HEC MONTRÉAL

MATH60629A MACHINE LEARNING I: LARGE SCALE DATA ANALYSIS AND DECISION MAKING

FEDERATED LEARNING

PROJECT REPORT

SIDDHANT ARORA 11302420
PHILIPPE BÉLIVEAU 11320035
KARAN JOSEPH 11304383

TABLE OF CONTENTS

INTRODUCTION

Machine learning is widely used in healthcare for monitoring and prognosis of rare and serious diseases. However, the full potential of Machine Learning is limited by several obstacles such as insufficient data, the likelihood of adverse attacks on the algorithm, and high learning costs. To address these issues, Federated Learning (FL) along with privacy techniques such as Differential Privacy (DP) is emerging as a promising approach.

What is Federated Learning?

Federated learning is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; instead, updates intended for immediate aggregation are used to achieve the learning objective. Data is generated locally and remains decentralized. Each client stores its own data and cannot read the data of other clients. A central server coordinates the training, but never sees raw data.

In the context of healthcare, clients can be different hospitals. Individual hospitals might not be comfortable in sharing their highly confidential and private data with other hospitals or they may not have a richer sample of dataset to work individually in solving a problem. This is where Federated Learning comes to rescue where the hospitals can work collaboratively towards a common Machine Learning problem without the need of sharing the data.

In Federated Learning, there is always a two-sided communication between the clients and the server. During this communication, an adversary can try to attack with an aim to retrace the original data. To prevent this from happening, Differential Privacy (DP) can be used to add an extra layer of security.

What is Differential Privacy?

Differential Privacy is a privacy-preserving technique for statistical data analysis that aims to protect the privacy of individual data contributors while still allowing accurate analysis of the aggregate data. The technique works by adding noise to the data before it is shared, which makes it difficult to infer any individual's data from the output. A mechanism is considered differentially private if the probability of any outcome occurring is nearly the same for any two datasets that differ in only one record. In hospitals, if the dataset relates to some x-ray scan images, then the noise can be added to the pixels of the images.

Problem Statement

This project aims to investigate the impact of privacy techniques on the performance of the Machine Learning model for medical image classification. Privacy techniques include the Federated Learning framework itself and Differential Privacy applied to the dataset. The objective is to compare the performance of the model under 3 different frameworks: typical centralized Machine Learning, Federated Learning, and Federated Learning with Differential Privacy. The hypotheses to be tested are: (1) Differential Privacy could affect the model's performance, (2) Centralized and Federated Learning could produce similar accuracy, but Federated Learning may require more training cycles.

RELATED WORK

As this research (V. Mothukuri, 2021) points out, the issue of privacy cannot be fully addressed by federated learning as it has its own vulnerabilities such as membership attacks, unintentional data leakage, generative adversarial network based inference attacks.

Furthermore, even if the data is anonymized, the collection of only a few data attributes may allow for patient re-identification (L. Rocher, 2019).

On the other hand, more neat privacy techniques are being explored, such as this paper which discusses Federated Learning’s patient privacy and security measures and its use of additional techniques such as blockchain and encryption to enhance IoMT (Internet of Medical Things) security and privacy (Ons, 2023).

The hospitals’ updates are susceptible to information leakage about the patients’ data due to model overfitting to training data (Carlini et al. 2019).

Differential privacy (Dwork and Roth 2014) limits an adversary’s certainty in inferring a patient’s presence in the training dataset.

Before optimizing the deep neural network, Gaussian random noise is added to the computed gradients on the patients’ data to achieve differentially-private stochastic gradient descent (DPSGD) (Abadi et al. 2016).

As per the analysis done in the research (Malekzadeh, 2019), differential privacy in combination with other techniques such as secure aggregation, homomorphic encryption can establish a better privacy-utility tradeoff (termed as *Dopamine* in the paper). Results on a diabetic retinopathy~(DR) task show that Dopamine provides a DP

guarantee close to the centralized training counterpart, while achieving a better classification accuracy compared to other baseline frameworks. Experiments on a benchmark dataset of DR images show that Dopamine can outperform existing base-lines by providing a better utility-privacy trade-off.

This is what we aim to see in our analysis too, wherein we expect to see similar results between Machine Learning and Federated Learning frameworks and a change in performance in the Federated Learning and Differential Private framework with more robust privacy.

ABOUT THE DATASET

Retinal optical coherence tomography (OCT) is an imaging technique used to capture high-resolution cross sections of the retinas of living patients. Approximately 30 million OCT scans are performed each year, and the analysis and interpretation of these images takes up a significant amount of time (Swanson and Fujimoto, 2017).

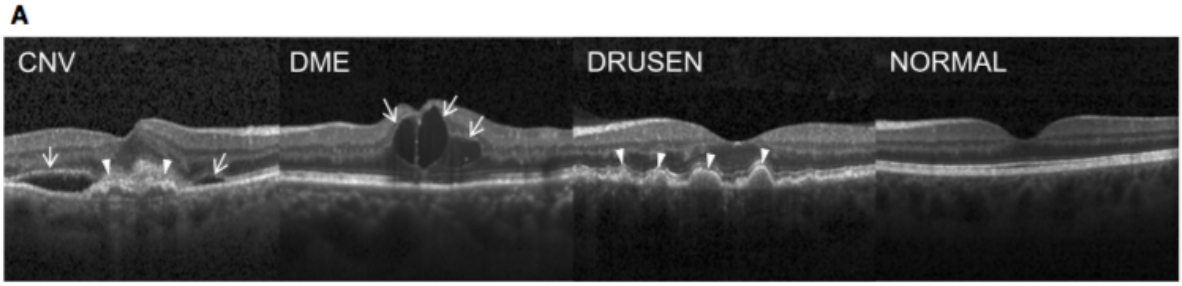


Figure 1: Sample images from the four classes

The data set contains about 84,500 images with labels and each image has a width of 800 to 1,000 pixels. Images are labeled as (disease)-(randomized patient ID)-(image number by this patient) and split into 4 directories: CNV, DME, DRUSEN, and NORMAL. The first 3 classes represent the scans of the retina which are abnormal and the last class represents the image of the normal retina.

The dataset was already divided into train, test and validation as follows: CNV (37206 train images), DME (11349 train images), DRUSEN (8617 train images) and NORMAL (26316 train images). Each class contains 242 test images and 8 validation images.

Data Preprocessing

Some data transformations have been applied to the images such as resizing the images, cropping the center of the resized images, adjusting the sharpness of the images, randomly flipping the images horizontally, normalizing the pixel value of the

images by using calculated mean and standard deviation, converting the input images to PyTorch tensor and so on. In the Federated Learning Differential Privacy framework, an additional transformation step is to add noise to the pixels of the images. Lastly, splitting of the data among the four clients is completely random.

MACHINE LEARNING MODEL

Model Architecture (Convolutional Neural Network)

We use convolutional neural networks (CNN) for our image dataset. In general, a Convolutional Neural Network (CNN) is a type of deep neural network that is commonly used for image classification, object detection, and image segmentation tasks. The main feature of a CNN is the convolutional layer, which is responsible for feature extraction from the input image.

We create Retina_Model with the following architecture:

```
Retina_Model(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (dropout1): Dropout2d(p=0.2, inplace=False)
  (dropout2): Dropout2d(p=0.2, inplace=False)
  (fc1): Linear(in_features=12544, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=4, bias=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1
)
```

Figure 2: Architecture of Retina_Model

1) First convolutional layer 3 input channels, 32 output channels, a kernel size of 3x3 and stride of 1.
2) Second convolutional layer 32 input channels (output of the first layer), 64 output channels, a kernel size of 3x3 and stride of 1.
3) Dropout 1 and 2 that will randomly set a fraction of the activations in the input tensor to zero during training, with probability of 0.2.
4) First fully connected layer that will transform the flattened input tensor into a tensor of size 128.
5) Second fully connected layer that will transform the output of the first fully connected layer into a tensor of size num_classes (number of classes) i.e. 4.
6) MaxPool creates a max pooling layer with a kernel size of 2 and stride of 2, reducing the spatial dimensions of the output tensor by a factor of 2.

During training and testing, Cross Entropy loss is used.

Hyperparameter Tuning and Regularisation

Hyperparameter tuning is done using Optuna on the validation set. Optuna is a hyperparameter optimization software framework that is able to easily implement different state-of-the-art optimization methods to perform hyperparameter optimization rapidly with great performance. By default, Optuna implements a Bayesian optimization algorithm (TPE - Tree-structured Parzen Estimators) but it can be easily switched to other existing algorithms in the package. We chose a Bayesian algorithm because it enabled us to optimize over a larger search space in a quick and efficient manner. This would have been either ineffective or time consuming if we chose a traditional hyperparameter optimization algorithm such as GridSearch or RandomSearch. Unlike traditional techniques, Bayesian optimisation algorithms have the flexibility to intelligently focus on regions that have the greatest potential to produce the best results on the validation set without evaluating the performance of all possible combinations of the hyperparameters. It does this by using the performance of the previous trials to choose the next round of hyperparameters in an experiment.

The following hyperparameters were optimized with their optimal values mentioned in parentheses: Learning Rate (0.03134), Momentum (0.9) and Dropout (0.2). Optimiser was kept as Stochastic Gradient Descent (SGD).

In terms of regularization (to prevent overfitting and improve generalization performance), the following two techniques were used: Dropout and Early Stopping. Dropout, as mentioned briefly before, is to randomly drop out some neurons in the network during training, forcing the remaining neurons to learn more robust features. Early stopping is to stop the training of the network when the validation error stops improving, even if the training error continues to decrease.

FEDERATED LEARNING ALGORITHM

Flower Client

In order to update the model parameters in Federated Learning, the server shares the model settings with the client, who updates their copy of the model using this information. The client then trains the model on its local data and sends the updated model settings or gradients back to the server. This is achieved with two helper functions within the Class Flower Client: set_parameters and get_parameters. Then, using a function called evaluate, when it is time to evaluate the model on the local data, the client receives the model parameters from the server and evaluates the model on the local data. Finally, it sends back the evaluation result to the server (Flower).

Simulation

Before starting the simulation, we define our strategy (fl.server.strategy.FedAvg) that allows us to achieve convergence in our Federated Learning framework. We want to minimize each local loss function (specific to each client with a different distribution), in order to minimize a global loss function). Therefore, we need to find a common statistical model that will solve this distributed optimization problem.

To do so, the strategy we implement is called FedAvg. FedAvg attempts to minimize communications by only considering periodic communications between the client and the server, which occur during rounds. Between rounds, a client performs several local steps. These local steps are gradient descent steps on the client's local loss function. At the end of the round, each client uploads its local model to the server, which averages these models and sends it back to the client. This process repeats for several rounds. Here, we do it for 10 rounds.

We also define other parameters in the strategy function, such as fraction_fit and min_fit_clients, which correspond to the percentage of clients available for training and the minimum number of clients to use for training. Another argument we specify is evaluate, evaluate refers to what we discussed above in Flower Client, in our case we are doing a server side evaluation, it works the same way that evaluation in centralized machine learning does.

Then, we can start the simulation using the class: fl.simulation.start_simulation specifying the number of clients (4 in our case), the number of training cycles and some configuration parameters.

We start the flower server using the gRPC connection which is a fast API. It allows client and server applications to send information to each other. We initialize the server settings by passing the initial parameters directly to the strategy using the parameters in our model. Then the training cycles begin, we define a function (fit_config) that allows the server to send configuration values to the client, such as the number of local hours.

Subsequently, we initiate the training process by selecting client samples, fitting the model on local data, transmitting model weights to the server, computing model averages on the server with a designated strategy, sending the revised weights back to the client, evaluating the local data, and transmitting the evaluation outcome back to the server. This is repeated for 10 rounds as mentioned above.

Laplace Mechanism for Differential Privacy

In order to achieve differential privacy, we add some noise to the pixels of the images and this is done via laplace mechanism.

$$\mathcal{M}_{\mathrm{Lap}}(x, f, \epsilon) = f(x) + \mathrm{Lap}\left(\mu = 0, b = \frac{\Delta f}{\epsilon}\right)$$

Figure 3: Laplace Equation

where mu is the expectation of the Laplace distribution and b is the scale parameter. Scale is sensitivity over epsilon. Sensitivity is a measure of how much the output of a query (e.g., the mean, sum, or variance) can change with the addition or removal of a single data point from the dataset. It is typically defined as the maximum possible change in the output of a query due to the addition or removal of a single data point. Epsilon, also known as the privacy budget, is a measure of the privacy guarantee provided by a differentially private algorithm. It represents the level of privacy protection offered by the mechanism.

In the context of our image dataset, the sensitivity is typically determined by the maximum possible change in the pixel values due to the addition or removal of a single image. Once the sensitivity is calculated which comes out to be 0.05, the Laplace mechanism adds noise to the pixel values of the images. We use epsilon = 0.2 The noise is generated from a Laplace distribution, which is a probability distribution with a heavy tail. The Laplace distribution is characterized by two parameters (as seen in the equation above): a location parameter (mu) and a scale parameter (b). The location parameter determines the center of the distribution, and the scale parameter determines the spread or concentration of the distribution. The scale parameter (b) of the Laplace distribution is chosen based on the sensitivity of the dataset and the desired privacy level.

RESULTS

The performance of the centralized machine learning model improves in terms of validation and training losses over the number of epochs. This is shown in the figure 4 below where we can also observe that there is no overfitting.

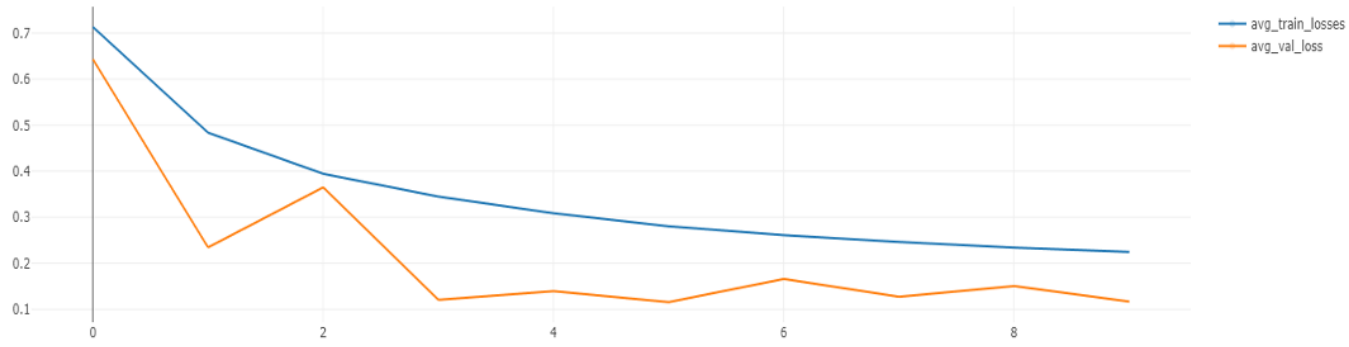


Figure 4: Training and Validation Losses for the centralized machine learning model over epochs

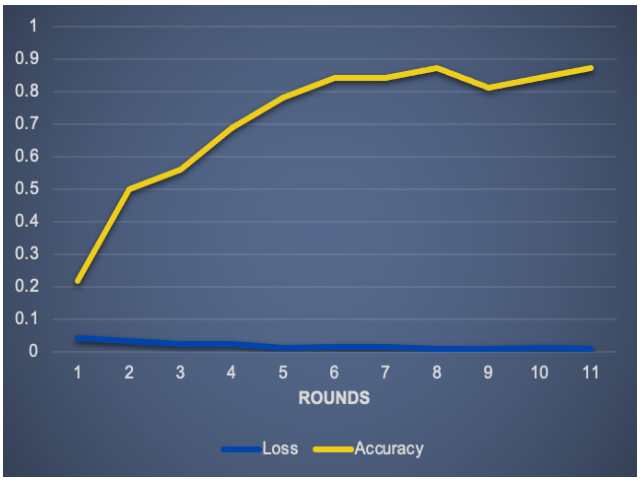
The accuracy obtained on the validation set is as follows:

1) Centralized Machine Learning: 94%
2) Federated learning: 93.5%
3) Federated Learning with Differential Privacy: 81.25%

These results are in line with our hypotheses. We see a slight decrease in accuracy (0.5%) for the Federated Learning framework compared to centralized Machine Learning framework. This could be due to the fact that in Federated Learning, each client only has access to their local data, which can be limited in size and diversity. This can result in a smaller and less diverse dataset compared to a centralized dataset, which can lead to lower model accuracy. Moreover, in Federated Learning, each party may have different data distributions or feature representations, which can lead to a more heterogeneous dataset compared to a centralized dataset. This can result in a more challenging learning task, which can lead to lower model accuracy. Lastly, as expected, with noise being added we expect the accuracy to go down even further and that's why the accuracy of Federated Learning with Differential Privacy model is the least.

On the validation set, we can also observe for the Federated Learning and Federated Learning with Differential Privacy models that the performance improves (lower loss and

higher accuracy) as the number of rounds increases. This is shown in Figures 5 and 6 below.



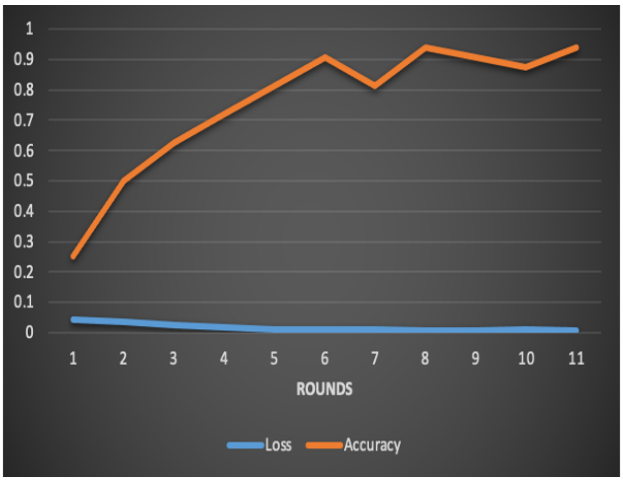


Figure 5 and 6: (Left): Loss and Accuracy across rounds for Federated Learning model. (Right): Loss and Accuracy across rounds for Federated Learning with Differential Privacy model.

Following is the part of classification report for all 3 frameworks on the test set which gives precision, recall and F1 Score for 4 classes along with overall accuracy:

	Centralized			Federated			Federated + Differential Privacy		
	Precision	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
CNV	0.876	0.996	0.932	0.699	0.996	0.821	0.814	0.979	0.889
DME	0.996	0.983	0.990	0.995	0.901	0.946	0.978	0.909	0.942
DRUSEN	1.000	0.868	0.929	1.000	0.591	0.743	1.000	0.636	0.778
NORMAL	0.992	1.000	0.996	0.927	1.000	0.962	0.812	1.000	0.896
Accuracy	0.962	0.962	0.962	0.872	0.872	0.872	0.881	0.881	0.881

Table 1: Results of the 3 frameworks on the test set

From the above table, model accuracy for the centralized Machine Learning framework is 96.2%, for Federated Learning framework it is 87.2% and when Differential Privacy is applied the accuracy obtained is 88.1%. Here, on the test set, the Federated with Differential Privacy model performs slightly better than the Federated without Differential Privacy model. The observed difference in performance between the two approaches may be attributed to various factors, such as the level of noise added to the data or the efficacy of the shuffling procedure used to generate each client's local dataset distribution. However, the exact cause of the discrepancy remains undetermined. Therefore, this finding should be interpreted with caution, as it requires further validation through repeated simulations to obtain a more accurate estimate of the approaches' true performance. Unfortunately, the limited time and resources available prevented us from conducting multiple simulation runs at this stage.

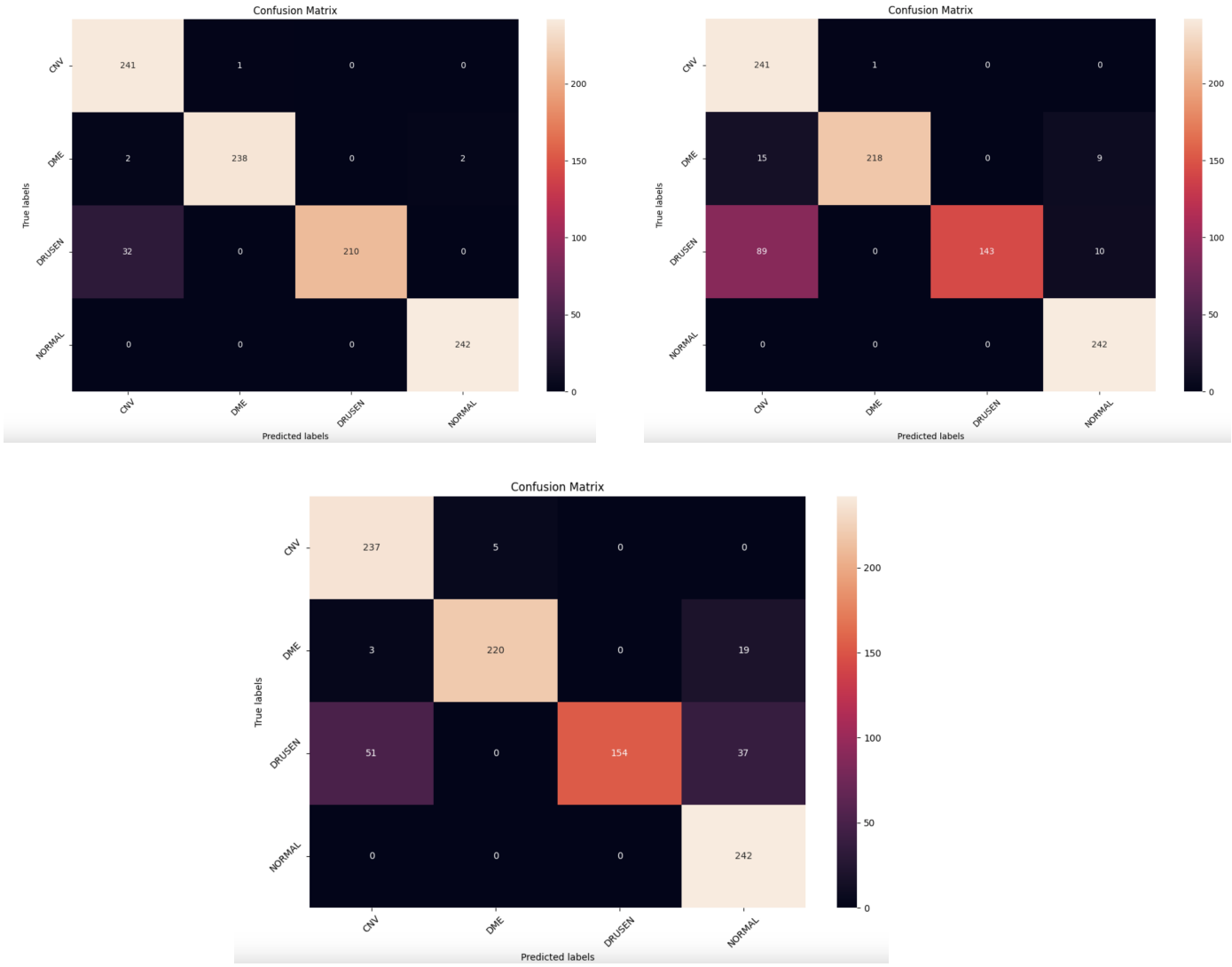


Figure 7, 8 and 9: (Top left): Confusion matrix for the centralized Machine Learning model. (Top Right): Confusion matrix for the Federated Learning model. (Bottom): Confusion Matrix for the Federated Learning with Differential Privacy model

From the above confusion matrices (Figures 7, 8 and 9), It can be seen that the model faces some difficulties in differentiating between CNV and DRUSEN classes because when it is actually DRUSEN, the model predicts as CNV. This could be due to the fact that the scans (images) of these both classes look very similar when compared with the other classes. Also, DRUSEN has the lowest number of training images giving the model less instances to learn more about this class. In the third framework of Federated Learning with Differential Privacy, the model also confuses a bit between DRUSEN and NORMAL which could be due to the added noise.

CONCLUSION

Machine Learning and Federated Learning models perform similarly whereas when we apply Differential Privacy, we see a decrease in accuracy because of the addition of noise. With differential privacy, it is important to think about privacy-accuracy trade-off. In general, the more noise that is added to the data, the greater the level of privacy protection, but the lower the accuracy of the resulting model. Conversely, reducing the amount of noise added can improve accuracy, but may also reduce the level of privacy protection. Thus, finding the right balance between privacy and accuracy is crucial in differential privacy federated learning. This balance can be achieved by carefully tuning the amount of noise added to the data (epsilon), as well as by selecting appropriate model architectures and hyperparameters.

In terms of deployment, with this Federated Learning Differentially private model, the hospitals can easily collaborate with other hospitals without the worry of sharing their sensitive data and being attacked by an outside adversary. The model could be used to detect eye diseases such as diabetic retinopathy, age-related macular degeneration, glaucoma etc. The federated learning model could also be integrated into telemedicine platforms, enabling remote eye exams and disease screening. It can be integrated into mobile apps that allow patients to take pictures of their eyes and receive real-time feedback on potential eye diseases. Thus, the model could be deployed as a cloud-based API that hospitals and other healthcare institutions could use to screen patients for eye diseases. The API could be integrated into electronic health record (EHR) systems, telemedicine platforms, and mobile apps, making it widely accessible to healthcare providers and patients.

Such a model can also be used in other sectors apart from Healthcare. For example, in Financial Services, Federated Learning and Differential Privacy could be used to build models for fraud detection, credit risk assessment, and customer behavior analysis. Banks and financial institutions could use these techniques to train models on their customers' financial data while keeping it secure. Similarly, in E-Commerce, online retailers can use it to build models that recommend products to customers based on their past purchases and browsing behavior. This can help retailers personalize their offerings without risking the privacy of their customers' data.

FUTURE DIRECTION

The main challenges of federated learning arise when it comes to achieving linear convergence rates. For example, clients may have different loss functions as their distributions vary, as well as clients may have different operating speeds due to variability in hardware and connectivity. Finally, low bandwidth channels and limited battery power reduce the number of communication cycles and send compressed updates (Mitra, 2021).

With the FedAvg strategy we use, convergence can be more challenging in real-world situations, as the more local steps a client takes, the more it drifts towards its own individual minimizer. Due to this client drift, algorithms such as fed-AVG may end up converging to incorrect points. Furthermore, this situation becomes even more dangerous due to the heterogeneity of the system. The number of local steps can vary significantly from one set of clients to another and from one run to another. This can lead to convergence to incorrect points. Other algorithms, such as the Fed-LIN algorithm, have been proposed to facilitate convergence in LF (Mitra, 2021).

Due to very limited research and literature available on hyperparameter tuning in a Federated Learning framework, in this project, for simplicity, the optimal hyperparameters found for the centralized Machine Learning model were used as it is for Federated Learning models. However, ideally, as a future point of action, it would be good to try hyperparameter tuning for Federated Learning too. Due to very long training times, the optimiser was not a part of hyperparameter tuning and it was set as Stochastic Gradient Descent. In the future, optimisers can be optimized too to find an optimal optimiser among, let's say for example, SGD, AdaGrad, AdaDelta, Adam etc.

To achieve differential privacy, we have used local differential privacy where the noise is added to the raw dataset. However, a fourth scenario can be tested where the noise is added to the computed gradients known as Differential Privacy Stochastic Gradient Descent (DP-SGD) and compare the performance with the existing 3 frameworks.

In a practical scenario, since the model faces a difficulty to differentiate between CNV and DRUSEN classes, it would be a good idea to collaborate with more hospitals (clients) especially with those who might have a richer repository of DRUSEN retinal scan images and then re-run and evaluate the results of the models again.

REFERENCES

- ACM Comput. Surv., Vol. 1, No. 1, Article . Publication date: November 2021. https://arxiv.org/pdf/2111.08834.pdf

- Mitra, Aritra. Rayana Jaafar, George J. Pappas, Hamed Hassani. Linear Convergence in Federated Learning: Tackling Client Heterogeneity and Sparse Gradients. https://arxiv.org/abs/2102.07053. 30 Aug 2021

- Ons Aouedi. Kandaraj Piamrat. “Handling Privacy-Sensitive Medical Data With Federated Learning: Challenges and Future Directions.”. IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 27, NO. 2. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9804708. February 2023.

- L. Rocher, J. M. Hendrickx, and Y.-A. De Montjoye, “Estimating the success of re-identifications in incomplete datasets using generative models,” Nature Commun., vol. 10, no. 1, pp. 1–9, 2019.

- V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, “A survey on security and privacy of federated learning,” Future Generation Computer Systems, vol. 115, pp. 619–640, 2021.

- Laplace Equation: https://en.wikipedia.org/wiki/Additive_noise_mechanisms

- https://towardsdatascience.com/state-of-the-art-machine-learning-hyperparameter-optimization-with-optuna-a315d8564de1

- https://debuggercafe.com/using-learning-rate-scheduler-and-early-stopping-with-pytorch/

- https://github.com/StefanieStoppel/pytorch-mlflow-optuna/blob/tutorial-basics/mlflow-optuna-pytorch.ipynb

- https://optuna.readthedocs.io/en/v2.0.0/tutorial/pruning.html#activating-pruners

- https://www.cs.toronto.edu/~lczhang/360/lec/w04/convnet.html

- https://medium.com/analytics-vidhya/predict-retinal-disease-with-cnn-retinal-oct-images-dataset-6df09cb50206

Codes: https://github.com/KaranJoseph/Retinal_OCT