A peer-to-peer Federated Learning Model for Face Recognition

Saurab Sirpurkar
Georgia Institute of Technology
Atlanta, Georgia, 30313
Email: ssaurab3@gatech.edu

Omkar Prabhu
Georgia Institute of Technology
Atlanta, Georgia, 30313
Email: oprabhu3@gatech.edu

Rohan Pawar
Georgia Institute of Technology
Atlanta, Georgia, 30313
Email: rohan.pawar@gatech.edu

Alice Dumay
Georgia Institute of Technology
Atlanta, Georgia, 30313
Email: adumay3@gatech.edu

Chima Okechukwu
Georgia Institute of Technology
Atlanta, Georgia, 30313
Email: cokechukwu3@gatech.edu

Group 1

ABSTRACT

With its key feature being privacy, federated learning is a popular method for clients to train a global model using localized data. In this project, we look into the application of federated learning for the face recognition task. The motivation behind using federated learning for face recognition comes naturally from the goal of preserving data privacy of clients whose faces the model will train on. Later, the global model can be deployed to facilitate seamless face-recognition based authorization software for these clients. For example, a model trained on athletes' personal photos on the edge can be used in allow seamless entry into respective locker rooms while keeping out others. Similarly, staff can access restricted hospital units seamlessly. In this project, we will look into a generic face-recognition model that is trained using the Federated Learning paradigm and evaluate its efficiency in absolute terms and in relative terms comparing with centralized training. Later, we will debate whether such a system is indeed feasible.

1 INTRODUCTION

Face recognition is a well studied topic in the area of deep learning, with models such as DeepFace [1] and FaceNet [2] achieving close to perfect accuracy on large datasets. Due to this availability of well known highly accurate models, it is usually considered as a solved problem, especially after the advent of deep-CNNs. Face recognition models find a plethora of use cases in real life, especially when some kind of authorization is required. However, we seldom see them deployed in reality. One of the primary reasons behind this scarcity is the problem of privacy. We know that face recognition models require faces of the involved participants in the training phase. However, freely sharing faces to a central server where the model trains on the fetched data can lead to several problems. First, if there are many participants in the system, the server has to be scalable to store all this data which increases the costs involved. Second, there is a large communication cost overhead each time a new participant joins the training as they need to send all their photos to the server. Third, any security breach at the server directly affects all the participants as they do not have control over their data anymore.

To solve all these problems, we propose the use of federated learning as described by McMahan et al. [3]. Federated learning is a model training paradigm that trains a global model by aggregating the gradient changes at participants' local models. The participants fetch the updated model in each round before training on the next batch. Figure 1 shows the communication dynamics, in the common use-case of edge devices. We adapt a similar methodology,

but instead of using a server-client model, we use a peer-to-peer architecture in the systems layer. However, the application layer still deploys a server-client like training architecture as we discuss in section 3.

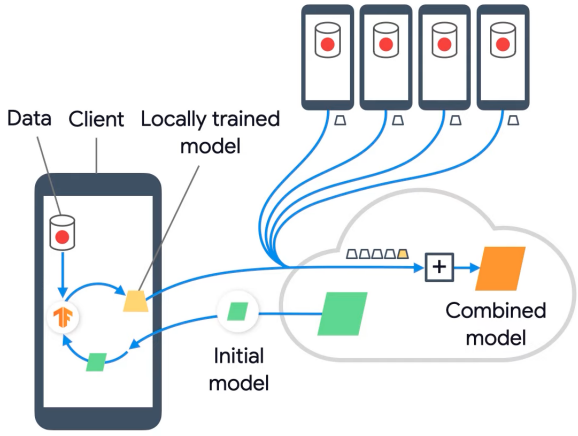

Fig. 1. The federated learning architecture in case of edge devices. Source: Google Cloud Tech [4]

2 MODEL AND DATASET

2.1 Dataset

We use VGGFace2 as our main dataset. It has 3.31 million images across 9131 actors, politicians and athletes etc. It is an ideal dataset for our task because of the pose-variations and illumination differences, which are important to train a deployable face-recognition model that works in the wild. The figure 2 displays these variances in a randomly chosen set of classes. Personally, we also observed a balanced ratio of male and female classes as well as young and old classes, thereby reducing chances of making our trained model biased towards a single demographic.

2.1.1 Class Distribution

As we can see from Figure 3, there is a high variance in the number of images that each person has in the dataset. To facilitate a good training with no class imbalance, we choose the classes that have the most images. All these classes come under the first three bins (600-762 images per class) in the Figure 3. The 6 most frequent faces, which are same as our chosen set for 6 clients experiment, are shown below the plot. As a future work, we can run more experiments that consider classes across multiple bins to see the effect these imbalances on the accuracy.

2.2 Transfer Learning

As we saw from the class distribution plot, there are only a few 100s of images of our chosen clients classes.

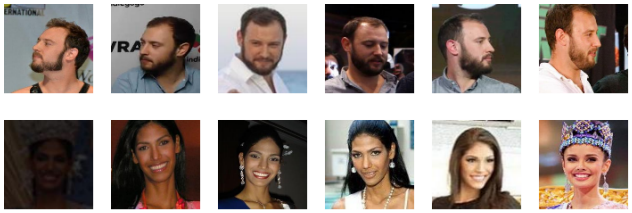

Fig. 2. VGGFace2 - Pose (top) and illumination (bottom) variations depicted using two random classes

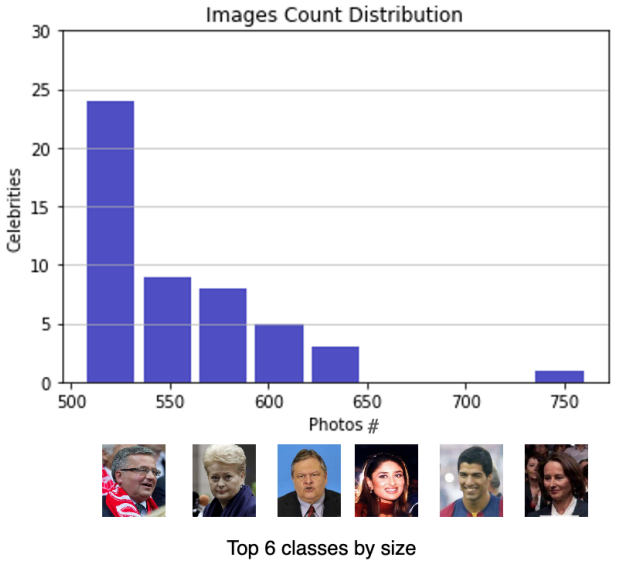

Fig. 3. VGGFace2 - Image count distribution across classes and 6 largest classes (chosen for clients = 6 experiment discussed in 5.1)

These images are not sufficient to train very deep neural networks. For this reason, we choose to employ *Transfer Learning* to train our model. As mentioned in [5], transfer learning is based on the observation that humans can apply previous learning experiences to newer tasks and quickly get good at them with less training, unlike regular deep neural networks that require training from ground-up for each new task. Based on this observation, transfer learning proposes reusing the the trained model parameters from a previous task to facilitate faster training over newer tasks using lesser samples. For our experiments, we use a VGG-16 model pre-trained on the ImageNet dataset with final dense layers excluded. Instead, we deploy our own dense layer with softmax activation to suit the output length. This architecture is depicted in Figure 4. The hyperparameter tuning and the exact number of clients used are discussed in section 4.

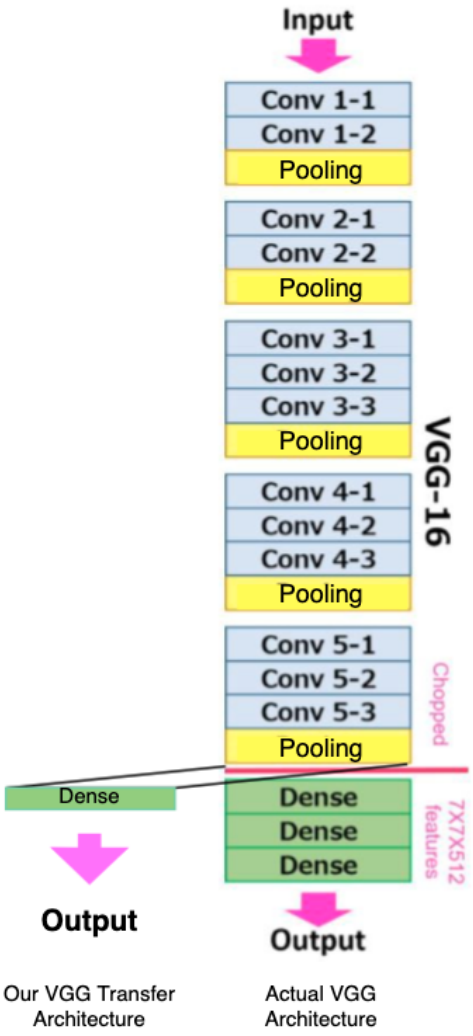

Fig. 4. Transfer Learning based on pre-trained VGG16

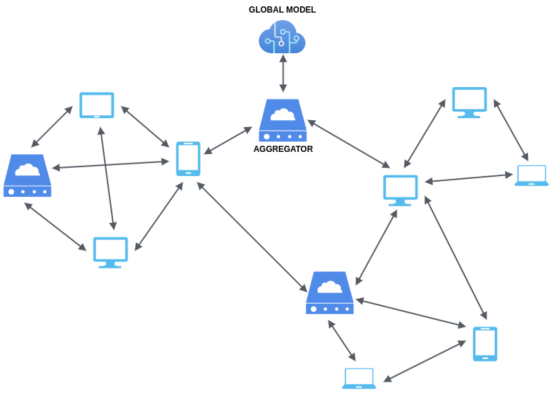

Fig. 5. An example P2P architecture in the systems layer

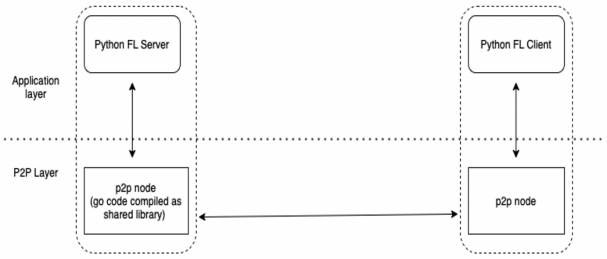

Fig. 6. Different levels of abstractions in the architecture - application layer and P2P layer

3 SYSTEM ARCHITECTURE

As mentioned in [6], even though regular deployment of a federated learning system has many advantages such as preserving privacy, increased data availability etc., one of the main issues in such deployments is the necessity for participants to agree on a central server which is hard in multiple scenarios. Moreover, any disruptions to this central server will completely disrupt the training process of all the participants. To get around these issues, we implement a peer-to-peer federated learning architecture for our project.

Within this architecture, there are two layers of abstractions to look at the system. On a systems level, as evident in Figure 6, the architecture is decentralized or peer-to-peer. However, on an application level, any of the nodes can act as a server in a given round with other nodes participating in the training as per usual. Even though we have a fixed server throughout the training process due to limitations of the root codebase, as a future work we can explore randomly selecting a server in each training round or employ a voting based system to elect the server in each round.

3.1 Monitoring UI

To be able to monitor our federated learning system in real-time, we developed a monitoring UI as shown in Figure 8 that can give us the details such as hardware specifications, a sample image from the class belonging to each client and the current training accuracy. A demo showing this monitoring UI can be found here: youtube.com/watch?v=l6chhhLZiIo

4 EXPERIMENTAL METHODS AND EVALUATION

4.1 CloudLab

CloudLab [7] is a testbed designed by University of Utah to facilitate large scale research in a cloud environment. It is specifically suitable to help with research on distributed systems such as in our case, offering fine control on different operations such as configuring hardware and

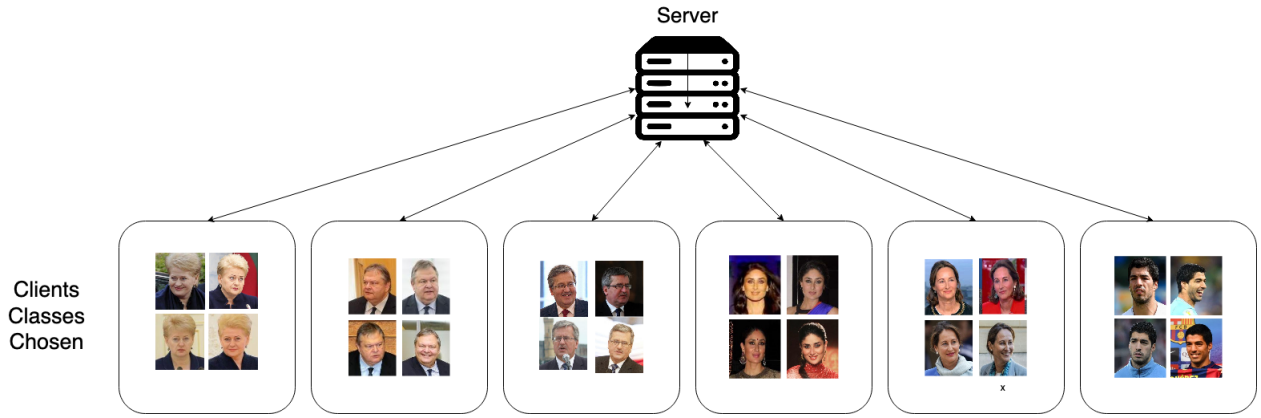

Fig. 7. Our federated learning architecture with 6 chosen classes

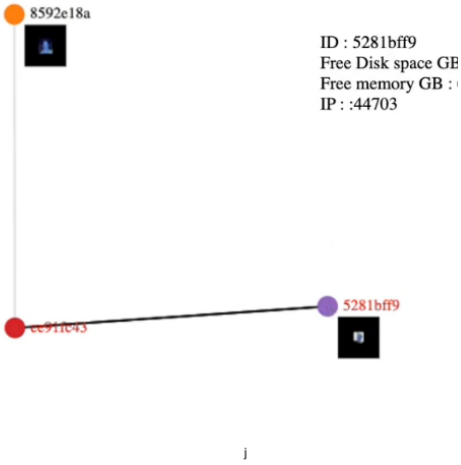

Fig. 8. A screenshot of our monitoring UI showing the face of the owner of each node and the hardware resources available

```
GPU 0: Tesla K80 (UUID: GPU-c3a6305e-95fe-9e80-fcbd-006d9a103f3e)
Sat Nov  6 06:26:00 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   49C    P8    31W / 149W |      3MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
Model name:          Intel(R) Xeon(R) CPU @ 2.30GHz
CPU MHz:             2299.998
13G
Avail
32G

------------------------------------------------------------
              RESOURCE USAGE STATISTICS
------------------------------------------------------------

Gen RAM Free: 12.4 GB  | Proc size   : 1.2 GB
GPU RAM Free: 11438 MB | Used       :     3 MB | Util       :     0% | Total
```

Fig. 9. Google Colab Hardware Specifications

storage, automating software installations etc. by providing a bare-metal level virtualization. This becomes specifically useful in repeatability of complex experiments that require distributed hardware like in cloud computing but with a higher control over all levels of the cloud stack.

4.1.1 Hardware Specifications

We used 40 cores of Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz, 2200 MHz to run our experiments on CloudLab. Due to access limitations, we used up to two nodes in total to run our experiments. Therefore, our server and clients were emulated as processes within these machines communicating using sockets. However, our code is easy to be deployed on any number of nodes, and hence it is easy to scale-out to the scenario where different machines are used for each client and the server.

For the experiments that are performed as Federated Learning emulations (see 4.2) on Google Colab, the environment's Hardware specifications are shown in the Fig. 9. A Tesla K80 GPU with 10GB GPU RAM was used along with a Intel Xeon(R) CPU with 12GB CPU RAM.

4.2 TensorFlow Emulations

Since we were limited by the number of nodes we could access, we ran some of the experiments as emulations on TensorFlow to utilize the GPUs for higher compute while avoiding the context switch that came with deploying all the processes in a few machines.

4.3 Experiments Conducted

Our main objective throughout the project was to build a face recognition system that contains a model trained using federated learning, thereby preserving the participant privacy. Therefore, most of our experiments are conducted by keeping this objective in mind.

We identified the need to experiment on two phases on such a face-recognition system, **training and prediction**, to comprehensively evaluate the usefulness and feasibility of such a model. We will discuss the experiments conducted in these two phases and the motivation behind them in the next two subsections.

4.3.1 Training Phase Experiments

We want to build a general face-recognition system that can train on any set of faces efficiently while maintaining good accuracy. For this, we need to take a close look at the training phase of the system. The experiments that we conduct here are the following:

1. Training accuracy across multiple rounds with different learning rates as shown in Figure 10
2. Effect of number of clients on the overall accuracy
3. Speed of training in federated learning vs centralized learning with the same model and number of classes
4. Bandwidth required in the server-client communication

4.3.2 Prediction Phase Experiments

Since we aspire to arrive at a system that can be trained using federated learning, but later be deployable centrally to identify the participants accurately, we need to conduct experiments investigating the prediction phase of our system. Therefore, we conduct the following experiments to evaluate the prediction phase:

1. Final global prediction accuracy and confusion matrix
2. Analysis of false positives and false negatives
3. Comparisons with accuracy from centralized-learning

4.4 Evaluation Metrics

Each experiment in the training and prediction phase require different metrics of evaluation. In training phase, we mainly look at the validation accuracy v/s the round number for the first experiment for each learning rate. Next, we see the steady-state validation accuracy as we scale up the number of clients. For the third experiment, we look into the average time (in seconds) between successive training rounds. Finally, we arrive at the bandwidth estimate using $\frac{size\,of\,model\,parameters}{time\,for\,each\,round}$ in Mbps.

In prediction phase, we look into the accuracy and F1 scores for the first experiment and inspect the confusion matrix. The second experiment is a subjective discussion about the classes where the model failed to produce good results and the possible reasons behind this. For the final experiment, we compare the accuracy and F-1 scores of centralized learning and federated learning.

5 EXPERIMENT RESULTS AND ANALYSIS

5.1 TRAINING PHASE

First, we first investigate the effect of learning rate on the global validation accuracy by fixing batch size as 32. As seen in Figure 10, we observe that having a high η such as 5e-4 leads to a poor training. As we decrease the learning rate to 5e-5, we see a similar jitter in the curve but the accuracy roof is much higher. As we decrease the learning rate, although this jitter disappears, we observe a lower accuracy roof. Therefore, to balance both these factors, we primarily use 5e-6 as our learning rate in all the experiments that follow.

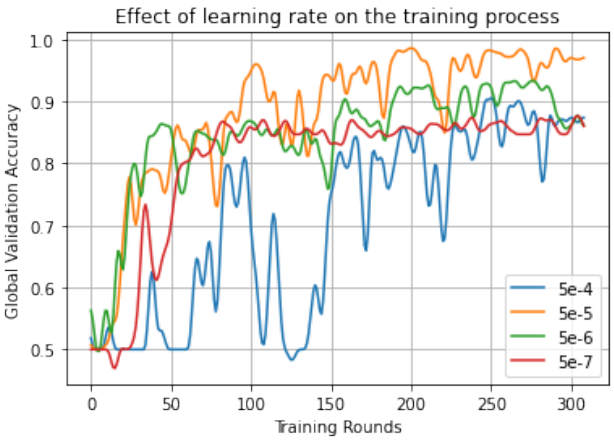

Fig. 10. Effect of learning rate on the training curves for 2-client architecture. The plots were smoothened with hamming window of length 10.

Fig. 11. Effect of number of participants on the training curves. The plots were smoothened with hamming window of length 10.

Next, we look into the effect of number of clients on the validation accuracy as the training goes on for 600 rounds. As we can see from Figure 11, adding more clients decreases the overall accuracy of the model. This is expected because the model can get confused between a higher number of pairs as the number of classes increased. We conduct a more detailed study on confusion matrices in 5.2.

Next, we look into the speed of learning. On average, to train the same number of samples, we found that Federated

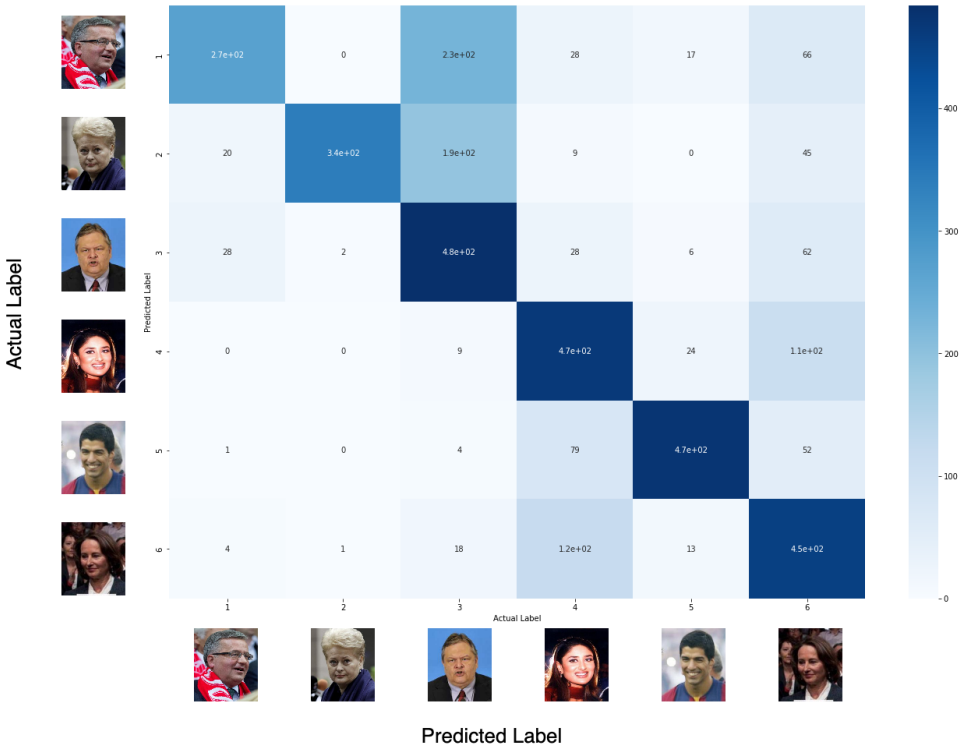

Fig. 12. Global confusion Matrix in case of 6-client Federated Learning

Learning takes 2-3x longer. One of the root causes behind this is the server-client communication overhead in each round. Moreover, one of the clients could take slightly longer to train its batch thereby making other clients wait (since we are waiting for all clients to respond in each round in our setup).

Finally, we observed that the bandwidth required mainly depends on the batch size and the validation overhead in each batch (if any). For B=16, with validation ran in each training round, we found that it takes 3.5s per round. Therefore, an upload and download speed in the order of 10s of Mbps is required to reduce the communication overhead to reasonable amounts.

5.2 PREDICTION PHASE

In the previous section, we looked at the experiments to quantitatively evaluate the desired qualities in a good federated learning system in the training phase. We looked into the effect of learning rate and number of clients on the model accuracy. Later, we also looked into the training time and bandwidth estimates.

These metrics cover the initial phase of such a model when all the participants are actively training. However, the end goal for developing such a system is to be able to deploy it in real world to facilitate seamless authorization and other use-cases that require face-recognition. Therefore, in this section we focus on the metrics that tell us how good such models can be when deployed in real-life scenarios.

In the rest of our discussion in this section, we consider our main model to be the trained VGG-16 model with three final layers excluded, followed by our dense layer. The batch size here is considered as 16 and the learning rate is chosen to be 5e-6 with Adam optimization. First,

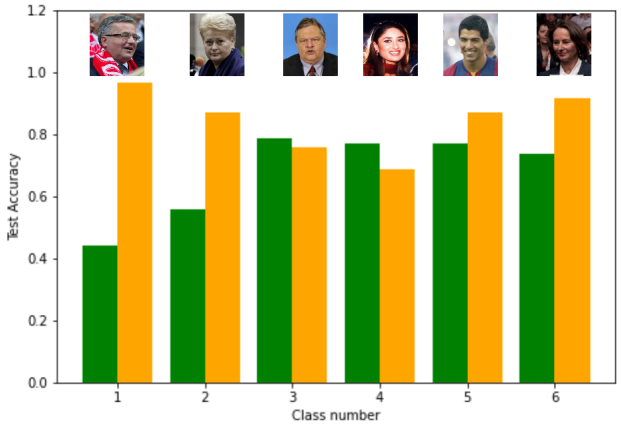

Fig. 13. Accuracy of our federated model on different classes (green) and the accuracy of centralized training (orange)

let us take a look at the performance of the model that trained for 600 rounds on the test sets of each face that we trained on. As shown in Figure 13, we find that the model consistently reaches close to 80% accuracy, except

in the first two classes where the accuracy is much lower. We discuss the possible reasons behind this when discussing about confusion matrix.

Next, we consider the confusion matrix 12 to observe where the most mistakes were. As we can see the first class was often mispredicted as class 3. We can see that there is some chance for this confusion based on the facial features similarity among these classes and due to the fact that they fall in the same demographics, age and gender wise. Similarly, the model often mispredicts class 2 to be class 3. Also, class 6 is sometimes predicted as class 4. We again see that both these classes share the same gender and age group, and there are some facial similarities that could confuse our model.

Finally, we evaluate the model performance against a centrally trained model ran on all the photos for 100 epochs. As shown in the figure 13, the centrally model has a consistently better performance over our federated learning model. This indicates that we could train on more rounds and smaller batch sizes in the future to see if we can improve our model accuracy.

6 FUTURE WORK

Since our model reaches 80% accuracy on most classes, as a next step, our goal would be to see if running even more training rounds (1000s) can further increase this accuracy.

As discussed in section 3, throughout our training process, we fixed a single node to be the server. This implies that this node cannot train its own data (although it is possible to make this happen, we do not implement such a server-is-a-client algorithm). As a future work, we can leverage the P2P nature of our systems layer where the aggregator changes in each round and can participate in training during the rest of the rounds.

Moreover, as discussed in section 2.1.1, to make our training process easier, we chose randomly from the largest available classes. However, one of the major challenges faced in federated learning is the issue of unbalanced datasets. As a future work, we can look into such imbalances and empirically evaluate the effect of a skewed distribution on the model performance.

Since we saw that the bandwidth required is in the oreder of 10s of Mbps for the training to work effectively, we can look into quantization and other techniques to compress the model updates while maintaining accuracy.

We can also leverage blockchain based techniques over the P2P network to establish the security and finality of the data.

7 CONCLUSION

In this project, we examined the use of federated learning to train a face-recognition system, with an objective to develop a system that can be deployed in real life to provide seamless authorization-based access to restricted areas and facilities with various use cases such as sport locker rooms, airports and hospitals. After empirically evaluating the efficiency of such a learning system in both training and testing phase, we observed that the federated learning model recognizes the faces of subjects fairly accurately when compared with the centralized training paradigm, reaching an acceptable accuracy overall. Therefore, we conclude that it is indeed feasible to develop such systems and deploy them in such use-cases.

References

[1] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L., 2014. "Deepface: Closing the gap to human-level performance in face verification". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1701–1708.

[2] Schroff, F., Kalenichenko, D., and Philbin, J., 2015. "Facenet: A unified embedding for face recognition and clustering". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 815–823.

[3] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A., 2017. "Communication-efficient learning of deep networks from decentralized data". In Artificial intelligence and statistics, PMLR, pp. 1273–1282.

[4] What is Federated Learning by Google Cloud Tech, howpublished = https://www.youtube.com/watch?v=x8yywunttoy, note = Accessed: 2021-02-05.

[5] Torrey, L., and Shavlik, J., 2010. "Transfer learning". In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, pp. 242–264.

[6] Roy, A. G., Siddiqui, S., Pölsterl, S., Navab, N., and Wachinger, C., 2019. "Braintorrent: A peer-to-peer environment for decentralized federated learning". *arXiv preprint arXiv:1905.06731*.

[7] Duplyakin, D., Ricci, R., Maricq, A., Wong, G., Duerig, J., Eide, E., Stoller, L., Hibler, M., Johnson, D., Webb, K., et al., 2019. "The design and operation of cloudlab". In 2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19), pp. 1–14.