Final Report: Reproducing FedUFO

Shreyashri Biswas and Quentin Oschatz

1 Motivation

Federated learning describes an approach to optimizing the weights of machine learning models in a decentralized approach, where training is done locally on a large number of devices without the underlying data ever being shared [2]. Originally, this idea was applied to mobile devices, where concerns over data privacy and the prohibitively large quantity of user data made central aggregation and training sub-optimal [2]. In order to support this type of highly decentralized optimization, the "FederatedAveraging" (or FedAvg) algorithm was introduced [2].

This algorithm was meant to address a number of shortcomings observed in existing distributed optimization algorithms core to the federated setting [2]. For one, communication was deemed more expensive than computation, meaning that reducing the frequency and amount of data exchanged was prioritized at the expense of having clients perform more training [2]. Additionally, unbalanced and non-IID data is common in federated environments, and many existing algorithms assumed balanced, IID data [2].

While the FedAvg algorithm managed to greatly improve the performance of distributed optimization in settings where data did not conform to those assumptions, there remains a significant performance gap [6].

The FedUFO algorithm proposed by Zhang et. al. in 2021 aims to narrow this gap by building on top of the FedAvg algorithm [6]. By proactively compensating for data skews during local training, the algorithm prevents local models diverging too far during each iteration [6]. This, in turn, ensures that the weights being averaged stem from a local optimization problems with aligned objectives [6].

2 Background

Federated learning has emerged as a popular approach to distributed machine learning that emphasizes data privacy and efficient communication. This literature survey focuses on the advancements in federated learning algorithms, specifically the FedAvg [2] and FedUFO [6] algorithms, as well as the challenges and solutions related to privacy preservation, convergence rates, and handling heterogeneous and non-IID data.

The Federated Averaging (FedAvg) algorithm [2] serves as a cornerstone for many recent developments in federated learning. It involves distributing a global model to local devices (clients) for a common initialization, followed by local training iterations. After local training, optimized weights are sent back to the central server, where a weighted average is computed, favoring clients with larger datasets. This iterative process continues until a satisfactory stopping criterion is reached.

Building on the FedAvg algorithm, [2] offers higher data security by performing local model updates on client devices and aggregating them on the server using a weighted average [1]. The secure aggregation protocol proposed in [1] ensures strong privacy guarantees by allowing the server to learn only aggregated model updates without identifying individual updates. FedUFO further adapts to non-IID data by incorporating additional regularizers to address skewed data distributions across clients [5].

Another advancement is the Federated Multi-Task Learning introduced by [3], which extends the FedAvg algorithm to simultaneously learn multiple related tasks. This approach enables efficient sharing of information across different tasks while preserving data privacy, leading to improved convergence rates and model performance, especially in the presence of heterogeneous data distributions.

In addition, [4] proposes a robust and communication-efficient federated learning algorithm for non-IID data, introducing a new optimization algorithm that integrates local adaptation and a communication-efficient model averaging scheme. This algorithm outperforms traditional federated learning algorithms in terms of convergence speed and final model performance, particularly when handling heterogeneous and non-IID data.

Both FedAvg and FedUFO adopt a client-centric weighted averaging approach during model aggregation, considering the varying dataset sizes of clients [2]. [7] investigates the impact of non-IID data on federated learning algorithms' performance, particularly the FedAvg algorithm. They propose a data partitioning strategy that mitigates the negative effects of non-IID data on convergence rates and model performance, and provide a theoretical analysis and empirical evaluation to demonstrate the benefits of their approach.

3 Experimental Setup

3.1 The FedUFO Algorithm

The FedUFO algorithm utilizes a basic optimization loop of several iterations of local optimization followed by global averaging. Two key additions are made to the local optimization problems to combat the issues caused by non-IID data which act as regularizers. After local training is completed, the models are further optimized by employing first a uniform adversarial loss, then a set of consensus losses. The former is implemented in the

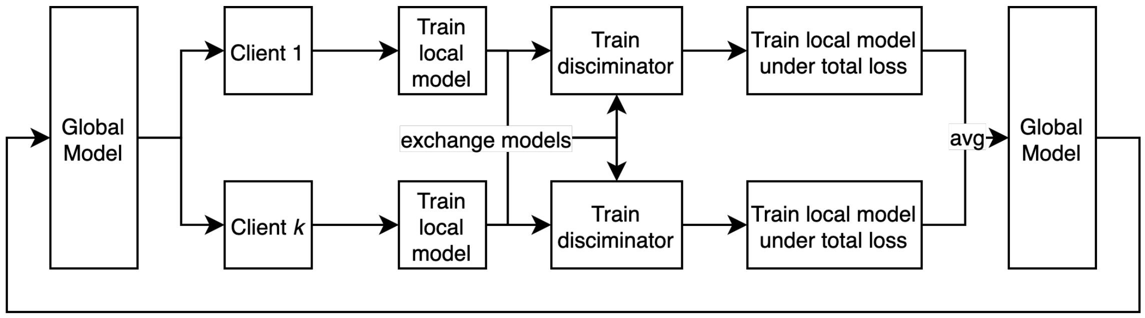


Figure 1: The basic training loop of FedUFO

form of a discriminator which attempts to learn differences in feature extraction between local models, whereas the latter uses a set of Kullback–Leibler divergence losses to align the results.

In order for these steps to be taken, additional data beyond the scope of the original FedAvg algorithm must be shared. After the initial local training step, all clients share their model weights between themselves so as to allow the further steps to be taken without breaching the ban on sharing raw data.

3.1.1 UAD Loss

Typically, deep neural networks used for classification tasks can be split into two parts: a feature extractor h_e and a classifier h_c [6]. The former extracts key features from the training data, such as shapes when dealing with images. Given different data, models may learn different feature sets, thus greatly increasing the risk of misalignment during averaging. In order to compensate for this, a discriminator model is introduced during training.

Given this discriminator — a simple two-layered neural network — and every other clients' local models, each client will first train the discriminator on the features $\tilde{\mathbf{f}_{\mathbf{i|j}}}$ each local model extracts from this client's data:

$$\tilde{\mathbf{f}_{\mathbf{i|j}}}(h_e(\mathbf{x_i}|\tilde{\mathbf{w}}^{\mathbf{e}}_{\mathbf{t,j}})), \forall j \in [1, K_c] \wedge j \neq k, \tag{1}$$

where $\mathbf{x_i}$ represents this client's data, $\tilde{\mathbf{w}}^{\mathbf{e}}_{\mathbf{t,j}}$ are the weights for the extractor at time t from client j, K_c being the number of clients, and k being the current client.

In order to have the discriminator attempt to predict which client produced which feature set, the gathered $\tilde{\mathbf{f}_{\mathbf{i|j}}}$'s as well as the local features $\hat{\mathbf{f}}_{\mathbf{i}}$ are fed into the model:

$$\tilde{\mathbf{d}_{\mathbf{i|j}}} = Softmax(D(\tilde{\mathbf{f}_{\mathbf{i|j}}}|\tilde{\mathbf{w}}^{\mathbf{d}}_{\mathbf{t,j}})), \forall j \in [1, K_c] \wedge j \neq k \hat{\mathbf{d}}_{\mathbf{i}} = Softmax(D(\hat{\mathbf{f}}_{\mathbf{i}}|\tilde{\mathbf{w}}^{\mathbf{d}}_{\mathbf{t,j}})), \tag{2}$$

with $\tilde{\mathbf{w}}^{\mathbf{d}}_{\mathbf{t,j}}$ being the parameters of discriminator D.

Using the gathered predictions, the uniform adversarial loss can now be calculated as follows:

$$\mathcal{L}_{uniform} = -\frac{1}{K}\sum_{j=1}^{K}\log(\hat{\mathbf{d}_{\mathbf{i}}^{\mathbf{j}}}). \tag{3}$$

It should be noted that this loss will be minimized when the discriminator predicts each client as an equally likely to have produced $\hat{\mathbf{f}}_{\mathbf{i}}$, i.e. that $\hat{\mathbf{d}_{\mathbf{i}}^{\mathbf{j}}} = \frac{1}{K}, \forall j \in [1, K]$.

Lastly, in order to train the discriminator, a cross entropy loss is used that utilizes all discriminator outputs:

$$\mathcal{L}^{d}_{cls} = -\log(\hat{\mathbf{d}_{\mathbf{i}}^{\mathbf{l_k}}}) - \frac{1}{K-1}\sum_{j=1, j\neq k}^{K}\log(\tilde{\mathbf{d}_{\mathbf{i|j}}^{\mathbf{l_k}}}), \tag{4}$$

where l_k is class of the feature representation of the current client k.

3.1.2 Consensus Loss

With the UAD loss explained in 3.1.1, we now move on to the second augmentation introduced by FedUFO. Following the work of the previous section, and with the UAD loss aligning the feature extractors, the consensus losses aim to align the other part of the models — the classifier.

The first consensus loss is the group consensus loss, which aims to align all the local classifiers in the current iteration. To achieve this, the algorithm first uses the local models from other clients to generate predictions on the current dataset, applying a softmax operation on the results:

$$\tilde{\mathbf{y}_{\mathbf{i|j}}} = Softmax(h(\mathbf{x_i}|\tilde{\mathbf{w}_{\mathbf{t,j}}})), \forall j \in [1, K]. \tag{5}$$

In order to compensate for unequally distributed classes, the results are then further multiplied by the proportion of that class in the total dataset:

$$\tilde{\mathbf{y_i}} = Softmax(\sum_{j=1}^{K} \mathbf{p_j} \otimes \tilde{\mathbf{y_{i|j}}}), \tag{6}$$

where $\otimes$ is an element-wise product, and $\mathbf{p_j}$ is a vector of fractions, where each element is the proportion of that class compared to the local dataset of that client.

Utilizing Kullback–Leibler divergence to measure the difference in result distributions, the group consensus loss can now be calculated as follows:

$$\mathcal{L}_{cgr} = KLDiv(\hat{\mathbf{y}}_\mathbf{i} || \tilde{\mathbf{y}}_\mathbf{i}), \tag{7}$$

where $\hat{\mathbf{y}}_\mathbf{i}$ is the prediction of the local model.

The second consensus loss is the global consensus loss, which aims to align the local optimization with the global optimization problem. It is calculated in a very similar way, by first getting results from the global model

$$\tilde{\mathbf{y_{i|0}}} = Softmax(h(\mathbf{x_i}|\tilde{\mathbf{w_t}})), \tag{8}$$

and then applying Kullback–Leibler divergence again:

$$\mathcal{L}_{cgl} = KLDiv(\hat{\mathbf{y}}_\mathbf{i} || \tilde{\mathbf{y}}_\mathbf{i|0}) \tag{9}$$

3.1.3 Training

Training is done in multiple steps. First, local models are trained as in the FedAvg algorithm, and subsequently shared to all clients. Next, the feature extractors are briefly trained under a combined UAD + cross entropy loss, followed by the discriminator (parameters of the discriminator and local model are frozen respectively). Next, the local model is trained again, this time under the following combined loss:

$$\mathcal{L}_{total} = \mathcal{L}_{cross-entropy} + \mathcal{L}_{uniform} + \mathcal{L}_{cgr} + \lambda \mathcal{L}_{cgl}, \tag{10}$$

where λ is a hyperparameter.

3.2 Challenges

A major challenge during implementation was interpreting a few ambiguous parts of the training loop. While many parts — such as the loss functions and parameters such as learning rate — were described with great detail, and a picture describing the training loop was included, key details such as how often the posterior model and the discriminator were to be trained were not properly addressed. Therefore, we had to employ a great amount of trial and errors to build a seemingly accurate version of their training loop. Given the decently large amount of time each test run took, and given that inaccuracies in the algorithm would usually only manifest well into the training, this process was time consuming and ultimately impossible to verify, as no reference implementation is publicly available.

3.3 Parameters

For our testing, similarly to the original paper, used a 2-layer discriminator. However, we found that the learning rate and momentum values for both the client and the server used by the authors [learning rate of `0.001` and momentum of `0.9`] did not yield good results on our implementation, yet a learning rate of `0.0001` and a momentum of `0.5` did. Additionally, while we would have liked to compare a range of values for the λ used in the total loss function, we settled to a value of 2 for all runs. With more time, it would be worth recreating the original paper's testing comparing different values. We also use 10% of all clients every round for the update, randomly sampled each iteration.

4 Results and Discussion

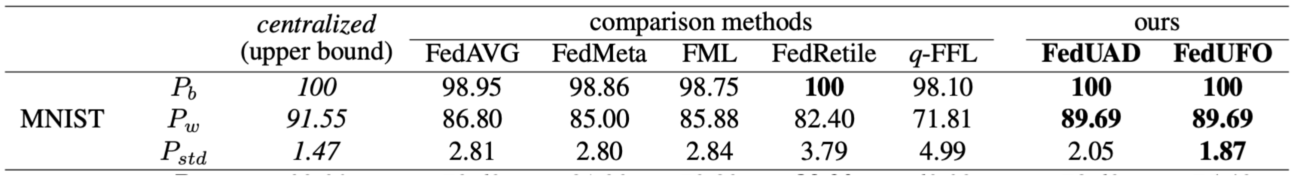
We ran multiple experiments in order to ascertain the effectiveness of the FedUFO algorithm. Using the MNIST dataset (and identical seeds for data sampling) under non-IID conditions, we primarily compare the performance of algorithm to that of the FedAvg algorithm. Additionally, we also ran the algorithm with only one of the two additional losses, UAD and the consensus losses, in order to better understand the influence of each component.

First, we would like to point out the wildly different run times of each algorithm. While it is somewhat obvious that additional and more complex loss functions add time complexity, the original paper only briefly mentions the additional overhead along with the high communication cost caused by sharing all models between each client. In our testing, we noted that the FedUFO algorithm is around 6 times slower than the baseline.

Next, it should be noted that the best, worst local performance and the stand deviation(std) of all the local performances are used to evaluate the performance fairness, which are termed as P_b, P_w and P_{std} in Table 2.

	FedAvg	FedUFO	FedUFO (only UAD)	FedUFO (only consensus)
Test Acc	0.8438	0.885	0.9176	0.8805

Table 1: Final test accuracy of multiple algorithms/variations on non-IID MNIST after 50 iterations



		centralized	comparison methods					ours	
		(upper bound)	FedAVG	FedMeta	FML	FedRetile	q-FFL	**FedUAD**	**FedUFO**
MNIST	P_b	*100*	98.95	98.86	98.75	**100**	98.10	**100**	**100**
	P_w	*91.55*	86.80	85.00	85.88	82.40	71.81	**89.69**	**89.69**
	P_{std}	*1.47*	2.81	2.80	2.84	3.79	4.99	2.05	**1.87**

Table 2: Accuracy values from the original FedUFO paper

The original paper uses test accuracy as its method of comparison. In table 1, after running the algorithm for 50 epochs, with 10 local iterations and the parameters specified in 3.3, we observe the the FedUFO algorithm does indeed outperform the baseline. We also see that only using UAD loss actually lead to an even higher accuracy value, with the consensus loss not performing as well without the feature alignment. While the actual values do not match those of the paper (seen in table 2), this is almost certainly down to the authors there using the best/worst local accuracy (in order to evaluate fairness), instead of the mean as we did. Looking at the worst local accuracy observed, we see that both our FedAvg and FedUFO variants slightly underperform by similar amounts, suggesting that this is most likely down to the inherent randomness of federated learning, or perhaps down to the fact that the original paper does not state how many iterations were used for training. The only stand-out value is the relatively high accuracy of the UAD-only variant.

The UAD accuracy values can be easily explained when looking at the convergence. In figure 2a, we see a comparison of training loss over time. The plots provide a few interesting insights. The stark difference the magnitude of the loss can be explained given the fact that a summation of losses are used during training, causing a higher loss floor. More interesting are the trends of each line. We can clearly observe that the UAD loss is converging, albeit a bit slower than baseline FedAvg, which is consistent with observations made in [6]. There, the authors say that should UAD loss will have "similar" convergence rates to the baseline, though figure 2a shows that, at least when comparing test accuracy, UAD seems to improve slightly slower.

Looking at the consensus losses in figure 2b, it is no surprise to find them diverging over time. Without proper feature alignment, attempting to force consensus becomes increasingly difficult as learned features diverge. Thus, this is consistent with expectations.

When inspecting the loss curve of the full FedUFO algorithm, the results shown in figure 2c do not match expectations. What should be a clear — albeit slower than the baseline — trend towards convergence mirroring the FedAvg algorithm shown in figure 2d instead shows wild fluctuations without signs of convergence. This is especially puzzling given that both the test accuracy and loss curves of its components seem to line up with expectations. It should be noted that loss values were observed to be relatively consistent across different clients.

When looking at the training accuracy over time shown in figure 3, although the results show all algorithms performing similarly, with the UAD loss seemingly showing slightly better stability, and the FedUFO algorithm more often than not ever so slightly ahead of the baseline. Curiously, the consensus losses also perform well here. We theorize that this is most likely due to MNIST being a fairly easy dataset for training, meaning that training loss reaching near 100% is feasible even for somewhat sub-optimal algorithms.

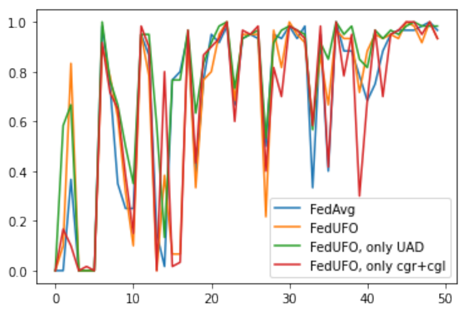
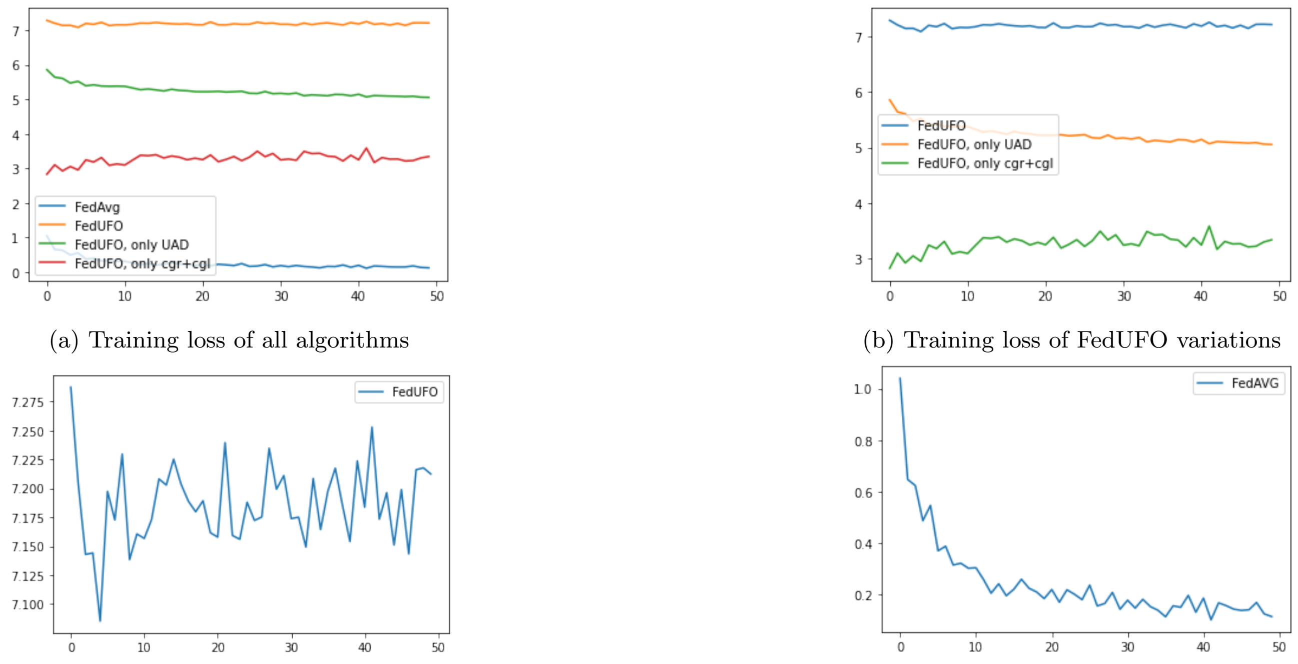


Figure 3: Training accuracy of all algorithms over the same non-IID dataset as in 2

Unfortunately, given that the original paper only concerns itself with testing accuracy and not testing loss, and that value seems to line up with their results. Hence, it is hard to determine the cause of the loss curve. As mentioned in 3.2, the paper did not adequately detail every step of training, forcing us to make several assumptions which may have compromised the algorithm. Furthermore, as discussed in 3.3, the exact parameters used in the original paper yielded very poor results for us, further supporting the theory that our implementation does not fully match the original authors'. Given that the run with only UAD loss provided reasonable loss curves, it is not unreasonable to suggest that the interplay between the loss functions — which is critical to this algorithm — was improper, either due to mentioned parameter misalignment or implementation differences. Without a reference implementation, however, this is impossible to verify.



(a) Training loss of all algorithms (b) Training loss of FedUFO variations

(c) Training loss of FedUFO (d) Training loss of FedAvg

Figure 2: Training losses of different combinations of algorithms over 50 iterations on a non-IID MNIST dataset. Each graph shows the same run with different algorithms being compared

5 Future Work

The logical next steps would be to run a more comprehensive set of testing, exhaustively checking different parameters and different ways to implement the ambiguous parts of the paper. It may be prudent to reach out to the original authors to either receive a reference implementation or to clear up said ambiguities. Then, the suite of tests we ran could be recomputed, as well as some tests that we were not able to reproduce due to time constraints, such as scanning through multiple values for λ. It would be of great interest to see the loss curves in particular, given that they are not present in the original paper.

Next, in order to confirm the real-world performance claimed by the authors, running the algorithm on several more realistic datasets, including but not limited to those shown in the paper (such as CUB200), and comparing the results to a broader selection of algorithms would aid in further confirming FedUFO's capabilities. Additionally, testing the algorithm on unbalanced datasets and different numbers of clients would be another good way to observe its robustness.

6 Division of Labor

- Implement FedUFO (Mostly Quentin)
- Testing pipeline and integrating FedAVG (Both)
- Gathering Results (Mostly Shreya)
- Report (Both)
- Presentation (Both)

References

[1] K. Bonawitz, H. Eichner, W. Grieskamp, *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, A. Singh and J. Zhu, Eds., ser. Proceedings of Machine Learning Research, vol. 54, PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: `https://proceedings.mlr.press/v54/mcmahan17a.html`.

[3] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4427–4437, ISBN: 9781510860964.

[4] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

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

[6] L. Zhang, Y. Luo, Y. Bai, B. Du, and L.-Y. Duan, “Federated learning for non-iid data via unified feature learning and optimization objective alignment,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 4400–4408. DOI: `10.1109/ICCV48922.2021.00438`.

[7] V. S. Mai, R. La, T. Zhang, Y. X. Huang, and A. Battou, “Federated learning with server learning for non-iid data,” en, IEEE 57th Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, US, 2023-03-24 04:03:00 2023. [Online]. Available: `https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=935346`.