Mid-Term Project Report

Aditya Varun V
Indian Institute of Technology, Hyderabad
Kandi, Sangareddy
ai22btech11001@iith.ac.in

Arsh Arora
Indian Institute of Technology, Hyderabad
Kandi, Sangareddy
bm22btech11004@iith.ac.in

Surya Saketh Chakka
Indian Institute of Technology, Hyderabad
Kandi, Sangareddy
ai22btech11005@iith.ac.in

Mayank Parasramka
Indian Institute of Technology, Hyderabad
Kandi, Sangareddy
ai22btech11018@iith.ac.in

Saketh Ram Kumar Dondapati
Indian Institute of Technology, Hyderabad
Kandi, Sangareddy
ai22btech11023@iith.ac.in

Abstract

In continuation of the previous report, this midterm report provides a comprehensive analysis of various model architectures and their performance on the classification task. The report also delves into the federated learning setup, highlighting the implementation and results obtained using the FedAveraging algorithm. We also discuss several techniques to improve results for non-IID data distributions across the clients in federated learning, which will be further explored in the next report. We also present methods for improving model performance, including strategies to address these issues in federated learning scenarios. Throughout the report, future directions for experimentation and optimization are discussed, with a focus on enhancing both model accuracy and efficiency. The source code for all experiments can be accessed via the following GitHub repository: GitHub Repository

1. Base Models: Architecture and Preliminary Results

1.1. ResNet

This architecture is a 1D Residual Network (ResNet) designed for classifying ECG signals from the PTB-XL dataset. It processes 1D temporal data and utilizes residual blocks for efficient training of deeper networks.

1.1.1 Architecture Overview

The network is composed of the following layers:

- **Initial Convolution Block:** The input layer receives a 1D ECG signal, followed by a convolution layer with 64 filters, kernel size of 7, and a stride of 2. Batch normalization and ReLU activation are applied, and the output passes through a MaxPooling layer to reduce the sequence length.

$$\begin{aligned}\text{Conv1D}(64, 7, \text{stride} = 2) \rightarrow \text{BatchNorm} \\ \rightarrow \text{ReLU} \rightarrow \text{MaxPooling1D}(3, \text{stride} = 2)\end{aligned} \tag{1}$$

- **Residual Blocks:** Each residual block consists of two convolutional layers. The input is added to the output via a shortcut connection. If the input and output dimensions differ, a 1x1 convolution is applied to the shortcut for dimensionality matching.
  The general form of the residual block is:

$$\begin{aligned}y = \text{ReLU}\big(\text{BatchNorm}(\text{Conv1D}(x))\big) \\ + \text{Shortcut}(x)\end{aligned} \tag{2}$$

  The first two blocks use 64 filters, followed by blocks with 128 and 256 filters, respectively. The residual block design enables efficient gradient flow, mitigating the vanishing gradient problem.
- **Global Average Pooling and Dense Layers:** After passing through the residual blocks, the sequence is globally

averaged to produce a feature vector. The feature vector is then passed through fully connected layers for classification:

$$\begin{aligned} \text{GlobalAveragePooling1D} &\rightarrow \text{Dense}(128, \text{ReLU}) \\ &\rightarrow \text{Dense}(\text{num\_classes}, \text{sigmoid}) \end{aligned} \tag{3}$$

1.1.2 Experiments and Results

The model was run on GPU/cuda over the complete PTB-XL dataset, and after 50 epochs, the training accuracy was 86.63%, and the validation accuracy was 65.52%.

1.1.3 Further Improvements

To improve the ResNet model for the PTB-XL dataset, several architectural enhancements can be made:

- Modify the initial stem using multiple smaller convolutional layers (e.g., 3x3) instead of a single large 7x7 filter. This helps capture finer details in the early stages of the network.
- Implementing bottleneck residual blocks, where 1x1 convolutions are applied before and after the central convolution, can reduce the number of parameters while maintaining model expressiveness.
- Adding Squeeze-and-Excitation (SE) blocks allows the network to recalibrate channel-wise feature importance, which is especially useful for focusing on critical ECG signal features.

These changes would enhance the model's capacity and efficiency, making it better suited for handling the intricacies of the PTB-XL data. Moreover, applying the *xresnet1d101* model used in Strodthoff *et al*. [5] would be a major improvement.

1.2. Convolutional Neural Networks (CNNs)

The following results were obtained from using a lightweight CNN model to perform preliminary classification on the PTBXL dataset.

1.2.1 Layers Overview

- **Conv1D:** This layer performs 1D convolution, applying filters to the input signal to extract features. The first layer has 64 filters and a kernel size of 7, which means it looks at 7 time steps at a time.
- **Batch Normalization:** This layer normalizes the output of the previous layer to stabilize and speed up the training.
- **MaxPooling1D:** This layer reduces the dimensionality of the data by taking the maximum value over a specified pool size (2 in this case), thereby downsampling the data.
- **Flatten:** This layer converts the 3D output from the previous layers into a 1D array, making it suitable for the Dense layer.
- **Dense:** These layers are fully connected layers. The first Dense layer has 128 units and uses the ReLU activation function, while the final Dense layer outputs the class probabilities using the sigmoid activation function.
- **Dropout:** This layer randomly sets a fraction (0.5) of input units to 0 during training to prevent overfitting.

1.2.2 Experiments and Results

The model was run on GPU/cuda over the complete PTB-XL dataset, and after 50 epochs, the training accuracy was 59.63%, and the validation accuracy was 52.55%.

1.2.3 Possible Improvements

These improvements are based on the Lightweight CNN Model implemented by Saglietto *et al*. [3]

1. **Increase Network Depth with More Convolutional Layers:** Adding more convolutional layers helps the model learn more complex and hierarchical features, which is essential for accurately capturing subtle patterns in time-series ECG data.
2. **Use Smaller Kernel Sizes:** Smaller kernel sizes (e.g., 3 or 5) are more effective for capturing fine-grained, localized patterns in ECG signals. This allows the network to extract relevant features without losing temporal precision.
3. **Incorporate Data Augmentation:** Applying data augmentation techniques, such as adding Gaussian noise, time scaling, amplitude scaling, and temporal cropping, enhances the model's generalization. This ensures the network performs well on unseen data by simulating real-world variability.

1.3. Transformers

The following is the architecture for the transformer to perform preliminary classification on the PTBXL dataset.

1.3.1 Architecture

The architecture for the transformer model is given in table 1. The model follows an architecture similar to the Vision Transformer.

1.3.2 Experiments and Results

The transformer model is yet to be sufficiently trialed and tested

Layer Type	Details
Position Embedding	Embedding Size = `input_shape[1]`
Element-wise Addition	Add position embedding
Transformer Encoder	**Block**
Layer Normalization	`epsilon = 1e-6`
Multi-Head Attention	Heads = 4, Projection Dimension = `input_shape[1]`, Dropout = 0.1
Dropout	Dropout = 0.1
Residual Connection	Addition of input and output
Feed-Forward Layer 1	ReLU Activation, 0.1 Dropout
Feed-Forward Layer 2	ReLU Activation, 0.1 Dropout
Residual Connection	Addition of input and output
Transformer Block	**Repeat 3 Times**
Average Pooling 1D	Pooling Operation
Dense Layer	Activation = ReLU
Output Layer	Activation = Sigmoid

Table 1. Transformer Model Architecture

1.3.3 Future Improvements

To further enhance the architecture, the following approaches will be explored:

- **Lim *et al.* [2]**: Proposes the use of an LSTM network to encode positional embeddings, effectively capturing the sequential dependencies in time series data, which could improve performance on the PTBXL dataset.
- **Masked Transformer for Electrocardiogram Classification** [7]: Introduces a novel transformer-based training strategy that leverages an encoder-decoder architecture based on the Vision Transformer model, specifically designed for electrocardiogram classification.

2. FL Setup

2.1. FedAveraging Results

As a preliminary testing, a simple FedAveraging algorithm was used. The dataset was divided into 5 parts by sampling IID. Due to computational constraints, the algorithm was run for 10 rounds, where in each round, 3 users were sampled randomly to update the weights. Each client receives and returns weights from and to the global model. The setup managed to achieve an 85.93 % training accuracy, and a 60% validation accuracy on 10% of the entire dataset using the CNN Architecture describer above. From Figure 1 and Figure 2, we observe that both CNN and ResNet architectures seem to perform similarly, with the ResNet architecture having a slightly higher validation accuracy. From this learning set up, we can see that, clients are able to successfully train the global model, without ever passing the actual data itself to the central model, but instead only sharing their trained weights.

2.2. Improvements

- **Parallelization:** The setup currently trains the 'local' models one by one. A huge improvement would be parallelizing this, using multiple cores can enable much quicker training.
- **GPU:** We can try integrating `GPUs` into the model for faster computation and better parallelization.
- **Decentralized FL:** We could switch from Centralized to decentralized FL, removing the dependency of a central server, which can also act as a bottleneck/ be vulnerable to attacks and breaches, and can distribute workload among clients better.

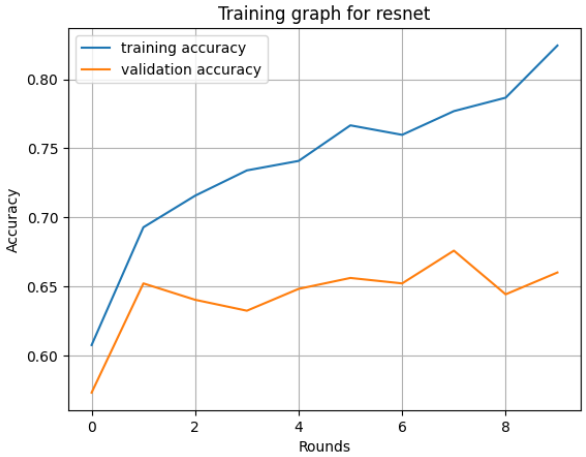

Figure 1. Training graph for the *Resnet* model

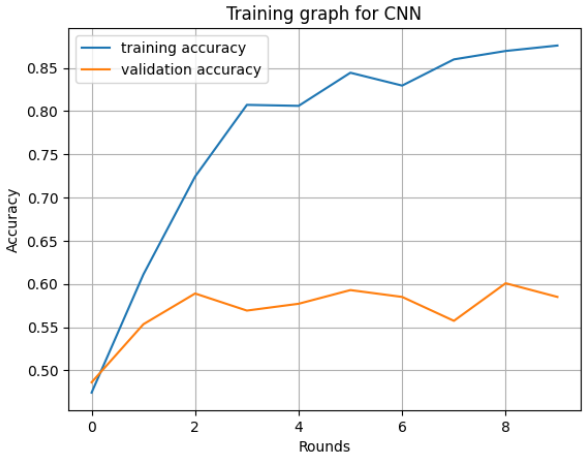

Figure 2. Training graph for the CNN model

3. FL Algorithms: Issues and Improvements

The following section discusses some common issues faced by FL algorithms, and ways to improve them. These techniques will be explored for the subsequent report.

3.1. Personalization Issues in Non-IID Situations

Federated Learning (FL) faces significant challenges with personalization due to non-IID data distributions across clients, which often leads to suboptimal performance of a single global model. FedAvg algorithm assumes that clients have similar data, but in non-IID situations, this assumption breaks down. Conflicting updates from clients with different data distributions can slow down learning and reduce the overall performance of the global model.

Potential Improvement Techniques

- **Personalized FedAvg**: [1] adapts the global model for individual clients by leveraging Model-Agnostic Meta-Learning (MAML) principles, enhancing client-specific performance in non-IID environments.
- **Data Sharing for Centralization vs. Accuracy**: [6] discusses sharing a portion of data among clients to find a balance between centralized training and federated learning, improving accuracy in non-IID scenarios.
- **Clustered Federated Learning**: [4] modifies the FL algorithm by grouping clients with similar data distributions, effectively addressing non-IID characteristics for better convergence and performance.

References

[1] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning, 2023. 4

[2] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021. 3

[3] Andrea Saglietto, Daniele Baccega, Roberto Esposito, Matteo Anselmino, Veronica Dusi, Attilio Fiandrotti, and Gaetano Maria De Ferrari. Convolutional neural network (cnn)-enabled electrocardiogram (ecg) analysis: a comparison between standard twelve-lead and single-lead setups. *Frontiers in Cardiovascular Medicine*, 11:1327179, 2024. Erratum in: Front Cardiovasc Med. 2024 Mar 14;11:1396396. 2

[4] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints, 2019. 4

[5] Nils Strodthoff, Patrick Wagner, Tobias Schaeffter, and Wojciech Samek. Deep learning for ecg analysis: Benchmarks and insights from ptb-xl. *IEEE Journal of Biomedical and Health Informatics*, 25(5):1519–1528, 2021. 2

[6] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. 2018. 4

[7] Ya Zhou, Xiaolin Diao, Yanni Huo, Yang Liu, Xiaohan Fan, and Wei Zhao. Masked transformer for electrocardiogram classification, 2024. 3