FedEx: An Exploration of Clustering Clients in Federated Learning

Carly Miles[1], Asad Khan[1], Louis Baranger[1], Matt Miller[1], Abhishek Saxena[1], Aarya Kulshrestha[1]

[1]University of Michigan

{carmiles, asadk, louisbar, mattrm, abhisa, akulshre}@umich.edu

Abstract

With the amount of computational power and data that exist in the pockets of people with smartphones around the world, machine learning models could be trained to higher accuracies, on a wider variety of problems, and with less environmental impact than through the use of large data centers for training. The question is: How do we harness this potential? Researchers McMahan et al. wrote the paper *Communication-Efficient Learning of Deep Networks from Decentralized Data* in an attempt to answer this question. In the paper, they propose an approach to Federated Learning, where a central server communicates with a group of clients to train a model. All client data remain private on their devices, model training occurs only on client devices, and expensive communication between the server and clients is limited. In this paper, we replicate the results of the MNIST experiment as described in the original paper by McMahan et al. (McMahan et al. 2023) on two different arrangements of data. We show that the algorithm proposed in their paper achieves high accuracy on distributed MNIST data classification. Additionally, we propose a novel algorithm that we structure to outperform the original algorithm on client data drawn dependently and/or from different distributions. This algorithm, called Federated Expert (FedEx), is structured to take advantage of clusters of clients that might have similar data and train expert classifiers on those clusters. However, initial experiments show that this extended algorithm does not improve performance on this type of data. Without further investigation, we cannot be certain of why this is, but analysis shows that this lack of performance is likely due to a flaw in the structure of the algorithm we use to combine the outputs of expert classifiers. Further work could be in restructuring the FedEx algorithm to optimize for accuracy as opposed to high prediction confidence, and testing our algorithm on real-world data.

Introduction

Personal devices, such as smartphones and tablets, are ubiquitous around the world today. These devices are used by billions of people every day and contain mountains of rich data and computational power that are inherently private to the owner of the device and cannot be accessed outside of the device. These data have the potential to be used for the training of machine learning models to enhance user experience and usability on these devices, and for applications in healthcare, security, and many other fields. Due to the private nature of these data, traditional methods of machine learning are not viable for using these data because they involve direct access to data on private mobile devices. This would be a clear breach of privacy for the user of the devices.

Federated Learning In their paper titled *Communication-Efficient Learning of Deep Networks from Decentralized Data*, McMahan et al. (McMahan et al. 2023) seek to solve this problem by using a decentralized machine learning approach. In this approach, devices collaboratively train a Machine Learning model without ever sharing raw data with a central server. Instead, each device independently updates a global model by contributing local model adjustments, which are aggregated on a central server. This preserves user privacy, as raw data does not need to be shared (McMahan et al. 2023).

We replicate and investigate the learning technique proposed by Federated Learning by re-creating results from the paper *Communication-Efficient Learning of Deep Networks from Decentralized Data.* The key result of the McMahan et al. (McMahan et al. 2023) study is the introduction of the FederatedAveraging algorithm which allows devices to train models locally while contributing to the learning of a larger model and maintaining data privacy.

In order to approach the data Federated Learning problem, we must take into account some key constraints. The first is that data found on mobile devices are inherently not independently sampled and identically distributed (non-IID), meaning that the data on individual devices may represent distinct distributions with strong within-device correlations and weak or no correlations across devices. This is because each device is most likely used by different individuals, so the data contained on these devices are going to differ from device to device (Kairouz et al. 2021). The next assumption made about problem is that client dataset sizes are unbalanced from one another due to heavier use on some devices over others. The solution to this as proposed by the original paper is to assign weights proportional to the dataset size on each device that will be utilized in the FedAvg algorithm. The final and most important constraint that we consider is the necessity to limit communication between the central server and client devices in a Federated Learning environment. Due to the nature of mobile devices being offline for a large portion of the time and running into other issues like limited battery and network bandwidth, updates to the client model are held up by communication costs between the central server and the client devices. Therefore, we prioritize heavy computation on devices over an increased number of communication rounds to minimize overall time to convergence.

Open Problems in Federated Learning The study *Advances and Open Problems in Federated Learning* identifies non-IID data as one of the possible sources of drawbacks in efficiency and effectiveness for Federated Learning. Because "the most common sources of dependence and non-identicalness are due to each client corresponding to a particular user, a particular geographical location, and/or a particular time window" (Kairouz et al. 2021), these differences in client devices are unpredictable and hard to simulate. Several types of non-identical distributions are expected to be found in a Federated Learning setup including *feature distribution skew*, *label distribution skew*, *same label different features*, *same feature different labels*, and *quantity skew* (Kairouz et al. 2021). We expect the two most prevalent sources of non-IID distributions in Federated Learning to be feature distribution skew and quantity skew due to the constraints identified in the Federated Learning setup.

We decided to focus on robustness to non-IID data as our extension to the original study because we found this to be the most relevant issue in the current architecture of Federated Learning. The proposed experimental setup in the original study made several assumptions in the experimental setup used to test the Federated Learning algorithm. The first of those assumptions is that the non-IID setup tested in the experiments in the original paper is representative of the non-IID setup in a real-world setting. We believe that the datasets used in the experimental setup, while technically non-IID, is different from the non-IID found that we would expect to find in a real world setting. Although the Federated Learning algorithm may achieve high test accuracies to the non-IID split of data from the MNIST and CIFAR-10 datasets, this will not translate as well to a real-world setting. Another issue that this experimental setup overlooks is the presence of a bad actor in the experimental setup. In a real-world setting, it is likely that there will be outlier clients with highly different datasets from any other client. It is expected that an outlier client will send back weights to the global model that can be opposite the other client weights sent back to the global model. This means that in the Federated Averaging algorithm these weights can cancel out other client weights sent back, resulting in a harmful update to the global model. For our extension, we propose a possible architecture that will prevent some of the drawbacks of having a non-IID dataset.

FedEx Specifically, we propose an extension to the original FedAvg algorithm that we hypothesize will improve the performance of the global model on non-IID data. Instead of sending all clients the same global model, we explore an approach that performs the FedAvg algorithm multiple times on subsets of the clients, and then combines the outputs of the resulting models. We chose this approach because training models on clients with similar data allows the individual models to become highly effective at a specific task, and their outputs can be efficiently combined with those of other "expert" models. After performing a small set of experiments, initial results are inconclusive that this approach is effective. Specifically, we highlight the weaknesses of the algorithm we use to combine expert model outputs as it reinforces mistakes made by the models.

Related Work

Several key studies analyze the non-IID nature of the datasets used in a Federated Learning setting. The first of which, *Advances and Open Problems in Federated Learning,* we discuss above. This paper introduces the types of non-IID data that can be found on client devices, and introduces possible strategies for dealing with non-IID data in a Federated Learning problem. This paper then goes into a possible multi-model approach for a Federated Learning problem, deviating from only having one global model on the central server. This multi-model approach was influential to the architecture of the FedEx algorithm we developed for our extension because it introduces the idea of combining the output of models trained on different, specific tasks in order to produce more accurate final output with potentially less computation. This work helped lead us towards developing our clustering extension, where expert models are trained on clients with similar data, whose outputs are then combined using a final online layer.

Another key related work that is similar to the path that we took for our extension is the study *Federated Learning on Non-IID Data: A Survey* by Zhu et al. (Zhu et al. 2021). This paper is very relevant to our extension because it is a survey of non-IID data in Federated Learning, discussing the different categories of non-IID data in Federated Learning as possible approaches to handling non-IID data. The portion of this approach that is most relevant to our extension is the identification of a clustering algorithm that could add robustness to non-IID data. They propose several clustering algorithms in the paper, the one most similar to ours uses a similarity metric to cluster different clients similar to how we do. At the time of developing our extension, we were unaware of this study, but this study introduces clustering based on cosine similarity, which is the same metric that we used for the clustering portion of our extension. While this paper mentions clustering in Federated Learning, they never implement any form of clustering at all and is purely theoretical.

The paper most relevant to our extension is *Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization Under Privacy Constraints* by Sattler et al. (Sattler, Müller, and Samek 2019) . This paper is also an extension of the original paper by McMahan et al. (McMahan et al. 2023). Just like the previous paper we discussed, we were unaware of it when developing our replication and our extension. This paper introduces a clustering framework that uses cosine similarity almost exactly the same as the framework for our clustering algorithm. The key difference between the clustering algorithm discussed by is that their algorithm is adapted to decide on a number of clusters dynamically based on several metrics, and uses the outputs from the clusters differently than we do in our extension. This approach is more refined than our approach and would be a good starting point for future work. The main distinction between this approach and our clustering algorithm is the online layer we developed to handle the outputs of each cluster.

Background

Datasets: The datasets we are using to model client data are the MNIST and CIFAR10. The MNIST dataset holds about 60,000 images of handwritten digits 0 through 9. It holds equal amounts of images per digit and is easily accessible through the PyTorch library. It is the primary source of information that was tested throughout the project, and we made use of its entirety through a training, validation, and testing split. It works well to test image recognition models.

CIFAR10 contains a different image classification task. CIFAR10 holds images from 10 different classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each class has 6,000 images for a total of 60,000 images. The CIFAR dataset was used mainly to create noise in the clustering experiment, and not all of the available images were used. It is also easily accessible through the Pytorch Library.

Data Distribution: The Federated Learning algorithm hinges on modeling real-world clients and the select data on each client device. For a classification task with data separation, the way we distribute data is extremely specific. There are two methods to model this specific task:

- **Independent and Identically Distributed Data (IID):** IID data represents a random distribution that is completely independent of one another. Each client in the model will, on average, have a robust snapshot of the diversity of the which ensures that the clients have a similar distribution of data.
- **Non-Independent and Identically Distributed Data (Non-IID):** This refers to a client simulation in which data cannot be considered statistically independent from others. There is some correlation between each selected data point and another. This idea is used to represent the characteristics of real-world data, as devices usually do not include examples from the entirety of the distribution but rather a few specific instances.

Federated Learning: Federated Learning is a proposed machine learning technique for a federation of distributed devices (clients) to aggregate prediction data onto a central server while maintaining client anonymity.

- **FedAvg:** Algorithm from McMahan et al. (McMahan et al. 2023) to implement Federated Learning by creating a global server model by averaging weights received from distributed client models.
- **FedEx:** Proposed algorithm as an extension to FedAvg, which, before aggregation, clusters similar clients as an attempt to perform better on Non-IID prediction tasks.
- **E:** Number of Epochs client models are trained with.
- **B:** Batch Size of images that are trained for each client model.
- **C:** Communication Rounds where weights from client models are reported and incorporated into central server.
- **G:** Number of clusters of similar clients that will be created within the FedEx algorithm.
- **Client Ratio:** Ratio of total clients that will be trained per communication round.
- **Test Accuracy:** Accuracy of the global server model on the prediction task of test split data from the current dataset, calculated after each communication round.

Methodology, Results, and Discussion

Replication Methods

In the original Federated Learning paper, the authors found that they were able to use the Federated Averaging algorithm to employ distributed training of machine learning models on different tasks, such as digit classification with MNIST data, while achieving similar performance to machine learning models trained on centralized data (McMahan et al. 2023). We are interested in replicating the algorithm proposed in that paper, Federated Averaging (FedAvg), to learn a classification model using the MNIST dataset. We hope to explore whether we can produce results similar to those presented in the paper that are competitive with model performance on centralized data. Because the original paper claims that the original FedAvg algorithm is robust to non-IID data, we will also replicate the simulation run in the original paper to analyze this hypothesis. In our paper, we will replicate FedAvg by simulating devices through a multithreaded program. Each thread will be in charge of coordinating the devices and aggregating the weights, and 100 threads will represent clients running independently on their own datasets. For the local training of each client, the same CNN architecture will be used as the paper, specifically it will be a "CNN with two 5x5 convolution layers (the first with 32 channels, the second with 64, each followed with 2x2 max pooling), a fully connected layer with 512 units and ReLu activation, and a final softmax output layer (1,663,370 total parameters)" (McMahan et al. 2023). The allocation of data to each client is based on the data allocation described in the original paper, for both the IID and non-IID case. For IID experiments, each client receives 600 training examples and the data is randomly allocated among all 100 clients. For non-IID, the data each client will receive will be distributed using the same strategy as the McMahan et al. paper (McMahan et al. 2023).

The initial step in replicating the Federated Averaging algorithm is modeling client data. There are two techniques to split the data amongst the devices: IID and non-IID, which are outlined in the core paper. For IID, splitting the data involves creating a random permutation of all images in the train split of the dataset and assigning 600 random images to each client. This way all images are chosen independently from the same distribution to create a balanced representation across all clients. As for non-IID, the technique begins by creating shards of 300 images with the same true label (i.e. the same digit). These shards are then shuffled, and two shards are randomly assigned to each client (i.e. shards of digits 1 and 7). Through this process, we are able to simulate clients with a limited scope of data which are not representative of the entire distribution, similar to the scope of real-world devices.

Using this data assignment, we run our experiments using the FedAvg algorithm as described in the paper. The algorithm, as shown below, trains a global model by executing a certain number of client-server communication rounds. This hyperparameter is specified by the variable C, and was frequently changed throughout the project to check algorithm efficiency. During each communication round, a number of clients of the total devices based on the client ratio are randomly selected to participate in a server update. The server sends its current global model to each client, and each device trains the model locally with specified E number of epochs, and B batch size. After all clients finish training, they each communicate the weights update back to the global server. The server then averages the updates from all active devices in the communication rounds and incorporates the change into its global model. We repeat this process until convergence, or until our target C communication rounds are met. For more detail, reference the pseudocode in the Appendix Algorithm 1.

Replication Analysis

The experimental results from McMahan et al. (McMahan et al. 2023) paper on FedAvg based on different configuration of batch size and epoch for IID and non-IID data is seen in Figure 1 in the Appendix and our experimental results replicating the experimental results from the original paper is seen in the Appendix Tables 1 and 2. McMahan et al. ran the experiment for 1000 communication rounds while we chose to run the experiment for 200 rounds due to limited computational resources (McMahan et al. 2023).

For IID data, our results in Figure 1 in the Appendix were consistent with the results produced by McMahan et al. in Figure 2 in *Communication-Efficient Learning of Deep Networks from Decentralized Data* based on test accuracies (McMahan et al. 2023). This chart shows a clear relationship between adjustments in batch size and number of epochs and the number of rounds it takes to converge on a test accuracy. While we were only able to train for 200 communication rounds, we can still see convergence to a maximum test accuracy for all splits of batch size (B) and epochs (E) except for $B = \infty$ and $E = 1$. Given more rounds of training, we expect that it would converge at the same rate as Figure 2 in the McMahan et al. study (McMahan et al. 2023). Looking closely at the chart, we can see that with $B = \infty$ all sizes of E take the most number of rounds to converge when compared with the other batch sizes. $B = 50$ and $B = 10$ produced similar results, but we noticed slightly lower accuracies and time to convergence for all splits of $B = 50$. From this chart, we are able to see that $B = 10$ produces the highest accuracy and least number of communication rounds to convergence, and as the number of epochs (E) goes up the test accuracy goes up and the number of rounds to convergence goes down.

For non-IID data, the results show a clear relationship between the batch size and convergence of the testing accuracies (Appendix Fig. 1b). When the model is trained on lower batch sizes ($B = 10$), it achieves higher accuracy across all epochs ($E = 1$, 5, 20) and converges faster than those trained with larger batch sizes. This is because the smaller batch size is able to provide more frequent parameter updates and improve generalization to the unseen data. On the other hand, when we used the entire training dataset as part of a batch ($B = \infty$), we noticed lower accuracy and greater fluctuations across all epochs. This instability is due to fewer, more drastic updates that fail to generalize across diverse local data distributions. This is particularly problematic with the non-IID data, where large batches exacerbate the differences between local models and the global model. Overall, the experiment demonstrates that smaller batch sizes with more epochs ($B = 10$, $E = 20$) offer the best trade-off between convergence speed, communication efficiency, and model stability in non-IID federated learning scenarios. This aligns similarly with what McMahan et al. also found when running the experiment with higher communication rounds (McMahan et al. 2023). In addition, when we compare results of the same configurations to the IID counterpart, we see a much higher level of variation in test accuracies throughout communication rounds. This must be because the non-IID data distribution affects model training and makes it more difficult to extract generalizable, learnable features from skewed data. This results in the instability seen in non-IID simulation compared to the more stable IID results.

Extensions

As highlighted in Kairouz et al. (Kairouz et al. 2021), the FedAvg algorithm has weak support for why and how it performs well on non-IID data. McMahan et al. (McMahan et al. 2023) claim its robustness to a specific case of non-IID data, and we hope to improve upon that robustness by training expert models on clusters of clients with similar local model weights, and then by adding an online layer to gate which model outputs are included in the final prediction, and with what confidence. The intuition behind this decision is based on our definition of non-IID. In this paper, we simulate non-IID data within the MNIST dataset by assigning two shards of 300 samples of data to each client, where a shard is a set of data that all have the same true label. In other words, each client in our non-IID simulation has data that maps to one of two true labels. With this in mind, we hypothesize that models trained on more specific tasks (i.e. trained to predict between one of a smaller set of labels) will have higher accuracy on that task. In combination with an online layer that learns to weight the models in a way that maximizes confidence in the final prediction, we hope that this method improves the algorithm's robustness on the type of non-IID data we are exploring. In order to determine the optimal clusters of clients while maintaining the restrictions of the original federated learning paper, we looked at the weights sent from the clients after one communication round. The restrictions of the problem require that the central server never accesses client data, so clustering the clients directly based on the similarity of their data is not an option in this context. Instead, we rely on the fact that local client weights are directly related to the data the model is trained on and all clients use the same algorithm to train their models, so clients with similar weights are likely to also have similar data. Therefore, in order to estimate which clusters of clients are most representative of clusters that might exist in actual client data, we run spectral clustering on the local model weights sent from the clients after running one FedAvg communication round. The number of clusters is a hyperparameter, G. After determining the cluster assignments, we train G expert models using the FedAvg algorithm such that each of the expert models communicate with only the clients in their corresponding cluster. Each of the clients in the clusters sends local weights to their corresponding expert models after each of their expert model's communication rounds, after which the expert model updates its weights using the FedAvg algorithm, until convergence is reached. After all G expert models have converged, the next challenge is performing inference. In order to combine the output of the expert models for inference and ensure that the global model's predictions have consistently high confidence, we propose a Federated Expert online layer, FedEx. On inference, the FedEx layer calculates the final prediction by taking the weighted average of the outputs of each of the expert models using the current weights. Then, it updates the weights based on the relative confidence of each of the expert models' predictions for the output label calculated using the previous round's weights. The resulting weight updates are a function of the output of each model and the current timestamp. See the full FedEx algorithm in the appendix under ALGORITHM 2.

Our extension focuses on improving the performance of the original FedAvg algorithm on a specific case of non-IID data as presented in the original paper. We hypothesized that by training expert models on clusters of clients with similar data and then combining the outputs of those models in a way that maximizes the confidence of our final prediction, we could improve the performance of the algorithm on non-IID data. After running experiments, we saw a noticeable decline in accuracy and loss when using our clustering and FedEx algorithms. As shown in Table 1, the clustering and FedEx approach achieved a maximal accuracy of 77.98% on the non-IID data, while FedAvg achieved an accuracy of 97.43% for the non-IID data. We did see evidence that clients with similar data were getting assigned to the same cluster, as shown in Figure 2 in the appendix.

Although our extension approach did not perform as anticipated on the non-IID data mentioned from our core paper, we set out to perform another experiment of non-IID data to further verify our approach. In addition to the non-IID approach defined earlier we introduce a data distribution to add noise. In this experiment, we define noise as adding 10,000 samples of data from the CIFAR-10 dataset to our training data and assigning 15% of clients in our experiment the CIFAR-10 data. When assigning CIFAR-10 data to the clients, we will utilize the non-IID definition previously defined - each client will be assigned two shards, where each shard is a set of data that has the same true label. Our experiment will still maintain 100 clients and use all 60,000 data samples from the MNIST dataset. Consequently, with the addition of noise there will be more data spread across each client, and we address this by increasing the shard size from 300 to 350 for each client.

The CIFAR-10 dataset has 10 classes similar to the MNIST dataset; however, both MNIST and the CIFAR-10 dataset will map to class labels 0-9. This means that even though class 0 in CIFAR-10 maps to airplane and class 0 in MNIST maps to the digit 0, the global and local models will be associating both inputs from the different datasets as class 0 in its model. Our proposed logic for this is that in real non-IID data settings two different clients might contain very different data, but might still map to the same output. Additionally, as described in the introduction, there might be bad actors and outliers in a federated learning environment, so the noise proposed in this paper does a better job in assessing the performance of FedAvg and the robustness to our extension compared to the experimental setup in the original paper. Our hypothesis is that FedAvg will not perform as well as our proposed extension on noisy data because it will attempt to average weights from distributions that are extremely different from each other. However, if a solution like clustering and FedEx is proposed, then the clustering algorithm will cluster similar clients together, (i.e MNIST client cluster(s) and CIFAR-10 client cluster(s)), such that each cluster is a model expert in its own distribution allowing for the FedEx to gate the cluster experts correctly to arise at an informed prediction. We hypothesize that our noise approach will work better than FedAvg despite receiving unpromising results from our extension previously because we believe that our noise added will make the data distribution significantly more different than before causing FedAvg to break and for our extension to succeed.

We ran experiments on the noisy data with the clustering and FedEx algorithms proposed, adjusting the hyperparameter G and compared the results to FedAvg on loss and accuracy. After running experiments, similar to beforehand, we saw a noticeable decline in performance when using our clustering and FedEx algorithms compared to FedAvg using the noisy data defined earlier. As shown in Table 2 in the appendix, the clustering and FedEx approach achieved a maximal accuracy of 80.20% on the noisy non-IID data, while FedAvg achieved an accuracy of 84.72% for the noisy non-IID data. We visualize how all the clients that train on CIFAR-10 data are assigned to the same cluster using our algorithm under Figure 3. In previous research, CNNs have shown to be very effective at classification tasks on MNIST data ($> 99\%$ accuracy), and with more time we would perform experiments to verify expert model performance and ensure that these models are performing well on the labels that consistently appear in the datasets of their corresponding clients (Chen et al. 2018). Using the process of elimination, this highlights the point that if our clustering algorithm is effectively clustering clients with similar data, it is most likely our FedEx algorithm that combines model output which is the bottleneck to our performance in handling non-IID data. With this in mind, and after analysis of the algorithm, we hypothesize that the reason the FedEx online layer is not improving performance and predicting the most accurate label from each of the expert models is because it is optimized to maximize confidence in each prediction, not accuracy. Our simulated data, where each cluster of clients all have relatively similar MNIST data with similar features, is not a situation in which the current version of FedEx will perform the best. Instead, we infer that vastly different clusters of clients that each train expert models with very high accuracy would lead to better performance by FedEx. Future work would run experiments to verify this hypothesis and simulate data with clusters that FedEx should perform better on, such as language tasks in different languages or images with color-based features and shape-based features.

Conclusions

The findings from our replication closely mirrored those established in McMahan's original paper, demonstrating the enduring effectiveness of the Federated Averaging (FedAvg) algorithm since its inception. Our experiments achieved a consistent 99% accuracy on the MNIST dataset, reinforcing the algorithm's reliability. One of our key areas of focus was the algorithm's robustness on non-IID data, where we conducted an experiment aimed at improving performance. However, the results of our clustering experiment did not align with our hypothesis; rather than improving accuracy, the results only decreased, making us conclude that our approach did not offer a performance boost. Specifically, our clustering extension received a capped accuracy of 77.98% on the MNIST non-IID data when FedAvg received a 97.43% accuracy on the same data. In addition, our clustering extension achieved a maximal accuracy of 80.20% on non-IID data with CIFAR-10 noise while FedAvg achieved a baseline accuracy of 84.72%. We see consistent decline in performance with our clustering extension even when we see that our clustering algorithm clusters similar clients together. Therefore, we hypothesize that the reason for the lack of performance boost with the clustering extension is because the FedEx online layer is built to maximize confidence in each prediction, as opposed to accuracy. So, mistakes made with high confidence by expert models are reinforced by FedEx. Looking ahead, we see potential in exploring several potential improvements. In addition to testing the model on larger and more complex datasets, one area of interest is experimenting with delayed stochastic gradient descent combined with a reinforcement learning gating mechanism that takes true labels into account. This would address the current issue we hypothesize with the FedEx layer proposed in this paper: it only focuses on maximizing confidence, not accuracy. Another exciting direction is applying the FedAvg algorithm to a natural language processing (NLP) task, allowing us to assess its performance on prediction tasks outside of image recognition. Moreover, while the MNIST and CIFAR datasets we used are relatively simple, real-world distributed devices often involve more intricate data with more varied classes and detailed images. Further research is necessary to understand Federated Learning performance in more complex systems.

Societal Impact

Federated Learning has the potential to revolutionize industries by enabling decentralized, privacy-preserving machine learning across various sectors, including healthcare and finance, all while ensuring that sensitive data never leaves its source. This approach addresses some of the most pressing ethical concerns in AI, enabling the creation of broad, generalized models without compromising user privacy. However, the decentralized nature of Federated Learning introduces challenges related to bias and noise across clients. Ensuring that the algorithm learns from a diverse, representative set of data and does not disproportionately prioritize biased or noisy devices is crucial to avoid skewing the global model. The fairness and equity of the model aggregation process are vital to maintain the objectivity of distributed learning algorithms and prevent unintended biases in the final model.

References

Chen, F.; Chen, N.; Mao, H.; and Hu, H. 2018. Assessing Four Neural Networks on Handwritten Digit Recognition Dataset (MNIST). arXiv:1811.08278.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; D'Oliveira, R. G. L.; Eichner, H.; Rouayheb, S. E.; Evans, D.; Gardner, J.; Garrett, Z.; Gascón, A.; Ghazi, B.; Gibbons, P. B.; Gruteser, M.; Harchaoui, Z.; He, C.; He, L.; Huo, Z.; Hutchinson, B.; Hsu, J.; Jaggi, M.; Javidi, T.; Joshi, G.; Khodak, M.; Konečný, J.; Korolova, A.; Koushanfar, F.; Koyejo, S.; Lepoint, T.; Liu, Y.; Mittal, P.; Mohri, M.; Nock, R.; Özgür, A.; Pagh, R.; Raykova, M.; Qi, H.; Ramage, D.; Raskar, R.; Song, D.; Song, W.; Stich, S. U.; Sun, Z.; Suresh, A. T.; Tramèr, F.; Vepakomma, P.; Wang, J.; Xiong, L.; Xu, Z.; Yang, Q.; Yu, F. X.; Yu, H.; and Zhao, S. 2021. Advances and Open Problems in Federated Learning. arXiv:1912.04977.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2023. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:1602.05629.

Sattler, F.; Müller, K.-R.; and Samek, W. 2019. Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization under Privacy Constraints. arXiv:1910.01991.

Zhu, H.; Xu, J.; Liu, S.; and Jin, Y. 2021. Federated Learning on Non-IID Data: A Survey. arXiv:2106.06843.

Individual Contributions

- **Carly:** I contributed to the development of the FedEx online layer and clustering code. I also wrote the overview and analysis of the extensions, and I wrote the algorithms used in the Appendix. Throughout the project I provided planning, debugging, analysis, and timeline support.
- **Abhishek**: The main area I worked on was creating the dataloaders to simulate IID and Non-IID data. In addition, I worked on designing and coding the clustering algorithm, specifically how to find the clusters themselves. I also ran the tests to visualize Non-IID performance with the FedAvg (Fig 1b). I also served as project manager to create/track deadlines and delegate tasks. For the paper, I worked on the replication methodology and analysis, as well as the conclusion.
- **Louis:** My main contributions to this project include the design of the clustering algorithm, the implementation of the multithreaded code, and the debugging of both logical and programming bugs. I designed the overall architecture behind FedEx, which was refined over time through group conversations during our meetings. The multithreaded program follows a standard manager-client architecture and leverages the threading library to ensure safe concurrency.
- **Aarya:** I contributed to the project by developing data loaders and implementing the replication code for the FedAvg algorithm. I coded and executed the experiments comparing communication rounds and test accuracy across various batch size and epoch combinations in Non-IID settings with Abhi. Additionally, I collaborated with the team to debug algorithmic and cluster-related issues. I also helped with coding and running extension experiments to explore alternative configurations and improve model performance. I helped with the replication methodology and analysis and proofreading the paper.
- **Matt:** I contributed to the project by helping with the development of the replication of the project helping setup and develop the training for Federated Learning and Federated Averaging algorithm. I helped code and run the experiments to replicate the IID chart for MNIST data in figure 2 with Asad. Additionaly, I collaborated with the team to help with parts of replication and extension that needed help on. Lastly, I contributed to Introduction and methodology of the paper.
- **Asad**: I collaborated in developing the replication code for FedAvg. Additionally, Matt and I coded and ran the replication experiment of FedAvg ran on IID data. In addition, I introduced adding noise in the form of CIFAR-10 data to our clustering extension and ran experiments to how it performed to baseline FedAvg. Furthermore, I contributed in writing the methodology section and creating figures for the paper.

Appendix

Algorithm 1

Algorithm 1: FEDERATEDAVERAGING

The K clients are indexed by k. B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
1: Server executes:
2: Initialize $w_0$
3: for each round $t = 1, 2, \ldots$ do
4:     $m \leftarrow \max(C \cdot K, 1)$
5:     $S_t \leftarrow$ (random set of $m$ clients)
6:     for each client $k \in S_t$ in parallel do
7:         $w_{t+1}^k \leftarrow$ ClientUpdate$(k, w_t)$
8:     end for
9:     $m_t \leftarrow \sum_{k \in S_t} n_k$
10:    $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
11: end for

12: ClientUpdate$(k, w)$: Run on client $k$
13: $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
14: for each local epoch $i$ from 1 to $E$ do
15:     for each batch $b \in \mathcal{B}$ do
16:         $w \leftarrow w - \eta \nabla \ell(w; b)$
17:     end for
18: end for
19: return $w$ to server
```

Algorithm 2

Definitions:

- T = Number of inferences (timestamps)
- G = Number of expert models
- D = Dimension of expert model output (number of classes for classification)
- $w_{ij}^{(t)}$ = Weight for expert model i on class j at timestamp t
- $\bar{m}^{(1)}, \ldots, \bar{m}^{(G)}$ = Outputs of expert models 1 through G
- $c^{(i)}$ = Confidence of expert model i

Algorithm 2: FEDEX WEIGHT UPDATE ALGORITHM

```
1: for $t = 1$ to $T$ do
2:     $p \leftarrow \arg\max_i \sum_{j=1}^{D} w_{ij}^{(t-1)} \bar{m}_j^{(i)}$
3:     for $i = 1$ to $G$ do
4:         $c^{(i)} \leftarrow \frac{\bar{m}_p^{(i)}}{\sum_{j=1}^{G} \bar{m}_p^{(j)}}$
5:     end for
6:     for $i = 1$ to $G$ do
7:         $w_{ip}^{(t)} \leftarrow \frac{w_{ip}^{(t-1)} \cdot t + c^{(i)}}{t+1}$
8:     end for
9: end for
```

Experiment Type	**Best Loss**	**Accuracy**
Baseline	1.48	97.43%
Cluster 1 (3 Clusters)	1.77	71.31%
Cluster 2 (3 Clusters)	1.75	72.11%
Cluster 3 (3 Clusters)	1.72	74.78%
Total (3 Clusters)		77.98%
Cluster 0 (4 Clusters)	N/A	N/A
Cluster 1 (4 Clusters)	1.93	52.91%
Cluster 2 (4 Clusters)	1.82	52.91%
Cluster 3 (4 Clusters)	1.82	64.12%
Total (4 Clusters)		9.01%

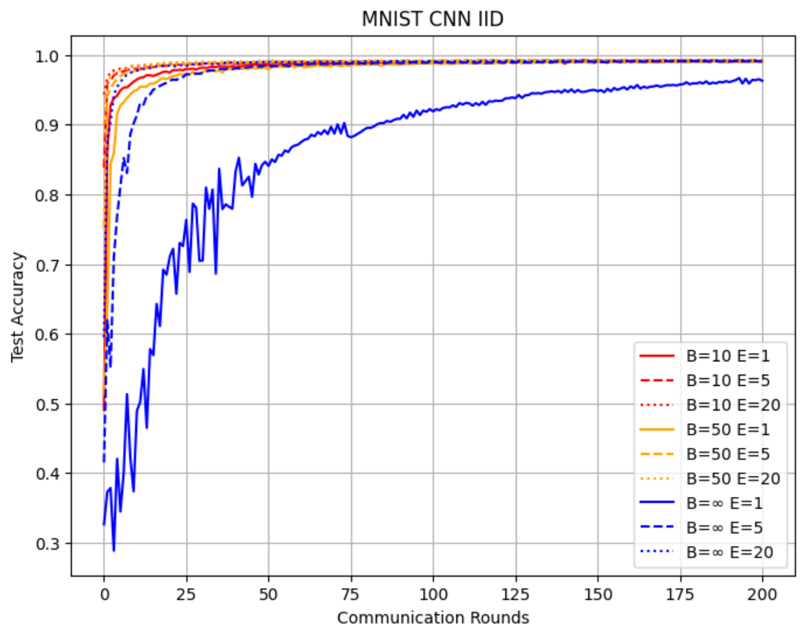
Table 1: Experiment with non-IID data (no noise).

Experiment Type	**Accuracy**
Baseline	84.72%
Cluster 0 (3 Clusters)	76.33%
Cluster 1 (3 Clusters)	40.04%
Cluster 2 (3 Clusters)	52.11%
Total	80.20%
Cluster 0 (4 Clusters)	54.52%
Cluster 1 (4 Clusters)	48.57%
Cluster 2 (3 Clusters)	45.82%
Cluster 3 (3 Clusters)	40.17%
Total	63.88%

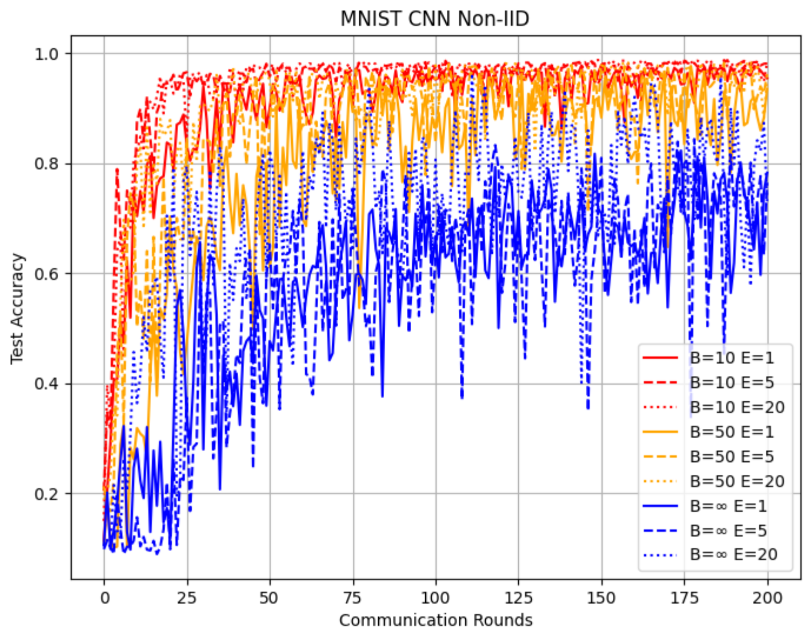
Table 2: Experiment with non-IID data (noise).

Tables

Figures

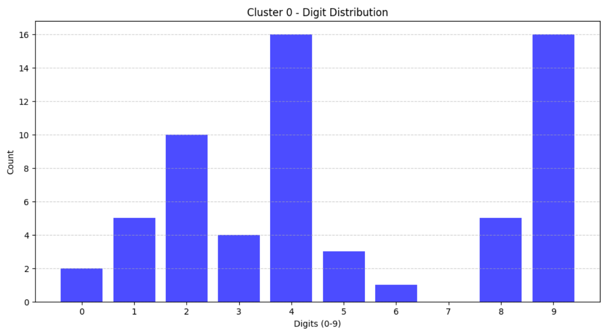


(a) FedAvg Test Accuracy vs. Comm Rounds on IID data

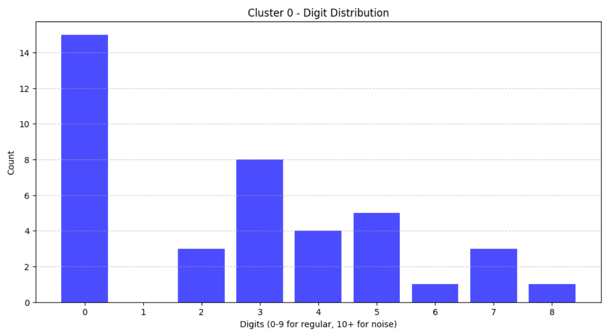


(b) FedAvg Test Accuracy vs. Comm Rounds on Non-IID data

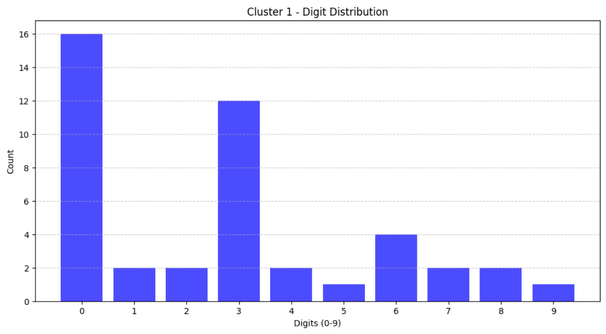
Figure 1: FedAvg Hyperparameter Tests



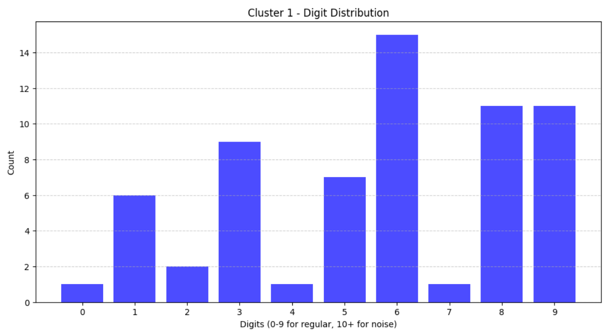
(a) Cluster 0 Digit Frequencies



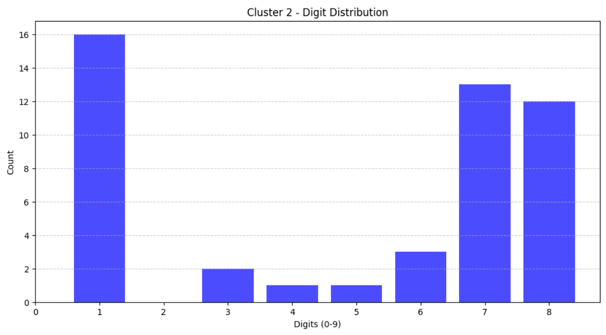
(a) Noisy Cluster 0 Digit Frequencies



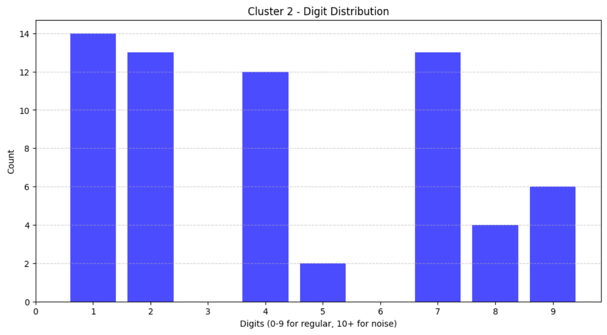
(b) Cluster 1 Digit Frequencies



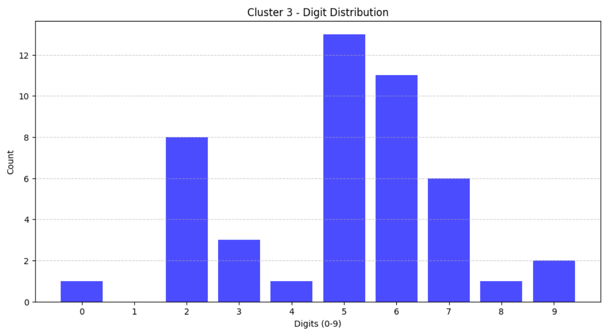
(b) Noisy Cluster 1 Digit Frequencies



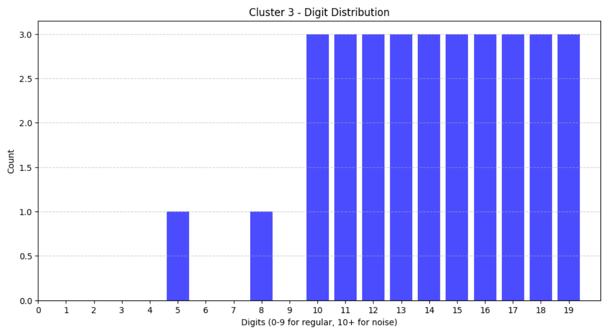
(c) Cluster 2 Digit Frequencies



(c) Noisy Cluster 2 Digit Frequencies



(d) Cluster 3 Digit Frequencies



(d) Noisy Cluster 3 Digit Frequencies

Figure 2: FedEx Cluster Visualizations by Digit

Figure 3: FedEx with Noise Cluster Visualizations by Digit