Federated Learning
Security, Privacy and Explainability in Machine Learning
Assignment 2

Group 16
Wojciech Michaluk - 12341799
Stephan Klein - 12114759

SummerSemester 2024

1 Introduction

We have chosen Topic 3.2.2: Federated Learning for Image Data. The objective is to evaluate the predictive effectiveness (e.g., accuracy) of federated learning compared to centralized learning. Federated learning aims to train a central model from data that remains distributed across multiple nodes. This approach ensures data privacy and security by keeping the data local while sharing only model updates.

In this project, we explore the application of federated learning using the PySyft and TensorFlow Federated libraries by training a CNN model on CIFAR10 [2] and compare its effectiveness and efficiency against centralized learning.

Our report contains a summary of the architecture, the conducted experiments and their results. For details about the implementation please consult the attached code package, which provides a set of documented Jupyter Notebooks with additional information.

2 Our solutions

2.1 Used CNN

For our purposes and our chosen dataset CIFAR10, we construct a small but efficient CNN architecture which allows fast convergence on limited hardware without GPU support, s.t it allows to conduct several experiments in a reasonable amount of time:

1. **Conv2D Layer**: 20 filters of size 5×5, ReLU activation, input shape $(32, 32, 3)$.
2. **MaxPooling2D Layer**: Pooling size 2×2.
3. **Conv2D Layer**: 50 filters of size 5×5, ReLU activation.
4. **MaxPooling2D Layer**: Pooling size 2×2.
5. **Flatten Layer**: Converts the 2D feature maps to 1D feature vectors.
6. **Dense Layer**: 500 neurons, ReLU activation.
7. **Dense Layer**: 10 neurons, Softmax activation for classification.

2.2 Architecture 1: Pysyft 0.2.9

We initially implement the federated learning scenario using (PySyft), which is based on PyTorch. Our approach involves training a CNN on the chosen dataset and distributing the dataset + training across several virtual nodes (workers).

Unfortunately, pysyft removed full support for federated learning in pysyft 0.3 (see [1]) and therefore we are limited to version 0.2.9 also dependent on an older version of torch (1.4.0), which were released between 2018 and 2020. Those limitations do not affect our academic scenario, apart from the slightly more complex setup, but should be considered when using pysyft in practice (the current promise is, federated learning will be back in upcoming pysyft 0.9 [1]).

Pysyft 0.2.9 supports Federated Learning for our academic setup via Virtual Workers, which allows to evaluate a federated scenario in a virtual way on a single execution host.

Documentation for Pysyft is sparse in general, however a good reference can be found in [3]. By following the example from chapter 5 we could execute federated learning on a set of Virtual Workers for the CIFAR10 dataset.

Based on the suprisingly good results, we suspect this older pysyft version does not implement any privacy preserving techniques like Differential Privacy so we choose to pursue more powerful and up-to-date architectures instead.

2.3 Architecture 2: Pysyft 0.8

We already noted that Federated Learning is not supported out of the box by a modern pysyft 0.8. The whole framework has been restructured, removing the elements like Virtual Workers and federated datasets we used in the first architecture.

The new architecture defines the entities Data Owner, Data Scientist, Datasite Server and Gateway Server and a general framwork to submit and execute code and data between them in a privacy preserving way. The examples provided in pysyft github do not include a federated scenario with multiple federated clients, they are limited to execution between two parties.

We still tried to federate the given xample for pytorch and to define a federated $@sy.syft\_function$ to be executed on multiple clients. Unfortunately, we did not succeed to define such a function, even after trying multiple approaches. (Some details are provided in attached code package) The lack of documentation presented an additional challenge.

We consider this architecture a failed experiment and we recommend interested users to wait for the upcoming release of pysyft 0.9 for using it in a federated learning scenario.

2.4 Architecture 3: Tensorflow Federated (TFF)

Finally, after the challenging experience with Pysyft we build a final architecture based on Tensorflow Federated. In contrast to pysyft, tensorflow federated provides extensive and up to date documentation and examples. TFF also offers advanced privacy preserving technologies like Differential Privacy as well as Model and Update Compression which we consider in our experiments.

For comparing Centralised and Federated Learning we construct two implementations provided in the corresponding jupyter notebooks.

For the **centralised benchmark model** we use a default implementation based on Tensorflow with the following steps:

1. Load the dataset via tf.keras.datasets
2. Normalize Pixel Values and One Hot Encode the target feature
3. Compile our CNN 2.1 with Adam Optimizer, CategoricalCrossentropy loss-function and CategoricalAccuracy metric

4. Fit the model for multiple epochs

For the **federated model** we first must create a federated dataset from the regular dataset loaded with TF, by slicing it by the number of federated clients and by the batch size. We utilize tensor slices for that.

Then the federated algorithm is built via: build_weighted_fed_avg or build_unweighted_fed_avg respectively where different optimizers and aggregators can be specified for our experiments.

To execute one round of federated learning we execute the IterativeProcess.next-function

For good comparison to centralised learning we don't execute the evaluation in a federated way (this would also be possible with TFF), instead we copy the weights to a centralized clone of the model and evaluate the holdout testing set in each round.

We conduct the following experiments to compare Federated Learning to Centralised Learning in various configurations:

1. Usage of different combinations of Client and Server Optimizers: TFF allows definition of two different optimizers. One for the training on the client, and an additional which is used when combining the results on the server side.
2. Comparing different hyperparameters for federated learning like batch size and the number of federated clients
3. Impact of activating : Differential Privacy with different levels of noise
4. Impact of : Model and Update Compression

3 Results

3.1 Pysyft 0.2.9

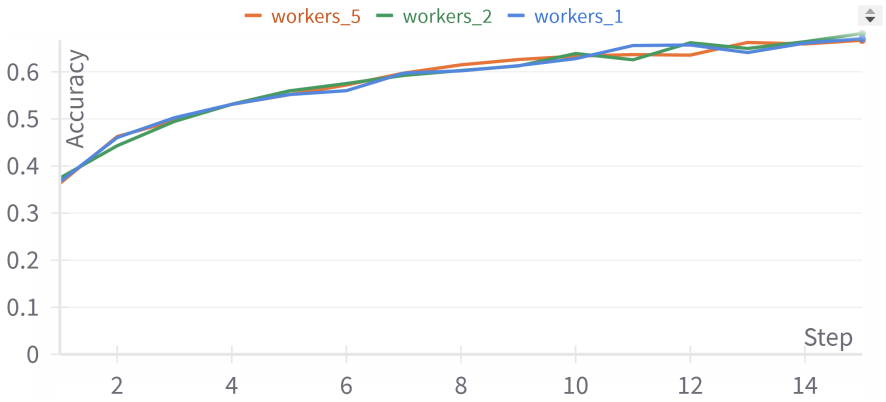

Figure 1: Accuracy by number of workers

Figure 2: Train loss

On the graph shown in Figure 2, it is demonstrated that independently of the number of workers, the CNN learns at a similar pace. The presented loss decreases similarly with an increased number of workers. Looking at the accuracy results in Figure 1, there are no changes in accuracy with an increased number of workers.

3.2 TensorFlow Federated

Note: All measures presented represent validation results of a holdout testset evaluated after a round of federated learning (unless otherwise mentioned)

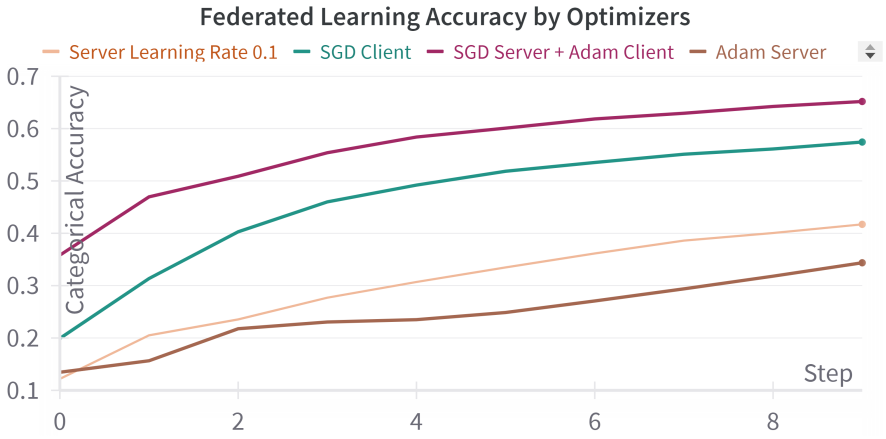

Figure 3: Various Optimizers for Federated Learning

To select the best model for federated learning, we started by testing various optimizer variants. Figure 3 shows that the best categorical accuracy was achieved with a combination of the SGD optimizer for the server (the central authority that orchestrates the federated learning process) and the Adam optimizer for the clients (devices that own and generate local data). Changing the learning rate did not yield better results, so we kept the default value. At the end we choose SGD and Adam as the baseline optimizers for federated learning.

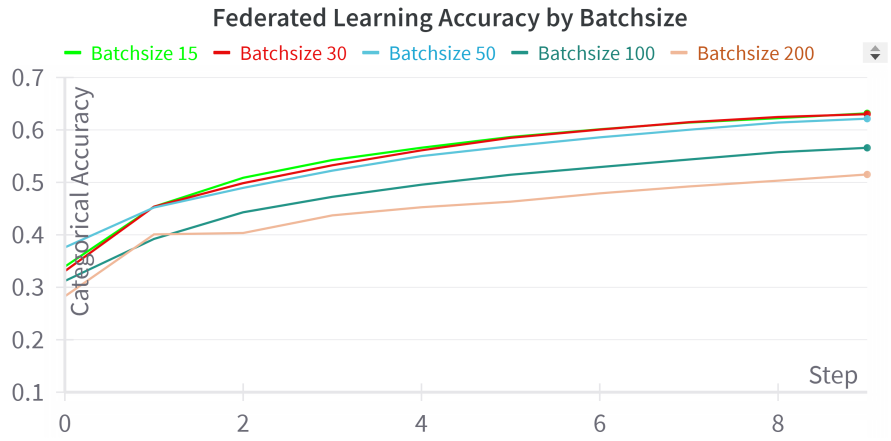

Figure 4: Influence of Batchsize on results in Federated Learning

Experimenting with batch size gave us the results presented in Figure 1. The best batch size for our model oscillates between 15 and 30 images per batch. A batch size of 50 images gave a slightly worse result. As the batch size increased above 50, the results worsened.

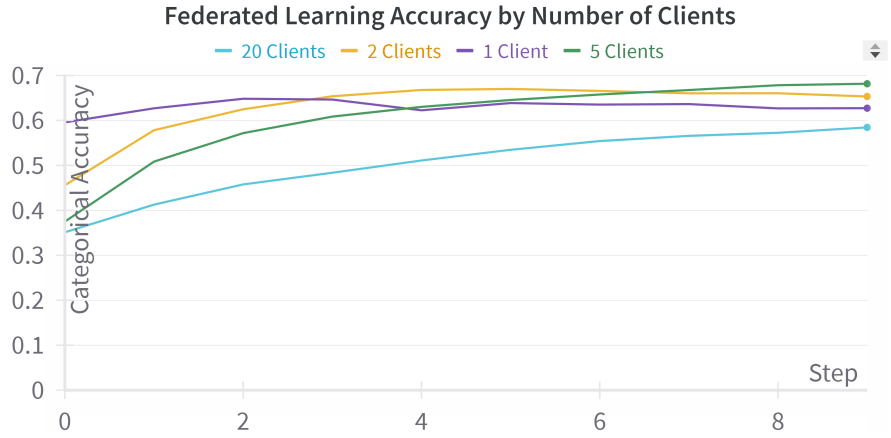

Figure 5: Most effective setup of Clients

Figure 5 shows that with a greater number of clients, the increase in accuracy slows down. Despite the initial success with a single client, the best result is achieved with 5 clients. Although it requires more time for training, the final results are better.

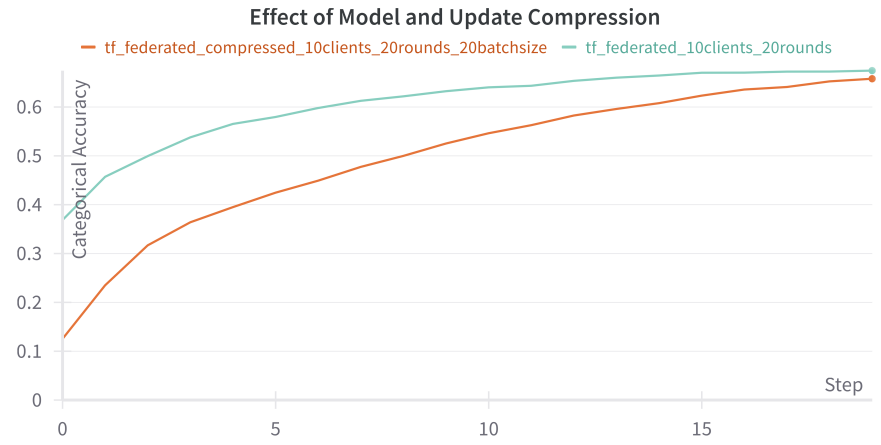

Figure 6: Effect of Model and Update Compression

Using TFFs Model and Update Compression to reduce the amount of transferred data, shows an effect on early rounds, however both models with compression and without compression converge at a similar point after 20 rounds of federated training.6

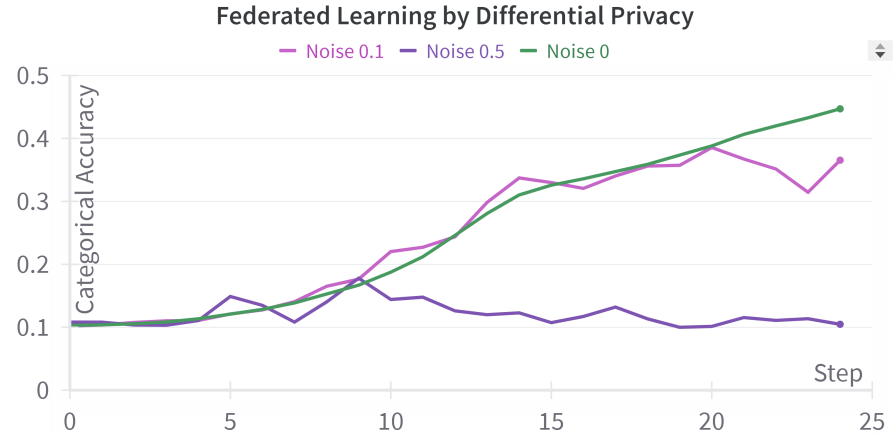

Figure 7: Trade-off between Differential Privacy and accuracy level

Data without noise is most effective for training. However, differential privacy is a desirable feature in federated learning to ensure the best security for data. Adding noise at a level of 0.1 results 7 in a decrease in accuracy from 45% to 38%.

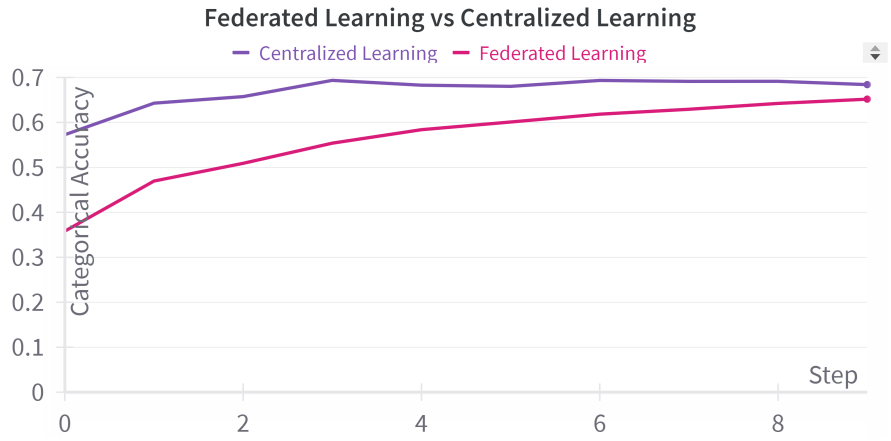

Figure 8: Federated Learning vs Centralized Learning

Comparison of Federated Learning and Centralized Learning for the best setup of both configurations. Centralized Learning achieves its maximum accuracy of 69% on the 3rd epoch, while Federated Learning reaches 65% after 10 epochs. This shows that the federated approach requires a much longer training time to achieve similar results 8 to the centralized approach.

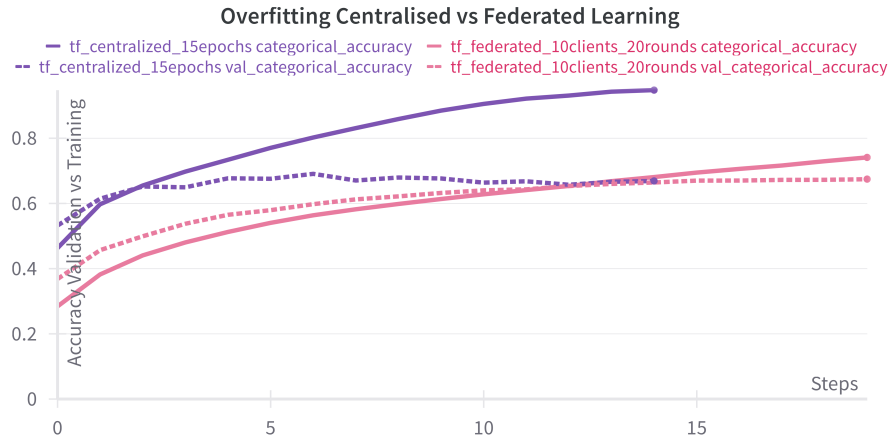

Figure 9: Overfitting in Federated Learning vs Centralized Learning

Further assessing the results we may look at the training vs validation (marked with _val suffix) accuracy. We observe that centralised learning is overfitting a lot faster than federated learning.

4 Results and Conclusions

Results presented on figure 1 showed for Pysyft 0.2.9 are highly different than for TensorFlow Federated. There is no decrease in accuracy with greater number of virtual workers. This might be a great success in real life scenario but it does not reflect real conditions. Without implementation of the differential privacy techniques, it is dummy federated learning. Additionally, the dataset is equally divided among several sources. This is hard to achieve in real scenarios.

In TensorFlow Federated, we divided the data equally between devices for easier testing. This library is reliable as it allows adjustment of optimizers, batch size, noise in differential privacy, and the number of clients. Our tests indicate that for the implemented model, a batch size between 15 and 30 is the most effective. Larger batch sizes worsen the results if tested model is not big enough. Increasing the number of clients generally results in an increased number of epochs required for training. This is due to the distribution of data across clients, which can lead to greater heterogeneity among clients. In our case, using 5 clients yielded the best results after 10 epochs of training. Our tests showed that tuning optimizers on both the server and client sides is beneficial. It leads to a faster descent of the gradient and helps in finding the minimum of the loss function more efficiently. Finally, we found that adding noise to ensure differential privacy involves a trade-off between accuracy and data privacy. Higher levels of noise weaken the original patterns, resulting in reduced accuracy.

Federated learning is a modern approach to training complex models using data collected from multiple devices. It typically results in slightly lower accuracy compared to centralized learning when using the same amount of data. However, it offers significant advantages, such as ensuring data privacy through mechanisms like differential privacy. Additionally, federated learning allows for the aggregation of much larger datasets for training, which can be beneficial for model performance.

References

[1] Federated Learning example with syft version 0.7 · Issue #7384 · OpenMined/PySyft (Mar 2023), https://github.com/OpenMined/PySyft/issues/7384, [Online; accessed 21. Jul. 2024]

[2] Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research) http://www.cs.toronto.edu/~kriz/cifar.html

[3] ur Rehman, M.H., Gaber, M.M. (eds.): Federated Learning Systems: Towards Next-Generation AI. Studies in Computational Intelligence, Springer Cham, Cham, 1 edn. (2021). https://doi.org/10.1007/978-3-030-70604-3, https://doi.org/10.1007/978-3-030-70604-3