Client Selection, Domain Generalization and Pruning in Federated Learning.

Silva Bashllari
Politecnico di Torino
Turin, Italy
[redacted]@studenti.polito.it

Neralb Cika
Politecnico di Torino
Turin, Italy
[redacted]tudenti.polito.it

Alejandro Mesa Gomez
Politecnico di Torino
Turin, Italy
[redacted]tudenti.polito.it

Abstract

The purpose of this work is to test different algorithms in the Federated Learning paradigm. In a time and age where the importance of data is gaining an ever increasing traction, so are the privacy concerns associated with it and federated learning comes as one of the possible solutions. However, there are as many challenges as opportunities associated with it. In this work, we test the IID and the non-IID settings, different client selection strategies, domain generalization techniques and pruning techniques. The experiments are tested on the FEMNIST dataset and the model used is a CNN. The code implementation can be found in this git account.

1. Introduction

1.1. The new paradigm of Federated Learning

Federated Learning is a new paradigm of machine learning introduced by Google in 2016. The core idea is to permit the training of machine learning algorithms without needing to collect all the data in a central server. In a time and age where the importance of data is gaining an ever increasing traction, so are the privacy concerns associated with it. Thus, federated learning comes as one of the possible solutions. Two of the major types of Federated Learning are: *cross-device*, in which there are generally many clients and each client has a relatively small dataset, and *cross-silo*, in which there are large entities, such as banks, hospitals, etc, which may be fewer in terms of the client number involved but, have large local datasets. The primary intuition on how federated learning works is as follows: each client trains a local model with its data and generates the weights. Then, each client communicates to the server these weights, upon which the server performs some sort of aggregation and averaging. Subsequently, the server communicates this averaged model back to the clients which retrain again on their local dataset. This process of communication rounds goes on until a stopping criterion is met, such as a maximum number of rounds or a desired accuracy is attained. Behind this intuitive description lie a lot of micro-decisions that determine the outcome of this process. For example, the way the averaging is done may be a simple average, a weighted one, etc. In addition, another decision is the way the clients are being selected. Moving towards realistic scenarios, not all clients will always be available in each communication round. Furthermore, the distribution of data across clients may differ and this in turn impacts the federated averaging. In addition, communication overhead is of paramount importance in Federated Learning given that the communication over the internet is a central element, as weights are constantly exchanged between clients and the server. Thus, micro-decisions that make this communication more efficient are of imperative importance.
These encompass a small and definitely non-exhaustive subset of an ever increasing list of challenges and opportunities that come along with the Federated Learning paradigm, some of which we will try to address in this work.

2. Related Work

Several approaches have emerged in recent years with regards to the Federated Learning paradigm in order to address the challenges, to optimize it and bring it closer to realistic scenarios [6]. As mentioned above, we tested the weighted Federated Average algorithm in both the IID and non-IID settings, given that statistical heterogeneity has been shown in previous works to be one of the main challenges [3] using the Federated Extended MNIST dataset. Regarding client selection decisions, which as mentioned in Section 1 are crucial, non-uniform client selection methods have been important, particularly in the work of Cho et al(2002) using Power of Choice selection strategies [2]. The main idea here is to train those clients that have a higher loss and thus reaching convergence faster. Furthermore, emphasis has been placed on domain generalization in Federated Learning, which involves the ability of a model trained on a specific dataset to make accurate predictions on unseen data from a slightly different domain. It is crucial that models generalize effectively, as their performance depends not

only on their ability to learn patterns in the training data, but also on their ability to adapt to new situations. Previous work in this regard was used, namely the RotatedFemnist dataset was built, where 1000 clients are generated and they are divided in 6 subsets [1]. More specifically, researchers have studied the domain generalization paradigm through "Leave-one-domain-out" strategy, in which, each test set accounts for one rotated domain, and the next five rotated domains are the training set. Furthermore, researchers at Oxford in 2022 developed the FedSR algorithm (Federated Simple Representation) which we experiment with in this work [5]. Exploiting concepts from representation learning, in which the input data is transformed into a more useful and compact form that captures relevant features, FedSR applies L2 norm regularizer and CMI regularizer to the RotatedFemnist datasets. Considerable attention has been devoted in recent years to discovering methods for reducing the size of models, for a variety of reasons, including. This is for a variety of reasons, such as but not limited to: edge devices do not have the same processing capacities as a central server does, the fact that clients have to continuously communicate weights with the server and in the cases that the models are DNN-s, this introduces a very large overhead, often with millions of parameters. Thus, pruning strategies have been thought as possibly very helpful in the case of Federated Learning [4]. Pruning removes from the model what are deemed by certain criteria as unnecessary weights or can go as far as to remove whole structures. For example, in the case of unnecessary or uninformative parameters, they are often set to 0, which reduces the size of the model and the amount of data that needs to be communicated, given the fact that it is lighter to transmit sparser matrices of parameters, thus optimizing the federated setting.

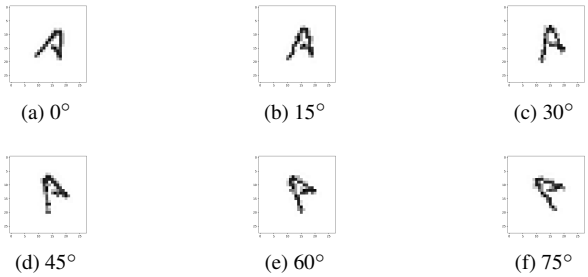
3. Methods

3.1. Federated Average

The algorithm for the federated averaging can be implemented, as discussed in Section 1 in a variety of ways. In this work, we are proceeding with a weighted average, thus the weights of the clients that have more samples are given a larger importance. Let w_i represent the weight associated with client i, and n_i represent the number of samples held by client i. The weighted average of model parameters θ across K clients can be expressed as:

$$\theta^{\text{new}} = \sum_{i=1}^{K} \left(\frac{n_i}{N} \right) \cdot \theta_i$$

Where: θ^{new} represents the updated model parameters; $N = \sum_{i=1}^{K} n_i$ denotes the total number of samples across all clients; θ_i signifies the model parameters from client i; The term $\left(\frac{n_i}{N}\right)$ calculates the proportion of samples held by client i relative to the total number of samples. Last but not least, in the "vanilla" version, we assume unbiased client participation, where clients are selected with a discrete uniform random probability distribution to be part of the traning for each round.



(a) 0° (b) 15° (c) 30° (d) 45° (e) 60° (f) 75°

Figure 1. A sample of the letter A rotated.

3.2. Domain Generalization

3.2.1 Rotated Femnist

There are several methods for generating artificial data to make models more robust and for this project, the artificial images are generated by rotations, specifically at 15°, 30°, 45°, 60° and 75°. To maintain the consistency of the project, the process is performed using the rotate function of the torchvision library, which is applied to the tensor only when an image is requested from the custom dataset FEMNIST. This set of images, together with the unaltered (0°) images, is distributed into a new set of 1000 clients, taken from N-IID distributions. The goal is for each angle to have approximately the same number of clients (approximately 170 per angle). The rotation process originally added artifacts (see Appendix C) to the corners of the images but using the parameter *fill* of the pytorch rotate function, we managed to eliminate them and an exemplary image is displayed in Figure 1.

3.2.2 Leave-one-out domain

As explained above, we have 6 domains and each domain has a certain degree of rotation. In leave-one-out, as the name suggests, you leave out one domain, which is used for testing purposes and the training is done in 5 domains. Thus, 6 different tests are made. The idea behind this method is to understand if one domain is more difficult to learn than others and which this domain is.

3.2.3 FedSR(Federated Simple Representation)

FedSR proposes learning a "simple" representation of the data. Specifically, using representation learning, a new probabilistic mapping of the data is created and two regularization methods are applied, namely: L2 regularizer on

the representation (not on the network parameters) and the Conditional Mutual Information. The final local objective function of each client *i* will be:

$$\overline{f}_i + a^{\text{L2R}} \cdot l_i^{\text{L2R}} + a^{\text{CMI}} \cdot l_i^{\text{CMI}} \tag{1}$$

L2 norm regularizer, intuitively will have the effect of encouraging z to be centered around 0 for all clients,thus helping with the marginal alignment of the representation. Its formula is the following:

$$l_i^{L2R} \approx \frac{1}{N_i} \sum_{n=1}^{N_i} \left\| z_i^{(n)} \right\|^2 \tag{2}$$

Conditional Mutual Information regularizer is calculated as follows:

$$l_i^{\text{CMI}} \approx \frac{1}{N_i} \sum_{n=1}^{N_i} \text{KL} \left[p(z|x_i^{(n)}) \,|\, r(z|y_i^{(n)}) \right] \tag{3}$$

where we quantify the difference between the two distributions. This operation intuitively involves aligning the conditional distribution of the representation z given y, a common and conventional technique in Domain Generalization.

3.3. Client Selection Algorithms

In the vanilla setting, the clients are selected at random with a discrete uniform probability distribution. However, moving towards more realistic scenarios choosing clients at random may not always be the optimal choice or possible. Real-world datasets can exhibit heterogeneity and in these cases, random selection may fail to capture the underlying patterns and result in sub-optimal model performance. In terms of availability, some clients may have more computational resources and a better access to a stable internet connection than others, and that has to be factored in. In this work, in addition to random selection, the following client selection methods were tested:

3.3.1 Non-uniform Client Selection

1. in which 10% of clients being selected with probability 0.5 at each round.
2. in which with 30% of clients being selected with probability 0.0001 at each round.

3.3.2 Power of Choice Strategies in Client Selection

The power of choice client selection method, as described in Section 2, depicts that biasing client selection towards clients with higher local loss achieves faster error convergence. First and foremost, the way the federated average is done in this setting, differs from the one described in section 3.1, in that here clients, at first, are selected not with a uniform probability distribution but the clients with a higher number of images have a higher probability of being chosen. In this implementation, out of a set of clients γ, a subset of clients **d** is chosen at each round as mentioned above and a subset **m** with the highest losses is selected in each round to be used to train the model. Then, the intuition behind this algorithm is best described using a concrete example. Assume you have *d=10*, (so, 10 clients per round) and *m=2* (the number of clients with the highest loss to be trained). In the first round, a random model is sent to the 10 clients and they test its accuracy and measure the local losses. The 2 clients with the highest losses are then used to retrain this model and compute the Federated Average. This is the new global model. It proceeds in this fashion until the stopping criterion is met, which, in this case is the number of rounds.

3.4. Pruning

Pruning involves the reduction of parameters in a neural network, making models more sparse by eliminating unnecessary connections or weights, as mentioned in 2. In this work, two types of pruning were tested: structured and unstructured. Structured pruning removes entire neurons or channels, reducing model size by targeting specific structures within layers. Unstructured pruning removes individual weights, leading to a more granular reduction in model size. Sparse matrices, achieved through pruning, contain fewer non-zero elements, making them easier to transmit due to reduced memory requirements and bandwidth usage. In this implementation, the extent of pruning is set by providing some input parameter "amount to prune". Pruning can be done at different stages. In this work, it starts only after 70% of the set number of rounds has passed, thus giving the model time to learn before starting the process. Furthermore, pruning is done either in linear or convolutional layers, separately, to study the impact further. For unstructured pruning, both convolutional and linear layers are tested. Whereas, for structured pruning, only convolutional layers are tested, given that the linear layers are usually the classifier ones, and removing entire structures from those in such a "shallow" network might cause entire classification classes to be lost. Pytorch has its own library for pruning allowing for it to be done by a set of mechanisms. For the purposes of this work, pruning is done using L1 norm and the less important weights are shrunk towards zero and eventually pruned. Lastly, pruning in Federated Learning can be done in a variety of ways, for example: in the federated model after aggregation and averaging or in the local models. The design choice in this work is to prune on the local models of each client after the last local epoch. Also, pruning can be done during inference time or training. Pruning at training reduces the size of models that need to be transmitted between the server and the clients during each iteration, which is the focus of this work.

4. Experiments and Results

4.1. The model and the dataset

The dataset: For the purposes of this work, the chosen dataset was the Federated Extended MNIST, a version of the Extended MNIST dataset, made suitable for the federated setting [1]. FEMNIST contains 28x28 images distinguished over 62 classes, digits from 0 to 9, 26 lowercase and 26 uppercase characters of the English alphabet. The total number of samples are 805,263. The total size of the dataset is around 19GB and due to our limited computational resources, we decided to utilize a sampled version of FEMNIST at 60 percent of its capacity. The dataset can be split in: an iid fashion, where all clients have the same underlying distribution and in an non-iid fashion, where data distributions vary between clients. For the majority of the experiments in the federated setting, the mean number of images per client is 189 in the training set and 21.5 in the test set, for iid and non-iid alike. Whereas for the centralized setting, there are 408,541 train images and 46,585 test images.
The model:Two 5x5 convolutional layers with 32 and 64 output channels respectively. Each convolutional layer is followed by both a 2x2 max pooling layer and an activation ReLU layer.Two fully connected layers, the fist one mapping the input to 2048 output features, and the second one mapping to the number of classes.

4.2. Centralized Setting Experiments:

A total of 40 experiments testing various hyperparameters were conducted, with 26 of them reported in this work (see Appendix A). Furthermore, the tests were conducted with two different seeds, namely: 42 and 1234. Trying different seeds is of imperative importance in experiments that have stochastic elements in them such as neural networks. The CNN is initialized using random values. Using different seeds allows researchers to understand how the model will react in a variety of initial conditions. For this dataset and model, it was observed that a very small learning rate, such as 0.0001 was not very fitting but increasing it to 0.01 and 0.1 yielded better results (see Appendix A). We also tested different numbers of batch size and decided to proceed with the value 32, given that no relevant pattern was detected. In terms of epochs, the best were around 10 and increasing it to 15 saw no significant improvement (see Appendix A).Further increases led to overfitting to the training data. Furthermore, introducing a small value of momentum, such as: 0.3 and 0.5 and a very small weight decay, 0.00001, proved useful to improve the test accuracy further. The best two scores of test accuracy from each random seed are 87 % with random seed 42 and 86% with random seed 1234 (see Appendix A).

4.3. Federated Setting Experiments:

Initially, we tested both the IID and non-IID settings in order to understand which local epochs and learning rates were best fitted in each scenario. In both settings, we kept these hyper-parameters static: 32 batch size, 20 rounds and 20 clients per round and, we tested the following epoch numbers: 1,5 and 10 and the following learning rates: 0.0001, 0.001, 0.01, 0.1, 0.2 and 0.5. For the IID, the best test accuracy of 82.4% was attained with a learning rate of 0.2 and local epochs 5 (see Appendix B). For non-IID, the best accuracy of 81.3% was attained with 0.1 learning rate and 5 local epochs (see Appendix B) and in both cases, as in the centralized setting, very low learning rates didn't provide good results. In N-IID, not only the smallest learning rates, but also the highest learning rates were not well fitted and caused the test accuracy to go near zero. Furthermore, we tested adding momentum and weight decay, the very same values that proved best for the centralized setting. In both IID and N-IID cases, just as in centralized, we managed to get the top accuracy scores having SGD with momentum and weight decay (see Appendix B). In the case of N-IID, the best accuracy was **80%** and for IID **88%**, reaching approximately the same accuracy as the centralized setting.
Regarding the experiments for the number of clients, 9 experiments were conducted with three different seeds and the following client numbers: 5, 10, 20 while keeping everything else static. In the end, we averaged the results with respect to the seeds, in order to have less fluctuations on the behaviour of the model, considering also the initial randomness that the seeds bring. In the IID setting, at the 20^{th} round, the test accuracy with 5 clients and 10 clients get to the same point, whereas, the one with 20 clients, it drops dramatically. The same can be observed in the N-IID setting. However, this is not expected in the N-IID setting, where at least in theory, more clients would infer more data for the model to learn from, and to achieve better generalization for the test accuracy. It appears that our results seems to be highly impacted by the choice of the seeds because the results vary substantially. For example with seed=0 in round 20, test accuracy is 80% whereas for seed=42 it drops to 5%. Considering the small scale experiment that we held, the sensitivity of the model to randomness, and the fact that each client has on average very few images as mentioned in Section 4.1, may explain to a certain degree the extremities of the accuracy.
Regarding the experiments for the number of local epochs per round, maintaining all the other hyperparameters as static and proceeding with 10 clients per round, we tested the following three values for local epochs: 1, 5, 10. As in the experiment for the client number, we averaged the results from three different seeds. In the N-IID scenario, it is clear that the more local epochs are held in one round, the

worse the test accuracy performs, particularly in the case of 10 local epochs, where the average test accuracy of the 3 seeds drops at about 25% (see Appendix B). In the IID setting, the results are less intuitive, but the difference between the three experiments is not as dramatic as in the N-IID setting.

4.4. Rotated FEMNIST Experiments

To evaluate the model in a data-centric configuration, several experiments were conducted by varying hyperparameters, such as: learning rate, batch size, and momentum. In the centralized setting, we have tested without accounting for generalization, which means that the client dataset are shuffled in a random manner, between the training set and the test set. The tested learning rates are: 0.001, 0.01 and 0.1. The highest accuracy was attained with a learning rate of 0.01 with a value of **79%**.Even 0.001 showed good results, while learning rate 0.1 did not. After discovering that the best learning rate in centralised was 0.01, we also found out the it was the best performing learning rate even in the federated setting. In general, as found in all the other experiments, batch size did not show any impact, and therefore it is not included in the reported experiments of the section. Given that the evaluation metric for testing the experiments is **max test accuracy** which is the maximum accuracy that the model can reach, we also present the round in which it was obtained. We select this metric because it wasn't possible to achieve a convergence in the accuracy for the model in the scale of the test performed, as shown in the Figure 10 (see Appendix (C)). As shown in this appendix, we firstly wanted to understand the effect of the seed in the experiments, fixing the number of rounds, local epochs and learning rate. However, we found a very small difference in the results. A fairly decent accuracy was obtained, with values between **72%** and **78%**, as shown in the Tables on Appendix (C).

4.5. Leave one out Experiments

Leave-One-Out experiments were performed with the highest achieving parameters from federated experiments. It was shown that the model does actually learn some domains better than others (see Appendix C). The most difficult domain to learn was **angle=0** with an accuracy of **60%**. We believe that this behaviour is to be expected, given that we can conceptualize the rotations as an added noise in the domain, and if the training dataset is made only with the rotated segments, then the model will try to learn the noise and the variances. In this way, the model captures less patterns during training, and reflects it in the testing domain without rotation(or noise). If the original domain is part of the training set, then the model is able to recognise the pattern inside the rotations and therefore is able to classify images with rotations outside of the analysed domain (see also Appendix (D)).

Table 1. Leave One out best experiment for centralized setings

angle	lr	epochs	seed	accuracy
0	0.010	5	1234567890	50.000
15	0.010	5	1234567890	75.000
30	0.010	5	1234567890	85.000
45	0.010	5	1234567890	94.000
60	0.010	5	1234567890	96.000
75	0.010	5	1234567890	98.000

4.6. FedSR Experiments

In general, we did not achieve very high test accuracy results. It is possible that we received such results due to the low number of rounds that our experiments performed. Nonetheless, we can mention that there exist some interesting patterns, observing that the domain with the highest achieving accuracy was **angle=30**. Also, as we mentioned in Section 3.2, FedSR helps in aligning the marginal and conditional distributions of the clients. We can observe from the experiments that FedSR eliminates the assumption that one domain is harder to learn than the others, consequently eliminating the variances across the clients, and achieving uniform accuracy across all the domains (see Appendix D.

4.7. Client Selection Experiments

In this section, from now and onward it will be denoted as: *client selection 0*, when it is done by using discrete random uniform probability distribution; *client selection 1* when 10% of clients are being selected with probability 0.5 at each round; *client selection 2* when 30% of clients being selected with probability 0.0001 at each round; *client selection 3* when power of choice strategies are used. This naming convention is consistent with the one in the code.

4.7.1 Non-uniform client selection

Using the same hyper parameters in client selection 0, 1 and 2 and runing the experiment for 40 rounds with 10 clients per round, it was observed that the IID setting appears to be more stable than the N-IID, in which more fluctuations are present as it can be observed in Figure 2. This is an expected result given that in the N-IID setting, clients have different distributions and giving more or less importance to some might create more variance in the test accuracy. This is always, if it is done at random, as in this case, and not by some strategy, for example giving more importance to clients that have a better representation of all classes. Furthermore, it appears that client selection 0 performs better

in the N-IID setting than the other two. In the IID setting, giving more importance to 10% of the clients to be selected, so client selection 1, works better (see Appendix E). However, in the almost 20% gap in the accuracy of the IID was a bit strange, and it could be due to the low number of images per client. Thus, a new IID dataset was sampled, in which, each client had on average 22,000 images. Training for 20 rounds and with 10 clients per round in this new dataset and then plotting the test accuracy, far less fluctuations were observed than the previous case. The three clients selection methods in this new IID dataset have about 5% points of accuracy as difference(see AppendixE).

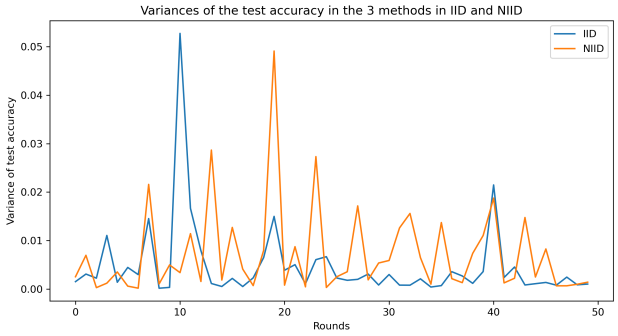


Figure 2. The variance of the test accuracy in Client Selection 0, 1 and 2 in IID and NIID.

4.7.2 Power of choice client selection

Initially, we experimented with 10 clients per round and the following values of m: 2,5,8 and 10. Maintaining all the other hyper-parameters as static, two different sets of experiments were conducted with 10 clients per round and 40 rounds, using two different seeds. In the first two experiments (see Appendix E), it can clearly be observed that the signal is too noisy, and it's very difficult to detect a clear trend in both cases. However, it can be observed than in both IID and N-IID scenarios, the m=2 is consistently providing lower accuracy results. Moreover, two tables are constructed in order to study the speed of growth of the accuracy in the baseline and in the different versions of **m** (see Appendix E). In the IID case, the baseline FedAvg reaches its peak accuracy at round 30 with seed 42 whereas all the other versions of **m** reach their peaks of $test\_accuracy \geq 0.9$ in by far less rounds, sometimes even in $\frac{2}{3}$ less rounds. Whereas, with seed 0 in IID, only **m**=10 gets an accuracy higher than FedAvg baseline in $\frac{1}{2}$ of the rounds that the baseline needs. In N-IID, similar trends were not noted, and the FedAvg baseline reaches earlier higher peaks. It has to be noted that these are local maxima and they do not infer some steady trends, given that the signal is very noisy, as noted earlier. The matter of the earlier peaks in IID with power of choice method could be due to the selection of the 10 clients not based on random uniform probability as in FedAvg but those that have more data have more chances to be selected. In N-IID, this may not be such a good idea because clients with more images are surely trained but, those clients may not have a good representation of all classes. To study the phenomenon further, and given that the theoretical expectation of power of choice algorithm is earlier convergence, the derivative of the test accuracy was singled out for IID FedAvg baseline and **m=2** and **m=5** for both seeds as exemplary, in order to understand the instantaneous rate of growth. It can be observed that the derivative of the test accuracy of the power of choice methods is always higher than that of FedAvg baseline but we cannot be sure if that is due to faster convergence to a higher accuracy or a very unsteady initial transient state in the power of choice algorithm (see Appendix E). Last but not least, another experiment was conducted with 40 clients per round and 80 rounds, hoping that in a longer series and with more images, patterns would be detected more easily. However, also in this case, the graphs were too cluttered and noisy to detect some clear pattern.

Table 2. Number of weights per layer in the CNN.

Layer	Number of Weights
conv1	832
conv2	51,264
fc1	6,423,808
fc2	127,038
Total	6,488,942

4.8. Pruning Experiments

Given the structure of our network described in Section 4.1, the network has a lot of weights, which are represented in Table 2. Thus, pruning might prove useful to reduce the number of weights and finding an acceptable trade-off with the drop in accuracy.

We performed a set of tests for the IID and N-IID settings, maintaining all the other hyper-parameters static besides the ones relevant here. Initially, we experimented with 10 clients per round for 40 rounds and the following three values for the unstructured pruning amount: 0.1, 0.01 and 0.001, for convolutional and linear layers, separately. As mentioned in section 3.4, the pruning process is set to start once 70% of the number of rounds has passed, permitting the model to learn before making it sparser. During the process, mean sparsity is measured across the clients that participate in that round and in the entirety of the network, regardless of which layer is set to be pruned. With respect to linear layers in the IID setting, it was detected that we could have a mean sparsity of up to 80% and still maintain an 80% accuracy with pruning amount 0.1 in the 33rd

Table 3. Test accuracy vs. Mean Sparsity in IID and N-IID with unstructured linear layers pruning.

Type	Amount_Prune	Mean Sparsity	Test_Acc
IID	0.1	99%	7%
N-IID	0.1	99.1%	5%
IID	0.01	58.9%	38%
N-IID	0.01	71.7%	5.5%
IID	0.001	25%	42%
N-IID	0.001	52.4%	5.5%

round. Naturally, the lower the pruning amount is, the less aggressive is the slope of the mean sparsity (see Appendix F). Regarding linear layers pruning in N-IID, with a prune amount of 0.1, the best trade off between mean sparsity and test accuracy goes to about 97% for both, in the 37th round. As it can be observed in 3, at the 40th round, in the N-IID setting there is consistently a higher mean sparsity and a lower accuracy than in the IID setting, for the same pruning amounts. In N-IID settings, where data distributions can be more skewed or heterogeneous, certain pruning techniques may result in higher sparsity due to the presence of outliers or specific patterns in the data. With regards to unstructured convolutional layers pruning, the mean sparsity barely moves beyond 7% and yet sharp accuracy drops are observed (see Appendix F). It must be emphasised that mean sparsity measures the sparsity in the entire network for all layers, regardless upon which layers pruning is being done. As noted in Table 2, the convolutional layers are not major contributors in terms of weights in the network compared to the fully connected linear ones. Yet, pruning the convoltuional layers' weights does not provide a good trade off between mean sparsity and test accuracy.

In structured pruning of convolutional layers, both in the IID and N-IID setting, it appears that like in the unstructured pruning, the mean sparsity is very close to 0 but unlike in unstructured pruning, the test accuracy does not appear to be impacted. More future tests need to be conducted in this direction to explore further implications.

As noted in section 3.3.2, power of choice client selection methods aim for a faster convergence. Consequently, we experimented merging unstructured pruning techniques on linear layers of IID setting, which proved to provide the best trade-offs between test accuracy and mean sparsity with amount to prune 0.1 and 0.01 with the power of choice client selection strategy with **m=2** and **m=5**.

As it can be observed in Figures 3 and 4, especially with pruning amount = 0.1, in round 40, we can have a mean sparsity of about 20% in the case of **m**=2 and about 50% in the case of **m**=5 with barely any drop in accuracy. This result, is very different from the case of vanilla IID FedAvg, in which a sharp decrease in accuracy was observed in round 40. However, unlike the vanilla IID FedAvg, here the mean

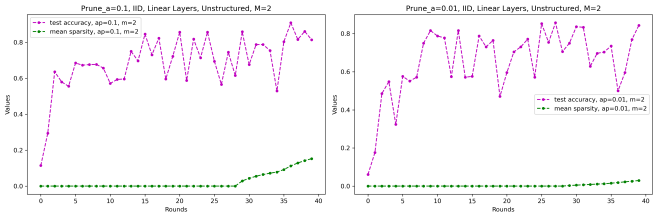


Figure 3. Test Accuracy vs. Mean Sparsity in linear layers unstructured pruning mixed with power of choice - IID, m=2.

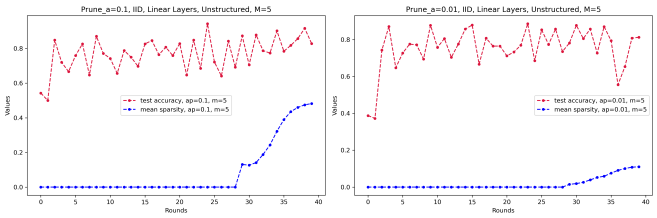


Figure 4. Test Accuracy vs. Mean Sparsity in linear layers unstructured pruning mixed with power of choice - IID, m=5.

sparsity is smaller. Probably, a contributing factor is not just the fact that we are training only on the clients with the highest loss but also the fact that unlike in vanilla FedAvg, we select **d** clients with a probability distribution based on the number of images and not just random uniform. Thus, in the IID setting, they might have more images to contribute and thus the network learns better without hampering the test accuracy, given their IID distribution. However, more tests would need to be done in the future, with more rounds and more clients, to fully understand how the two signals of test accuracy and mean sparsity behave.

5. Discussion

In this paper, the following set of the challenges and opportunities regarding the Federated Learning paradigm were addressed: heterogeneity in data distributions, domain generalization techniques, client selection strategies and pruning. Regarding the FEMINIST dataset in both centralised and federated learning, it was understood that a small learning rate was not functional and adding SGD with momentum and weight decay allowed the results to improve. In terms of client selection methods, what we can state with certainty is the fact that a higher variance was noted in the N-IID case as opposed to the IID one. However, due to computational limitations, it is difficult to speak with statistical confidence on the certainty of the results and to make statements such as "x method is better than y". We had to limit ourselves to: understanding the trend in time through visualizations of signals where possible, as opposed to just values in fixed moments in time, simple linear transformations such as taking the mean of three different seeds

as in section 4.3, descriptive statistics as taking the variance as in the case of clients selection as in section 4.7 or finding the rate of growth with the derivative to study the results. Nevertheless, the time element, in this case depicted by the number of rounds, is too short to see clear trends. In addition, it is also too impacted by initial noise and fluctuations, such as in the case of the Power of Choice Client Selection, which prevent the observation of trends. Thus, future tests are necessary, not only in terms of a higher number of rounds, but, also with more seeds, sufficient to do hypothesis testing and then denote with some statistical confidence when comparing methods.

References

[1] Sebastián Caldas, Sai Praneeth Kumar Duddu, Peter Wu, Tianrui Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: a benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018. 2, 4

[2] Y. J. Cho, J. Wang, and G. Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020. 1

[3] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. 2019. 1

[4] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W. H. Lee, K. K. Leung, and L. Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2

[5] A. Tuan Nguyen, Philip Torr, and Ser-Nam Lim. FedSR: A simple and effective domain generalization method for federated learning. In *Advances in Neural Information Processing Systems*, 2022. 2

[6] OpenMined. Federated learning: Exploring types of federated learning. https://blog.openmined.org/federated-learning-types/, 2024. 1

A. Appendix

Table 4. Learning Rate Tuning in Centralised Learning.

Nr	seed	B.S	L.R	M	W.D	Ep	Test Acc
1	42	32	0.0001	0	0	5	33
2	42	32	0.001	0	0	5	79
3	42	32	0.01	0	0	5	84
4	42	32	0.1	0	0	5	**85**
5	1234	32	0.0001	0	0	5	31
6	1234	32	0.001	0	0	5	78
7	1234	32	0.01	0	0	5	84
8	1234	32	0.1	0	0	5	**85**

Table 5. Batch Size Tuning in Centralised Learning.

Nr	seed	B.S	L.R	M	W.D	Ep	Test Acc
1	1234	64	0.001	0	0	5	70
2	1234	64	0.01	0	0	5	82
3	1234	64	0.1	0	0	5	85
4	42	64	0.001	0	0	5	70
5	42	64	0.01	0	0	5	84
6	42	64	0.1	0	0	5	86

Table 6. Epochs Number Tuning in Centralised Learning.

Nr	seed	B.S	L.R	M	W.D	Ep	Test Acc
1	1234	*32*	0.01	0	0	10	85
2	1234	32	0.1	0	0	10	86
3	1234	32	0.01	0	0	15	86
4	1234	32	0.1	0	0	15	85
5	42	32	0.01	0	0	10	85
6	42	32	0.1	0	0	10	86
7	42	32	0.01	0	0	15	84
8	42	32	0.1	0	0	15	85

Table 7. The best experimental Results in Centralised Learning.

seed	B.S	L.R	M	W.D	Ep	Test Acc
42	*32*	0.01	0.3	0.0001	12	87
42	*32*	0.01	0.5	0	12	87
1234	32	0.01	0.5	0	10	86
1234	32	0.01	0.5	0	15	86

B. Appendix

Table 8. IID Learning Rate and Epochs Number Tuning.

L.Rate	0.0001	0.001	0.01	0.1	0.2	0.5
Test_a-1ep	0.039	0.058	0.509	**0.765**	0.078	0.078
Test_a-5ep	0.059	0.162	0.75	0.794	**0.824**	0.088
Test_a-10ep	0.078	0.58	**0.784**	0.078	0.078	0.078

Table 9. N-IID Learning Rate and Epochs Number Tuning.

L.Rate	0.0001	0.001	0.01	0.1	0.2	0.5
Test_a-1ep	0.036	0.143	0.482	**0.696**	$\sim$0	$\sim$0
Test_a-5ep	0.146	0.396	0.771	**0.813**	0.125	0.125
Test_a-10ep	0.054	0.423	0.66	**0.75**	$\sim$0	0.071

Table 10. IID Weight Decay and Momentum Tuning.

L.R	0.01	0.01	0.1	0.1
Epochs	5	5	5	5
W.D	0	0.00001	0	0.00001
Mom	0.5	0.3	0.5	0.3
Rounds	20	20	20	20
Test_a	0.725	0.725	**0.88**	**0.88**

Table 11. N-IID Weight Decay and Momentum Tuning.

L.R	0.01	0.01	0.1	0.1
Epochs	5	5	5	5
W.D	0	0.00001	0	0.00001
Mom	0.5	0.3	0.5	0.3
Rounds	20	20	20	20
Test_a	0.718	0.718	**0.8**	**0.8**

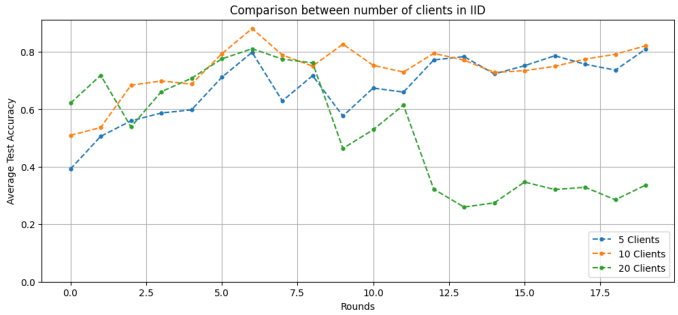


Figure 5. Comparison between different number of clients in IID.

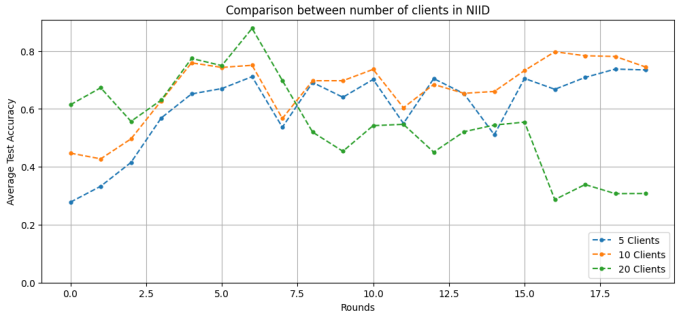


Figure 6. Comparison between different number of clients N-IID

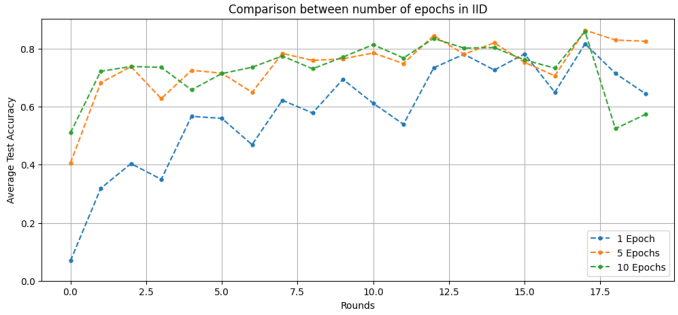


Figure 7. Comparison between different number of epochs IID

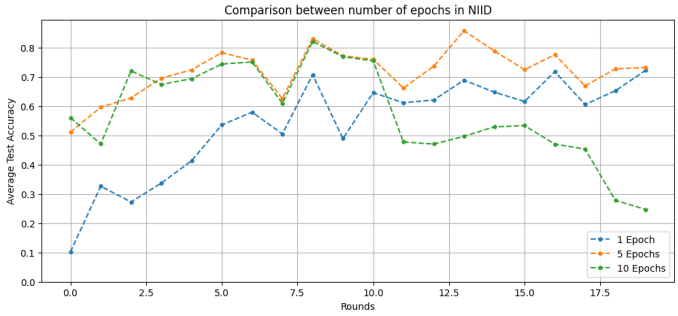


Figure 8. Comparison between different number of epochs N-IID

Table 12. N-IID Client Number Experiment with 3 seeds.

Seed	Client_Number	Test_Acc
0	5	0.94
0	10	0.8
0	20	0.8
42	5	0.57
42	10	0.6
42	20	0.06
0123456789	5	0.68
0123456789	10	0.828
0123456789	20	0.05

Table 13. IID Client Number Experiment with 3 seeds.

Seed	Client_Number	Test_Acc
0	5	0.8
0	10	0.82
0	20	0.88
42	5	0.85
42	10	0.81
42	20	0.05
0123456789	5	0.77
0123456789	10	0.81
0123456789	20	0.06

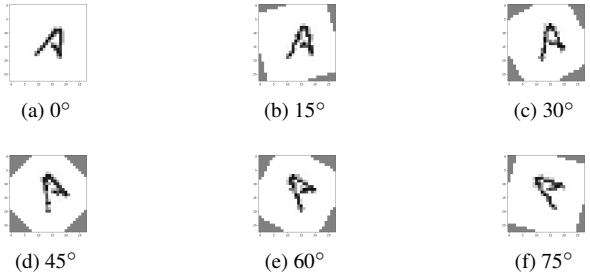
Table 14. IID Epoch Number Experiment with 3 seeds.

Seed	Client_Number	Test_Acc
0	5	0.59
0	10	0.78
0	20	0.87
42	5	0.55
42	10	0.81
42	20	0.75
0123456789	5	0.78
0123456789	10	0.88
0123456789	20	0.09

Table 15. N-IID Epoch Number Experiment with 3 seeds.

Seed	Client_Number	Test_Acc
0	5	0.52
0	10	0.57
0	20	0.63
42	5	0.67
42	10	0.69
42	20	0.35
0123456789	5	0.96
0123456789	10	0.93
0123456789	20	0.06

C. Appendix



(a) 0° (b) 15° (c) 30°

(d) 45° (e) 60° (f) 75°

Figure 9. A sample of the dataset with rotation artifacts.

Table 16. Centralized domain generalization experiments

lr	epochs	seed	Accuracy
0.001	20	42	74
		1234567890	72
	100	42	75
		1234567890	75
0.010	20	42	79
		1234567890	78
	100	42	78
		1234567890	77
0.100	20	42	4
		1234567890	4
	100	42	5
		1234567890	5

Table 17. Domain generalization with seed variation

LocalEpochs	Lr	rounds	numClients	seed	epochs	max test accuracy	round max accuracy
5	0.010	50	100	42	5	0.914	41
5	0.010	50	100	0	5	0.917	15
5	0.010	50	100	1234567890	5	0.917	38

Table 18. Best experiments for federated domain generalization

numClients	LocalEpochs	rounds	Lr	seed	epochs	max test accuracy	round max accuracy
50	5	50	0.010	1234567890	5	0.941	49
		100	0.010	1234567890	5	0.941	49
100	5	50	0.010	1234567890	5	0.917	41
		100	0.010	42	5	0.914	41

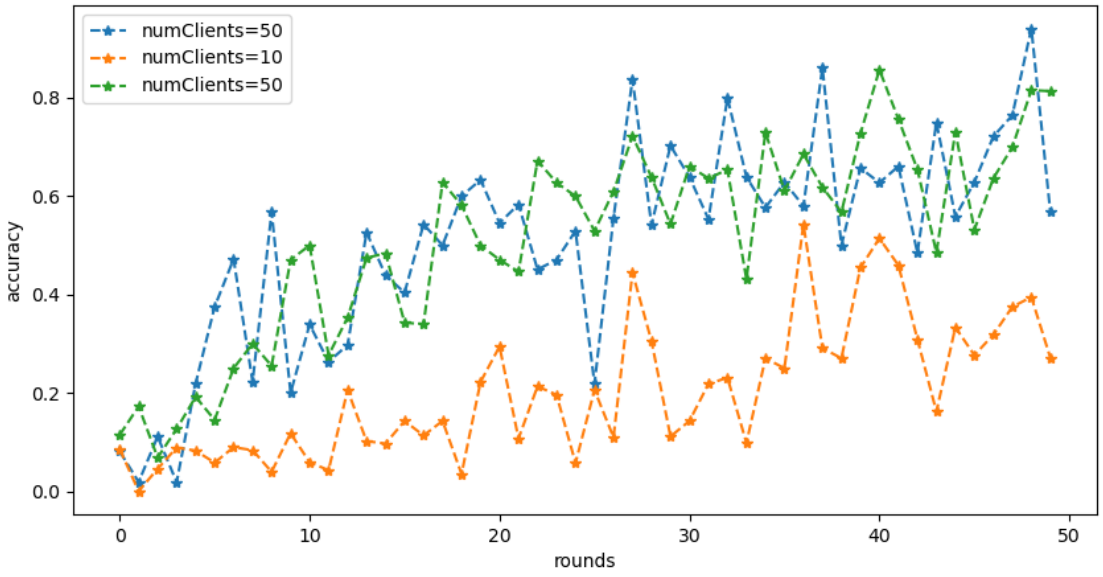


Figure 10. Domain generalization accuracy per round

D. Appendix

Table 19. Leave One out for federated setings

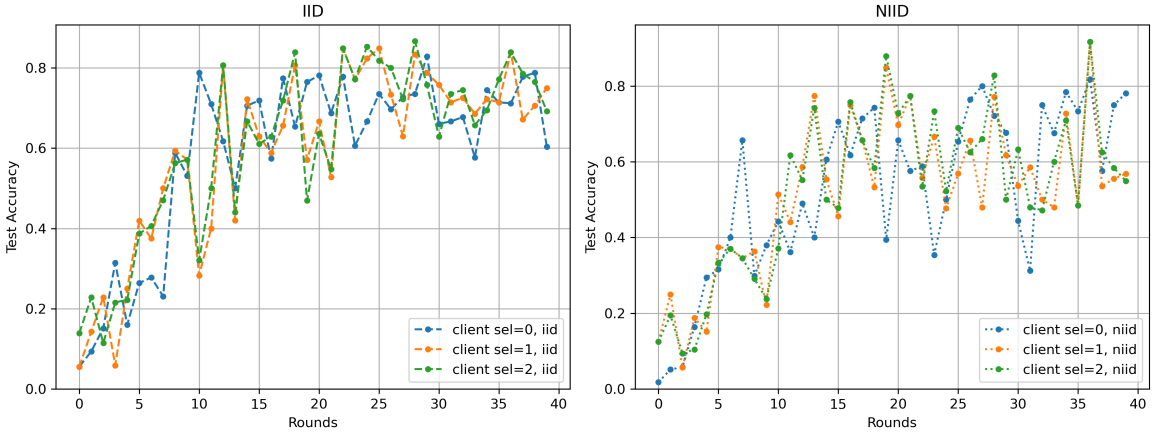
Seed		LocalEpochs	Lr	Rounds	Number of Clients	Epochs	Max test accuracy	Round max accuracy
42	0	5	0.010	50	50	5	0.609	49
	15	5	0.010	50	50	5	0.918	15
	30	5	0.010	50	50	5	0.872	22
	45	5	0.010	50	50	5	0.972	37
	60	5	0.010	50	50	5	0.971	19
	75	5	0.010	50	50	5	0.972	50
1234567890	0	5	0.010	50	50	5	0.742	44
	15	5	0.010	50	50	5	0.857	13
	30	5	0.010	50	50	5	0.857	12
	45	5	0.010	50	50	5	0.919	33
	60	5	0.010	50	50	5	1.000	16
	75	5	0.010	50	50	5	0.971	25

Table 20. FedSR leave one out experiments, with 20 rounds, 20 epochs, 100 clients

Lr	angle	max test accuracy	round max accuracy
0.010	0	0.140	3
	15	0.171	18
	30	0.242	13
	45	0.094	1
	60	0.135	18
	75	0.147	6

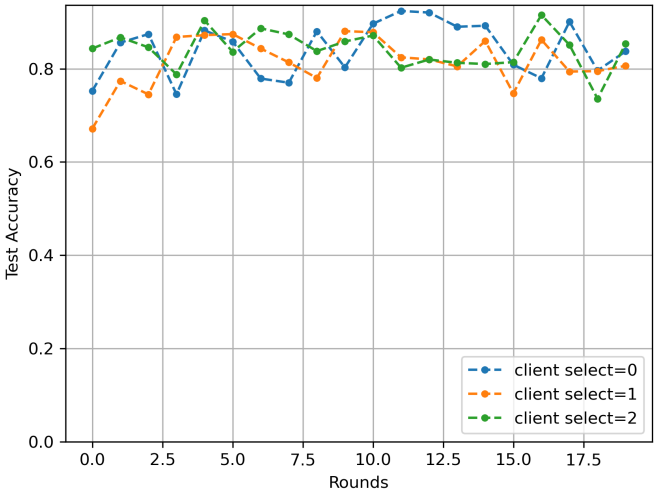
E. Appendix

The impact of the client selection method.

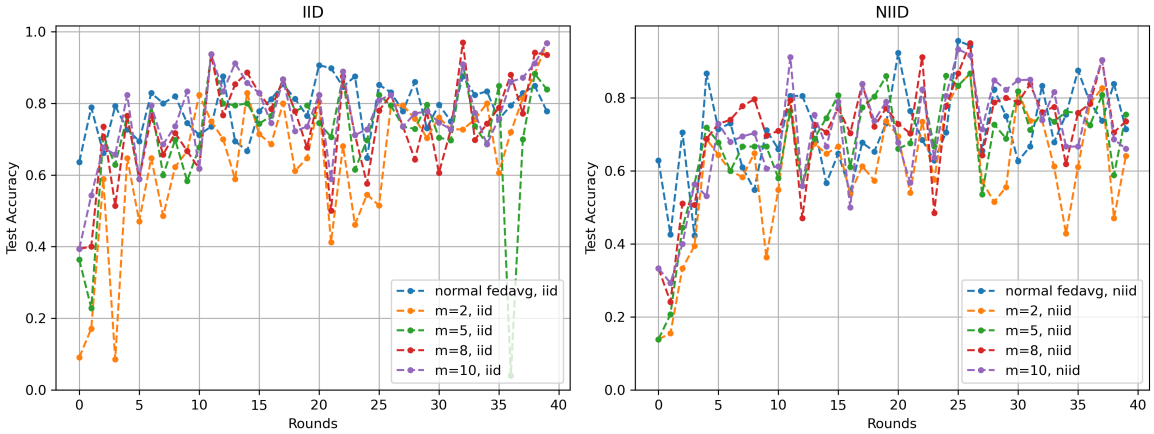


Local Epochs=5, Rounds=20, Client number=10 and Learning Rate=0.01

The impact of the client selection method in IID setting.



Power of Choice Client Selection Algorithm (seed 0).



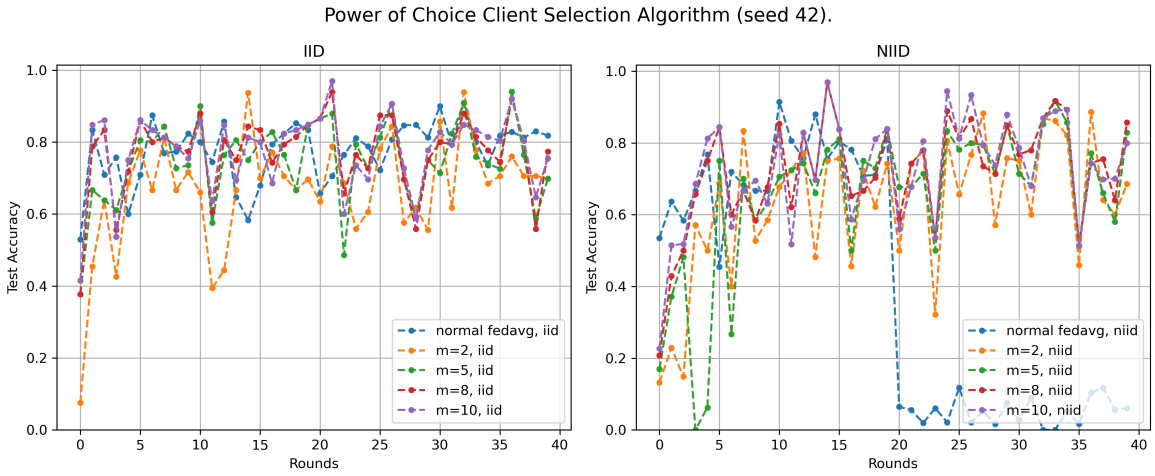


Figure 11. Power of Choice Selection Algorithm in IID and non-IID settings with seed=42.

Table 21. Power of choice vs. baseline FedAvg 10 clients per round with 40 rounds in IID.

Exp_nr	Max_acc	Round_nr	Method	Seed
0	**0.9**	**30**	**baseline**	42
1	0.93	14	m=2	42
2	0.9	10	m=5	42
3	0.93	21	m=8	42
4	0.96	21	m=10	42
5	**0.89**	**21**	**baseline**	0
6	0.96	39	m=2	0
7	0.9	34	m=5	0
8	0.96	32	m=8	0
9	0.93	11	m=10	0

Table 22. Power of choice vs. baseline FedAvg 10 clients per round with 40 rounds in N-IID.

Exp_nr	Max_acc	Round_nr	Method	Seed
0	**0.91**	**10**	**baseline**	42
1	0.88	27	m=2	42
2	0.91	33	m=5	42
3	0.96	14	m=8	42
4	0.96	14	m=10	42
5	**0.95**	**25**	**baseline**	0
6	0.86	26	m=2	0
7	0.86	19	m=5	0
8	0.95	26	m=8	0
9	0.93	27	m=10	0

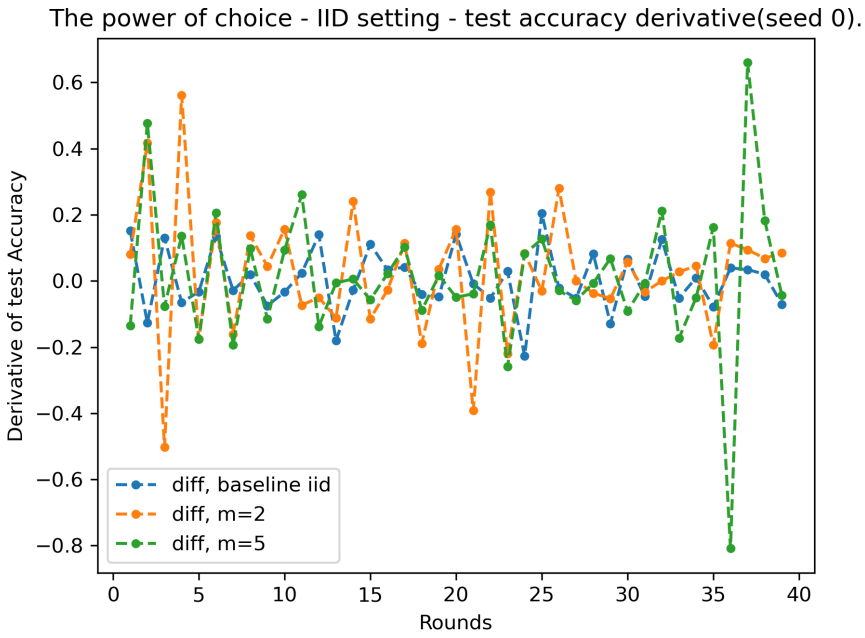


Figure 12. The derivative of test accuracy - Power of choice vs. FedAvg in IID (seed 0).

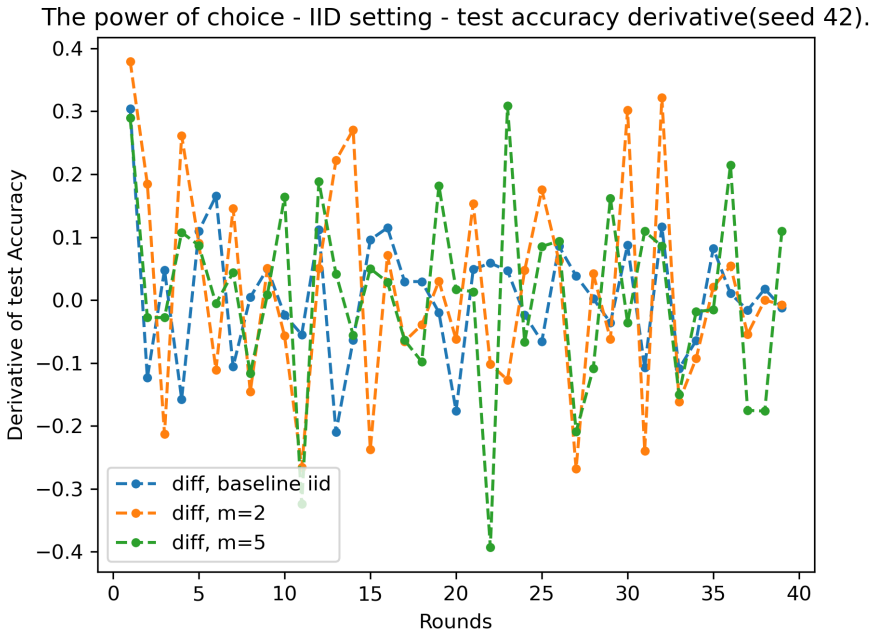


Figure 13. The derivative of test accuracy - Power of choice vs. FedAvg in IID (seed 42).

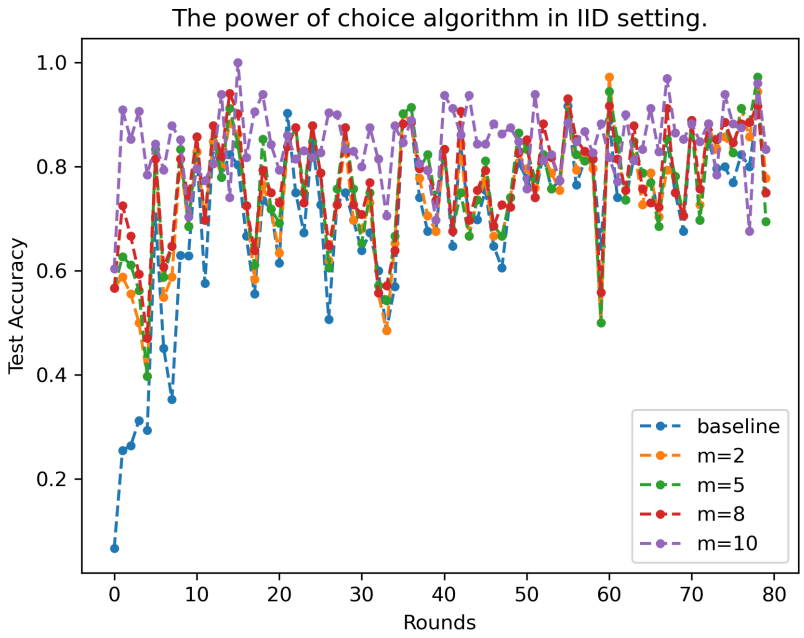


Figure 14. Power of choice vs. FedAvg with 40 clients and 80 rounds - IID.

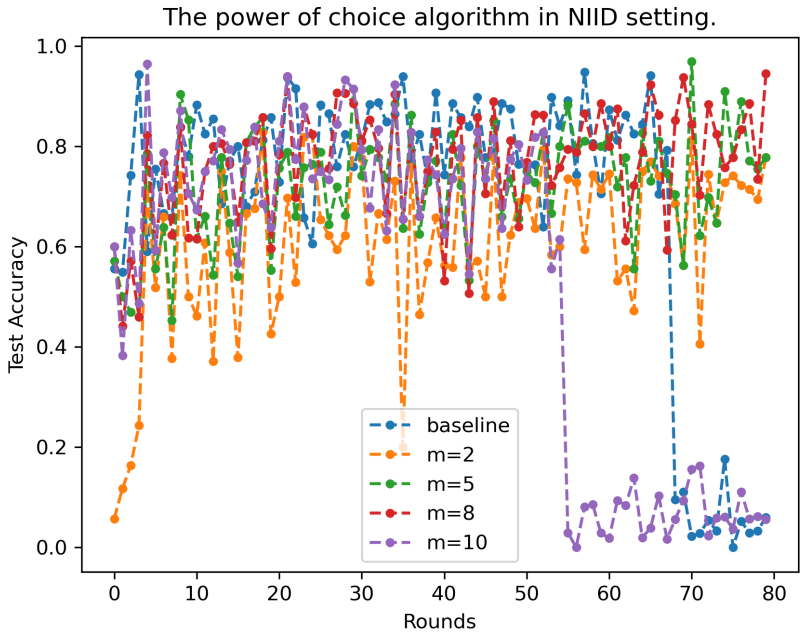


Figure 15. Power of choice vs. FedAvg with 40 clients and 80 rounds - NIID.

F. Appendix

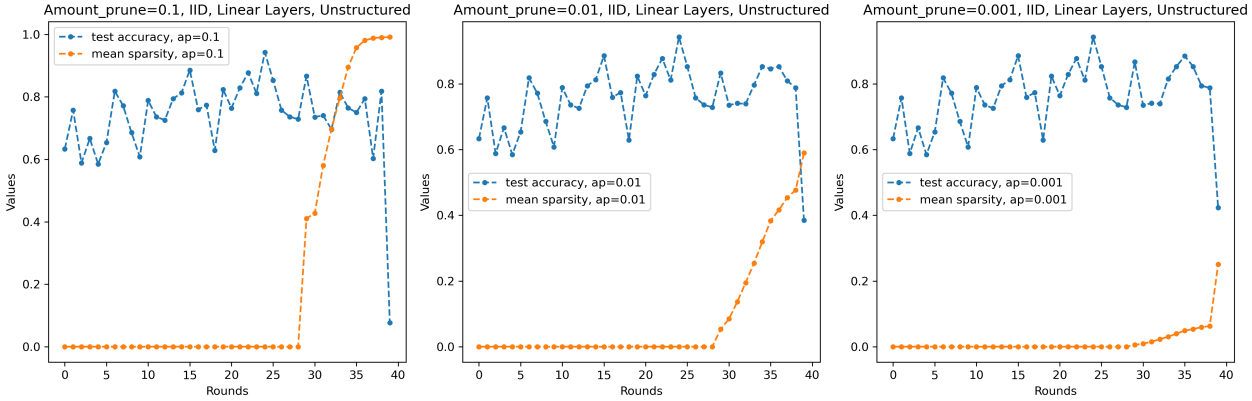


Figure 16. Test Accuracy vs. Mean Sparsity in linear layers unstructured pruning - IID.

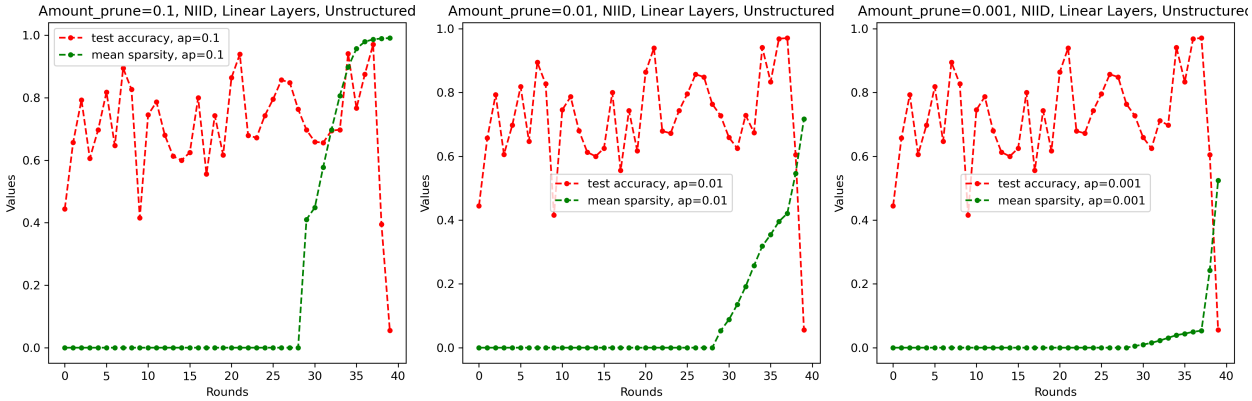


Figure 17. Test Accuracy vs. Mean Sparsity in linear layers unstructured pruning - NIID.

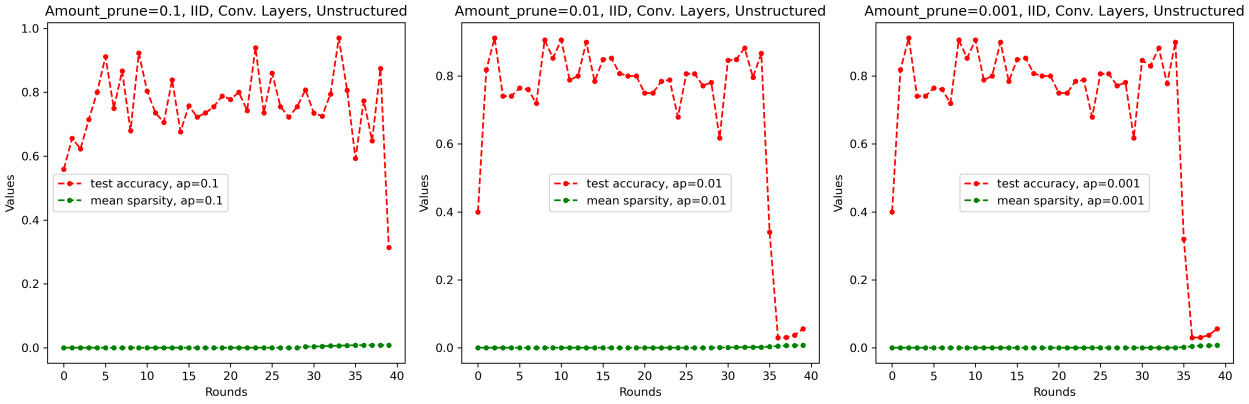


Figure 18. Test Accuracy vs. Mean Sparsity in convolutional layers unstructured pruning - IID.

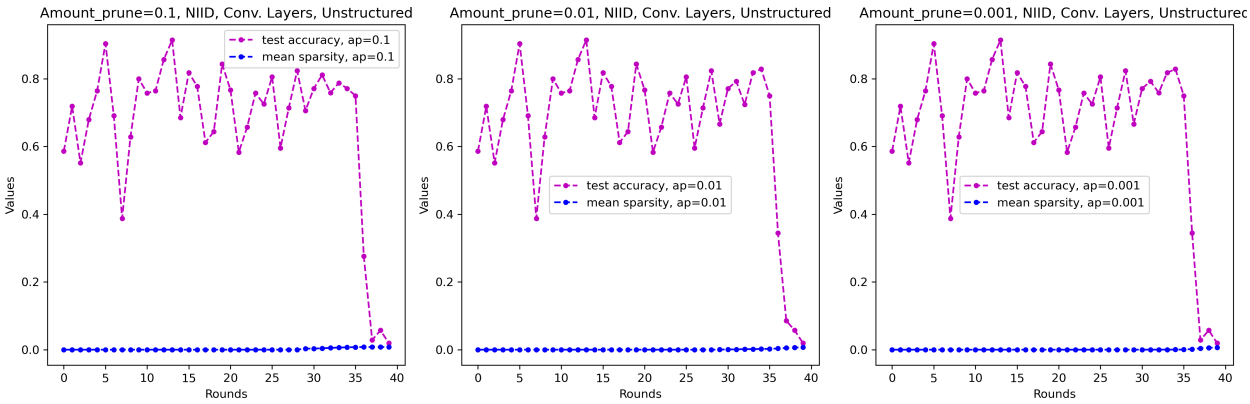


Figure 19. Test Accuracy vs. Mean Sparsity in convolutional layers unstructured pruning - NIID.

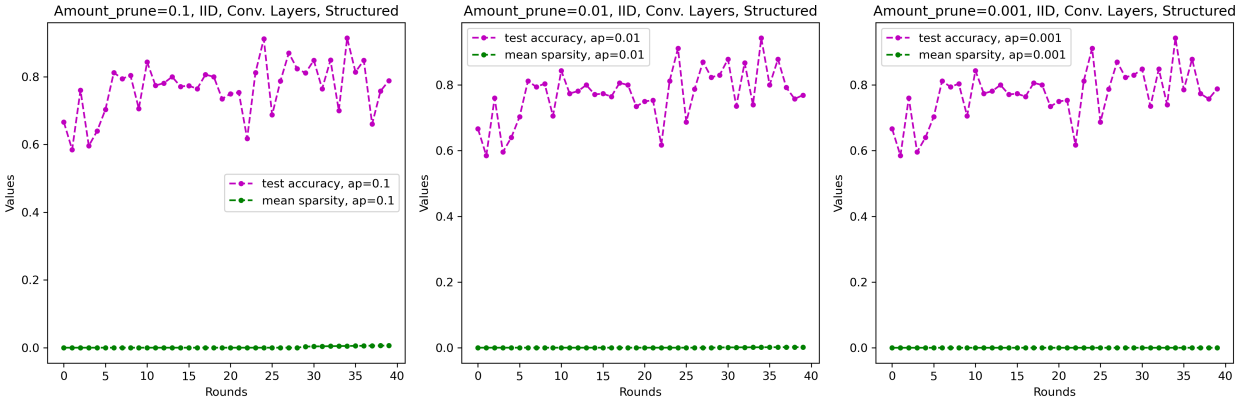


Figure 20. Test Accuracy vs. Mean Sparsity in convolutional layers structured pruning - IID.

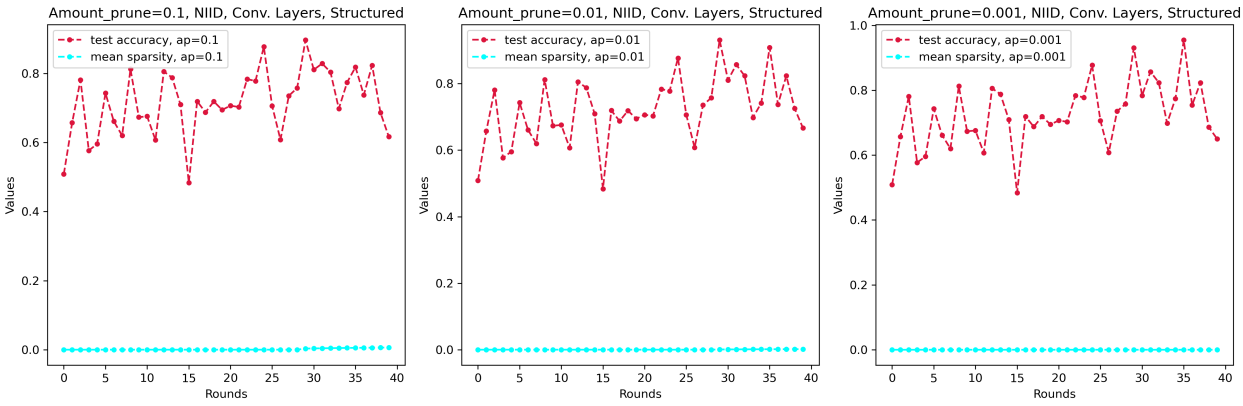


Figure 21. Test Accuracy vs. Mean Sparsity in convolutional layers unstructured pruning - NIID.