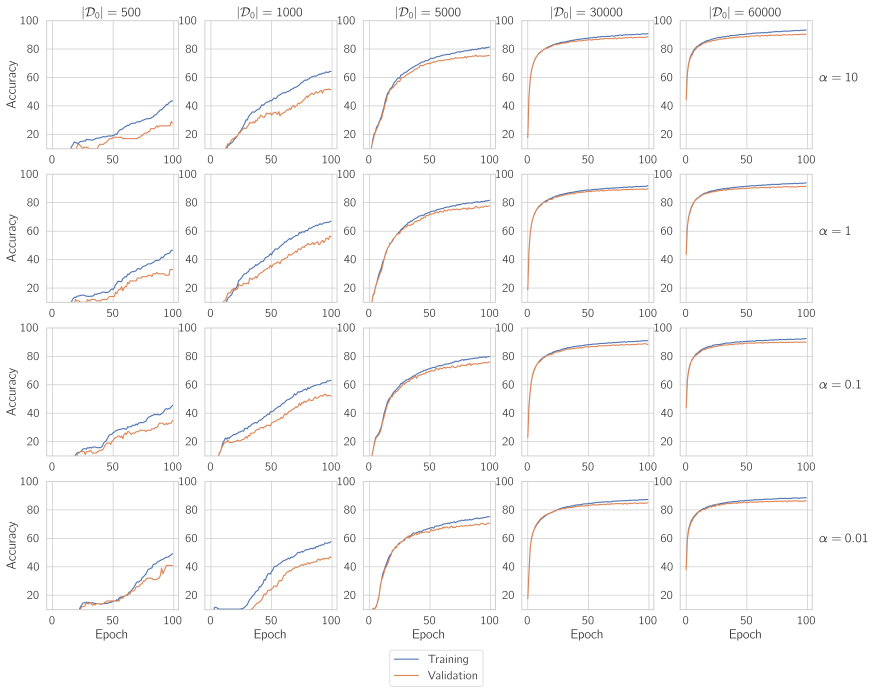


Figure B.18: EMNIST: FedED-w0 student training accuracy with the CNN3 model and CE loss for 1 seed.

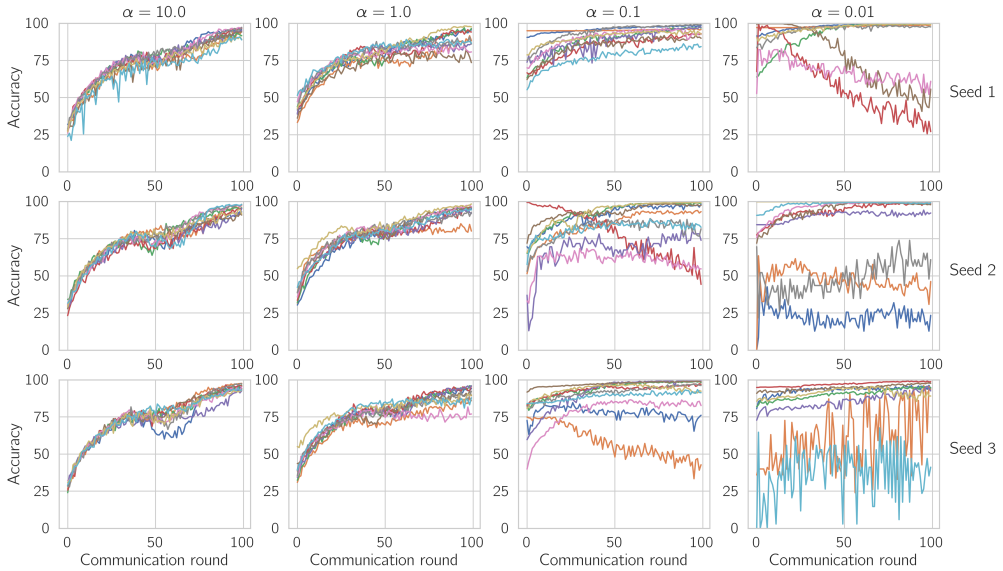


Figure B.20: CIFAR-10: FedProx local training accuracies with the Resnet-18 model for 3 different seeds.

B.3 CIFAR-10

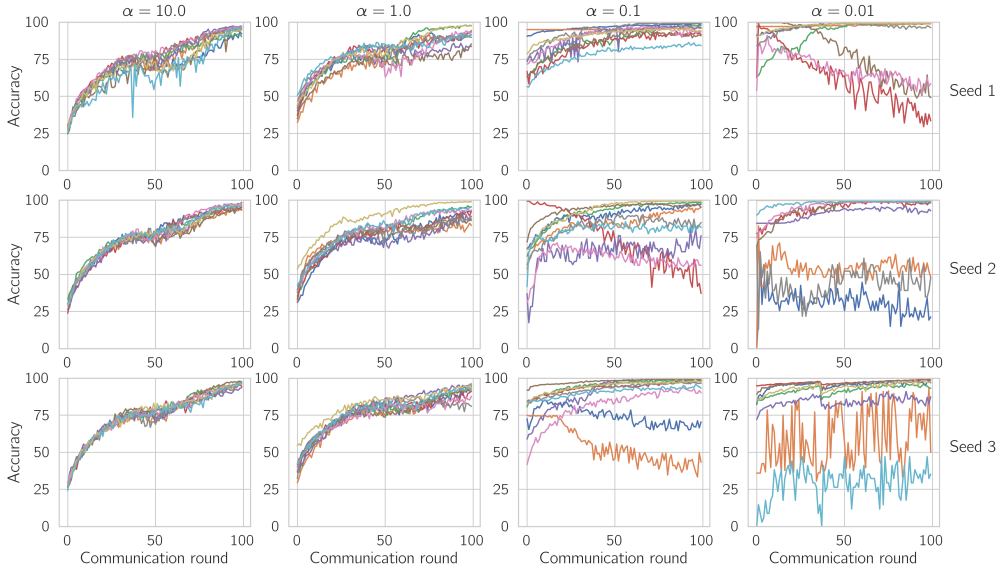


Figure B.19: CIFAR-10: FedAvg local training accuracies with the Resnet-18 model for 3 different seeds.

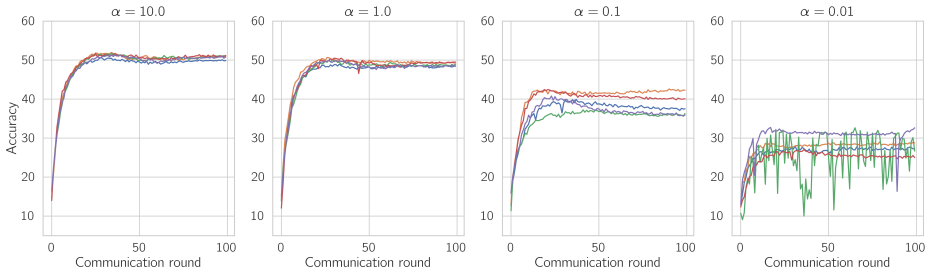


Figure B.21: CIFAR-10: FedAvg global test accuracy with the Resnet-18 model for 5 different seeds.

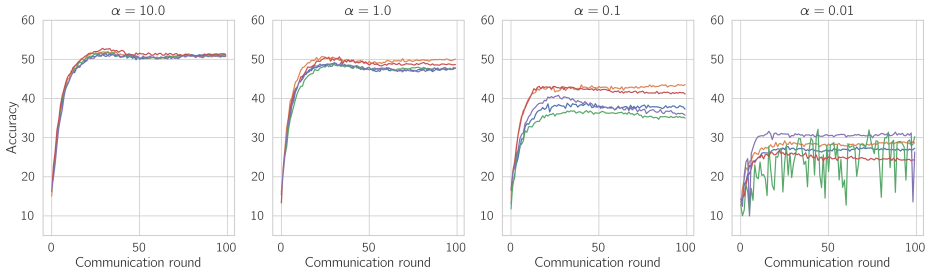


Figure B.22: CIFAR-10: FedProx global test accuracy with the Resnet-18 model for 5 different seeds.

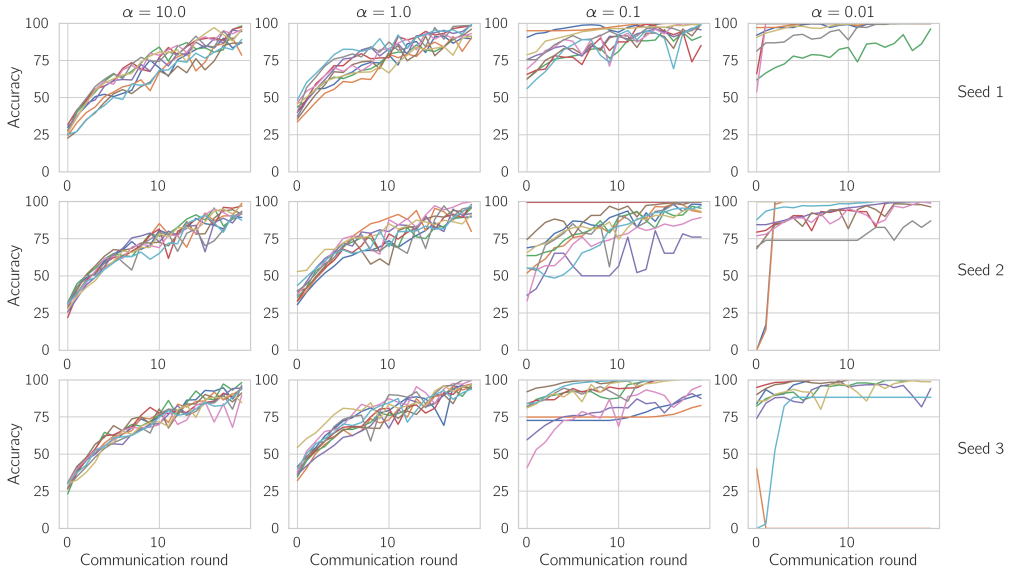


Figure B.23: CIFAR-10: FedED local training accuracies with the Resnet-18 model for 3 different seeds.

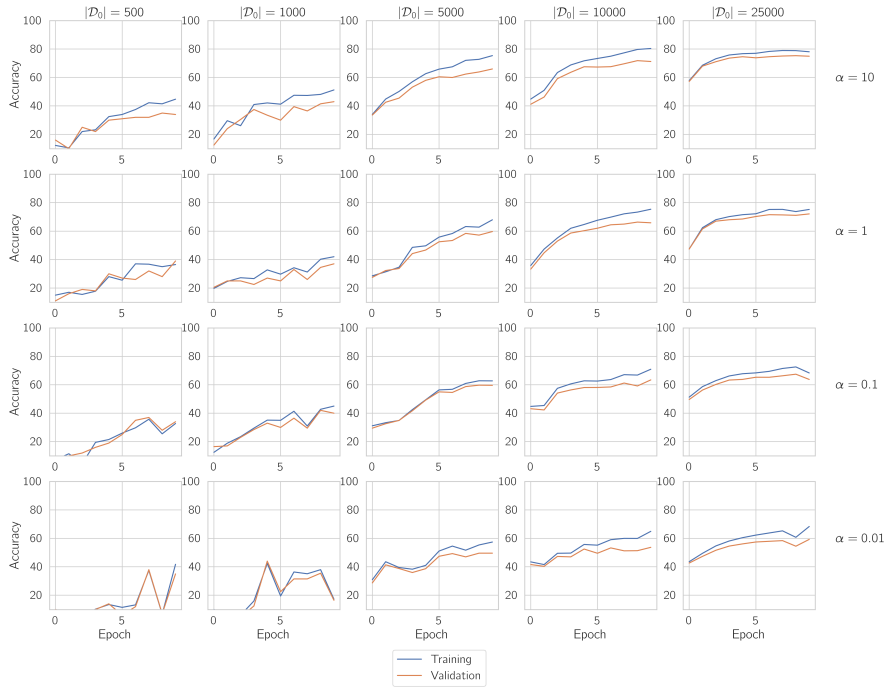


Figure B.24: CIFAR-10: FedED-w0 student training accuracy with the Resnet-18 model and MSE loss for 1 seed.

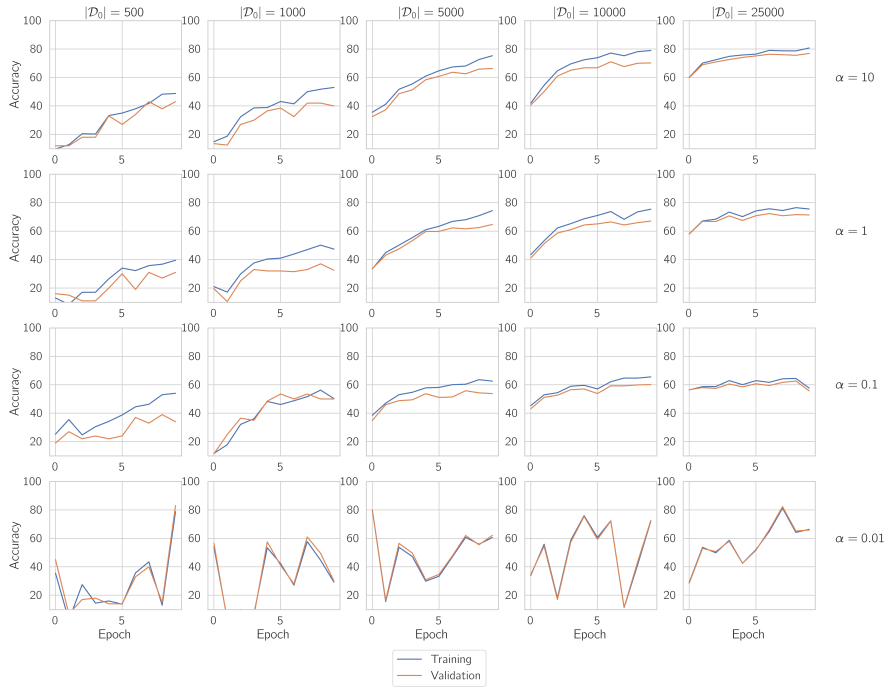


Figure B.25: CIFAR-10: FedED-w1 student training accuracy with the Resnet-18 model and MSE loss for 1 seed.

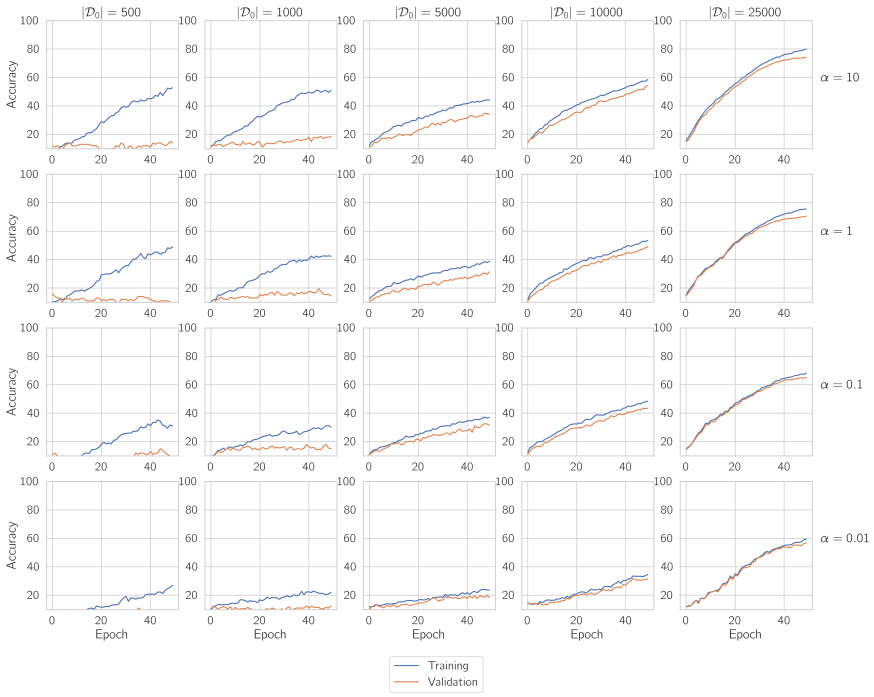


Figure B.26: CIFAR-10: FedED-w2 student training accuracy with the Resnet-18 model and MSE loss for 1 seed.

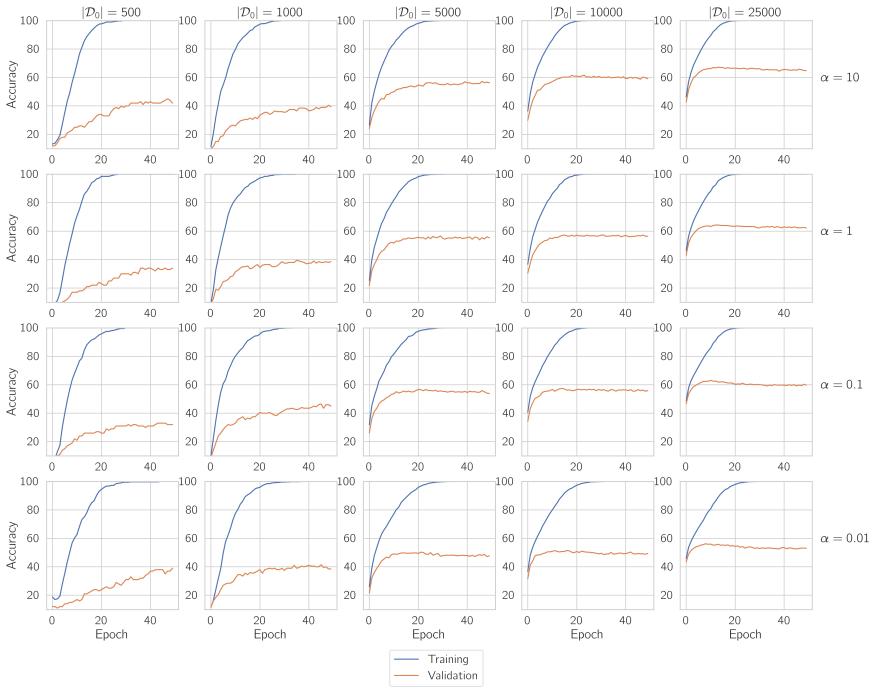


Figure B.27: CIFAR-10: FedED-w0 student training accuracy with the Resnet-18 model and CE loss for 1 seed.

DEPARTMENT MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY