FEDERATED LEARNING FROM NON-IID DATA

IMPROVING ACCURACY THROUGH DATA-AUGMENTATION AND COMMUNICATION EFFICIENCY

Master Thesis

To obtain the degree of Master of Science in Embedded Systems, within the field of software, specialized in Distributed Systems, at the Delft University of Technology, to be defended publicly on Wednesday February 23, 2022 at 14:15.

By

Izaak CORNELIS

Student Number	4392701	
Project duration	February 2021 - February 2022	
Thesis Committee	Dr. L. Chen	TU Delft, supervisor
	Dr. G. Lan	TU Delft

Abstract

Federated learning allows multiple parties to collaboratively develop a deep learning model, without sharing private data. Models can be generated from the most up-to-date data while taking unique and not publicly available data into account. However, the distributed nature of federated learning causes problems too, and clients are not guaranteed to hold independently identically distributed (iid) data, causing performance degradation.

This work analyzes existing methods of generating such skewed datasets and finds that the Earth Movers Distance (EMD) can be used to compare them. A novel scheme called *phase-shift* is introduced, which allows clients to communicate more frequently, without increasing communication, hereby reducing drift caused by non-iid data. Finally, we propose a data-driven approach that can reduce the data skew by supplementing local datasets with augmented data. A novel method of balancing unaltered and augmented data is introduced, taking the skew of the dataset into account.

Empirical analysis shows that phase-shift can reduce the instantaneous communication load on the system by 37.5% without suffering a performance loss or reducing convergence rate. Evaluation of data augmentation on a heavily skewed cifar10 dataset shows that accuracy is improved by 10%. Finally, phase-shift and data augmentation are combined, resulting in a 13% accuracy improvement, surpassing algorithms such as `FedNova` and `FedProx` when dealing with label-heterogeneity.

Contents

1

Introduction

Learning requires good data, and a model is only as good as the data supplied to the learning algorithm. The *Information processing inequality* dictates: post-processing cannot increase information. Thus, to derive sufficient models, the amount of information in the data must also be sufficient.

Datasets that include the most recent, extensive and feature-rich samples are distributed. Such data is decentralized, collected under different circumstances, and stored on devices with varying capabilities and connectivity. Creating a single data set from multiple sources might be impossible or unethical due to practical or privacy-related reasons.

Federated learning (FL) has been proposed [2] to address these issues. It allows the derivation of a model from many sources without the raw data becoming public. It allows models to be generated from the most up-to-date data while taking unique and not publicly available data into account. Federated Learning has sparked new research in privacy conserving collaborative learning, and has been applied in practice [9] [1] [12].

1.1. Federated learning

In federated learning, data never leaves devices to preserve privacy. Thus, learning must happen locally, and clients can only learn from their own local dataset. Local data is not representative of the full learning problem, nor complete, nor large enough to support local training only. Clients send their trained model to a central server, called the federator. The federator aggregates these multiple models into a single model, combining the unique insights provided by individual clients. This new model is then sent back to clients to spark a new round of training. The process of common model initialization, local learning, and aggregation continues until convergence. The algorithm that defines these steps, as proposed by Brendan McMahan et al. [2], is called `FedAvg`.

1.2. PROBLEMS IN FEDERATED LEARNING

Decentralized data is thus an inherent feature of federated learning. However, it is also an inherent problem. Research has shown that FL under-performs when compared with centralized machine learning (ML), given a realistic data set [2] [10] [11] [13]. Realistic is a keyword here, and what makes a federated dataset realistic? Data is generated decentralized, by unique devices, giving unique insights but also causing problems. These problems can be divided into four categories:

1. **Label-heterogeneity or label-skew** In a typical classification problem, the dataset should consist of an equal number of samples from each class, which is not guaranteed in FL.

2. **Quantity-heterogeneity** Devices can hold a different number of samples, causing some clients to learn more per round than others, causing issues at the aggregation stage

3. **Feature-heterogeneity** features can be unequally distributed.

4. **Resource-heterogeneity** Devices can have different resource constraints, such as computational power, or connectivity.

The first three are collectively called data-heterogeneity, of which the first two pose the greatest challenge [11] [16] [41].

As said by Hsieh et al. [10]: *"Intuitively, there is a tug-of-war among different data partitions, with each partition pulling the model to reflect its data.."*

The problem is known as the non-iid data problem, as data among partitions (devices) is not independent and identically distributed. Some recent studies have tried to find solutions for quantity-heterogeneity [17] [31], or for resource-heterogeneity [4] [21] [36] [33] [25]. Furthermore, non-iid data makes it difficult to distinguish a malicious update from a genuine update, causing problems for security as well [3] [7] [35]. All these works have found that label heterogeneity worsens the impact of other problems. For instance, a typical method to deal with resource heterogeneity is to group faster or slower clients. However, the type of data might be correlated with the type of device, and can thus cause additional label-skew. The same can be argued for quantity skew. Therefore, it might be worth attacking the label-skew problem separately.

1.3. PROBLEM DEFINITION AND RESEARCH QUESTIONS

This work considers the non-iid label distribution problem in federated learning. To focus on this problem specifically, all other forms of heterogeneity ignored. This allows us to clearly see the impact of label skew, and thus the following assumptions are made:

1. Each client has the same number of samples

2. Each client has the same capabilities, both in communication and computation

3. Each participating client is available whenever requested by the federator

4. Each client participating in a round of federated learning is guaranteed to return a result to the federator at the end of the round.

5. Features are uniformly distributed.

6. Each client has a *non-uniform* distribution of the classes of the problem

1.3.1. MEASURING SKEW

Many papers related to Federated learning discuss the effects of non-iid data and perform empirical evaluations of proposed ideas to deal with these effects. However, no standard way of generating a skewed dataset exists, making it hard to compare results or to judge the challenge posed by the degree of skewness. This is further hampered by the lack of a standard way to measure - independent of the distribution method - the degree of skewness. Ideally, such a method would exist, any two distributions, with the same degree of skew, would result in similar performance. One solution has been proposed, called Earth-Moving-Distance (EMD) [40], and tested for a single distribution method. Results are promising, but research into other distributing methods is missing. Herein lie the first two research questions:

RQ1 **Can EMD be used to quantify the degree of skew, such that two non-identical distributions, but with the same degree of skew, result in similar performance?**

RQ2 **Can the existing distribution methods, for artificially creating non-iid label distributions, be compared using EMD?**

Answering these questions should aid in reviewing related work, and lead to an understanding of how factors such as the number of classes, or the number of clients, affect the level of skew.

1.3.2. REDUCING COMMUNICATION

A simple method to improve performance is to reduce the number of iterations through the local dataset [10]. However, this increases the total data transfer, and the instantaneous communication load on the server, limiting the number of clients that can partake in a round [25]. Several methods that reduce the communication load of federated learning exist. The standard is to use compression schemes such as gradient sparsification [25] [20] [8], but this comes at the cost of computational overhead. On the other hand are methods that are specific to federated learning, such as client scheduling or client selection [33] [6] [30]. However, actively selecting clients can result in a biased data selection. Furthermore, these methods rely on increasing convergence rate and thus do not decrease the instantaneous load on the system. Herein lies the following research question.

RQ3 **How can the communication frequency be increased, without increasing the instantaneous communication and computational load on the system?**

Answering the question will have implications for research into dealing with non-iid data and reducing communication.

1.3.3. REDUCING SKEW

Several solutions have been introduced to deal with non-iid data, such as the algorithmic solutions `FedNova` [32], `FedProx` [17] and `Scaffold` [13]. However, far greater improvements are obtainable by attacking the problem at the root cause: data-skew. Data-driven approaches such as `data-sharing` [40] and `FedMix` [38] reduce skew by sharing a public dataset, or encoding and redistributing private data. These solutions come at the cost of huge communication overhead. Duan et al. [6] propose an alternative and use data augmentation to deal with global data skew. Clients' data samples are augmented to increase the number of samples such that the global data distribution appears uniform. Data augmentation has several advantages over the previous data-driven methods. It completely mitigates the need for data transfer, removing the communication overhead. Furthermore, it does not rely on the existence of a publicly available dataset.

However, the research of Duan et al. is lacking in two areas. First, the test setup used to show the improvement over `FedAvg` is flawed. Data augmentation increases the number of samples in each clients' local dataset, but this is not corrected for in the `FedAvg` algorithm. The proposed solution performs more learning iterations per round than the baseline `FedAvg`, causing an unfair advantage. Second, research into how the amount of augmented and unaltered data should be balanced is lacking. Thus, two additional research questions can be defined:

RQ4 Can data augmentation be used to complement each client's local dataset, to reduce skew and increase performance? or is it preferable to spend the additional learning on unaltered data instead?

RQ5 How can the right balance between unaltered and augmented data be achieved?

1.4. CONTRIBUTIONS

The contributions to federated learning are threefold. First, the existing methods of artificially creating non-iid label skew are analyzed and compared. Research questions one and two deal with this topic, and aim to understand how different distribution methods impact performance. EMD is put forward by related work as a measuring unit to define skew and tested in this work to quantify its ability to measure skew of other existing distribution methods. This work can serve as a reference when comparing results from related work.

Second, a novel method, dubbed *phase-shift*, is introduced, capable of reducing communication, by shifting away from a single centralized model, and allowing clients to operate out of phase from each other. Phase-shift is a unique contribution to the field of federated learning and has interesting implications for further research.

Finally, an existing method of dealing with non-iid labels skew through data augmentation is verified and improved. A novel method of balancing unaltered and augmented data is introduced, that links this balance to the skew of the dataset.

1.5. ORGANIZATION

This work is organized as follows. First, the background and related work are discussed in chapter 2. The ability of EMD to compare existing data distribution methods is researched in chapter 3. Chapter 4 introduces phase-shift; a novel scheme that can reduce communication without causing a performance drop. Using the results from chapter 3, chapter 5 discusses the ability of data-augmentation to reduce label-skew, and how unaltered and augmented data should be balanced. The results from chapter 4 and 5 are combined and compared against related work in chapter 6. Finally, the implications, key findings and future work are discussed in chapter 7.

2

Related work and background

2.1. Federated learning

In federated learning, K devices (clients), each with a dataset D_k, aim to optimize the following problem.

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \quad \left[F(\boldsymbol{w}) := \sum_{k=1}^{K} p_k F_k(\boldsymbol{w})\right] \tag{2.1a}$$

$$\text{where} \quad p_k = \frac{n_k}{\sum_{k=1}^{K} n_k} \tag{2.1b}$$

$$F_k(\boldsymbol{w}) = \frac{1}{n_k}\sum_{\xi\in D_k} f_k(\boldsymbol{w};\xi) \tag{2.1c}$$

In this equation, each local objective function F_k is weighted by the number of samples n_k a particular client k has in its dataset D_k. The local loss function f_k is defined by the model and ξ represents a sample from the local dataset.

Federated learning proceeds in rounds, and during a round t a selection of clients run T_k iterations of a local solver (usually SGD) and communicate the result back to the central server (the Federator). The federator has to combine the different models using some aggregation function, before sending the result to a new selection of clients to be used in the next round. The aggregation function most commonly used is `FedAvg` [2], which takes the weighted average of the model-updates. Clients' models are weighted by the number of samples they possess, as can be seen in equation 2.2.

$$\texttt{FedAvg:} \quad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \sum_{k=1}^{K} \frac{n_k}{\sum_{k=1}^{K} n_k}\left(\boldsymbol{w}_t^{(k)} - \boldsymbol{w}_t\right) \tag{2.2}$$

Where w_{t+1} is the central model at round $t+1$, and $\boldsymbol{w}_t^{(k)}$ is the updated model after local learning on client k. This equation assumes that all clients participate, but `FedAvg` can be changed to allow for partial client selection. In cases where the number of clients is very large, this might be become necessary [25].

2.2. INTUITIVE EXPLANATION OF THE PROBLEM

As explained in the introduction, federated learning performs worse when compared to centralized ML, given a realistic dataset. The root cause for the difference in accuracy between federated learning and centralized learning is non-iid data. However, the link between cause and problem is not directly clear.

Each client has a unique dataset, and thus a specific loss function over this data. This causes the local optima for the model weights to be different among clients, which in turn causes the clients to drift apart before synchronization.

The clients pull the local model away from the global optimum at different rates, causing the average of the client updates to be away from the global optimum. This phenomenon has been dubbed *weight-drift* or *weight-divergence* [40] [13]. Take a look at figure 2.1 for a visual indication.

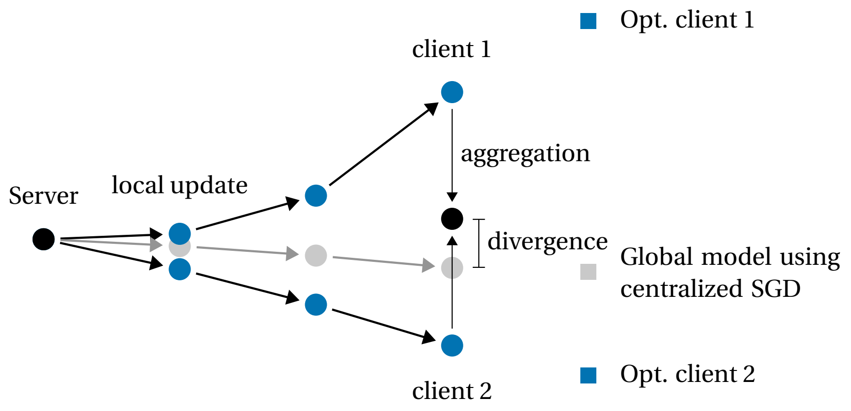


Figure 2.1: Weight divergence intuition

A better understanding of the issue allows us to look at how this might be solvable. The problem can be attacked in several areas:

1. **Reducing skew:**The root cause is non-iid data, and complementing the local datasets with a (parts of) a public data set can move the local optima closer to the global optimum. e.g. `data-sharing` [40]

2. **Local objective:** The SGD update rule can be modified to steer the local updates back to the global optimum. e.g. `SCAFFOLD` [13], `FedProx` [17]

3. **Aggregation:** The updates of all clients can be combined in such a way that the result is closer to the global optimum. e.g. `FedNova` [32].

4. **Increasing communication** Finally, the number of local SGD updates can be reduced, limiting the drift away from the global optimum. This has been the most common method to improve performance for `FedAvg` when faced with non-iid data. `SkewScout` uses frequent communication to reduce the negative effect of non-iid data.

2.3. RELATED WORK: DEALING WITH NON-IID DATA

As discussed before, the problem of non-iid data can be attacked from different angles. Existing methods of dealing with non-iid data will be categorized based on this classification scheme.

2.3.1. REDUCING SKEW

The simplest method to attack the problem is to remove the root cause: the missing samples that cause a skewed distribution. A publicly available dataset could be used to complement local datasets, as proposed by Zhao et al. [40]. First, they show that weight divergence causes the performance drop on non-iid data. Weight divergence is bounded by the earth mover's distance or EMD for short. EMD measures the difference between two distributions and, in this case, compares local distributions to the combined global distribution. They argue that to reduce weight divergence, EMD must be lowered. A small subset of a globally shared dataset is sent to clients to complement their local dataset. Experiments show that accuracy can be increased by 30% on Cifar10 with only 5% globally shared data.

This solution requires a public dataset that matches the problem, and, for clients to download the public dataset, increasing communication. The communication overhead could be minimal if the number of clients is small, and clients participate multiple times in the learning process. However, if the number of clients is very large, and clients are not likely to partake more than once, the overhead becomes infeasible. Finally, overfitting the shared dataset could become a problem, especially if the dataset is kept small to reduce communication.

To avoid these problems with a public dataset, Duan et al. [6] choose to augment local data instead. A self-balancing federated learning framework, called Astraea, is introduced. Unlike the previous solution that focuses on local non-iid data, Astraea deals with a globally skewed dataset and uses augmentation to supplement data and make the global dataset uniform.

Two problems can be identified in this solution. First, no theoretical or practical rule exists for deciding the balance between unaltered and augmented data. Furthermore, the meta-parameter that controls this balance has no connection to the degree of skew, so a new setting needs to be found when the distribution changes. Secondly, the experiment setup is an unfair comparison to the baseline `FedAvg`. By augmenting, additional data is introduced, and Astraea is allowed to learn on the original unaltered and the augmented dataset, whereas the baseline is only allowed to learn from the original dataset. Thus, Astraea performs more learning iterations to cover both datasets. A fair comparison should have upsampled the unaltered dataset such that `FedAvg` is allowed to perform the same number of learning iterations. Due to this flaw in the test setup Duan et al. are unable to show that data-augmentation is useful, and whether or not more computation on existing data can solve the problem as well.

Another alternative is for users to exchange data, such that their local distributions appear uniform. A straightforward implementation of this is a privacy concern and goes against the idea of Federated Learning. Shin et al. [26] propose an XOR mix-up method that allows exactly this to happen while guaranteeing privacy. The federator collects the Clients' xor-encoded samples and decodes them using its own samples to form a uniform dataset. Unlike regular federated learning, the server performs all training, and the process consists of a single round. Although privacy is preserved, client data must be transferred resulting in large communication overhead. Furthermore, the federator must have access to a publicly available dataset for decoding.

Yoon et al. [38] remain closer to the original scheme of FL, and avoid the need for a publicly available dataset. `FedMix` is proposed, and instead of using XOR operations, clients send and receive averaged local data. Instead of collecting this data only at the server, data is redistributed to supplement local datasets. Again, this method comes at the cost of large communication overhead.

2.3.2. LOCAL OBJECTIVE

Several works attack the problem of non-iid data by changing the local objective function. Karimireddy et al. [13] comes to a similar conclusion as Duan et al. and argue that non-iid data causes clients to drift to a local optimum instead of the global optimum. To correct for this drift, a control variable $\boldsymbol{c}^{(k)}$ is introduced and maintained similar to the local model $\boldsymbol{w}^{(k)}$. This new variable is updated locally and aggregated at the federator, same as the local model. The objective function is changed to include both the local and global versions of the control variable, as can be seen in equation 2.3. The control variable is derived from a convergence analysis of `FedAvg` on non-iid data and calculated accordingly. `Scaffold` is shown to improve 1.4% in accuracy on a highly skewed EMNIST dataset.

$$\boldsymbol{w}^{(k)} \longleftarrow \boldsymbol{w}^{(k)} - \eta\big(\nabla f_k(\boldsymbol{w}^{(k)};\xi) + \boldsymbol{c} - \boldsymbol{c}^{(k)}\big) \tag{2.3}$$

Where η is the learning rate. The updated local objective allows for quicker convergence, but requires double the communication per round, as the control variable is aggregated as well. The improvement of 1.4% is not nearly enough to rectify the loss due to non-iid data, and the effectiveness of `Scaffold` on other datasets, such as cifar10 is missing.

Instead of correcting for client drift directly, Li et al. [17] choose to limit how far the local model is allowed to drift from the global model. Similar to `Scaffold`, `FedProx` bases its findings on a convergence analysis of `FedAvg` on non-iid data. The objective function is altered to penalize drift from the global model, and `FedProx` instead minimized the function shown in equation 2.4. Here, the meta-parameter μ controls how much the objective function is penalized for drift. This parameter requires tuning, and Li et al. provide no rule to help select this value. Their theory shows that choosing $\mu > 0$ should improve convergence when learning over non-iid data, but practice shows that this is not the case for non-iid label distributions.

$$F_k(\boldsymbol{w}^{(k)} + \frac{\mu}{2}||\boldsymbol{w}^{(k)} - \boldsymbol{w}||^2 \tag{2.4}$$

2.3.3. AGGREGATION

The standard `FedAvg` uses the weighted average of clients' models to arrive at a centralized model. [32] show that this causes objective inconsistency when dealing with non-iid data. It takes into account that clients might need a different number of learning steps per round. Clients that perform more steps, will return a larger update and thus more significantly impact the global model, even when the datasets of all clients are the same size. An algorithmic modification called FedNova is proposed, which normalizes the client updates at the aggregation stage, to ensure that the updates are not biased. Although this mainly deals with quantity or resource heterogeneity, results indicate that it performs well under label heterogeneity as well.

2.3.4. COMMUNICATION

Some authors argue that the non-iid data issue can only be solved through close communication. Hsieh et al. [10] introduces SkewScout, algorithm that controls communication based on the degree of skew. Skew is measured by sending the model trained on one client, to another, and measuring relative performance loss. This idea is tested decentralized learning, not in an FL setting, but the idea could be interesting to FL as well. Although SkewScout controls communication by the degree of skew, total communication is increased.

2.4. RELATED WORK: COMMUNICATION EFFICIENCY

Communication is inherent to federated learning and should be treated as a limited resource. Communication efficiency in federated learning is a research field in and of itself, and several works have introduced new methods to reduce communication.

To address the communication issue, compression methods have been introduced in distributed learning and Sattler et al. [25] adapt these techniques to a federated learning environment. STC is introduced, a top-k gradient sparsification with a novel mechanism to enable both up-and-downstream compression. Communication can be reduced significantly, and interestingly STC performs very well under non-iid label conditions. Much larger improvements than those found by researchers focusing on non-iid data specifically are found, raising questions about the validity of these results. Any theory as to why STC can deal with non-idd data is missing. Finally, other works that use gradient sparsification do not report similar performance improvements [8] [20] [24].

Wang et al. [30] uses deep Q-learning to select a subset of devices at each round. The learning process is rewarded for increased validation accuracy, and penalized for the use of more rounds. Using this technique, a 49% reduction in communication is found for Cifar10, compared to FedAvg. However, rewarding on convergence rate might force the algorithm to prefer clients whose dataset is more iid. A correlation might exist between the type of data held by clients, and their ability to increase accuracy over all data. The proposed learning algorithm might create a bias, that ignores clients that hurt convergence, even if these clients hold crucial data.

In addition to the already discussed augmentation, Duan et al. [6] introduces a mediator based multi-client scheduling scheme which uses client distribution to schedule clients. The federator talks to the mediators, while only the mediators communicate with the clients. Clients are assigned to a mediator such that their combined distribution appears uniform. Using this scheme, communication can be reduced by speeding up convergence.

Wang et al. [33] propose an algorithm that tries to find the best trade-off between the number of local updates, and the global model aggregation, subject to a resource-constrained loss function. Even dynamically changing constraints are considered. Essentially a dynamic meta-parameter optimization is proposed. However, this idea underperforms when faced with non-iid data. This makes sense: allowing clients to perform more local updates, skews the local models more to fit the data.

2.5. DATA DISTRIBUTION METHODS

All related works that perform some empirical validation introduce some an artificial label skew in an otherwise iid dataset. Different authors use different methods and different meta parameters to distribute data. Furthermore, it is unclear how parameters such as the number of clients or the number of classes impact the distribution method. This makes it difficult to judge the level of non-iidness used to evaluate proposed solutions. Please refer to chapter 3 for an overview of the different distribution methods used. Here, the different methods are compared, and the impact of the number of clients and classes is researched. Specifically, take a look at table 3.1 for an overview of the distribution methods used by related work.

3
EMD ANALYSIS

Many papers related to federated learning use different methods of generating a non-iid distribution for empirical validations. This involves creating partitions from an existing dataset such as Cifar10 while ensuring all samples appear in exactly one partition. A standardized way of measuring the *degree of skew* has not yet been defined, and thus it is hard to compare results from different papers. This chapter aims to find such a measurement unit for skew that can be used to compare these results, i.e., two non-identical distributions with the same degree of skew should result in similar performance.

Furthermore, it should be possible to calculate the skew with knowledge of how the data was distributed, not necessarily with knowledge of the distribution itself. This allows for existing data distribution methods to be compared.

3.1. Measurement unit for skew

Quantifying the degree of data skew can by done by comparing the local distributions to the global distribution and quantifying the differences. Li et al. [18] defines the degree of skewedness as:

$$\Gamma = F^* - \sum_{k=1}^{K} v_k F_k^* \tag{3.1}$$

Where F^* and F_k^* are the minimum values of the global (centralized) and local cost functions. v_k is the weight or value assigned to client k, and K is the total number of clients. This definition is difficult to compute as it requires the model to be learned in both the FL and the centralized setting. It also relies on the weight assigned to each client, which is solution-dependent.

The unit is closely tied to federated learning and the difference between centralized and federated learning, and thus makes sense as a unit of measurement. It can also quantify the skew of any distribution. However, it gives no method to distribute data to reach a certain degree of skew, which makes it difficult in testing.

Another method of measuring skew between distributions is proposed by Duan et al. [6], who use KullbackLeibler divergence:

$$D_{KL}(p_k||p) = \sum_{i=1}^{M} p_k(Y=i)\log\left(\frac{p(Y=i)}{p_k(Y=i)}\right) \tag{3.2}$$

where M is the number of classes in the global data set, p is the probability distribution over the classes in the global distribution, and p_k is the probability distribution over the data in client k.

However, no theoretical reason as to why this method is used is provided. Furthermore, the KullbackLeibler divergence is a measurement between two distributions, and it can be calculated between every local and the global distribution. One option to arrive at a single value is to aggregate the results using `FedAvg`.

A more convenient method is proposed by Zhao et al. [40] that mitigates the learning and directly compares the distributions by using the Earth Mover Distance (EMD or Wasserstein distance)

$$EMD = \sum_{k=1}^{K} \frac{n_k}{\sum_{k=1}^{K} n_k} \sum_{i=1}^{M} \left| p_k(Y=i) - p(Y=i) \right| \tag{3.3}$$

Where K is the number of clients and n_k is the number of examples in the training data of client k, EMD turns up in the bound on weight-divergence, and Zhao et al. show that it is a good indicator of performance when using their distribution method, i.e. two different distributions with the same EMD result in very similar performance. This raises the question, would the same result have been obtained if different distributing methods had been used?

3.2. Calculating EMD

Many papers use different methods to distribute a non-iid dataset. It would be convenient to calculate the value for EMD to allow for some degree of comparison. The most prominent distribution methods are *limit-labels-sampler*, where clients are only allowed a fixed number of label-types, i.e. classes. The Dirichlet distribution (*dirichlet-sampler*) is also used to generate a non-iid dataset. And finally, a probabilistic *q-sampler* is used by at least one paper.

3.2.1. Limiting the number of classes per client: limit-label-sampler

[4], [21], [25], [18], [17], [38] [1] give each client examples of a maximum of t out of the M available classes. This can give an extreme skew to the data, especially with low values for t.

[10] and [13] extend on this by defining a second parameter that controls the fraction of the dataset that is partitioned per label. Let f , $0 \leq f \leq 1$ be that fraction. This distribution method will be referred to as *limit-labels-f* or $ll(t,f)$ or simply $ll(t)$ if $f = 1$.

[1] [18] and [17] state that the number of samples per client follows 'a power law' but give no further indication how this implemented or what settings were used

DEFINITIONS AND ASSUMPTIONS

Some definitions and assumptions need to be made for the value for EMD can be calculated. First, the global distribution is assumed to be uniform, i.e.:

$$p(Y=y) = \frac{1}{M} \quad \forall\ y \in \{1..M\}$$

Secondly, the following sets need to be defined. Each client k gets assigned a set of labels that are given to it with priority, let Ω_k be this set. Let $\mathcal{K}$ bet the set of clients, and $\mathcal{Y}$ be the set of labels.

each set Ω_y is subject to several rules.

$$\left|\{k | k \in \mathcal{K}, y \in \Omega_k\}\right| = \frac{tK}{M} \qquad \forall y \in \mathcal{Y} \tag{3.4a}$$

$$\left|\Omega_k\right| = t \qquad \forall k \in \mathcal{K} \tag{3.4b}$$

Equation 3.4a dictates that each label is given to a fixed number of clients, equal for all labels. Since the size of a set must be an integer number, $\frac{nK}{M}$ must be an integer as well. This means that the number of clients K must be an integer multiple of the number of classes M. The second equation dictates that each client is given exactly t labels with priority. This last constraint means that all clients will have the same EMD. Since a weighted average of the same value will always result in the original value, the number of samples per client does not affect EMD.

CALCULATING EMD

A client gets an iid fraction $\frac{f-1}{K}$ of the available samples for each label. The remaining fraction is given to a limited number of clients $\frac{tK}{M}$ (size of Ω_k). Thus if a label y appears in the set Ω_k the client gets an additional fraction $\frac{fM}{tK}$. Note that, since we care about the relative proportions, K can be dropped from the iid and non-iid fractions.

$$p(Y=y | y \in \Omega_k) = \frac{\frac{fM}{t} + 1 - f}{t\frac{fM}{t} + M(1-f)} = \frac{f}{t} + \frac{1-f}{M} \tag{3.5a}$$

$$p(Y=y) | y \notin \Omega_k) = \frac{1-f}{t\frac{fM}{t} + M(1-f)} = \frac{1-f}{M} \tag{3.5b}$$

The next step is to calculate the value for EMD, as shown in equation 3.6a. Since all clients hold the same value for EMD, the sum over clients can be dropped. This simplifies to 3.6b, and simplifies even further to 3.6c if $f = 1$, when all data is partitioned in labels.

$$EMD = t\left|p(Y=y|y \in \Omega_k) - \frac{1}{M}\right| + (M-t)\left|p(Y=y|y \notin \Omega_k) - \frac{1}{M}\right| \tag{3.6a}$$

$$EMD = 2f - \frac{2tf}{M} \tag{3.6b}$$

$$EMD \overset{f=1}{=} 2 - \frac{2t}{M} \tag{3.6c}$$

Interestingly, the number of clients does not affect EMD. Although the label limit and the number of clients affect EMD, this cancels when the ratio between the two is kept constant. Limiting to 20 labels out of a hundred results in the same EMD as when limited to two out of ten.

3.2.2. Probabilistic data distribution: q-sampler

[3] defines skew by using a probability parameter q. The clients are divided into M groups, where M is the number of classes in the data set. A sample with label $y \in \{1..M\}$ is assigned to a member of group l with probability q, and to any other group with probability $\frac{1-q}{1-M}$. This distribution method will be referred to as *q-sampler*.

The EMD can be calculated to allow comparing. The global distribution is again assumed to be uniform. All clients are expected to have the following distribution (proof in section 3.2.5):

$$p_k(Y=y) = \begin{cases} q\,, & y = k \\ \frac{1-q}{m-1}\,, & y \neq k \end{cases} \tag{3.7}$$

Since all clients hold the same value for EMD, the sum over clients from equation 3.3 can be dropped. This leads to the following expected value for EMD.

$$EMD = \left|q - \frac{1}{M}\right| + (M-1)\left|\frac{1-q}{M-1} - \frac{1}{M}\right| \overset{q \geq \frac{1}{M}}{=} 2q - \frac{2}{M} \tag{3.8}$$

In this equation, M has the same influence as in equation 3.6b but is not opposed by a label limit, so EMD decreases with M.

3.2.3. Probabilistic data distribution: Dirichlet distribution

[16], and [32], use the Dirichlet distribution to generate non-iid data sets. More specifically: a vector is sampled from the Dirichlet distribution, $\mathbf{p}_y \sim Dir_K(\alpha)$, and a proportion of $p_{y,k}$ of the data with label y is assigned to partition k. The EMD of this distribution method is hard to calculate exactly. Thus this is done empirically, by taking the average EMD of several data sets generated using the Dirichlet method. For $M = 10$, $K = 10$, and $\alpha = 0.5$ (as defined in [16]), EMD becomes 0.86 with a standard deviation of 0.059. Figure 3.1 shows the result for EMD for a larger range of values. The the number of classes M and clients K asymptotically impact the value for EMD, stabilizing at $M = 20$ and $K = 15$.

The Dirichlet distribution method will be referred to *dirichlet-sampler* or $Dir_K(\alpha)$.

The Dirichlet distribution method introduces data-quantity skew as well, i.e. the clients are not expected to hold the same amount of data. This is especially the case for lower values of α (higher skew). Figure 3.2 shows boxplots for partition sizes, for different values for α and different number of clients. A normalized partition size of one corresponds to the quantity of data a client would get if data is uniformly distributed. Or in other words, the partition size of a single client would get if everybody is given the same amount of data. The data shows that choosing a smaller value for alpha not only increases EMD but also increases the data-quantity skew. The number of clients has little effect on the degree of quantity skew, but outliers appear further with a larger number of clients.

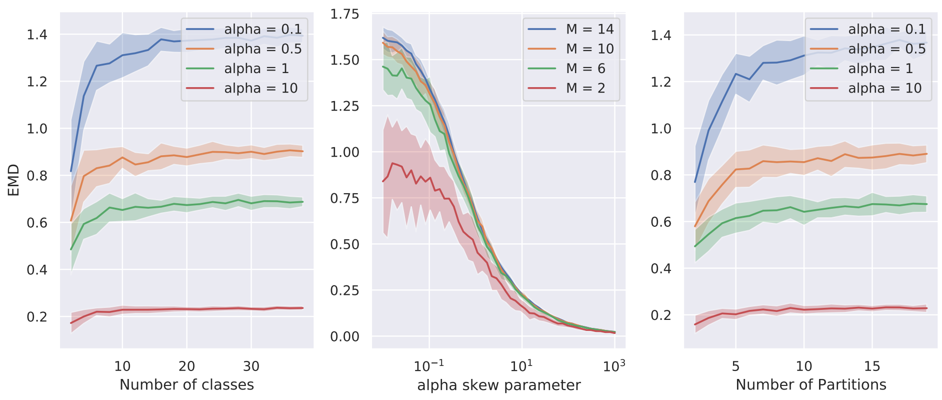


Figure 3.1: EMD of the Dirichlet distribution method for different values of α, M and K

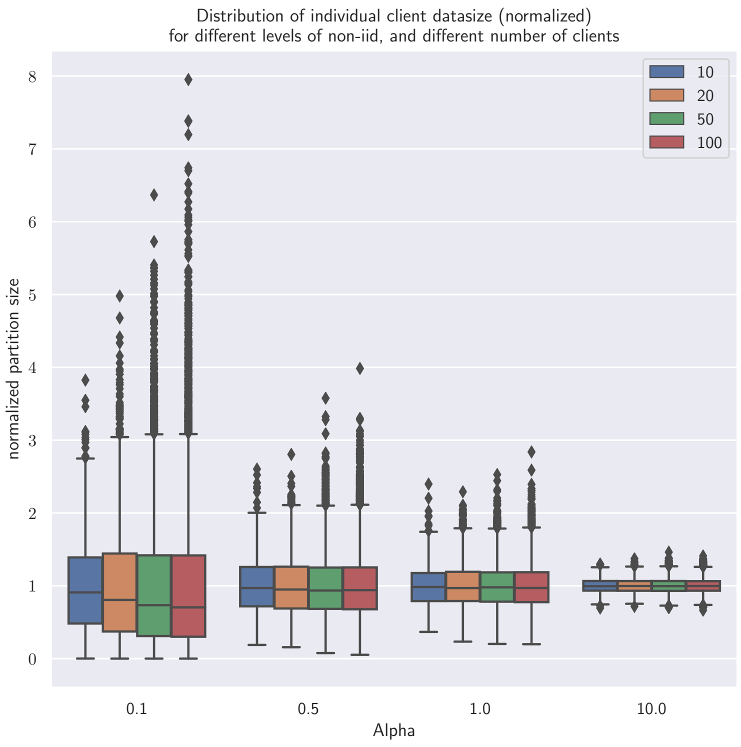


Figure 3.2: Boxplots for individual clients' partition sizes, when using the Dirichlet distribution method

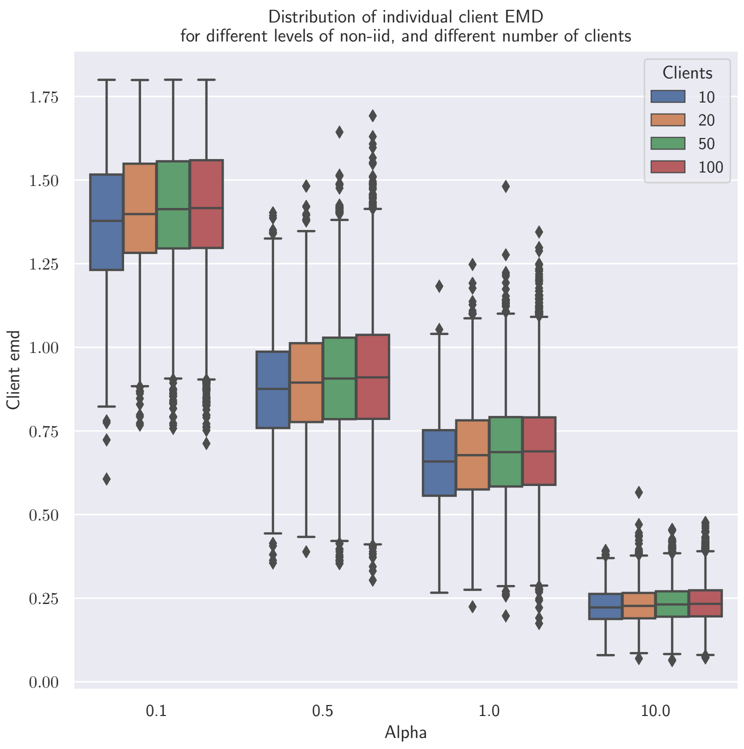


Figure 3.3: Boxplots for individual clients' EMDs, when using the Dirichlet distribution method

Unlike previous distribution methods, Dirichlet does not guarantee that clients have the same level of non-iidness, or the same value for EMD, as can be seen in figure 3.3. Clients can have plus or minus 0.5 EMD from the average, for lower values of alpha ($\alpha = 0.1, 0.5, 1$). Even at an average EMD of lower than 0.25, the spread is about half an EMD. Again, the number of clients has little effect on the spread of the range of values. Finally, further empirical testing has shown that there is little correlation between individual clients' data sizes and EMDs.

Dirichlet can introduce label-skew, quantity skew and also a spread in how skewed individual clients are.

3.2.4. EMD BASED PARTITIONING

An alternative to the methods described above is proposed by [40]. Previous methods went about generating a distribution using some arbitrary scheme, and discovering what the resulting EMD is afterwards. The other way around is possible as well, where a vector $\boldsymbol{p}$ with the probabilities of drawing a sample of each label is generated, given some EMD. This problem is defined formally in eq. 3.9, however, the implementation of the solver is omitted from the paper. When this vector $\boldsymbol{p}$ is obtained, it can be used to generate a distribution for all clients, by circular-shifting the vector one position for each client.

This ensures that all data is distributed, with each client holding unique samples.

$$\begin{array}{lll} \text{solve} & \boldsymbol{p} & = & [p_1, p_2, \ldots, p_M] \\ \text{s.t.} & \sum_{y=1}^{M} |p_y - \frac{1}{M}| & = & EMD \\ & \sum_{y=1}^{M} p_y & = & 1 \end{array} \tag{3.9}$$

The proposed method of EMD based partitioning lacks an implementation. However, the method should be included in the empirical analysis. A simple solver is therefore proposed that can solve equation 3.9 to within a certain tolerance. The basic idea is to apply Robinhood; if the EMD is too large, take samples from a class that has many, and give some to a class that has less, and anti-Robinhood; if the EMD is too small, take from the poor and give to the rich. This idea is defined more rigorously in algorithm 1.

In theory, this distribution method can produce all possible distributions that result in a certain value for EMD. However, it is likely that the solver has a bias towards a certain type of distribution and can thus not be viewed as a replacement for the other methods.

Algorithm 1 Generate a vector of probabilities that has a certain EMD

```
 1: procedure EMD($\boldsymbol{D}$)
 2:     return emd of distribution $\boldsymbol{D}$
 3:
 4: procedure ROBINHOOD-ANTI-ROBINHOOD($target\_emd$)
 5:     $\boldsymbol{D} \leftarrow$ random vector with length $M$ that sums to one
 6:     $emd \leftarrow EMD(\boldsymbol{D})$
 7:     while $|target\_emd - emd| > tol$ do
 8:         $i, j \leftarrow$ two random, unequal indices from $\{1..M\}$
 9:         if $D_i < D_j$ then
10:             $i, j \leftarrow j, i$
11:         if $emd < target\_emd$ then        ▷ emd too small, take from poor give to rich
12:             $\Delta \leftarrow Random(0, D_j)$
13:             $D_i \leftarrow D_i + \Delta$
14:             $D_j \leftarrow D_j - \Delta$
15:         else                               ▷ emd too big, take from rich and give to poor
16:             $\Delta \leftarrow Random(0, (D_i - D_j)/2)$
17:             $D_i \leftarrow D_i - \Delta$
18:             $D_j \leftarrow D_j + \Delta$
19:         $emd \leftarrow EMD(\boldsymbol{D})$
20:     return $\boldsymbol{D}$
```

3.2.5. PROPOSAL: COMBINED LIMIT-LABELS AND Q-SAMPLER

The fourth method of data distribution is a mix of the limit-label and q-sampler. This method will be referred to as limit-labels-q, or $ll\_q(t, q)$. The same assumptions and definitions as in 3.2.1 are applied here.

When distributing the data among clients, a sample with label y is given to any client k that satisfies $y \in \Omega_k$ with probability q. Since there are $\frac{tK}{M}$ clients that satisfy this rule, the probability of giving the sample to a specific client is $\frac{qM}{tK}$. Otherwise, the sample is given to any of the remaining clients, each having probability $\frac{1-q}{K-\frac{tK}{M}}$ of getting it.

Let $Gets(k, y)$ represent the event of a k client getting a sample with label y. The probabilities of $Gets(k, y)$ can now be formally defined.

$$p(Gets(k,y)|y \in \Omega_k) = \frac{qM}{tK} \tag{3.10a}$$

$$p(Gets(k,y)|y \notin \Omega_k) = \frac{1-q}{K - \frac{tK}{M}} \tag{3.10b}$$

For calculating EMD $p_k(Y = y)$, e.g. the probability of random sample of client k having label y, is needed. Thus equation 3.10 needs to be normalized.

$$p_k(Y = y|y \in \Omega_k) = \frac{p(Gets(k,y)|y \in \Omega_k)}{t \cdot p(Gets(k,y)|y \in \Omega_k) + (M-t) \cdot p(Gets(k,y)|y \notin \Omega_k)} \tag{3.11a}$$

$$p_k(Y = y|y \notin \Omega_k) = \frac{p(Gets(k,y)|y \notin \Omega_k)}{t \cdot p(Gets(k,y)|y \in \Omega_k) + (M-t) \cdot p(Gets(k,y)|y \notin \Omega_k)} \tag{3.11b}$$

If the equations 3.11 are filled in and simplified we arrive at the following probabilities, and the subsequent value for EMD.

$$p_k(Y = y) = \begin{cases} \frac{q}{t}, & y \in \Omega_k \\ \frac{1-q}{M-t}, & y \notin \Omega_k \end{cases} \tag{3.12}$$

$$EMD = t \cdot \left| \frac{q}{t} - \frac{1}{M} \right| + (M-t) \left| \frac{1-q}{M-t} - \frac{1}{M} \right| \overset{q \geq \frac{t}{M}}{=} 2q - \frac{2t}{M} \tag{3.13}$$

3.3. Translating EMD to existing methods

All distribution methods discussed in the previous section allow for the calculation of EMD given their settings. However, when generating distributions using any of these methods, it is more convenient to calculate their settings given some value for EMD. This would be relatively straightforward if the number of given variables (i.e. EMD) equals the number of free variables (e.g. t and f for *limit-label-sampler*), which is not necessarily the case. The techniques derived here will be used in the next section to perform an empirical analysis on the ability of EMD to predict the relative performance of federated learning on non-iid datasets.

Calculating the settings for the limit-label-sampler can be done by solving equation 3.6b for f. All possible solutions be found trying all possible values for t and checking if f is between 0 and 1.

The previous technique can not be applied to the Dirichlet sampler. Instead, a solver is needed that finds the right value for α such that the EMD of the resulting distribution is close to the requested EMD. A simple solver that can do this is the bisection method.

The same technique as with the limit-label-sampler-f can be applied here. By solving equation 3.13 for q and requiring that $t \in \{1..M-1\}$ and $\frac{t}{M} \leq q \leq 1$, all solutions can be found.

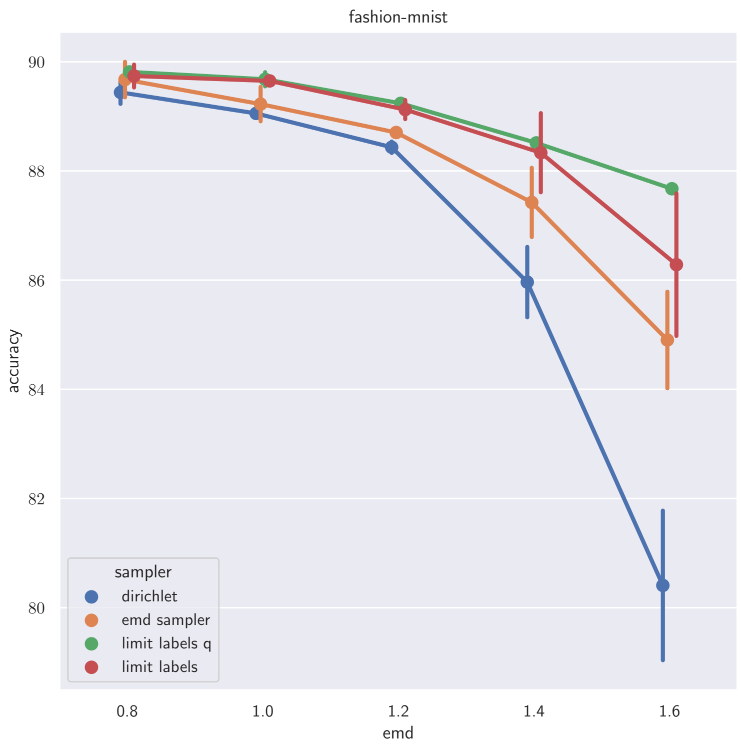


Figure 3.4: Accuracy on the fashion-mnist dataset for different distributions methods and different values for EMD

3.4. EMPIRICAL ANALYSIS

The EMD of existing distribution methods has been derived to see if EMD can allow for comparing the result of different papers. To show this is the case, the following experiment has been set up.

1. Several test values for EMD are defined: $[0.8, 1.0, 1.2, 1.4, 1.6]$.

2. Each solver for the different distributions methods is tasked with finding a distribution that has an EMD close to these values.

3. Number of local epochs(E) is set to 8, the number of rounds to 12.

4. All other experiment settings and implementation details are listed in section 4.4.

It is expected that the different distribution methods result in similar performance given the same EMD. Results for both the CIFAR10 and the Fashion-MNIST dataset are shown in figures 3.5 and 3.4.

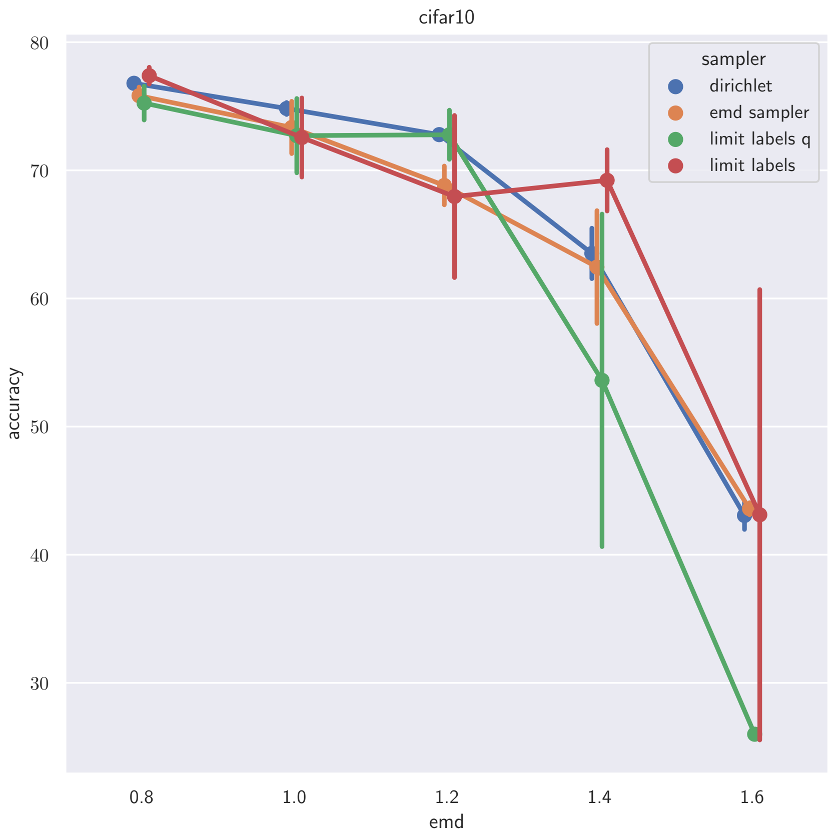


Figure 3.5: Accuracy on the cifar10 dataset for different distributions methods and different values for EMD

3.4.1. Results: fashion-mnist

Results for the experiment with the fashion-mnist dataset are shown in figure 3.4. All samplers follow the same trend and cause a drop in performance at the same EMD value of 1.2. For higher values, however, they drop off at different rates. The limit-labels-q is very consistent and appears to generate the easiest distribution. The limit-label sampler follows the same trend initially but is less consistent at higher values for EMD. The two are expected to perform similar since they both use the limit-label-sampler. However, the number of repetitions is only three, in the same range as the number of settings that result in an EMD of 1.6.

Dirichlet generates the hardest distribution to learn from. This is expected since in addition to label-skew it also results in quantity skew.

3.4.2. Results: cifar10

The results for cifar10 are visible in figure 3.5, and appear to be more consistent. All four samplers follow the same trend, with the drop-off starting at EMD-1.0. Both limit-label samplers show a large spread. Take the limit-label-sampler, which can result in an EMD of 1.6 with $ll(1, 0.89)$ and $ll(2, 1.0)$. The spread indicates that the two solutions for EMD-1.6 result in different performances. $ll(2, 1.0)$ results in all having only samples from two classes, whereas $ll(1, 0.89)$ results in 11% of data being distributed uniformly. $ll(2, 1.0)$ is likely the harder of the two, weighing even more heavily than the quantity-skew caused by the Dirichlet sampler.

3.4.3. DISCUSSION

The results from the previous two sections indicate that EMD is a good indicator of performance, but fails to capture the extreme skew of a pure limit label sampler (i.e $f = 1.0$ or $q = 1.0$). Showing the results for the different solutions to the same EMD value separately would have painted a clearer picture.

It can also be concluded that EMD values below 1.0 provide an insufficient level of non-iid-ness to draw general conclusions from. It does not seem that EMD captures all information that leads to performance degradation when faced with skewed data. Which distribution method and which dataset also have an impact.

The result of EMD causing variation in accuracy is not a contradiction of the findings of [40]. The authors found that it is a great indicator for performance. However, only a single distribution method was used. Our results show that, within a single distribution method, EMD is s great indicator as well. But it has no use in this case, as the existing hyper-parameters already are a great indicator of performance. EMD was derived from the bound on weight-divergence, it determines the bound, but a bound is not a direct performance indicator.

3.5. EMD OF RELATED WORK

With a tool to compare existing distribution methods, the difficulty of different distribution methods can be compared. Results are collected in table 3.1. What stands out from most results is the minimal improvement over `FedAvg`. Large improvements are only reported by the data-sharing method and `STC`. The limit-label sampler is used most often, but only some papers extend this sampler to a fraction of skewed and uniform data. The results from the previous section indicate that non-iid starts to affect performance at EMDs greater than 1. When using the limit-label-sampler, the drop-off for cifar10 only starts to appear at EMD-1.4. A few papers, `skewscout` and `FLTrust`, operate below this level of EMD-1.4. Finally, `Astraea` does not provide the method with which imbalance was created, so their result of 5.6% is without context and does not allow for comparing.

3.6. CONCLUSION

The limit-label(-q) and the EMD-sampler all result in distributions where each client is expected to have the same number of samples and the same value for EMD. Furthermore, the global distribution is always assumed to be uniform.

Realistic datasets are not guaranteed to adhere to any of these assumptions. Clients could have different values for EMD (global-label-skew), and a different number of samples (quantity-heterogeneity). The Dirichlet distribution method does not result in clients having the same number of samples, not is likely to keep clients at the same value for

EMD, this makes the Dirichlet distribution method the most realistic. In a realistic dataset, the global dataset is likely not uniform either. These types of data heterogeneity are outside the scope of this research. Quantity heterogeneity has been thoroughly researched in several papers, e.g. [32] [17], and non-uniform global distributions have been touched upon by at least one paper [6]

These problems are not captured by measuring EMD and suggest that further research should be done to classify the difficulty level of types of challenges.

Finally, the proposed method of measuring skew (eq. 3.3) takes the Federated-Average (i.e. weighted average by the number of samples) of the clients' EMDs. However, some papers propose different aggregation methods. The equation for calculating EMD 3.3 is derived from the bound on weight-divergence, which in turn is derived from `FedAvg`. Other aggregation methods will likely result in a different value for EMD, and possibly better capture the problems that non-iid data causes.

EMD is not the ideal unit for measuring the skew in a distribution. Different distributions, but with the same EMD, can result in different performances. However, EMD does allow for a ballpark estimate and allows for the comparison of different distribution methods. If another distribution method is introduced, its EMD can be calculated to get a sense of difficulty compared to existing methods.

Table 3.1 is shown on the next page

Table 3.1: EMD of related work

Paper	Solution	Sampler	EMD	Improvement over FedAvg			Dataset
[3]	FTtrust	$q(0.5)$	0.8	NA			MNIST FMNIST CIFAR10
[4]	TiFL	$ll(2)$	1.6	worse [1]			**MNIST**
		$ll(5)$	1.0				FMNIST
[10]	SkewScout[2]	$ll(2,0.2)$	0.3	NA			CIFAR10
		$ll(2,0.6)$	1.0				
		$ll(2,0.8)$	1.3				
[13]	Scaffold	$ll(2,1.0)$	1.9	1.4%			EMNIST
		$ll(2,0.9)$	1.7	1.4%			(M=62)
[16]				FedProx	Scaffold	FedNova	
		$Dir_{10}(0.5)$	0.9	-0.3%	1.6%	-1.4%	MNIST
		$ll(1)$	1.8	2.3%	0%	0%	FMNIST
		$ll(2)$	1.6	0.9%	-0.7%	-3.3%	**CIFAR10**
		$ll(3)$	1.2	-1.2%	-0.5%	-3.9%	
[17]	FedProx [3]	$ll(2)$	1.6	0.0%			MNIST
[21]	FedCS	$ll(2)$	1.6	NA			MNIST FMNIST
[25]	STC	$ll(1)$	1.8	64%			CIFAR10
		$ll(2)$	1.6	31%			
		$ll(3)$	1.2	14%			
[32]	FedNova	$Dir_{16}(0.1)$	1.4	5.63%			CIFAR10
[40]	Data-sharing	$ll(1)$	1.8	25%			MNIST
		$ll(2)$	1.6	NA			**CIFAR10**
[6]	Astraea	NA	NA	5.59%			EMNIST
[38]	FedMix	$ll(2)$	1.6	7.4%			**CIFAR10**
		$ll(20)$	1.6				CIFAR100

If ambiguous, improvements over FedAvg are provided for the dataset in **bold**.

[1] Abstract claims TiFL performs better in heterogeneous conditions, but care full analysis of the provided graphs shows this is not the case for label-heterogeneity (i.e. non-iid label skew)

[2] Skewscout is aimed decriminalized learning, where data can be shared freely. Comparison to FedAvg is therefore impossible.

[3] Number of samples is non-uniform as well, but unclear how. FedProx does improve when faced with stragglers, even in non-iid settings, but shows no improvement without stragglers

4

COMMUNICATION EFFICIENCY

Since federated learning is distributed, communication is inherently necessary. This of course comes at a cost, and ideally, communication is kept at a minimum. The most simple parameter that can reduce the amount of communication is the number of local epochs E. Increasing E results in more local learning per round, and if the total amount of local learning is kept the same, it results in less communication. In practice, increasing E hurts convergence, and thus more rounds are necessary to reach the same accuracy.

This chapter looks into a communication scheme that reduces both the total amount of data transfer and the instantaneous communication load, without sacrificing on accuracy or convergence.

4.1. PROBLEM OF WEIGHT DIVERGENCE

Recently, a number of papers [32], [13], [40] have theorized that weight-divergence is at the core of the problems caused by non-iid data. The authors state that, due to the differences in the local data distributions, the local models diverge from the global optimal model at different rates, causing the average of the local models to diverge as well. Figure 4.1 shows a 2D representation of the problem.

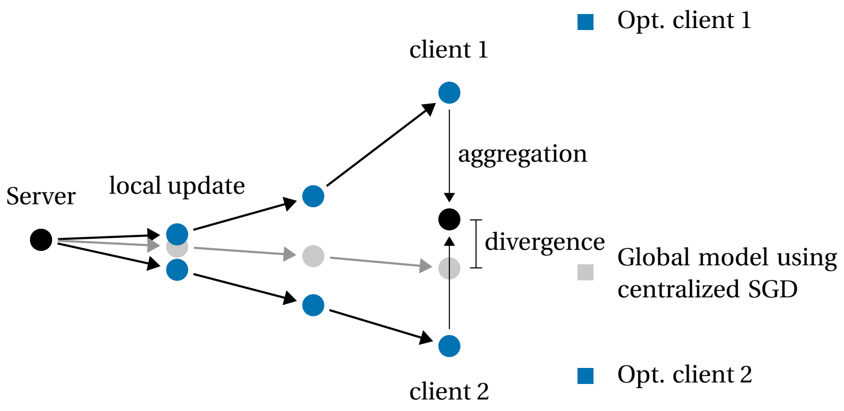


Figure 4.1: Weight divergence intuition

Zhao et al. [40] went a step further and derived a bound on the weight divergence. This is the same bound where the value of EMD appears. The main take-away that is useful for this chapter is the dependence on the number of local steps T, which in turn is directly proportional to the number of local epochs E:

$$\begin{aligned}||\boldsymbol{w}_{rT}^{(f)} - \boldsymbol{w}_{rT}^{(c)}|| \leq & \sum_{k=1}^{K} \frac{n^{(k)}}{\sum_{k=1}^{K} n^{(k)}} (a^{(k)})^{T} ||\boldsymbol{w}_{(m-1)T}^{(f)} - \boldsymbol{w}_{(m-1)T}^{(c)}|| \\ & + \eta \sum_{k=1}^{K} \frac{n^{(k)}}{\sum_{k=1}^{K} n^{(k)}} \sum_{i=1}^{M} ||p^{(k)}(y=i) - p(y=i)|| \sum_{j=1}^{T-1} (a^{(k)})^{j} g_{max}(\boldsymbol{w}_{mT-1-j}^{(c)})\end{aligned} \tag{4.1}$$

Table 4.1: Variables used in eq. 4.1

r	round number
T	Number of local SGD steps
M	Number of classes
K	Number of clients
$\boldsymbol{w}_{rT}^{(f)}$	Weights as result of Federated learning at round r
$\boldsymbol{w}_{rT}^{(c)}$	Weights as result of centralized learning after rT steps
$n^{(k)}$	size of the data set of client k
η	learning rate
$p^{(k)}$	distribution of data at client k
p	distribution of global data
$a^{(k)}$	$1 + \eta \sum_{i=1}^{M} p^{(k)}(y=i)\lambda_{\boldsymbol{x}\|y=i}$
f_i	probability for class i
$g_{max}(\boldsymbol{w})$	$max_{i=1}^{M} \|\|\nabla_{\boldsymbol{w}} \mathscr{E}_{\boldsymbol{x}\|y=i} \log f_i(\boldsymbol{x}, \boldsymbol{w})\|\|$

As equation 4.1 shows, the weight-divergence is bounded from above by a function

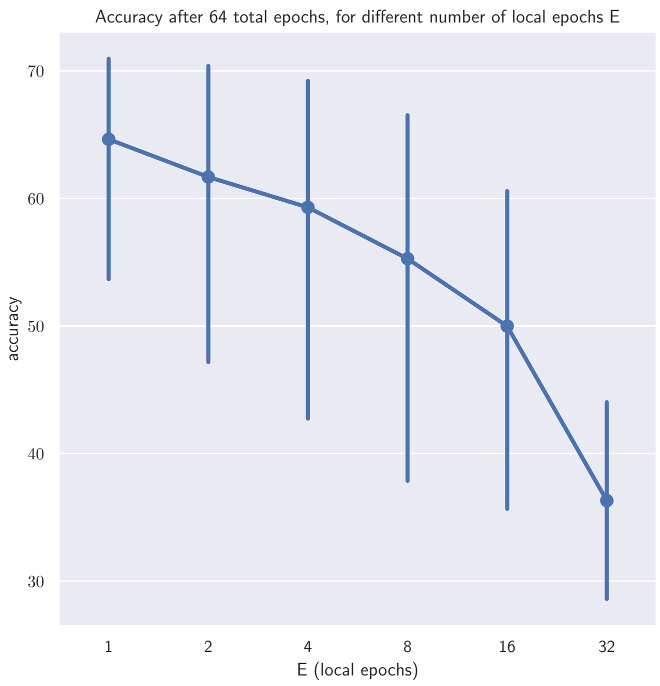


Figure 4.2: Impact of E on the accuracy after a fixed number of total epochs (cifar10)

that exponentially increases with T and thus with E. Thus, it is expected that increasing E harms performance, especially if the EMD of the distribution is high.

To show that this is indeed the case the following test setup has been designed. The cifar10 dataset is distributed among 20 clients using the limit-label distribution method, with EMD set to 1.0. All clients participate in every round, and perform 1 local epoch per round (E), for 64 rounds. The experiment is repeated, but each time E is doubled, and the number of rounds is halved, to keep the total amount of computation identical. All other experiment details can be found in section 4.4. The results are shown in figure 4.2. Here it is visible that increasing E hurts the convergence rate per locally computed epoch. This is consistent with the convergence rate found in existing works [13], [16].

With this conclusion, it seems that increasing E is an ineffective solution to reducing communication. However, through an elaborate scheme, E can be kept the same, while decreasing the amount of communication.

4.2. PHASE-SHIFTED FEDERATED LEARNING

In ‘regular ’ FL every model download from the server to the clients has a corresponding model upload. A possible reduction in communication can be found here: reducing the number of model uploads per download. The most simple version of this idea is displayed in figure 4.3, where every upload can be paired with two downloads. The first download $\boldsymbol{w}_t$ serves the same purpose as in regular FL, as a common initialization of the local models. However, the second download $\boldsymbol{w}_{t+1}$ serves as a drift correction for the local model. This can be done by simply taking a weighted average, and should decrease weight divergence.

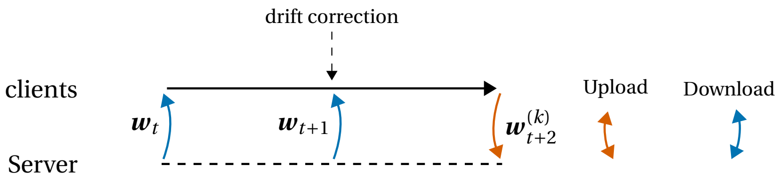


Figure 4.3: Step 1: decreasing the number of uploads per download

Of course, $\boldsymbol{w}_{t+1}$ needs to be an update over the initial $\boldsymbol{w}_t$, and has to come from somewhere. This is where the second part of the proposed idea comes in. The clients can be split into two groups and operate out of phase with each other. The result, or aggregation of the first group, can be used for drift correction of the second group. This idea has been visualized in figure 4.4. Federated Average is used to calculate the average of the local model and the drift-correction-model, where the drift-correction-model is weighted by the number of samples that were used to generate it, and the local model by the number of samples the client has.

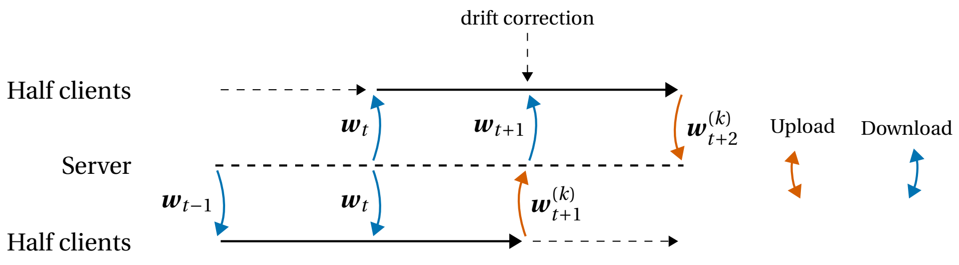


Figure 4.4: Step 2: splitting the clients and operating out of phase

The final scheme allows phase-shift to be compared to regular federated learning. Both are displayed in figure 4.5, where the number of local learning iterations per round is kept the same between the two versions. With this figure, the reduction in communication can be calculated. In the time window displayed in figure 4.5 `FedAvg` performs 4 uploads from clients to the server and 4 downloads from the server to the clients. Assuming both uploads and downloads require the same amount of data, `FedAvg` performs 8 ‘communications’ in total. In the same time window, phase-shift performs 8 downloads, each to only half of the clients. So effectively the same data is transferred as with regular `FedAvg`, and these can be counted as 4 ‘communications’. The same logic can be applied to the 4 uploads, which results in 2 additional 'communications'. Putting the up-and-downloads together results in 6 'communications', two less than `FedAvg`, a 25% reduction. The assumption that up and downstream communication is the same size holds for `FedAvg` [25].

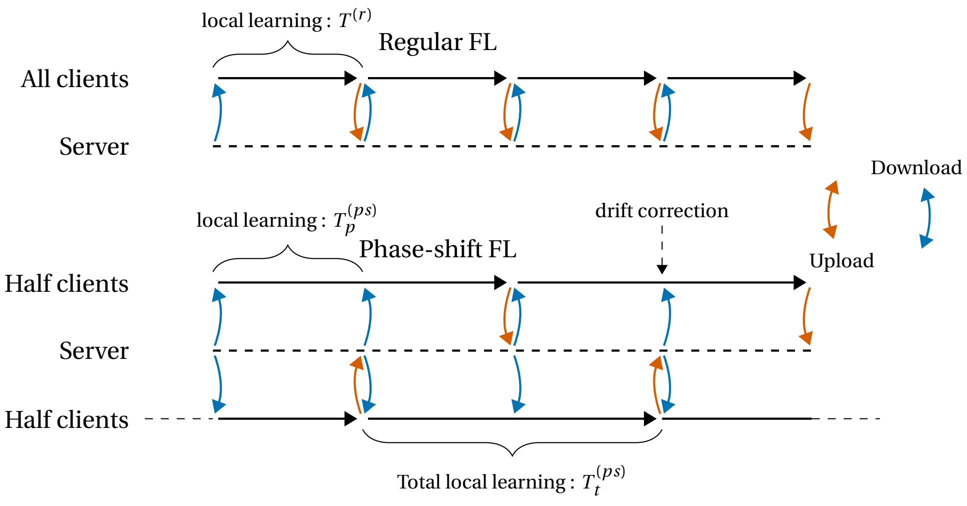


Figure 4.5: two-phase phase shift, maximum communication saving with $T^{(r)} = T^{(ps)}$

4.2.1. MODES OF OPERATION

Phase-shift can operate on a continuum of two extremes. First is maximum communication savings, where the algorithm is configured to use the same epoch settings as the baseline `FedAvg`. Due to the apparent number of local epochs being the same, performance is expected to only take a minimal hit. With two phases, 25% communication can be saved. This configuration is shown in figure 4.5.

In communication reduction mode, i.e. keeping $T^{(ps)}$ equal to $T^{(r)}$, the communication reduction follows equation 4.2, where n is the number of phases. Thus, in the case of two-phase federated learning, communication can be reduced by 25% compared to the baseline federated learning, while effectively keeping E the same.

$$\frac{Comm^{(ps)}}{Comm^{(r)}} = \frac{n+1}{2n} \tag{4.2}$$

The other extreme is where, instead of communication being saved, communication is constrained to the same amount as `FedAvg`. At this end, the performance of phase-shift is expected to be better than `FedAvg` due to more frequent communication. To achieve this, the scheme from figure 4.5 can be scaled in time such that the amount of communication per unit of time is equal to regular FL. This is displayed in figure 4.6. Notice that, although the total time of local learning is increased, the model is corrected halfway. This results in a smaller number of local updates without drift correction and thus should result in better performance when faced with non-iid data.

In general, the clients can be split in n parallel streams, with $n = 2$ corresponding to figure 4.6. The number of local learning steps for phase-shift, while keeping the amount of communication per unit of time equal to the baseline FL, follows eq. 4.3. The increase in total local learning follows eq. 4.4. Thus, for two phases, the number of local epochs can be reduced by 25%.

$$T^{(ps)}(n) = \frac{n+1}{2n} \cdot T^{(r)} \tag{4.3}$$

$$T_t^{(ps)}(n) = n \cdot T_p^{(ps)}(n) = \frac{n+1}{2} \cdot T^{(r)} \tag{4.4}$$

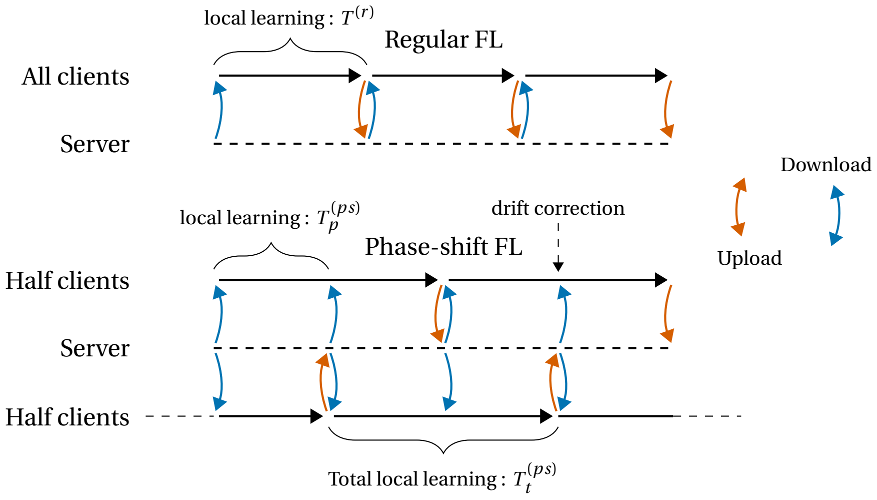


Figure 4.6: Two-phase phase-shift when communication is constrained to the same value as regular `FedAvg`

4.3. IMPLEMENTATION

The phase-shift communication protocol has been implemented in FLTK, using python and PyTorch. It has been implemented such that a hyper-parameter n determines the number of phases. To achieve this, a queue Q of length n is used to keep track of which set of clients is in which phase. The queue can be seen as a list where adding an item results in shifting all elements to an incremented index.

Algorithm 2 pseudo code the **server** side, for n-phase phase shift, based on FedAvg

```
1:  Server
2:      initialize $w_1$, $\mathscr{K}$, $m$, $n$ ▷ model, set of clients, clients per round, number of phases
3:      $Q \leftarrow$ queue of length $n$, initialized to $n$ sets of $m$ clients each, indexable by $1 \dots n$
4:      $N \leftarrow 0$
5:      for round $t \in \{1,2,\dots\}$ do
6:          $\mathscr{K} \leftarrow \mathscr{K} \cup Q.pop()$                ▷ Make finished clients available again
7:          $Q.add$(random set of $m$ clients from $\mathscr{K}$)
8:          $\mathscr{K} \leftarrow \mathscr{K} \setminus Q[1]$
9:
10:         ParFor client $k$ in $Q$                         ▷ parallel loop over all clients in $Q$
11:             if $k \in Q[n]$ then
12:                 $\boldsymbol{w}_{t+1}^{(k)}, n^{(k)} \leftarrow$ CLIENTRETURN($k$, $w_t$, $N$)
13:             else if $k \in Q[0]$ then
14:                 CLIENTUPDATE($k$, $w_t$)
15:             else
16:                 CLIENTCORRECTION($k$, $w_t$, $N$)
17:         $N \leftarrow \sum_{k \in Q[n]}$
18:         $w_{t+1} \leftarrow \sum_{k \in Q[n]} \frac{n^{(k)}}{N} \boldsymbol{w}_{t+1}^{(k)}$                    ▷ FedAvg aggregation
19: End
```

The client has three functions it uses, depending on which phase it is in. *clientUpdate* simply acts as a normal training round, same as in FedAvg, but it does not return the local model. *clientCorrection* implements the drift correction by taking the FedAvg of the local model and the new global model, after which it proceeds the same as *clientUpdate*. *clientReturn* does exactly the same as *clientCorrection* but returns the local model to the server. Local models are persistent between rounds.

In the last round the model from all clients is collected, instead of only from the clients in the final phase. This increases communication slightly from equation 4.2, but is negligible if the number of rounds is large enough. It can be seen as, if FL runs for n rounds, communication is reduced for $n-1$ rounds, and runs at full communication for the final round. This has been omitted from the pseudocode, to keep it more concise.

Finally, if one wants to extend the algorithm to include regular FedAvg, a special case in *clientCorrection* should be added if n equals 1, and only use the global model.

The pseudo code can be seen in listings 2 and 3.

Algorithm 3 pseudo code the **client** side, for n-phase phase shift, based on FedAvg

```
1:  Client
2:      $w \leftarrow$ empty model
3:      $n \leftarrow$ number of samples in client $k$
4:      function CLIENTUPDATE($k$, $w'$)
5:          $w \leftarrow w'$
6:          TRAIN(k)
7:      function CLIENTCORRECTION($k$, $w'$, $N$)
8:          $w \leftarrow \frac{N}{n+N} w' + \frac{n}{n+N} w$                ▷ Drift correction
9:          TRAIN(k)
10:     function CLIENTRETURN($k$, $w'$, $N$)
11:         CLIENTCORRECTION($k$, $w'$, $N$)
12:         Return $w$, $n$
13:     function TRAIN(k)
14:         arbitrary learning function that saves an updated model in $w$
15: End
```

4.4. STANDARD SETUP

Most experiments in this work use the setup described in this section. All experiments are run in an altered version of FLTK [5], a python implementation for truly distributed federated learning using `pyTorch`. The meta-parameters, common to most experiments, are listed in table 4.2. Parameters such as the number of rounds and the number of epochs are specified per experiment.

Table 4.2: Meta parameters used in all experiments

Parameter	symbol	value
number of clients	K	20
number of classes	M	10
optimizer		SGD + momentum
learning rate	η	0.001
momentum		0.9
batch-size	B	16
aggregation function		FedAvg
client selection		full client participation
repetitions		4

All experiments are repeated four times, each time with the same four seeds. This allows possible solutions for dealing with non-iid data to be tested on identical distributions and thus should result in a fair comparison.

Most related works use the *limit-label-sampler* to distribute data among clients. This distribution method limits the number of classes each client is given. This can be extended to allow for only a fraction of data to be distributed in this way, to other fraction

Table 4.4: Google Cloud Platform virtual machine settings

Property	Value
Region	`europe-west4`
Machine Family	General Compute
Machine Series	N1
RAM	3.75 GB
Machine Type	`g1-small`
Boot Disk	Ubuntu 20.04
Boot Disk Type	Balanced Persistent Disk
Disk Size	10 GB
Firewall	`Allow HTTP traffic and Allow HTTPS traffic`
Networking External IP	Ephermeral

being distributed uniformly. Details on this sampler can be found in chapter 3, section 3.2.1. This sampler is used unless otherwise specified. Most experiments in this work are conducted at EMD-1.4, and three different limit-label-sampler settings can achieve this level of non-iid-ness. Using the method described in section 3.3, the settings listed in table 4.3 can be derived. The sampler is altered such that when using a pure limit-label-sampler (i.e. fraction non-iid 1.0), the clients are given at least 1 sample of each class. The four random seeds have been chosen such that each sampling setting appears at least once.

Table 4.3: limit-label-sampler settings that can achieve EMD-1.4

label-limit	fraction non-iid	resulting EMD
1	0.78	1.4
2	0.86	1.4
3	1.00	1.4

4.4.1. TEST ENVIRONMENT

All experiments are run using the Google Cloud Platform. Each client, and the single federator, is assigned a virtual machine with a single virtual-core (Intel Skylake CPU). All virtual machines use the settings given in table 4.4. No GPU is used.

4.4.2. NEURAL NETWORKS

Three datasets are used throughout this document, `CIFAR10`, `MNIST`, and `Fashion MNIST`. For each dataset, the commonly used *CNN* networks are used. Their partial python implementations are shown appendix A.

4.5. EMPIRICAL ANALYSIS

With an empirical analysis, two questions need to be answered.

1. *When using phase-shift to reduce communication, does performance remain unaffected compared to* `FedAvg`? Phase-shift has no added value if the performance takes a significant hit by reducing communication.

2. *If traditional techniques are used to reduce communication, does performance suffer?*. If the same communication reduction can be achieved by increasing the number of epochs per round, without decreasing performance, then phase-shift has no added value either.

However, if (1) phase-shift can maintain performance while reducing communication, and (2) the same reduction can not be achieved using `FedAvg`, it can be concluded that phase-shift has some added value.

4.5.1. EXPERIMENT SETUP

To allow for a larger number of phases the number of clients will be set to 40, the number of phases can then be set to whole fractions of 40, to keep the number of clients per phase identical. The communication reduction, and the corresponding number of phases to achieve this, can be seen in the first and second line of table 4.5. The epoch increase required to match the communication reduction when using the conventional method is shown on the third line. Note that a single-phase corresponds to regular `FedAvg`. The baseline number of epochs E has been set to 8 and 16. The experiments with phase-shift will all be performed with these values for E. However, in order for `fedAvg` to reach the same communication savings, E must be increased by the values listed in the third line of table 4.5, e.g. $E = 8, 10.7, 12.8, \dots$. The number of total epochs will be kept at 120 for $E = 8$ or 128 for $E = 16$, or the closest round value when using fractional values for E.

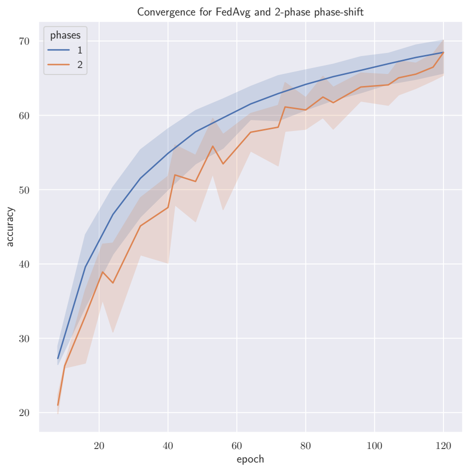
The idea of phase-shift is based on weight divergence, which only occurs if the distribution is sufficiently non-iid. Experiments from chapter 3 have shown that EMD-1.4 offers a significant challenge. Since the limit-label-sampler is used by most related works, the same sampler will be used here. All other details can be found in section 4.4.

4.5.2. TERMINOLOGY

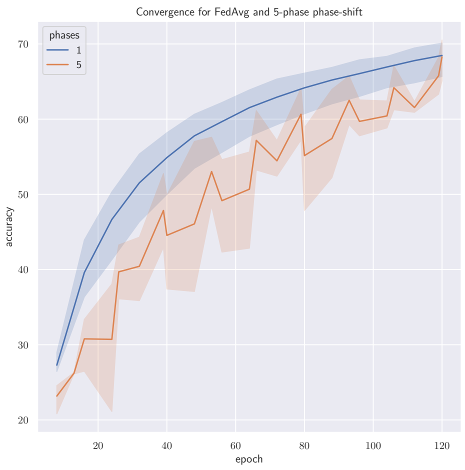
With the introduction of a new scheme, several new terms are required to describe the results.

1. **A phase of clients** is used to refer to a set of clients that operate in the same phase. For instance, in figure 4.5, each long black arrow represents a set of clients in the same phase. A client can only participate in a single phase at any moment in time, thus, any two phases that are active at the same time have no overlap.

2. **Distribution of a phase**. Each client in a phase of clients has a dataset. The distribution of a phase is used to describe the distribution of the combined dataset of a phase.

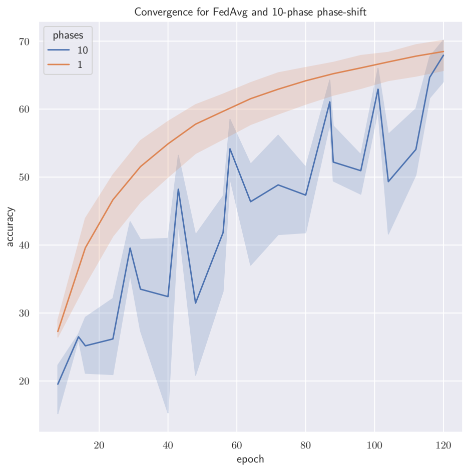


(a) Convergence of 2 phases compared to `FedAvg`

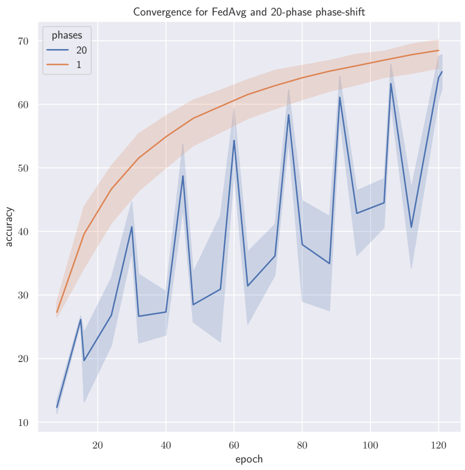


(b) Convergence of 5 phases compared to `FedAvg`

Figure 4.7: Phase-shift convergence on cifar10. `FedAvg` (single-phase) and phase-shift with 2 and 5 phases.



(a) Convergence of 10 phases compared to `FedAvg`



(b) Convergence of 20 phases compared to `FedAvg`

Figure 4.8: Phase-shift convergence on cifar10. `FedAvg` (single-phase) and phase-shift with 10 and 20 phases.

4.5.3. CONVERGENCE

The convergence of phase-shift has an interesting behavior. This can be seen in figures 4.7 and 4.8, where single-phase (`FedAvg`) is compared to phase-shift with 2, 4, 10 and 20 phases. Convergence for `FedAvg` is as expected, a smooth curve that asymptotically moves to a final accuracy. Phases-shift behaves very differently, initial convergence is jagged and far worse than `FedAvg`, but suddenly jumps to a higher accuracy in the final round. This is no coincidence but is due to how phase-shift operates.

The experiments are done with full client participation, so for both `FedAvg` and phase-shift, 40 clients are active every round. However, in `FedAvg` all 40 clients return their results for aggregation, whereas in phase-shift only $\frac{40}{n}$ clients return their results, where n is the number of phases. Thus, in the example with 10 phases, only 4 clients return their model for aggregation to the federator at the end of a round. Since the data is significantly non-iid, these 4 clients will have a non-uniform combined dataset, and the resulting model will have a preference for this partial dataset. The aggregation of this model does help as a drift correction, as the final result indicates, but can not represent the full central model as with traditional `FedAvg`. This means in phase-shift no real central model exists until, crucially, all clients return their model for aggregation in the final round. This explains the sudden convergence in the final round for phase-shift.

However, the information for a global model at every round must exist, but it is not necessary to return this information to the federator at every round, as can be concluded from the figures.

The 'jaggedness' of phase-shift increases with the number of phases, starting relatively smooth with two phases in figure 4.7a, and ending in unstable convergence with 20 phases, in figure 4.8b. Two contributing factors cause this behavior. First is the number of rounds that a phase of clients has before returning the model. For two phases, a phase of clients has two rounds to fit the model to their data, they get corrected after the first round, but not reset to a central model. This causes the local model to fit closer to the local data. This is even more pronounced if the number of phases is higher, and clients have more rounds to fit their model to the local data. This first effect can be concluded from the initial convergence being more stable, where phase-shift is not completely initialized yet. Even when set to 20 phases, the clients returning their model after the first round will have only had a single round to fit the model to their data, and thus this effect is less pronounced in the initial phases.

The second contributing factor is the number of clients in each phase. When configured to two phases, 20 clients return their model at the end of every round, whereas with 20 phases only two clients return their model. With two phases, the combined dataset of the clients in a single phase is more likely to be uniform, and distributions of the two phases are likely to be similar. This is contrasted by 20-phases where, due to the EMD-1.4, no two phases are likely to have a similar distribution. With each phase returning a model that is fitted to a very specific sub-dataset, a back-and-forth pulling effect takes place, with each phase trying to pull the global model to their dataset.

As can be seen in figures 4.7 and 4.8, the unstable convergence is corrected in the last round, where all phases return their models. This holds for 2, 5, and 10 phases, but the convergence for 20 phases is too unstable to be corrected in the final round.

Table 4.5: The number of phases needed to reach certain percentage of communication saving (second line) and the epoch increase if the same communication reduction is achieved through conventional methods (third line)

	Communication reduction (%) over baseline ($E = 8$ or $E = 16$)	0	25	37.5	40	45	47.5
Phase shift	Number of phases	1	2	4	5	10	20
FedAvg	epoch increase (%)	0	33	60	66	82	91
	Epoch with base E=8	8	10.7	12.8	13.3	14.5	15.2
	Epoch with base E=16	16	21.3	25.6	26.6	29.1	30.5

4.5.4. Results: $E = 8$

The experiment described in the previous section is first carried out at baseline $E = 8$. Phase-shift uses phases to reduce communication, and can thus keep the epochs per round at 8, whereas `FedAvg` has to increase E to match the communication reduction. Figure 4.9 shows these results. Three pairs of lines are visible, each corresponding to a dataset. In order of lowest to highest accuracy: Cifar10, fashion-mnist and mnist.

Mnist

Mnist is the easiest dataset to classify; it reaches the highest accuracy, as shown by the top blue and brown lines. It is also barely affected by decreasing the commutation budget. Both techniques traditional (i.e. increasing the number of epochs per round) and the new phase-shift scheme can maintain an accuracy similar to the baseline value (1-phase, 0% communication saving). However, phase-shift can match the baseline accuracy at 4 phases or 37.5% less communication. `FedAvg`, when operating at 37.5% less communication, loses 0.5%.

For higher communication savings the final accuracy suffers slightly both for phase-shift and the traditional communication-saving method. At 20 phases, phase-shift loses performance as expected.

Fashion-mnist

Fashion-mnist is the second hardest dataset to classify, as can be seen by the green and red lines in figure 4.9. Similar to mnist, phase-shift can match the baseline accuracy at 4 phases or 37.5% less communication, while the same communication reduction achieved through communicating less frequently results in a 1% drop in accuracy. Both schemes again lose accuracy when constraining communication even further. And again, at 20 phases, phase-shift loses significant performance.

Cifar10

Cifar10 is the most challenging dataset, as can be seen by the bottom blue and orange lines. When saving communication by increasing the number of epochs a more significant accuracy drop occurs. Phase-shift is only loses 0.5% accuracy when operating at 45% less communication, whereas `FedAvg` loses 3% accuracy. The same drop-off in performance can be seen when phase-shift is configured to 20 phases.

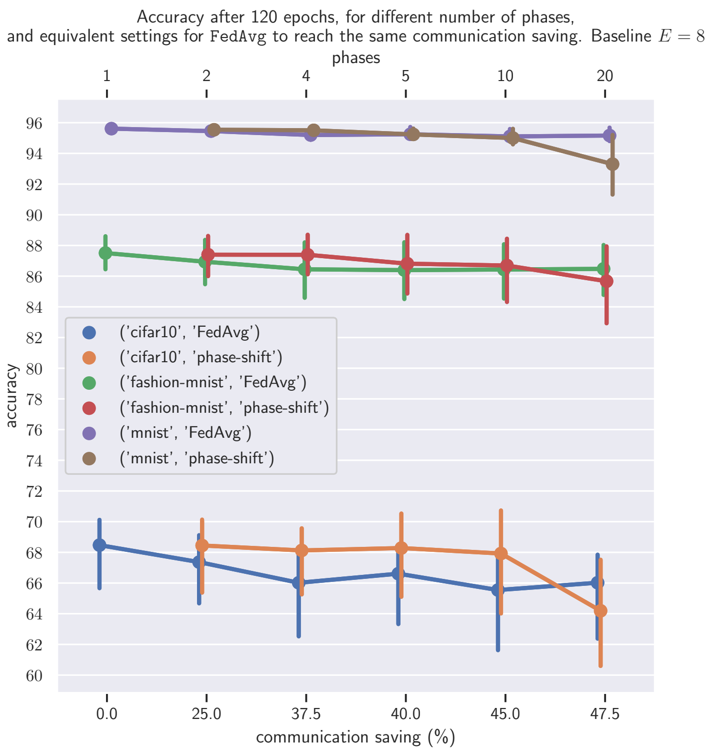


Figure 4.9: Accuracy after 120 epochs, with baseline $E = 8$. Results for phase-shift with 2, 4, 5, 10, and 20 phases, and for traditional methods to decrease communication.

4.5.5. RESULTS: $E = 16$

The experiment is repeated with a baseline of $E = 16$. Again, phase-shift uses this value for all experiments, whereas `FedAvg` has to increase E to match the communication reduction of phase-shift. Results are shown in figure 4.10. Mnist tells a similar story as with baseline $E = 8$, and no significant difference between the traditional method and phase-shift can be found. Phase-shift on fashion-mnist behaves similarly as well and can match the baseline at 4 phases, or 37.5% communication reduction, without lowering performance, whereas *FedAvg* loses a little over 2%. Only Cifar10 behaves differently from the previous experiment, and a drop in accuracy is noticeable from 5 phases and higher. However, phase-shift can match the baseline accuracy at 4 phases, whereas *FedAvg* loses 3% at the same communication reduction.

4.6. DISCUSSION

Phase shift operates best when configured to four phases. The optimum number of phases is likely determined by how many clients take part in a single phase, and how uniform the distribution of a single phase is. This is closely tied to the number of active

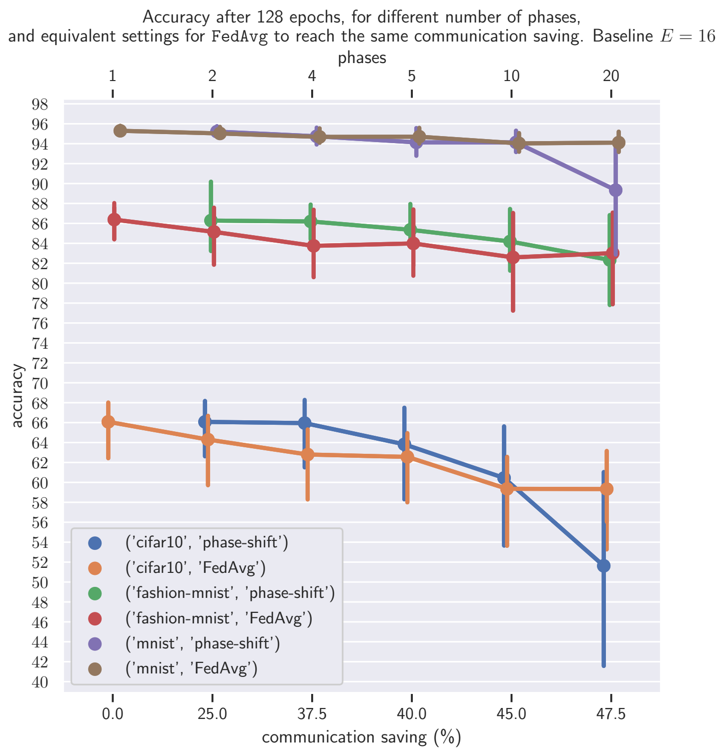


Figure 4.10: Accuracy after 128 epochs, with baseline $E = 16$. Results for phase-shift with 2, 4, 5, 10, and 20 phases, and for traditional methods to decrease communication.

clients per round, and the results indicate that, at EMD-1.4, at least 10 clients are needed per phase. This makes it likely that a higher number of clients can operate at a higher number of phases, as a random subset of clients is more likely to be uniform.

4.7. CONCLUSION

The results in the previous section indicate that phase-shift is a viable option, both for decreasing communication, or reducing the impact on convergence rate when using larger values for E. Specifically, phase-shift loses no % accuracy on cifar10 while using 37.5% less communication. If the same reduction in communication is achieved through conventional methods (i.e. increasing E) 3% in accuracy is lost. These results were achieved when operating at four phases, with a total of 40 clients active per round.

Phase shift can operate between a continuum of two extremes, *communication*- and *epoch*- reduction. This allows it to be applicable in reducing communication or improving accuracy when dealing with non-iid data.

Phase-shift operates differently from `FedAvg`, in that the function of the central model has changed. In regular federated learning, the central model is representative of what the system is currently capable of, whereas with phase-shift the central model only serves as a drift-correction. The central model is only obtained in the final round, where clients in every phase upload their local model to the server.

This has the added benefit of decreasing the computational load on the server, as fewer clients return their model per round.

4.8. FURTHER RESEARCH

One attribute of phase-shift is left undiscussed in this chapter; the difference in distribution between phases. Since all clients participate, which clients operate in which phase is locked from the first round on, which results in n sets of clients, or phases. The dataset of each phase will certainly not be iid, and some EMD can be calculated between the distribution of data in the first and second sets. One possible way to mitigate this issue is to allow for partial client participation. When not selecting all clients every round, clients can be part of both sets. And on average the EMD between any two phases will be lower.

5 AUGMENTATION

The most simple solution to the non-iid data problem is to tackle the root cause: the missing samples that result in a skewed dataset. This has been done before, Zhao et al. [40] reduced the skewness, or EMD, of the clients' datasets by introducing a common data set, in addition to the client's data. This dataset is used to reduce the skew of each client's data by extending the local data set with samples from the global set.

This has obvious drawbacks. First, such a dataset might not be available, second, it requires data transfer and thus introduces communication overhead, and finally, it might be prone to over-fitting. However, the solution should not be ignored as the results are promising.

An alternative to the common dataset is to augment the clients' existing data using various augmentation techniques. This method has at least one implementation in FL; `Astraea` [6], which shows promising results. Data augmentation is a well studied field within machine learning, and it can be used to extend different types of datasets, e.g. images [27], time-series [34] or text [19]. Data-augmentation exists in other forms as well, such as simple augmentation or completely artificially generated samples using General Adversarial Networks (GANs) [22] [29], this can even be applied in the medical field, where correct classification is arguably most important [28]. GANs have been shown to work in decentralized and in Federated Learning, even in non-iid settings [23] [37] [39].

Data augmentation comes at the cost of additional computation and can increase execution time, especially in the case of GANs. Recent works have shown that CPU overhead can be reduced by data-refurbishing [15] or by data-subsampling [14].

These techniques can be applied to deal with non-iid data distributions. Instead of drawing missing samples from a shared dataset, each client extends its data by augmenting local samples to reduce the skew. Thus, samples of a class of which a client has too few can be augmented and treated as additional samples to make the dataset appear uniform. This does require that each client has at least one sample of every class. This technique of data augmentation to combat non-iid data is not new, however, the authors of [6] do not provide a rule for the amount of data that needs to be added using augmentation. The amount of data augmentation does not bear any relation to the degree of skewness of the dataset. Herein lies an opportunity for improvement. Furthermore, the test setup favors the augmentation solution. The local datasets are up-sampled, so a single epoch contains more data. However, both the regular and augmented versions of FL operate using the same number of local epochs. This raises the question, *is the improvement in accuracy solely due to the augmentation of data or simply because it is allowed to perform more local learning*? Furthermore, is it favorable to learn on unaltered or augmented data if clients are allowed more local learning, and *how can the right balance between unaltered- and augmented data be achieved*?

This chapter is structured as follows, first, the algorithm for extending a dataset will be defined and analyzed. The resulting algorithm is then tested in a federated setting.

5.1. OVERSAMPLING METHOD

The method used to over-sample the dataset should take the skew of the distribution into account, as more skew is likely to require more data augmentation. However, data augmentation should be kept at a minimum to avoid over-fitting. As shown before, EMD is a good metric to predict the relative performance of federated learning on differently distributed datasets. The lower the EMD of a data-set, the better the performance of the network is likely to be. EMD can be used as a control parameter for extending a dataset, where an *augmented EMD* is given, and the dataset is strategically up-sampled to achieve this EMD. The term *augmented-EMD* will used to refer to the EMD of a dataset *after* augmentation.

An augmented-EMD does not define a distribution, thus an oversampling method is needed to determine a new distribution with the required EMD. A simple method would be to extend the dataset class-by-class until the target EMD is reached. However, it is likely smarter to start with the class of which a client has the least amount of samples and to start filling in the missing samples from the bottom up.

The idea of filling up from the bottom is shown in figure 5.1. As shown, the distribution has some value for EMD. To generate a new distribution, the classes in red are up-sampled until a number of L samples is reached. This is done for all k classes below the dotted line. The EMD' of the new distribution is certainly smaller than EMD as the new distribution is closer to uniform. The problem now becomes to find L and k for some given *augmented-EMD*.

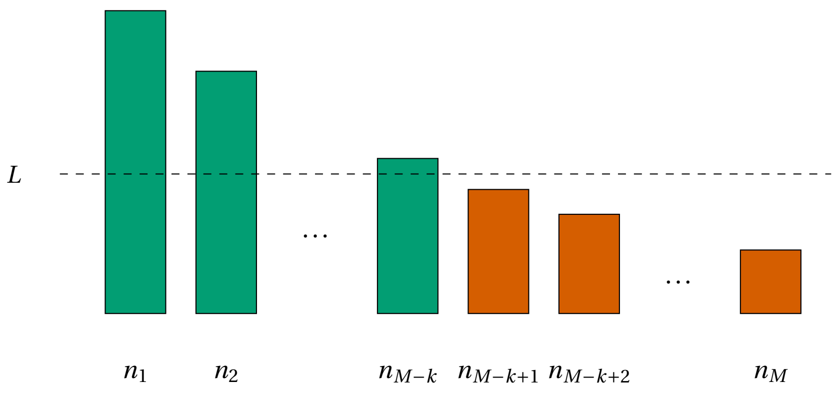


Figure 5.1: Distribution of a client, with the height of each bar representing the number of samples n_x of a particular class

5.1.1. FINDING L AND k

A solution to this problem is to express the value of EMD in terms of L and k and to solve for these parameters. It is assumed that the number of samples of each class are sorted, such that $n_i \geq n_{i+1} \forall i \in \{1..M-1\}$, to match figure 5.1. The augmented-EMD can now be calculated using the standard equation for calculating EMD 3.3:

$$EMD = \sum_{i=1}^{M-k} \left| \frac{n_i}{kL + \sum_{j=1}^{M-k} n_j} - \frac{1}{M} \right| + k \left| \frac{L}{kL + \sum_{j=1}^{M-k} n_j} - \frac{1}{M} \right| \tag{5.1}$$

Let $s_{M-k} = \sum_{i=j}^{M-k} n_j$, and since $n_{M-k} \geq L$, eq. 5.1 becomes:

$$EMD = \frac{s_{M-k}}{kL + s_{M-k}} - \frac{M-k}{M} + \frac{k}{M} - \frac{kL}{kL + s_{M-k}} \tag{5.2}$$

which simplifies to

$$EMD = \frac{2k}{M} - \frac{2kL}{kL + s_{M-k}} \tag{5.3}$$

Solving eq. 5.3 for L results in:

$$L = \frac{2ks_{M-k} - EMD \cdot s_{M-k}M}{2kM + EMD \cdot kM - 2k^2} \tag{5.4}$$

5.1.2. ALGORITHM

Equation 5.4 has the free variable k, which means that L can't be calculated directly. However, k is an integer between $1 \leq k < M$. This allows the construction of the algorithm shown in listing 4.

Each client is given the same *augmented-EMD*, and if it finds that its own EMD is larger, it calculates L using algorithm 4. The client then over-samples any class with less than L samples up to this level. This ensures that a client only over-samples classes of which it has few, furthermore, augmented-EMD acts as a control parameter that balances the trade-off between skew and the chance of over-fitting.

Of course, this method introduces extra data. To allow for a fair comparison to regular Federated learning and the proposed idea, the amount of local computation should remain the same in both cases. Therefore, the number of local epochs E will always refer to the size of the dataset *before* augmentation.

Algorithm 4 Procedure to calculating the minimum number of samples L

```
1:  procedure CALCULATE-L($D = [n_1, n_2, ... n_M]$)
2:      $D \leftarrow sorted(D, increasing)$        ▷ sorted distribution of class quantities
3:      $s_0 \leftarrow 0$
4:      $k \leftarrow M$
5:      for $i \in \{1..M-1\}$ do
6:          $k \leftarrow k-1$
7:          $s_{M-k} \leftarrow s_{M-k-1} + D_i$
8:          calculate $L$ using eq. 5.4
9:          if $D_{i+1} \leq L$ then        ▷ Sufficient condition for $D_j < L \ \forall\ j \in \{M-k+1..M\}$
10:             break
11:     return $L$
```

5.2. AMOUNT OF DATA ADDED

The method of data augmentation proposed in the previous section introduces extra data. The amount of data added to a client is of great interest as its ratio of true data v.s. augmented data will have an impact on over-fitting. To calculate this amount the following predicate needs to be considered.

$$\frac{n_i}{\sum_{j=1}^{M} n_j} \leq \frac{1}{M} \qquad \forall i \in \{M-k+1 \dots M\} \tag{5.5}$$

If eq. 5.5 holds, all classes in red in figure 5.1 (i.e. all those below L) have a chance $p \leq \frac{1}{M}$ of being drawn.

Furthermore, it needs to be shown that moving data around between the classes in red does not affect EMD, as long as the predicate still holds. Suppose we have two classes with n_i and n_j samples each, with $i, j \in \{M-k+1 \dots M\}$ and $i \neq j$. We want to move a data quantity a from n_i to n_j with a such that eq. 5.5 still holds. Their contribution to the total value of EMD is as follows.

$$\left|\frac{n_i - a}{N} - \frac{1}{M}\right| + \left|\frac{n_j + a}{N} - \frac{1}{M}\right| = \tag{5.6}$$

$$\left(\frac{1}{M} - \frac{n_i - a}{N}\right) + \left(\frac{1}{M} - \frac{n_j + a}{N}\right) = \tag{5.7}$$

$$\left|\frac{n_i}{N} - \frac{1}{M}\right| + \left|\frac{n_j}{N} - \frac{1}{M}\right| \tag{5.8}$$

So we can conclude that data can be moved arbitrarily between classes with the number of samples below L, without affecting EMD, as long as eq. 5.5 still holds. Thus, the

Table 5.1: ratio of unaltered data to total data for different sampling methods, at EMD-1.4

Augmented EMD	EMD-0.0	EMD-0.4	EMD-0.8
$ll(1, 0.78)$	0.13	0.39	0.64
$ll(2, 0.88)$	0.24	0.46	0.68
$ll(3, 1.00)$	0.30	0.50	0.70
mean	0.22	0.45	0.67

data of the classes in red can be redistributed (i.e. moved around) such that they all have the same level L'. To calculate the amount of data, we first calculate L, k using the augmented EMD, and then calculate L' with the original EMD. The fraction of unaltered data to total data (augmented + unaltered) follows equation 5.9.

$$\frac{\text{unaltered data}}{\text{total data}} = 1 - \frac{k(L - L')}{N + k(L - L')} = \frac{N}{N + k(L - L')} \tag{5.9}$$

EXAMPLE

Consider the case where a client is given data according to the $ll(2, 0.9)$, with $M = 10$, resulting in an EMD of 1.44. Say, we decide to augment this client up to an EMD of 0.4. It is clear how $k = M - t = 8$ follows from the distribution method. If the client has a total of N samples, $S_{M-k} = N * (f - t\frac{1-f}{M}) = 0.92N$ (eq. 3.10). Thus $L = 0.1725N$, and $L' = 0.01N$, from which follows the fraction of unaltered data is 0.47.

EXPECTED RATIOS

In section 4.4 the standard setup is described. Most experiments are conducted at EMD-1.4, using one of three samplers: $ll(1, 0.78)$, $ll(2, 0.86)$ and $ll(3, 1.0)$. Using the method described above, the expected ratios for unaltered to total data can be derived. Results are shown in table 5.1. Interestingly, the ratio of unaltered data to total data increases with the label limit. As before, the parameter k is the number of classes that need augmentation, and in the case of the limit-label sampler, k equals $M - t$. So a lower label-limit t results in more classes requiring augmentations, hence the ratio increases with t.

As will be explained in section 5.4.1 the algorithm will be tested at EMD-0.4, 0.8, 1.2 and 1.4. The results for these settings are repeated, but only the average of the possible settings is shown in table 5.2. As expected a lower value for EMD results in less data augmentation need to reach a certain augmented EMD. Similarly, augmenting a distribution to a lower EMD requires more augmentation, so the ratio increases with EMD and lowers with augmented EMD. The lowest ratio (i.e. the least amount of unaltered data) is reached at the highest EMD and the lowest augmented EMD.

5.3. IMPLEMENTATION

The augmentation algorithm is implemented alongside FLTK, using python and PyTorch. For MNIST and fashion-MNIST augmentations such as horizontal flip (vertical could change a 6 into a 9 for MNIST), affine-transformation, perspective transformations, and

Table 5.2: Average ratio of unaltered to total data for different values of EMD and augmented EMD. The number of settings possible for the $ll(t,f)$ sampler to reach the specified EMD is shown in the last column.

	augmented EMD			
emd	0.0	0.4	0.8	nr. solutions
0.4	0.78	NA	NA	7
0.8	0.45	0.66	NA	6
1.2	0.29	0.51	0.72	4
1.4	0.22	0.45	0.67	3

noise are applied. Each is transformation has some non-zero chance of being applied, and the meta-parameters are random as well. For instance, if applied, the affine-transformation applies some rotation between -20 and 20 degrees. This ensures the augmentations are combined in unique ways and lowers the sensitivity to overfitting.

Cifar10 has color channels, and thus additional transformations can be applied, such as color-jitter and grey-scale.

5.4. EMPIRICAL ANALYSIS

The proposed solution will henceforth be called `FedAug`. `FedAug` does not provide any method of selecting the right value for the meta-parameter: *augmented EMD*. Empirical analysis should provide insight into the right value to balance augmented and unaltered data.

The original experiments by [6] used a flawed setup that favored the proposed solution. Augmented data was used in addition to unaltered data, and thus the local dataset increased. This allowed `Astraea` to perform more learning iterations. A fair setup should either down-sample the augmented dataset to reduce it back to the original size, or the baseline should be allowed to perform the same number of computations.

The latter method is used in the experiments for `FedAug`. The number of local epochs E is increased, such that both the unaltered data and augmented data can be included without down-sampling. Thus, the empirical analysis should also provide insight into the usefulness of data-augmentation in the first place, or if more learning on unaltered data is always preferable.

5.4.1. EXPERIMENTAL SETUP

A total of 20 clients are given data according to the limit-labels distribution method. Four types of distributions are considered, at $EMD = 0.4, 0.8, 1.2$ and 1.4. `FedAug` is then tasked to augment these distributions to *augmented-EMD*= $0.0, 0.4, 0.8$. If the augmented EMD is higher than the EMD of the original distribution the experiment is skipped. Finally, each client runs 4 local epochs, for 8 rounds in total.

All Experiments are repeated four times, and standard deviations are reported. All other settings are given in section 4.4.

The augmented EMDs of 0.0, 0.4 and 0.8 result in different ratios of unaltered to augmented data, depending on the EMD of the original distribution. These ratios are provided in table 5.3. Note that these ratios are independent of the datasets used in each experiment, as the same random-seed is used to distribute Cifar10, MNIST and Fashion-MNIST.

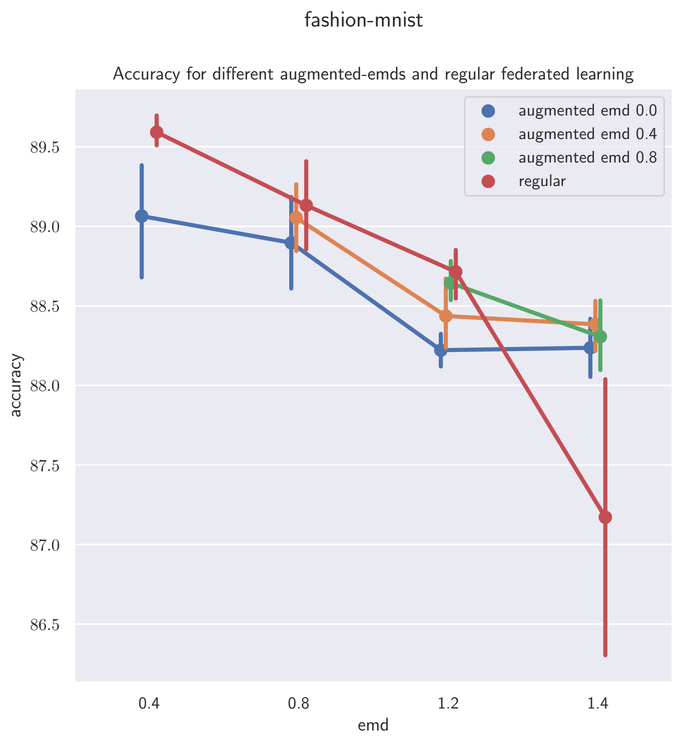


Figure 5.2: Results for augmentation of the fashion-MNIST dataset. Highest achieved accuracy after 10 rounds of 4 epochs each

As expected, a positive correlation can be found between the Augmented EMD and the percentage of unaltered data. Augmenting the distribution of EMD-1.4 to EMD-0.0 results in 20% unaltered data v.s. 80% augmented data, whereas augmenting to EMD-0.8 results in 60% unaltered data.

The resulting ratios are very similar to those predicted by theory in section 5.2, table 5.2. The real ratios are closest to the predicted value for EMD-0.8, 1.2, 1.4, while further away from those predicted for EMD-0.4. This is dictated by chance, as the number for possible solutions for the lower EMD values (see table 5.2) is greater than the number of repetitions in the experiment. Since not all settings that result in the same EMD also result in the same ratio, the true ratios differ from the average expected ratio.

Table 5.3: Ratio of unaltered data, to total data (unaltered + augmented) for different EMDs and augmented EMDs

	Augmented EMD		
emd ↓	0.0	0.4	0.8
0.4	0.6±0.1	NA	NA
0.8	0.5±0.1	0.7±0.1	NA
1.2	0.2±0.1	0.5±0.1	0.7±0.1
1.4	0.2±0.1	0.4±0.1	0.6±0.0

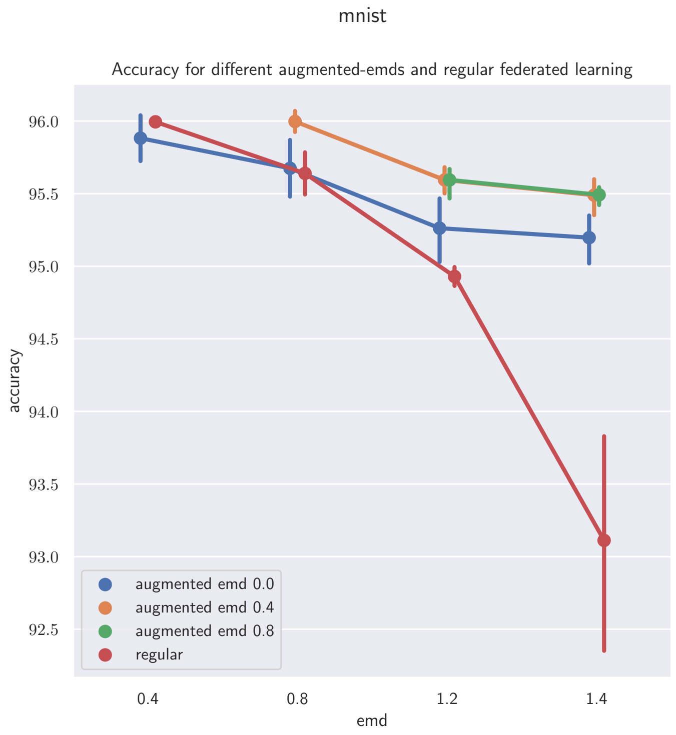


Figure 5.3: Results for augmentation of the MNIST dataset. Highest achieved accuracy after 8 rounds of 4 epochs each

5.4.2. RESULTS: MNIST

Results for the MNIST dataset are visible in figure 5.3. First, take note of the y-axis scale. Regular `FedAvg` only loses about 3% accuracy when going from EMD-0.4 to EMD-1.4. `FedAvg` performs better ar EMD-0.4, where it is more beneficial to learn from unaltered data, and augmenting to an even lower EMD has no benefit.

This changes for higher values, at EMD-0.8 `FedAug-0.4` outperforms `FedAvg`, and continues to do so for EMD-1.2 and 1.4. Augmenting to 0.8 performs very similar, and is therefore preferable to FedAug-0.4, as it should be less sensitive to over-fitting.

There seems to be no benefit to augmenting completely to EMD-0.0, as it is always outperformed by either FedAvg or by augmenting to higher EMDs. Augmenting to such a low EMD likely results in over-fitting.

FedAug-0.8 performs 2.4% better than FedAvg at EMD-1.4, with lower variance. At higher EMDs it is thus more beneficial to replace some learning on unaltered data with learning on augmented data, to reduce the skew of the local dataset.

5.4.3. RESULTS: FASHION-MNIST

Results for the fashion-MNIST dataset are visible in figure 5.2. Again, take note of the y-axis scale. Regular FedAvg only loses about 2.5% accuracy when going from EMD-0.4 to EMD-1.4. However, unlike the previous results, FedAvg performs better, up to EMD-1.2. At these lower values for EMD, the disadvantage of learning from skewed data is not great enough to be worth replacing unaltered samples with augmented ones.

At EMD-1.4 FedAvg falls off and FedAug performs better. Again, FedAug-0.8 seems to provide a good trade-off between unaltered and augmented data. It performs 1.2% better than FedAvg at EMD-1.4, with lower variance.

5.4.4. RESULTS: CIFAR10

Results for the cifar10 dataset are visible in figure 5.4. Cifar10 proves to be more difficult than previous datasets, as FedAvg loses about 16% accuracy when going from EMD-0.4 to EMD-1.4. However, FedAvg still performs better at EMD-0.4 and 0.8, but falls off at EMD-1.2 and 1.4. The same reasoning as with (fashion-)mnist can be applied here to explain these results, as the effect of non-iid data only starts to significantly hurt performance at EMD-1.2 and 1.4. Once this drop has occurred, it seems beneficial to replace unaltered data with augmented data, to reduce the skew.

Cifar10 seems to be more suitable for augmentation as FedAug-0.0 performs very similar to augmented EMD-0.4 and 0.8. This is likely due to cifar10 being the most complex dataset, and augmentations can thus offer more variety, such as color effects. Since augmented EMD-0.8 uses the least amount of augmented data, it should be preferred to reduce the chance of over-fitting. From this we can also conclude that the quality of the augmentations is likely a great impact on the augmentation level needed, i.e. better data augmentation is likely able to reach the same performance, at a higher augmented EMD.

FedAug achieves 10% higher accuracy than FedAvg at EMD-1.4, at significantly lower variance.

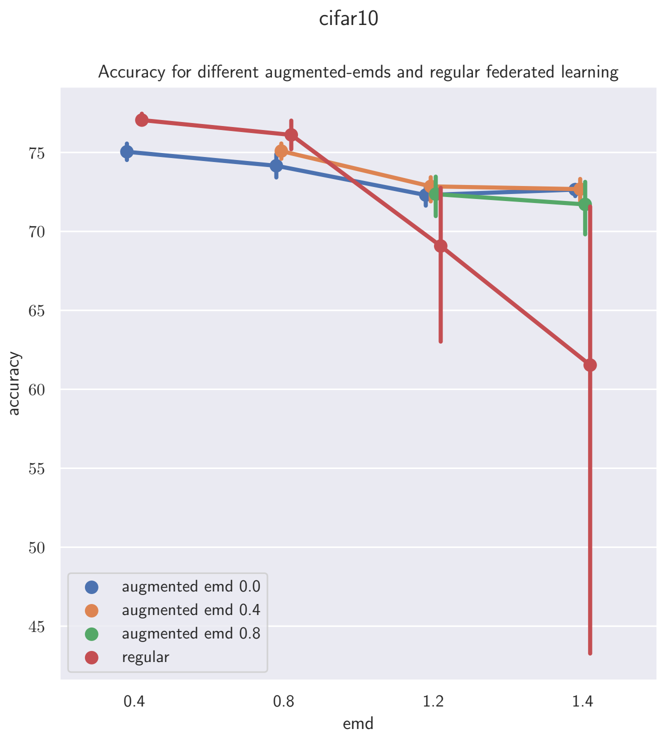


Figure 5.4: Results for augmentation of the Cifar10 dataset. Highest achieved accuracy after 40 rounds of 4 epochs each

5.5. Discussion and Conclusion

One point of contention is the method with which augmentation attempts to solve problems caused by non-iid data. EMD is used to introduce the problem (datasets are distributed to reach a specific EMD) and also used to solve it (upsampled to reach a specific EMD). One could argue that this skews the results in favor of the augmentation algorithm. However, any method used to introduce skew into a distribution results in some value for EMD, even if EMD was not used to generate that distribution.

In its current implementation, `FedAug` only augments those classes of which it has too little samples, as seen in figure 5.1. It could be that the solver starts to recognize the augmentations, to help it classify certain classes. However, in the test environment, all clients have the same EMD, and the global distribution is uniform. Any learned bias towards augmentations of a certain class is thus cancelled by another client with an 'opposite' distribution, at the same EMD. Furthermore, as found by several related works, Federated-Learning is barely affected by non-iid feature distribution [11] [16] [41].

`FedAug` can counter the effects of non-iid data, but only for EMDs above 1.2. At these values for EMD `FedAug` both increases accuracy and decreases variance in performance. On cidar10, accuracy is increased by 10% and the standard deviation by 9%. This indicates that `FedAug` is a great tool in the toolbox when dealing with non-iid data.

`FedAug` is not the first method to use augmentation, but it is the first to provide a more systematic way to balance augmented and unaltered data. Furthermore, the existing method `Astraea` used a testing setup that favored their solution. Finally, no distribution method was specified, and thus no EMD can be calculated. Their gain over `FedAvg` of 5.6% is consequently without context, and can not be compared to `FedAug`.

As discussed in chapter 3, only distributions where all clients have similar values for EMD, and a similar number of samples are considered. `FedAug` could perform well in these cases as well, where only clients with very skewed, or little data are augmented. Thus, `FedAug` could be of help when dealing with quantity-heterogeneity as well.

6
COMBINING RESULTS

6.1. DESIGN RATIONAL

The results from `FedAug` and phase-shift can be combined and should result in further improvements over `FedAvg`. The respective chapters of these solutions have shown in which cases they operate best. In chapter 5 it was shown that `FedAug` works best at higher EMD values. For this reason, an EMD of 1.4, using the limit-label-sampler, has been selected as a testing point. The results also showed that an augmentation EMD of 0.8 offers a good balance between unaltered and augmented data. This is further supported by the results of chapter 3, where it was shown that non-iid data starts to degrade performance above EMD-0.8-1.0.

The results of phase shift indicate that a minimum of ten clients are needed per phase. Due to limitations in the test setup, the number of clients will be kept at 20, and thus two phases will be used. Phase-shift can operate in a continuum of two extremes: *communication reduction*, where 25% less communication is used at the cost of a slight performance reduction and *epoch-reduction*, where the same amount of communication is used but accuracy is increased. Since this work focuses on non-iid data, we will use the epoch reduction mode.

Any new idea is incomplete without it being compared against related work. In this chapter the proposed augmentation algorithm and the phase-shift solution are compared against the existing solutions `FedProx` and `FedNova`.

6.2. IMPLEMENTATION

`FedNova` and `FedProx` have been implemented in the existing FLTK framework. The code is based on an existing implementation available on github [31]. `FedNova` updates the aggregation function to better approximate the global objective function. `FedProx` limits how far the local model is allowed to drift away from the initial model using a meta-parameter μ. The authors of `FedProx` proposed an automatic tuning algorithm for this parameter, which has been used in this implementation.

Table 6.1: Epoch and round settings for two-phase phase-shift (2P), `FedProx`, `FedNova`, and FedAvg

solution	E	rounds	total epochs
2P	9	12	108
`FedNova`, `FedProx`, FedAvg	12	9	108

6.3. EXPERIMENT SETUP

In order to constraint all algorithms to the same amount of communication, the epoch settings listed in table 6.1 will be used. The experiments are repeated for datasets Cifar10, Fashion-MNIST and MNIST. All other experiment settings can be found in section 4.4.

6.4. RESULTS

Figure 6.1 shows the results for the experiments described above. For cifar10 2P-`FedAug` is able to achieve a 13 % higher accuracy than baseline FedAvg. `FedNova` improves 7% and `FedProx` loses 1%. A similar story can be told for datasets Fashion-MNIST and MNIST, except for `FedNova` on Fashion-MNIST, where it loses 7%. 2P-`FedAug` consistently achieves a higher accuracy on all datasets, whereas `FedNova` improves on Cifar10 and MNIST, but loses on Fashion-MNIST. One possibility for the bad results of `FedProx` could be the automatic tuning algorithm for the μ parameter, which was only introduced as an afterthought by Li et al. and not included in the main work.

6.5. DISCUSSION

`FedNova` and especially `FedProx` aim to solve non-iid related problems in the case of heterogeneous data quantities. In this work, the problem is constrained to label heterogeneity only, so it is understandable that these solutions perform worse than advertised in their respective papers. However, both papers do claim to improve on this specific case, so they have been included in this work.

A clearer picture of 2P-`FedAug`-0.8 would have been achieved if `Scaffold`, `STC`, or `FedMix` would have been implemented as well (see table 3.1). `Scaffold` aims to reduce the variance between clients, `STC` applies compression to up-stream and down-stream communication, and `FedMix` used a privacy preserving data sharing method to reduce skew. Both have not been implemented due to time constraints. However, all three have been tested on Cifar10 in existing works, using the limit-label-sampler to achieve similar EMD values. [16] tested `Scaffold` on EMD-1.2 and 1.6 where it achieved 0.5% and 0.7% lower accuracy than `FedAvg`. `STC` was able to improve upon FedAvg by 31% at EMD-1.6 and by 14% at EMD-1.2. Finally `FedMix` shows a 7.4% increase in accuracy when dealing with EMD-1.6

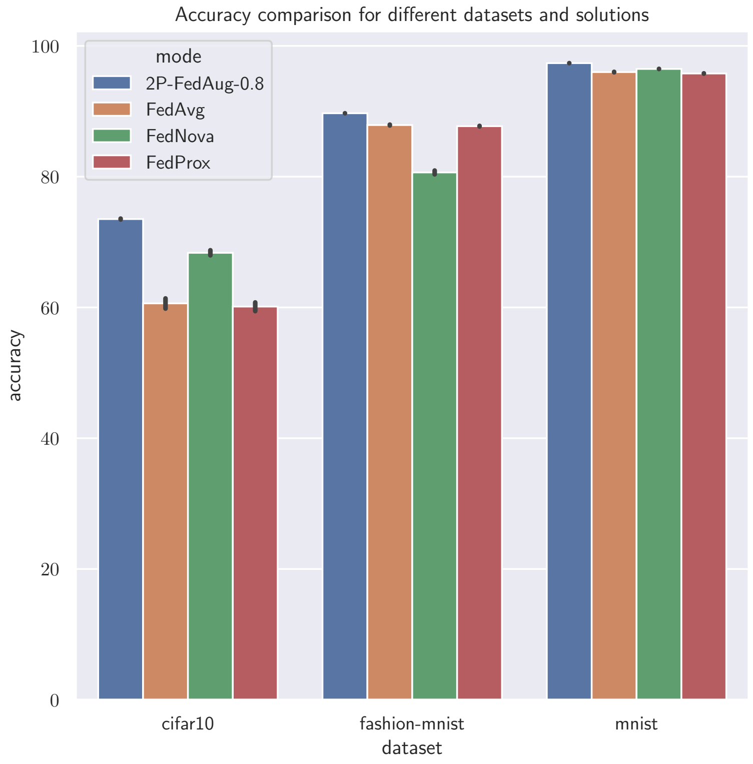


Figure 6.1: Accuracy comparison for different detests and solutions, EMD-1.4

6.6. CONCLUSION

The results show that when the problem is constrained to label-heterogeneity only, two-phase FedAug-0.8 is consistently better than FedProx and FedNova. 2P-FedAug achieves 13% accuracy than FedAvg on cifar10, at EMD-1.4, whereas FedNova gains 7% and FedProx loses 1%.

The results of Scaffold [13] and, the meta-analysis paper [16] both indicate that the proposed solution is likely better in dealing with non-iid data as well.

STC is both able to reduce communication, and their results indicate that it is better able to deal with non-iid data. An implementation of STC should confirm this but is left to future research.

It should be noted however, that both phase-shift and FedAug can be implemented alongside any of these algorithms, either to further reduce communication or to improve performance under non-iid data. This is left to further research as well.

7 Conclusion

Federated learning is an upcoming machine learning platform, that offers greater privacy protection while having the benefit of having multiple devices working together. However, data across these devices is typically not uniformly distributed, resulting in non-iid data. Clients' data being skewed makes it harder to collaboratively learn on the same machine learning problem.

Many researchers have studied the effect of non-iid data on a range of problems related to federated learning. However, no standard method of artificially creating skewed datasets exists, nor is there a unit with which to compare them. Earth movers distance (EMD) was proposed to measure the skew but insufficiently researched to draw more general conclusions. Thus the following two research questions have been defined.

RQ1 Can EMD be used to quantify the degree of skew, such that two non-identical distributions, but with the same degree of skew, result in similar performance?

RQ2 Can the existing distribution methods, for artificially creating non-iid label distributions, be compared using EMD?

These questions have been thoroughly researched in chapter 3. First, the existing data distribution methods have been analyzed and translated to EMD. This allowed us to generate distributions through different methods, but with the same value for EMD. Experiments show that performance is very similar, independent of the distribution method, for EMDs below 1.0. However, EMD does not capture the problems caused by quantity skew, and thus can not be reliably used to compare the Dirichlet sampler to other methods. Furthermore, EMD is not able to properly capture the extreme skew caused by a pure limit-label sampler, where a subset of clients holds all data from a class. Since much of the related work uses the limit-label-sampler, the ability of EMD to allow for the comparison of all distribution methods is reduced.

Chapter 4 introduced the novel scheme *phase-shift*, which allows for a reduction in communication, without losing accuracy, or reducing the convergence rate. Phase shift serves as an answer to the third research question:

RQ3 How can the communication frequency be increased, without increasing the instantaneous communication and computational load on the system?

Phase-shift is able to achieve this by moving away from a single centralized model, aggregated at the end of every round. Instead, only the result of a few clients is aggregated and redistributed to still running clients, to serve as drift correction. This means that no real central model exists, until the final round, where all active clients return their model. Experiments show that phase-shift is reliably able to reduce instantaneous, end thereby total communication by 37.5% without losing on accuracy or convergence. Furthermore, no additional computation is required, and only minimal changes to the original `FedAvg` algorithm are needed.

Chapter 5 discussed the ability of data-augmentation to reduce skew by supplementing local data. Based on the results from related work, two research questions were defined:

RQ4 Can data augmentation be used to complement each client's local dataset, to reduce skew and increase performance? or is it preferable to spend the additional learning on unaltered data instead?

RQ5 How can the right balance between unaltered and augmented data be achieved?

Results indicate that it can be beneficial to replace more learning from unaltered data, with learning on augmented data, to reduce skew. However, this only holds true if the data is sufficiently skewed, and experiments show that is the case at EMD-1.4. Chapter 3 showed that performance of federated learning drops once the distribution reaches above EMD-0.8-1.0, independent of distribution method or dataset. Therefore, EMD has been selected as a control parameter for balancing the amount of unaltered and augmented data. Testing indicates that augmenting a dataset up to an EMD of 0.8 results in a good balance between the two.

7.1. LIMITATIONS

This work purely considers the problem of label heterogeneity and does not consider the impact of quantity skew, or client unreliability. Quantity skew could cause issues with phase-shift, where phases hold different quantities of data. Furthermore, phase-shift requires clients to partake in several sequential phases, thus client unreliability could have negative consequences. However, including other types of heterogeneity would have clouded the impact of label-skew, limiting the reliability of possible conclusions.

Resource heterogeneity or constraints could negatively impact Data-augmentation as it induces a computational overhead. The impact of this is outside the scope of this work. Furthermore, data augmentation relies on each client having at least one sample of every class in the problem. Thus `FedAug` can not deal with distributions where a subset of clients hold all samples from a single class. Possible solutions are discussed in future work.

Furthermore, the related work tested in chapter 6 does not only focus on this problem but includes quantity skew as well. Implementing other methods such as `data sharing` or `FedMix` would have painted a clearer picture.

Finally, this work only considers a relatively small number of clients. This is mostly due to limitations in the test setup, as each client requires a virtual machine. Using a large number of clients would result in an expensive setup.

Due to time constraints, most experiments in this work have been conducted at EMD-1.4. Including extreme EMDs such as 1.6 or 1.8 would paint a broader picture of the capabilities of the proposed solutions. Results in chapter 3 show that performance starts to drop significantly at EMD-1.4, hence the selection of this value.

7.2. FUTURE WORK

Our research into EMD has shown that the unit is limited in its ability to unify different distribution methods. Further research into what causes the different distribution methods to diverge above a certain EMD might lead to a better understanding of the problems with non-iid data. Furthermore, other measuring units might be better able to capture those attributes of a distribution that cause performance degradation. Finally, the impact of quantity skew and EMD-heterogeneity should be quantified as well.

Phase shift requires clients to partake in multiple sequential rounds, and in a sense 'invests' in the result of a client after these rounds. Unreliable clients are therefore more likely to hurt performance. Further research should be done to see if the advantage of allowing more clients per round outweighs this disadvantage. Furthermore, the impact of the distribution of data within a phase of clients is still an open question. Finally, the idea of phase-shift could be extended to asynchronous federated learning.

Augmentation can be applied in dealing with quantity skew as well. Furthermore, GANs can be used instead of algorithmic data augmentation, likely reducing the chance of overfitting, and increasing overall accuracy. Finally, `FedAug` requires that clients hold at least one sample of every class. This is a disadvantage, and thus `FedAug` does not cover the case where a subset of clients hold all the samples of a specific class. This could be solved with a publicly shared dataset, or with privacy-preserving data-sharing solutions such as `FedMix` or `FedXOR`, but requires further research.

A
NEURAL NETWORKS

Three datasets are used throughout this document, `CIFAR10`, `MNIST`, and `Fashion MNIST`. For each dataset, the commonly used *CNN* networks are used. Their partial python implementations are shown in listings 3, 1, and 2.

Listing 1 CNN network for the MNIST dataset, defined using `pyTorch`

```python
class MNIST_CNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)
```

Listing 2 CNN network for the Fashion-MNIST dataset, defined using `pyTorch`

```
class FashionMNISTCNN(nn.Module):
    def __init__(self):
        super().__init__()

        self.layer1 = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=5, padding=2),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.layer2 = nn.Sequential(
            nn.Conv2d(16, 32, kernel_size=5, padding=2),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(2))

        self.fc = nn.Linear(7*7*32, 10)
```

Listing 3 CNN network for the CIFAR10 dataset, defined using `pyTorch`

```
class Cifar10CNN(nn.Module):
    def __init__(self):
        super().__init__()

        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(32)
        self.conv2 = nn.Conv2d(32, 32, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(32)
        self.pool1 = nn.MaxPool2d(kernel_size=2)

        self.conv3 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(64)
        self.conv4 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
        self.bn4 = nn.BatchNorm2d(64)
        self.pool2 = nn.MaxPool2d(kernel_size=2)

        self.conv5 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.bn5 = nn.BatchNorm2d(128)
        self.conv6 = nn.Conv2d(128, 128, kernel_size=3, padding=1)
        self.bn6 = nn.BatchNorm2d(128)
        self.pool3 = nn.MaxPool2d(kernel_size=2)

        self.fc1 = nn.Linear(128 * 4 * 4, 128)
        self.fc2 = nn.Linear(128, 10)
```

BIBLIOGRAPHY

[1] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design, 2019.

[2] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, 54, 2017.

[3] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. *arXiv*, dec 2020. URL http://arxiv.org/abs/2012.13995.

[4] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. Tifl: A tier-based federated learning system, 2020.

[5] Bart Cox and Izaak Cornelis. *FLTK - Federation Learning Toolkit*. GitHub, 2021.

[6] Moming Duan, Duo Liu, Xianzhang Chen, Yujuan Tan, Jinting Ren, Lei Qiao, and Liang Liang. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. *Proceedings - 2019 IEEE International Conference on Computer Design, ICCD 2019*, pages 246–254, 7 2019. doi: 10.1109/ICCD46524.2019.00038. URL http://arxiv.org/abs/1907.01132http://dx.doi.org/10.1109/ICCD46524.2019.00038.

[7] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning, 2021.

[8] Pengchao Han, Shiqiang Wang, and Kin K. Leung. Adaptive gradient sparsification for efficient federated learning: An online learning approach. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 300–310, 2020. doi: 10.1109/ICDCS47774.2020.00026.

[9] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction, 2019.

[10] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. The Non-IID Data Quagmire of Decentralized Machine Learning. *arXiv*, sep 2019. URL http://arxiv.org/abs/1910.00189.

[11] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael GL D, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrì Gascón, Badih Ghazi, Phillip B Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečn, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X Yu, Han Yu, and Sen Zhao. Advances and Open Problems in Federated Learning. Technical report, 2021.

[12] Georgios Kaissis, Marcus Makowski, Daniel Rückert, and Rickmer Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2, 06 2020. doi: 10.1038/s42256-020-0186-1.

[13] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning, 2021.

[14] Michael Kuchnik and Virginia Smith. Efficient augmentation via data subsampling, 2019.

[15] Gyewon Lee, Irene Lee, Hyeonmin Ha, Kyunggeun Lee, Hwarim Hyun, Ahnjae Shin, and Byung-Gon Chun. Refurbish your training data: Reusing partially augmented samples for faster deep neural network training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 537–550. USENIX Association, July 2021. ISBN 978-1-939133-23-6. URL `https://www.usenix.org/conference/atc21/presentation/lee`.

[16] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated Learning on Non-IID Data Silos: An Experimental Study. feb 2021. URL `http://arxiv.org/abs/2102.02079`.

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

[18] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. ON THE CONVERGENCE OF FEDAVG ON NON-IID DATA. Technical report, 2020.

[19] Pei Liu, Xuemin Wang, Chao Xiang, and Weiye Meng. A survey of text data augmentation. In *2020 International Conference on Computer Communication and Network Security (CCNS)*, pages 191–195, 2020. doi: 10.1109/CCNS50731.2020.00049.

[20] Amirhossein Malekijoo, Mohammad Javad Fadaeieslam, Hanieh Malekijou, Morteza Homayounfar, Farshid Alizadeh-Shabdiz, and Reza Rawassizadeh. Fedzip: A compression framework for communication-efficient federated learning, 2021.

[21] Takayuki Nishio and Ryo Yonetani. Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. *IEEE International Conference on Communications*, 2019-May, apr 2018. doi: 10.1109/ICC.2019.8761315. URL http://arxiv.org/abs/1804.08333http://dx.doi.org/10.1109/ICC.2019.8761315.

[22] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017. URL http://arxiv.org/abs/1712.04621.

[23] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. Fedgan: Federated generative adversarial networks for distributed data, 2020.

[24] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. FetchSGD: Communication-efficient federated learning with sketching. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8253–8265. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/rothchild20a.html.

[25] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robucommunication-efficient learning of deep networks from decentralized datast and communication-efficient federated learning from non-iid data, 2019.

[26] MyungJae Shin, Chihoon Hwang, Joongheon Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Xor mixup: Privacy-preserving data augmentation for one-shot federated learning, 2020.

[27] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48, 2019.

[28] Shobhita Sundaram and Neha Hulkund. Gan-based data augmentation for chest x-ray classification, 2021.

[29] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30:1882–1897, 2021. doi: 10.1109/TIP.2021.3049346.

[30] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 1698–1707, 2020. doi: 10.1109/INFOCOM41043.2020.9155494.

[31] Jianyu Wang, Qianghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. *Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization*. GitHub, 2020.

[32] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization, 2020.

[33] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019. doi: 10.1109/JSAC.2019.2904348.

[34] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, Aug 2021. doi: 10.24963/ijcai.2021/631. URL http://dx.doi.org/10.24963/ijcai.2021/631.

[35] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6893–6901. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/xie19b.html.

[36] Zirui Xu, Zhao Yang, Jinjun Xiong, Jianlei Yang, and Xiang Chen. Elfish: Resource-aware federated learning on heterogeneous edge devices, 2019.

[37] Ryo Yonetani, Tomohiro Takahashi, Atsushi Hashimoto, and Yoshitaka Ushiku. Decentralized learning of generative adversarial networks from non-iid data, 2019.

[38] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ogga20D2HO-.

[39] JianFei Zhang and YuChen Jiang. A data augmentation method for vertical federated learning. *Wireless Communications and Mobile Computing*, 2022:1–16, 01 2022. doi: 10.1155/2022/6596925.

[40] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated Learning with Non-IID Data. *arXiv*, jun 2018. URL http://arxiv.org/abs/1806.00582.

[41] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey, 2021.