A Fairness and Performance Evaluation of Federated Aggregation Strategies Subject to Heterogeneous Client data and Differential Privacy Constraints

Adarsh Mital Caleb Berman

Abstract

Federated Learning's popularity has grown in recent years due to its privacy preserving benefits and its natural extension to mobile edge computation. One of the fundamental components that can differ amongst different Federated Learning approaches in a parameter server architecture is how the server averages each of the client model gradients. We study how different gradient averaging methods (FedAvg, FedCostWAvg, Q-FedAvg) alter the accuracy, wall-clock time, and fairness of a CNN based Neural Network architecture for the task of handwritten digit recognition on a dataset partitioned by unique heterogeneous users. To match our approach to a real world Federated Learning context in which privacy is valued, we insert differential privacy into our models. Our results show that subject to this real world constraint, Federated Averaging achieves the highest test set accuracy and lowest test set accuracy variance after 50 training rounds for this task, while Q-Fair Federated Averaging achieves the smallest wall-clock training time and the lowest test set loss variance.

1. Introduction

Internet of things (IOT) devices make up an increasingly greater proportion of data sources and computing environments for machine learning tasks. Their everyday use in conjunction with the suite of sensors they are often equipped with allow them to capture vast amounts of personal data, much of which users hope to keep private. Yet, there are a growing number of Machine Learning based applications that can make deeply impactful and personalized predictions from this data. Thus, there is a heightened need for balancing individual user privacy with total utility when it comes to making use of this sensitive data.

Fleshed out most notably by McMahan et al., Federated Learning is a distributed learning approach that seeks to take advantage of these edge data collection and computing environments by sending a centralized model to an edge device where it can perform update steps on data and then send the updated weights back to the server where they are aggregated to a centralized model. In other words, Federated Learning allows user data to remain on the devices it was collected on while only sharing gradients between a centralized 'server' model and local 'client' model. This is a marked privacy advantage compared to centralized learning, where a general model is given access to all client data (though this can be anonymized, data leakage can still occur).

In a Federated Setting, four phenomena are commonly observed: Edge devices having Non-IID data, an unbalanced number of training examples among devices, the number of participating devices participating in training frequently outnumbering the number of training examples per client, and communication delays due to devices being offline. Non-IID data as well as an unequal numbers of training examples complicates the task of **how to aggregate the new models that each client returns to create the final global model**. Embedded in this task is an inherent trade-off between simply minimizing the aggregate client loss and ensuring a more 'fair' performance across all clients (Li et al., 2020). As a result, several client model aggregation methods have been proposed.

1.1. Federated Averaging

The first of these strategies proposed in the original work implementing federated learning is known as Federated Averaging, otherwise known as FedAvg. Federated Averaging seeks to minimize an overall global loss function, $f(w)$ that is a weighted average of the individual losses of each model trained on client K.

$$f(w) = \sum_{j=1}^{J} \frac{n_j}{N} F_j(w) \tag{1}$$

Where J is the number of clients to be sampled, n_j is the size of the data on client j's to be trained on, N is the total examples trained across all clients, and $F_j(w)$ is the loss function of client j.

The intention of Federated Averaging's client loss weighting scheme is such that individual client models that have seen

more training examples on their device should contribute a higher amount to the overall global loss function. To implement the minimization of this loss function, one takes a weighted average of each of the sampled model weights with the same weighting scheme as the total loss (see algorithm 1 for more detail).

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.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

1.2. Federated Cost Weighted Averaging

One limitation of Federated Averaging is that while it takes into account how many training examples each model has seen on a client, it doesn't take into account how much each model actually *learns* from those examples. This is the motivation behind Federated Cost Weighted Averaging (FedCostWAvg), an aggregation method recently developed by Machler et al. (Machler et al., 2021) who reported state of the art performance using this technique on a federated tumor image segmentation dataset.

In order to incorporate how much each client model learns during a training round, the global loss function includes a weighting term proportional to how much each individual client's loss function decreased. The overall global FedCostWAvg loss function is given by the following:

$$f(w) = \sum_{j=1}^{K} (\alpha \frac{n_j}{N} + (1 - \alpha) \frac{k_j}{K}) F_j(w) \tag{2}$$

Where $k_j = \frac{F_j(w_{t-1})}{F_j(w_t)}$ at fitting round t for each client j, and K is the total amount of loss change among the sampled clients given by: $K = \sum_{i=1}^{K} k_i$. Through the α parameter (bounded by 0 and 1) one can tune the degree with which the global loss depends on the reduction in loss or the amount of data the model as seen. To minimize this loss the same weighting scheme is applied to averaging the weights learned by each client model.

1.3. Q-Fair Federated Learning

While Federated averaging has been shown to be successful in training a global model that performs accurately on average, one essential aspect of federated learning that its objective does not take into account is data often being non-IID or heterogeneous amongst the edge devices it will be evaluating on. That is to say federated averaging can result in a shared global model that performs well in expectation but has high performance variance across clients. A model aggregation strategy known as Q-federated averaging seeks to create a global model that performs more fairly amongst the heterogeneous data it does inference on by introducing a new Federated Learning objective loss function (The q-FFL objective) that penalizes clients with higher loss functions more through minimizing the objective shown in equation 3.

$$f_q(w) = \sum_{j=1}^{J} \frac{p_j}{q+1} F_j^{q+1}(w) \tag{3}$$

Where q is a parameter to tune the amount of fairness we impose on the final centralized model and $p_k = \frac{n_k}{N}$ from the federated averaging objective.

With this federated loss scheme, the global loss is no longer simply a weighted average of the sampled client losses due to the $q+1$ exponent term. To address this Li et. al. proposed Q-Federated Averaging (Algorithm 2) where the weights are inferred from the upper bound of the local Lipschitz constants of the gradients of F_j^{q+1}.

Algorithm 2 q-`FedAvg`

Input: $K, E, T, q, 1/L, \eta, w^0, p_k, k = 1, \cdots, m$
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with prob. p_k)
  Server sends w^t to all selected devices
  Each selected device k updates w^t for E epochs of SGD on F_k with step-size η to obtain $\bar{w}_k^{t+1}$
  Each selected device k computes:

$$\Delta w_k^t = L(w^t - \bar{w}_k^{t+1})$$
$$\Delta_k^t = F_k^q(w^t) \Delta w_k^t$$
$$h_k^t = q F_k^{q-1}(w^t) \|\Delta w_k^t\|^2 + L F_k^q(w^t)$$

  Each selected device k sends Δ_k^t and h_k^t back to the server
  Server updates w^{t+1} as:

$$w^{t+1} = w^t - \frac{\sum_{k \in S_t} \Delta_k^t}{\sum_{k \in S_t} h_k^t}$$

end for

1.4. Differential Privacy

There has been a significant amount of work in Machine Learning research about the possibility of membership inference attacks in centralized settings. This is the capability of an adversary able to determine if a specific data point/user

is part of a training set. Though Federated Learning has an aforementioned privacy advantage to centralized data center training, Federated Learning tasks can still be prone to such inference attacks (Naseri et al., 2021), which can necessitate the use of Differential Privacy, a mechanism that provides a statistically bounded guarantee on how much influence a single user's data on the result of the algorithm.

Formally, if equation 4 is true for a mechanism M and two neighboring databases, D_1 and D_2 that differ in only a single record and for all possible output $S \subset Range(A)$, M provides (ϵ, δ) differential privacy.

$$P[M(D_1 \in A] \leq e^{\epsilon} P[M(D_2 \in A] + \delta \tag{4}$$

ϵ primarily controls the privacy/utility trade-off with higher values of ϵ generally indicating higher utility but lower privacy.

In Federated Learning settings, Differential Privacy is commonly instituted primarily in two different contexts: Local Differential Privacy (LDP) and Central Differential Privacy (CDP). In LDP, noise is added by each client before server side aggregation, while in CDP, noise is added after aggregation.

In this paper, we perform LDP with an SGD optimizer, with each client selected for training performing the following algorithm prior to aggregation. This algorithm is known as DP-SGD (Abadi et al., 2016).

Function DP−SGD (*Clipping norm S, dataset D, sampling probability p, noise magnitude σ, learning rate η, Iterations E, loss function $L(\theta(x), y)$*) **:**
  Initialize θ_0
  for *each local epoch i from 1 to E* **do**
    for *$(x, y) \in$ random batch from dataset D with probability p* **do**
      $g_i = \nabla_\theta L(\theta_i; (x, y))$
    end
    $Temp = \frac{1}{pD} \Sigma_{i \in batch} g_i min(1, \frac{S}{\|g_i\|_2}) + N(0, \sigma^2 I)$
    $\theta_{i+1} = \theta_i - \eta(Temp)$
  end
return θ_E

1.5. Purpose

In recent years, there has been literature that has evaluated federated aggregation algorithms primarily through the lens of maximizing total accuracy (be it global accuracy across all clients, or for more personalized models, accuracy on their own client test set)(Nilsson et al., 2018)(Ek et al., 2020) given a publicly available dataset, and a consistent model architecture. Accompanying this, some authors have opted to focus more on the 'fairness' aspect of Federated Learning, in the goal of ensuring more uniform performance across different clients (Li et al., 2020) (Divi et al., 2021). In the same time frame, researchers have tried to minimize overall training/test loss through re-imagining conventional LDP and CDP (Geyer et al., 2017) in Federated Learning and Differential Privacy settings.

In this paper, we hope to unify these tracks by evaluating both the fairness and global performance of various federated aggregation methods, with the added constraint of imbuing our models with differential privacy–to mirror a real world scenario. As far as our knowledge, this paper is one of the first evaluations into how differential privacy infused federated aggregation methods differ in utility and fairness. Additionally, this would be the first fairness related evaluation of the very recently proposed FedCostWAvg aggregation method.

2. Experimental Methods

2.1. Dataset

In order to perform experiments on a dataset representative of a real-world federated learning environment we opted to utilize the Federated Extended MNIST dataset (FEMNIST)(Cohen et al., 2017). The FEMNIST dataset is made up of handwritten digits 1 through 10 partitioned by the person who wrote the digits. This partition allows us to capture non-iid behavior typical of a federated learning set-up. Namely, in this case that although each user is writing the same digits, they should have differences in handwriting style, distribution of labels, and amount of labels available. We access the dataset through the TensorFlow Federated API where the images are already preprocessed into 28x28 Tensors of 1 color channel. The plots in figure 1 and figure 2 are produced as evidence that the dataset follows the desired properties for our federated learning simulations. For training and evaluation we use the test/train splitting already preprepared by the TensorFlow Federated package where random 10 examples for each client are held out for the test set. We then further split the test set for each client in half randomly devoting half of the images for validation and the other half for final testing metrics.

2.2. Model Architecture

All of our experiments, centralized or federated, utilize the same convolutional neural network (CNN) model architecture. This is adapted from the canonical PyTorch MNIST tutorial (exa, 2020), following the methodology of prior Federated Learning evaluation papers that employed a standard PyTorch/Tensorflow architecture (McMahan et al., 2017), and a Performance Evaluation of Federated Learning Algorithms. This architecture, implemented in TensorFlow along with all of our experiments, consists of two convolutional

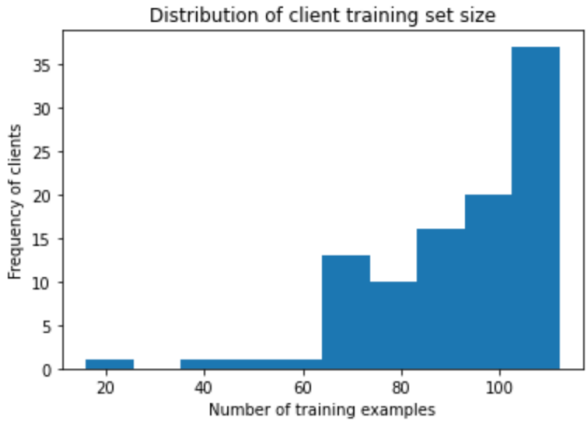

Figure 1. Distibution of training set size for the clients we ran our simulations on. The mean training set size was 91.04 with a standard deviation of 16.65. This plot evidences the unbalanced number of training examples of the dataset

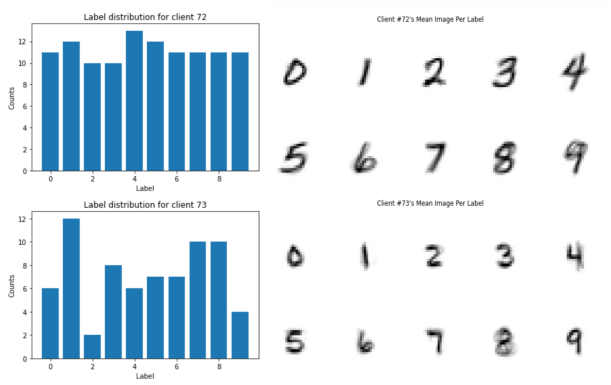

Figure 2. Distribution of label value and mean label image for two selected clients. The left plot evidences the unbalanced label distribution amongst the clients while the right shows how the distribution of each clients images is somewhat heterogeneous

layers, a 2D maxpool layer, one hidden linear layer, and one final output linear layer. For regularization we include two dropout layers at different stages of the CNN, one with a rate 0.25 and the other with 0.5. For all hidden layers we utlize a ReLu activation function.

2.3. Differential Privacy

To implement differential privacy, we make use of the TensorFlow-Privacy framework (TF, 2021). This provides a tested and widely used mechanism for injecting a model and accompanying optimizer with a specified amount of differential privacy. Namely, there are three parameters which control the amount of DP you inject in the training process: l2-norm clip (which is used to bound the gradients for updating weights), noise multiplier (the most indicative parameter in the amount of guaranteed privacy), and microbatches (which controls how many smaller batches each training batch is split into before clipping each microbatch gradient). We follow the official TensorFlow tutorial for implementing DP-SGD, which is instantiated for each client model in a federated setting, and for the main model in a Non-Distributed setting.

2.4. Non Distributed Model Benchmark

In order to benchmark the performance of our federated learning algorithms against a more conventional setting, we train a non-distributed model in which all the data is centrally hosted and trained upon. As SGD (Stochastic Gradient Descent) is one of the more commonly used optimizers in Federated Learning we utilize TensorFlow's SGD optimizer to maintain structural homogeneity.

To find hyperparameters that allow the Non-Distributed model to learn efficiently enough in order to be compared to our FL aggregation methods, we perform a k-fold cross validation grid search (with k =3) over training batch size and learning rate. This is also motivated by work that highlights that despite sharing the same training set and optimizer, a non-distributed setting may require different hyperparameters for optimal performance than a federated setting (Nilsson et al., 2018). We observe batch size of 32 and learning rate of 0.1 as best performing.

Table 1. Grid search Validation Set loss over batch size and learning rate (Noise = 0.2), after 15 Epochs

Batch size	Learning rate 0.001	Learning rate 0.01	Learning rate 0.1
8	2.6	1.43	0.46
16	2.28	1.55	0.37
32	2.29	1.9	**0.285**
64	2.29	2.18	0.31
128	2.29	2.213	0.437

We add Differential Privacy to our non-distributed model in the pursuit of structural similarity to the Federated setting via the aforementioned DP-SGD TensorFlow implementation. It is valuable to determine how the amount of noise (noise multiplier in TensorFlow Privacy) added affects our ability to learn given a relatively standard model architecture and non adaptive optimizer. We discover that from trying noise multiplier values of .2, .4, .6, .8, 1, and 1.2, the upper end of which was used as a noise value in the canonical TF tutorial, that .2 is the highest we can set our noise to ensure consistent learning across epochs. We use a l2-norm clip of 1 as well as a relatively computationally efficient mini batch number of 8.

Given a noise multiplier of .2, a batch size of 32, and a

learning rate of .15, we train on a 100 clients' aggregated data for 30 epochs and evaluate performance on an unseen test set from those clients.

2.5. Federated Learning Simulations

In order to implement the simulations of the several Federated averaging algorithms we utilize the Flower python framework (Beutel et al., 2021). This framework is chosen in particular due to it's ease of implementing new strategies of federated model averaging. For FedAvg and Q-FedAvg the Flower implementations are used. Since there is no existing FedCostWAvg implementation in flower we write our own using the framework. Due to computation limitations we simulate our federated learning experiments using a distributed network of 100 unique clients. Each client has the same model architecture outlined above locally stored such that it can send and receive weights to the server. Additionally each client's model is locally trained using DP-SGD to incorporate differential privacy.

During each training round we select 10 clients from the set of 100 randomly from a uniform distribution to be trained for 5 epochs each and then aggregated into the global model. After each training round we make two evaluative measurements of model performance. First we sample another 10 clients randomly from a uniform distribution and report their accuracy and loss on their local test set. Secondly, we construct an overall global test set consisting of an aggregation of all the 100 clients test sets and measure the global model performance on this set of examples. For all federated learning strategies we perform 50 training rounds. Additionally for each strategy we measure the training wall-clock time to complete the 50 rounds of training and evaluation. These times represent training using CPU due to lack of reliable access to GPUs to train on.

In our federated learning experiments we select the same optimal noise multiplier and the mini-batch size parameters for DP-SGD as for the non-distributed model to ensure consistency across our comparisons.

Following the methods in Li et al. (Li et al., 2020) to find optimal batch size and learning rate for the federated learning strategies we perform a grid search for these hyperparameters, tuning them to produce optimal performance on the global validation set after 25 training rounds of 10 clients being trained 5 epochs. We then use the same optimal learning rate and batch size for FedCostWAvg and Q-FedAvg. Through this method the optimal batch size is seen as 8 and optimal learning rate is 0.1.

For the alpha parameter in the FedCostWAvg strategy, we use a value of $\alpha = 0.5$ as it was suggested by Machler et al. to be the optimal value found in their experimentation. (Machler et al., 2021)

Lastly for selecting q, the fairness parameter in Q-FedAvg, we initially perform a grid search across the set of 0.0001, 0.001, 0.01, 0.1, and 1 and select the q that resulted in the lowest validation loss on the global validation set after 25 training rounds. At first we selected $q = 0.0001$ as it was the q that performed the best for this search. However after further investigation it was realized that the number of training rounds to convergence of performance scales with the value of q in this setting. Thus other q-fairness parameters could reach similar levels of validation loss performance, but require many more iterations to converge. To investigate the performance and fairness of strategies with a higher q value, we train an additional model with $q = 0.001$ for 100 training rounds such that it converges to a stable test-set accuracy value comparable to the optimal q found in our grid search.

To measure model fairness we measure the variance of the test set loss and accuracy across 10 uniform randomly sampled clients as performed in Li et al. In order to incorporate these measurements from a variety of clients, we average these values across the last 5 rounds of training for each federated aggregation strategy.

3. Results

3.1. Federated aggregation strategy comparison

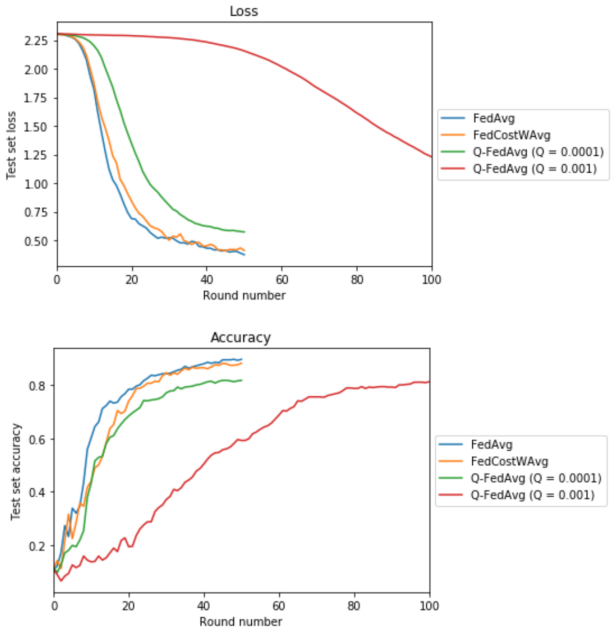

Figure 3. Comparison of Federated learning aggregation strategies test-set convergence as a function of training round. At the end of 50 rounds, FedAvg had the optimal test-set accuracy and loss, with FedCostWAvg not far behind.

The overall test-set performance during the course of training for our Federated Aggregation algorithms with differential privacy can be found in figure 3. In Figure 3 we directly compare each federated aggregation strategy in terms of test set accuracy and loss. We observed from these curves that FedAvg, FedCostWAvg, and Q-FedAvg with Q = 0.0001 all roughly converge in a similar number of rounds. FedAvg achieved the lowest loss value and highest accuracy (89.7 %) indicating that it has superior performance among these aggregation strategies for the FEMNIST dataset. This contradicts the findings of Machler et al. who reported that FedCostWAvg outperforms FedAvg, as we found that FedAvg slightly outperforms FedCostWAvg. This could be due to a number of potential reasons. Firstly, Machler et al. utilized DICE as a loss function in contrast to our use of categorical cross-entropy loss. Additionally they report only the DICE score and the Haussdorff95 distance of both federated aggregation strategies as a function of training round, and not overall accuracy. Perhaps while FedCostWAvg outperforms FedAvg in image segmentation tasks and metrics, this performance does not generalize to the task of overall image classification. These results could additionally be attributed to the confounding factor of using a differentially private optimizer for training our models. Most federated aggregation comparison papers do not do so in a Differential Private setting.

Our results additionally show that in the same number of rounds, Q-FedAvg with q = 0.0001 achieved a final test-set accuracy of 81.8%. This is a significant reduction in accuracy compared to FedAvg and FedCostWAvg. Additionally for Q-FedAvg with q = 0.001, we found that at round 50 the global model achieved an accuracy of only 38.2% and it took until training round 100 for it to reach similar performance to the initial Q-FedAvg model, achieving 81.1% accuracy. While we found no prior research on comparing FedAvg and Q-FedAvg on an MNIST-based dataset, in Li et al. the authors compare the average accuracies of Q-FedAvg and FedAvg for a variety of other federated learning datasets. Our results differ from theirs in two major aspects. Firstly, in their results they report that across the 5 different federated datasets they evaluated the aggregation strategies on, QFedAvg and FedAvg generally achieved very similar accuracy performance, with the greatest accuracy discrepancy between the two algorithms being under 2% for a synthetic dataset of their own construction using a linear classifier. This contrasts with our findings that after 50 rounds of training and using a very minimal q value, our accuracy results differ by roughly 8%. Furthermore, while the authors state that theoretically one would expect the rate of convergence to decrease, they don't observe significant decreases in convergence speed compared to FedAvg in their results. Whereas in our results we observe significant decreases in convergence speed for a one magnitude change in q.

For an idea of comparative final accuracy to preexisting MNIST-related federated learning experiments, FedAvg's 89.7 % accuracy and FedCostWAvg's 88.2 % are well below the 99% benchmark achieved on the non-IID MNIST experiments performed by McMahan et. al in the introductory FedAvg paper. However, in a differentially privacy constrained, CNN EMNIST evaluative experiment (Andrew et al., 2021) a maximum accuracy of 86 % was achieved across several thousand rounds with minimal gradient clipping, indicative to us that our models' accuracies are in line with what one would expect given the addition of differential privacy.

It is important to note that there is additional hyperparameter tuning that could have been appropriate for wholly maximizing global model accuracy(fraction of clients selected for training, model dropout rate), but this was not the end goal of our experiments.

Table 2. Measured federated aggregation method wall-clock training time for 50 rounds of training

AVERAGING SCHEME	WALL-CLOCK TIME (SECONDS)
FEDAVG	4419.67
FEDCOSTWAVG	4502.44
Q-FEDAVG	**4358.57**

Lastly, our measurements of wall-clock training time in table 2 indicate that for the same number of rounds, Q-FedAvg achieves the fastest training wall-clock time, over 2 minutes faster than the slowest performing strategy, FedCostWavg. Given that theoretically Q-FedAvg is the most computationally intensive strategy among the 3 we are unsure of what caused this discrepancy. Additionally it's important to note that the discrepancy among wall-clock times for these strategies is smaller (3.3%) than the discrepancy observed between strategy accuracies (9.7%). As such optimizing for performance will likely be more important for most use cases of federated learning.

3.2. Fairness Evaluation

To quantify the fairness of these aggregation methods, we opted to focus on the average variance of accuracy and loss across clients over the final 5 rounds of training. The idea here is that a model that has less variance in performance amongst clients is promoting a more uniform distribution of model performance and is thus being more 'fair' by not prioritizing the performance of some clients over others, a notion supported by Li et al.

Table 3 presents our measured variance values for test set accuracy and loss. While both variants of Q-FedAvg achieved the lowest variance of loss among the clients as expected, we

Table 3. Fairness evaluation of federated aggregation methods over last 5 evaluation rounds

Averaging Scheme	Average Variance of Accuracy	Average Variance of Loss
FedAvg	**0.014**	0.22
FedCostWAvg	0.018	0.73
Q-FedAvg (Q = 0.0001)	0.035	0.20
Q-FedAvg (Q = 0.001)	0.036	**0.0299**

unexpectedly found that the clients trained with FedAvg had the lowest average variance in terms of accuracy. This result could suggest that while the loss objective for Q-FedAvg ensures that the loss among distributed clients has lower variability, this doesn't always translate into low variability in terms of actual model performance. Additionally this could be due to the fact that QFedAvg had a worse overall performance than FedAvg and thus, depending on the data, can take on have vastly different performance whereas FedAvg always performs generally well. Furthermore the data is aggregated from measurements on only 50 clients, and using a larger distributed network may result in a resolution to the descrepancy we measure between the variance of loss and accuracy.

3.3. Comparison of Non-Distributed and Federated Learning

Figure 4 presents our non-distributed model's test accuracy and performance to the federated algorithms' across iteration number. An iteration is defined as training over a singular client example. The non-distributed model appears to converge quite a bit faster than our four federated ones, and achieves a lower loss and higher accuracy after approximately 30 federated rounds' worth of iterations. Given the more efficient data access of the non-distributed model, its superior convergence speed within a reasonable training time is line with previous documentation (Nilsson et al., 2018) (Ek et al., 2020). However it does appear that both FedCostWAvg and FedAvg may not have finished converging after 30 rounds, with neither their loss or accuracy curve appearing to fully level off. Over the course of several tens of thousands more iterations, which could be an appropriate time frame considering prior work extending into hundreds of rounds.(Geyer et al., 2017), differentially private FedAvg and FedCostWavg may well perform on par with a non distributed approach in final performance.

3.4. Privacy Calculation

TensorFlow-Privacy provides a function (compute-dp-sgd-privacy()) to calculate the ϵ guarantee of the amount of differential privacy one injects in the training process given the size of the training set, epochs trained, batch size, and noise multiplier. The lower the ϵ value, the stronger the privacy guarantee. After 30 epochs of training our non-distributed model, the calculated epsilon value was 1360, This privacy guarantee would be on the very low end among federated learning with differential privacy experiments (Wei et al., 2020). However, ϵ has an inverse relationship to the number of epochs run in a centralized setting. Therefore it is forecastable that with an optimizer that uses an adaptive learning rate, a more complex model architecture, as well a more optimally tuned batch size, our privacy guarantee would be orders of magnitude stronger.

Unfortunately, it is much more complex to calculate ϵ in a federated setting. Non withstanding, the noise multiplier of .2 we use for each client is within the same order of magnitude TensorFlow used in a differentially private, federated implementation of FEMNIST digit classification, which additionally had substantially poorer convergence speed relative to our FedAvg and FedCostWAvg implementations.

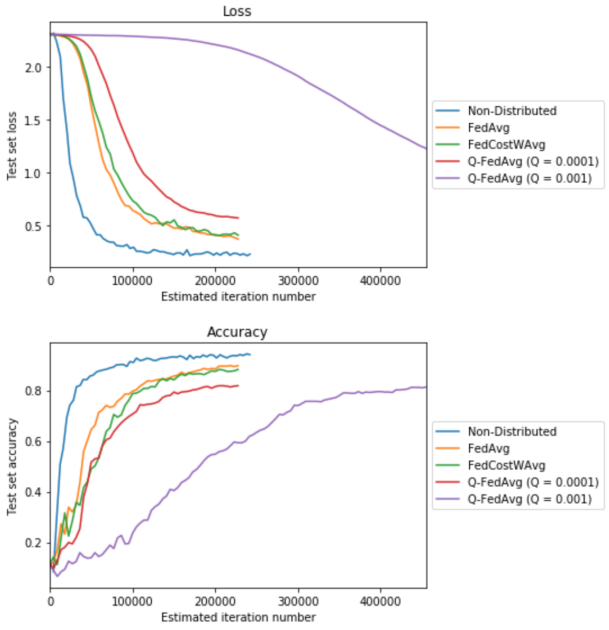

Figure 4. Comparison of Federated learning and Non-distributed learning test-set convergence as a function of number of training examples iterated through. For federated learning strategies, the number of iterations was estimated by taking the average training set size of each client and multiplying it by the number of epochs per client, the number of clients per training round, and the number of training rounds.

4. Conclusion

Federated Learning is enjoying a boom in both real world use cases as well as published research, with much of the cited work in this paper coming from the last 3-4 years (and several papers from just the last year). This is unsurprising given its theoretical efficiency advantages in both data collection side computation and privacy considerations. Re-

cent literature has provided provided valuable direction on how to strengthen its privacy guarantee through differential privacy, aggregation methods to maximize accuracy, and a more fairness oriented approach to its performance across clients. In this paper we seek to provide an investigation into how these three fundamental research areas might intersect in a real world, Federated Learning task.

We identify a non-IID dataset with clear client partitions, and train three federated aggregation algorithms and one non-distributed one with the constraint of differential privacy, seeking to compare performance and fairness. We show that FedAvg and FedCostWAvg can approach the performance of a non-distributed model in this setting, but that FedCostWAvg doesn't outperform FedAvg in a non image-segmentation task. Additionally we demonstrate that QFedAvg's convergence can be significantly worse than FedAvg and can indeed depend on the size of q, the fairness parameter. Lastly we report that QFedAvg may not result in the lowest variance of client accuracy, even though it does so for the loss function.

While this paper does indeed incorporate all three aforementioned Federated Learning avenues of exploration, there is plenty of room for future projects to expand upon our work and others'. Prada et al. engage in a similarly intersectional approach into how fairness and differential privacy can be incorporated into federated learning, and propose a framework by the name of FPFL that theoretically provides fair and private federated learning (Padala et al., 2021). A project seeking to build upon our work might be wise to experiment with different optimizers beyond SGD in a differentially private federated setting to minimize iterations required for training, and thus minimize ϵ. On a slightly different path, one could experiment with how various levels of privacy guarantee can alter the fairness behavior of federated aggregation methods, as Padala et al. observe a trade-off between fairness and privacy, and seeking to optimize both may become the new norm.

Acknowledgements: We wish to thank Daniel Beutel, one of the developers of the Flower, for his guidance in navigating this exciting framework.

References

Pytorch mnist example. https://github.com/pytorch/examples/blob/master/mnist/main.py, 2020.

Implement differential privacy with tensorflow privacy. https://www.tensorflow.org/responsible_ai/privacy/tutorials/classification_privacy, 2021.

Abadi, M. et al. Deep learning with differential privacy. 2016. doi: https://arxiv.org/pdf/1607.00133.pdf.

Andrew, G. et al. Differentially private learning with adaptive clipping. 2021. doi: https://arxiv.org/pdf/1905.03871.pdf.

Beutel, D. et al. Flower: A friendly federated learning framework. 2021. doi: https://arxiv.org/pdf/2007.14390.pdf.

Cohen, G. et al. Emnist: an extension of mnist to handwritten letters. 2017. doi: https://arxiv.org/pdf/1702.05373v1.pdf.

Divi, S., Lin, Y., Farrukh, H., and Celik, Z. New metrics to evaluate the performance and fairness of personalized federated learning. 2021. doi: https://arxiv.org/pdf/2107.13173.pdf.

Ek, S. et al. Evaluation of federated learning aggregation algorithms: Application to human activity recognition. 2020.

Geyer, R. C., Klein, T., and Nabi, M. Differentially private federated learning: A client level perspective. *CoRR*, abs/1712.07557, 2017. URL http://arxiv.org/abs/1712.07557.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. Fair resource allocation in federated learning. *International Conference on Learning Representations*, 2020. doi: https://arxiv.org/pdf/1905.10497.pdf.

Machler, L. et al. Fedcostwavg: A new averaging for better federated learning. 2021. doi: https://arxiv.org/pdf/2111.08649.pdf.

McMahan, H. et al. Communication-efficient learning of deep networks from decentralized data. 2017. doi: https://arxiv.org/pdf/1602.05629.pdf.

Naseri, M., Hayes, J., and De Cristofaro, E. Local and central differential privacy for robustness and privacy in federated learning. 2021. doi: https://arxiv.org/pdf/2009.03561.pdf.

Nilsson, A. et al. A performance evaluation of federated learning algorithms. 2018. doi: https://doi.org/10.1145/3286490.3286559.

Padala, M., Damle, S., and Gujar, S. Federated learning meets fairness and differential privacy, 2021.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., and Poor, H. V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020. doi: 10.1109/TIFS.2020.2988575.