Federated Learning across varying architectures with local parameter drift correction

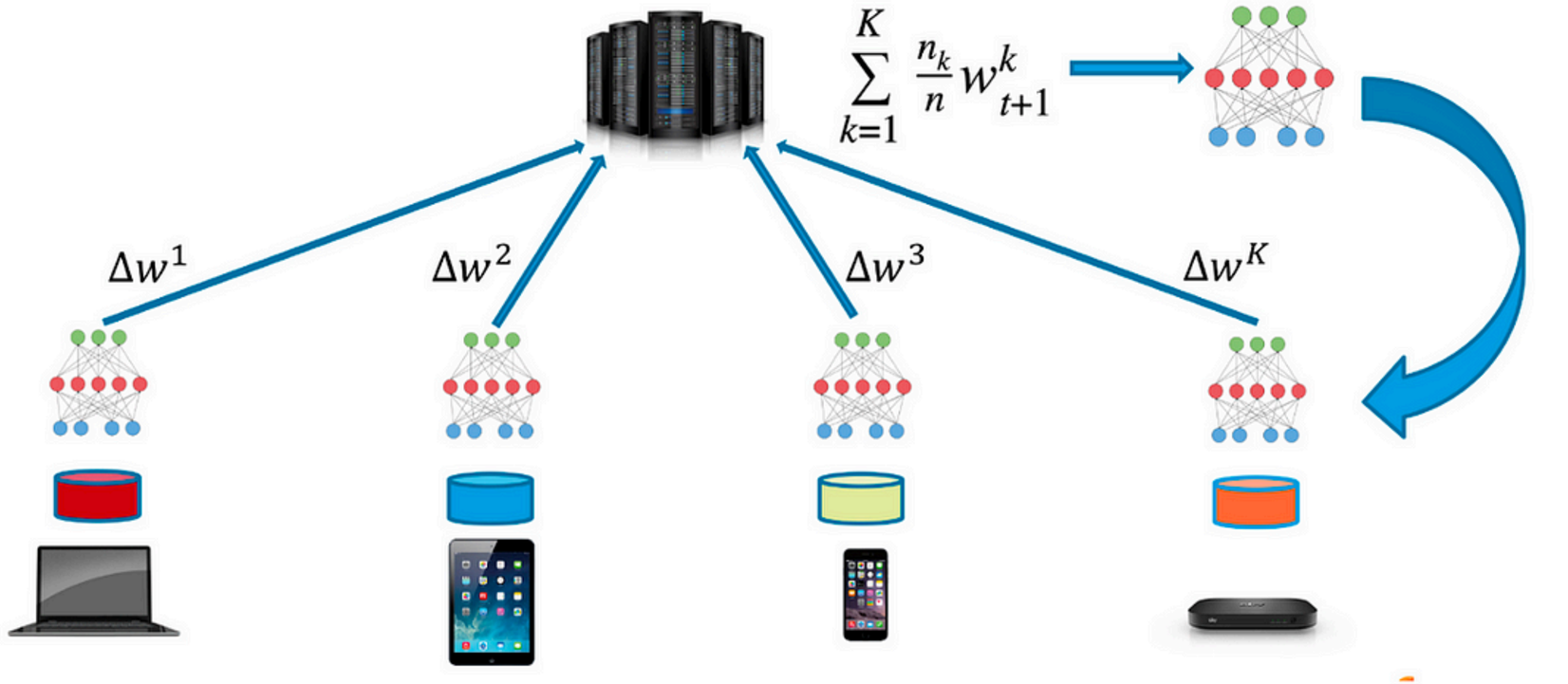
Brendan Jobus - 18321740

Literature

FedAvg

- FedAvg is an aggregation strategy that serves as a backbone to the vast majority of FL approaches
- It defines an efficient way to communicate between clients and the central server, and the process of aggregating the models, using SGD
- It works in rounds
  - in each, the server selects a subset of clients and sends them the global model
  - Each client then optimises the global model on their loss function and dataset
  - At the end of the round, the server receives all of the parameters and performs a weighted average on them to create the new global model

How does it work?



Literature

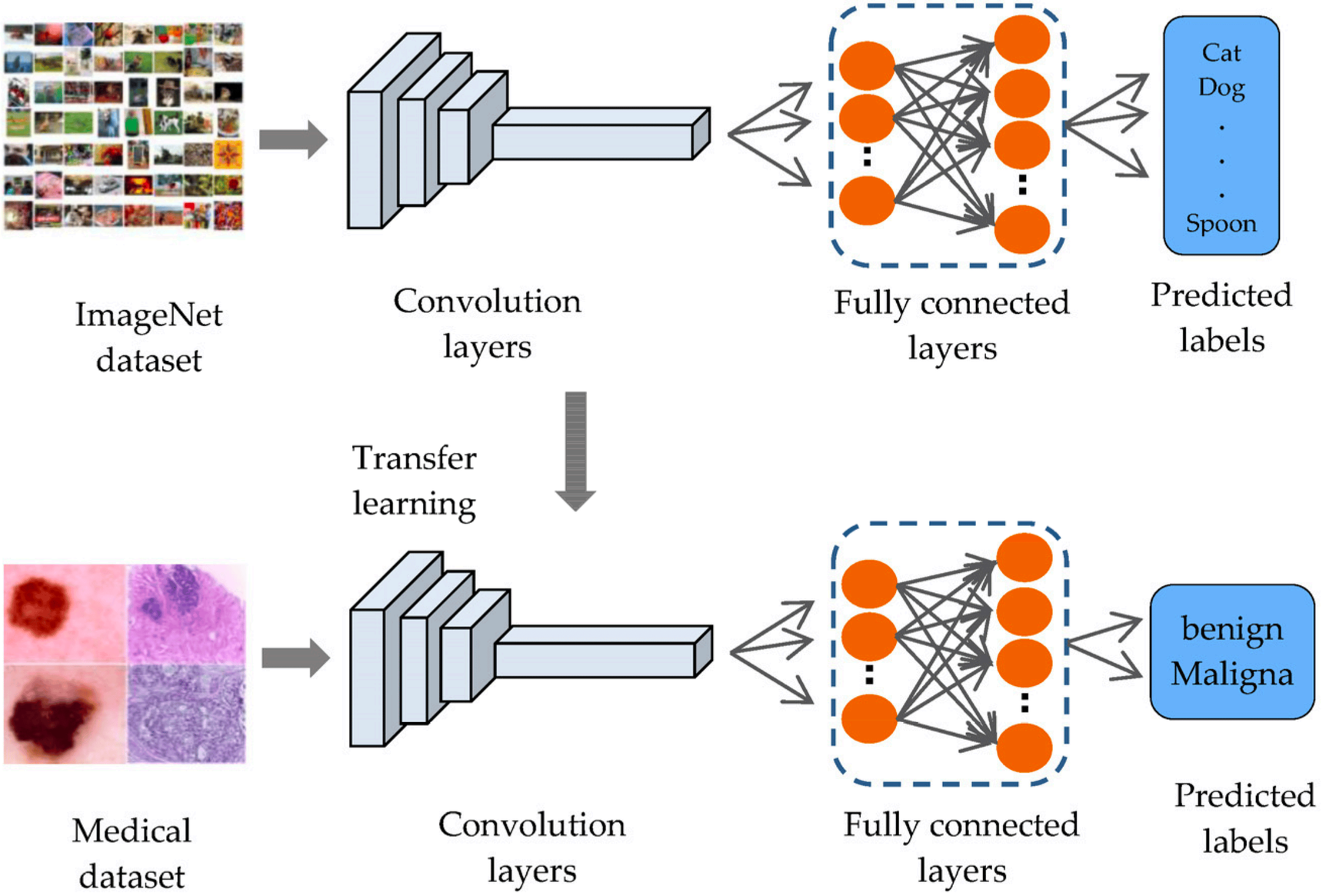
Drift Correction

- Various methods of dealing with parameter drifts, such as SCAFFOLD and FedProx
- These models look to minimise the local drifts, $min\,||\,\theta_i - w\,||$
- FedDC approaches the problem slightly differently
  - Look to instead learn the local drift
  - Set a new local drift term h_i which satisfies $w - \theta_i$
  - Now instead $min\,||\,h_i + \theta_i - w\,||$
- FedDC was shown to increase model accuracy when using non iid data and speed up convergence

Literature

Knowledge Distillation and Transfer Learning

- Knowledge Distillation and Transfer Learning are the back bone of FedMD
- Transfer learning is a method of personalisation that decreases the discrepancy between the global model and the local models
- Pre-train a model on a public dataset first, then train this model on the private data to personalise it
- Transfer Learning can be thought of as a model based method for dealing with data heterogeneity



https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.mdpi.com%2F1424-8220%2F23%2F2%2F570&psig=AOvVaw09fXxrVqXWxKw3-AXonP1Y&ust=1711546662203000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCJiHw6uGkoUDFQAAAAAdAAAAABAD

Literature

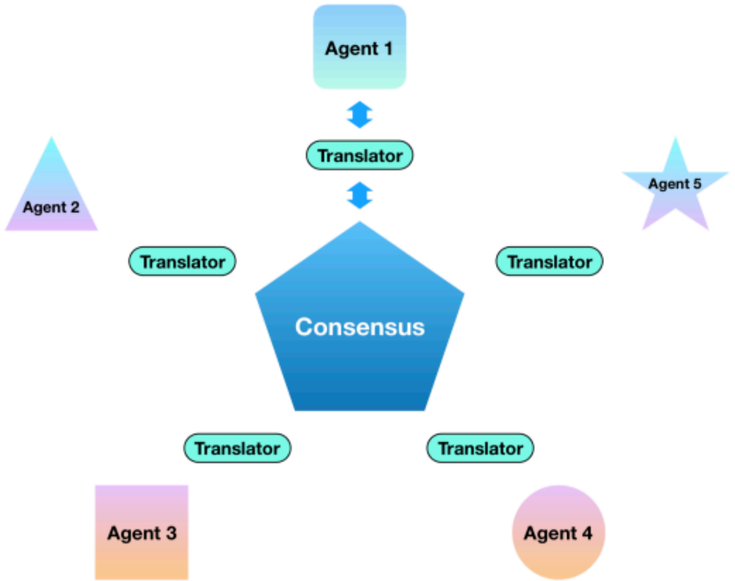
Knowledge Distillation and Transfer Learning

- Knowledge Distillation in Neural Networks was originally designed to make large ensemble approaches more lightweight
- Treats the logits or class scores as learned maps from input to output
- To learn lower level features, it greatly helps to utilise a labelled dataset to train the smaller model on
- Knowledge Distillation for FL can be thought of as an architecture based solution for personalisation, because by distilling knowledge we can move it between server and/or client regardless of architecture

Literature

FedMD

- FedMD utilises both transfer learning and knowledge distillation in an to both combat heterogenous data sources and to create personalised models
- It uses transfer learning fairly basically, having each client model undergo TL before beginning the collaborative training
- Uses knowledge distillation to allow us to perform the averaging step of FedAvg with models of varying architecture
- Does this by taking a small subset of the public dataset and predicting it, then taking the corresponding logits and using them effectively as the global model
- Retrain each local model with the smaller dataset and the averaged logits as the y_train


Agent 1
Translator
Agent 2
Agent 5
Translator
Translator
Consensus
Translator
Translator
Agent 3
Agent 4

Motivation

- One of Federated Learnings largest issues is non IID data
- Can reduce FL model accuracy by up to 55% for highly skewed distributions as shown by Zhao et al
- When looking through the vast array of approaches for FL implementations, the variety was clearly due mainly to variance in requirements AND ability to deal with non IID data with these limitations

Motivation

- When looking through some of the approaches towards solving this issue, many of the other approaches deal with this problem in a more complex manner, in contrast, FedDC was remarkably simple
- Examples of other contemporary approaches include SCAFFOLD and FedProx
- FedDC was still built upon FedAVG and so inherits some of its downsides
- Wanted to see if the advantages found in the paper could be found when merging it with another state of the art FL implementation

Problem Statement

- Look to improve FedMD's ability to cope with non iid data by applying FedDC's drift correction methodology
- Look to increase rate of convergence of FedMD

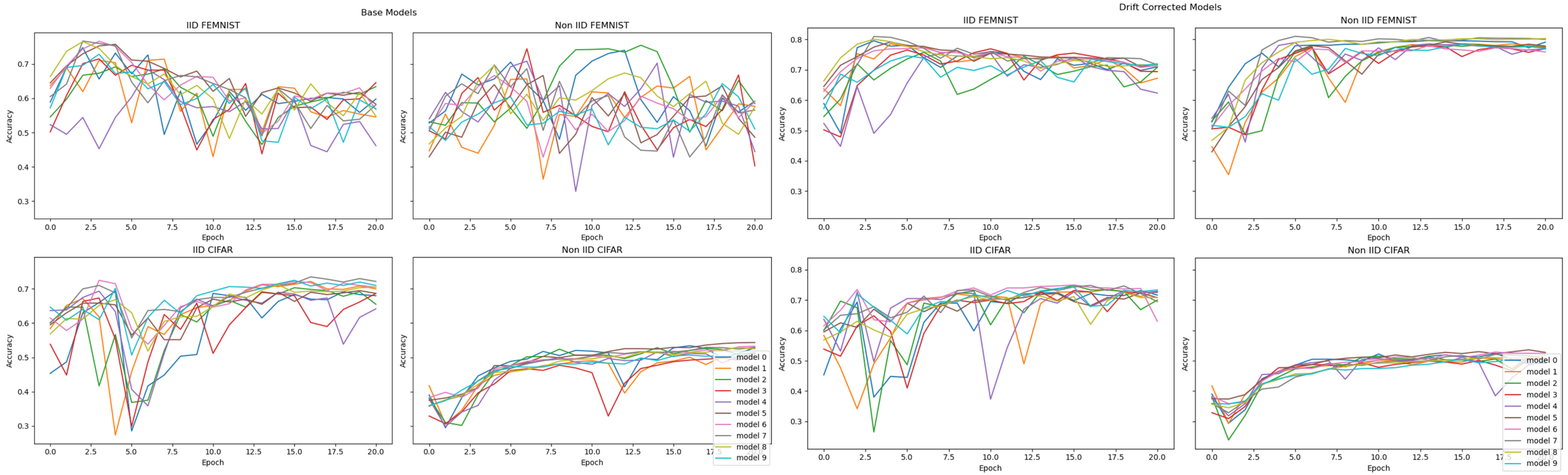
Technical Content

- Adapting FedDC was quite difficult due to how they implemented it
  - Implemented with PyTorch at a very low level
- FedMD was implemented with tensorflow
- Subclassed Tensorflow
- Creating a custom train_step function
- Allowing our train_step function to operate as a standard train_step until we want to do collaborative training
- This allows the regular FedMD model and the drift corrected models to be initiated off of the same process i.e. can clone the model after TL to run both models off of and allows for comparison between the two

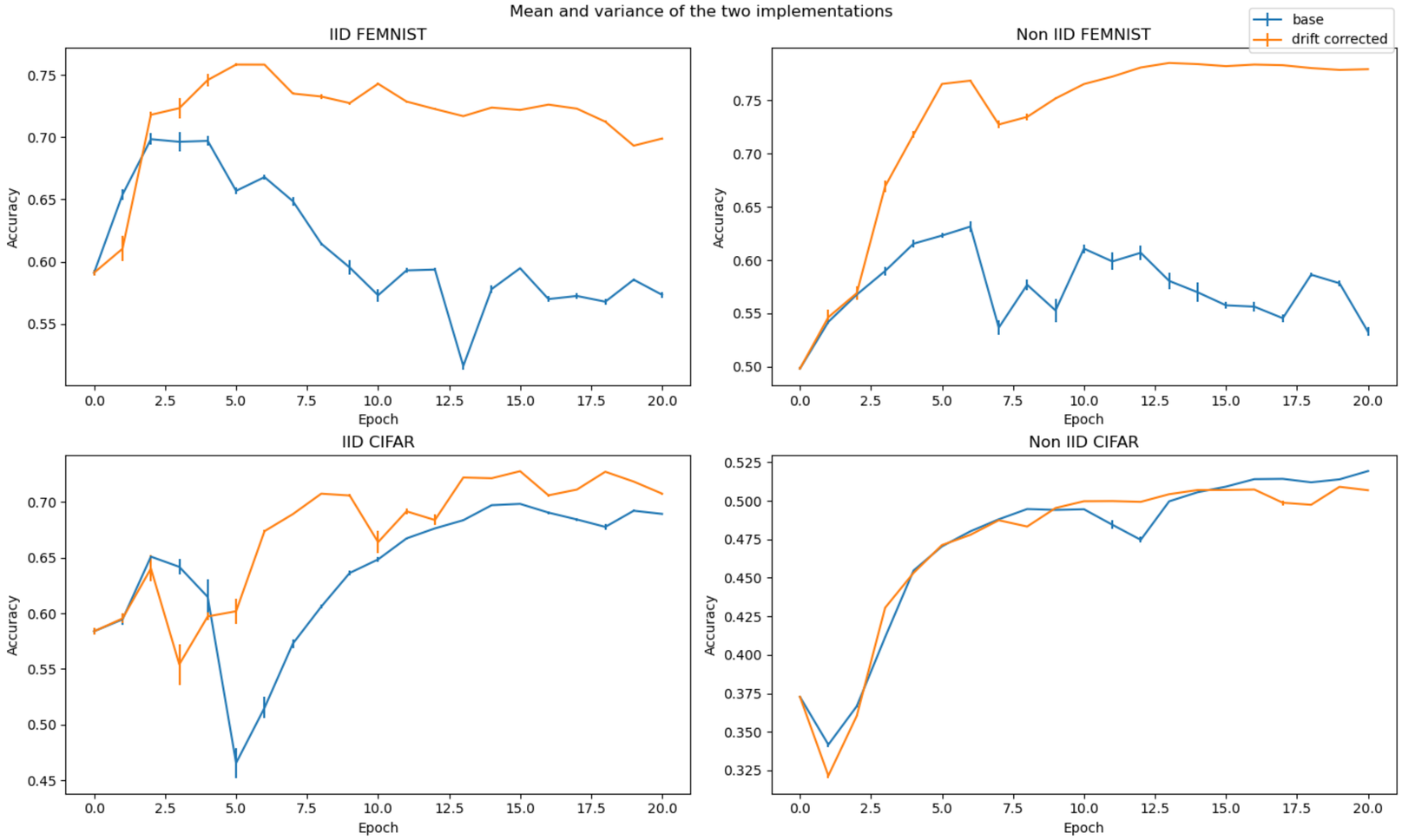
Technical Content

- Had to create a custom process for cloning the models due to tensorflow not providing such functionality in the case where you have your own code

Testing/Evaluation



Testing/Evaluation



Analysis

- For the FEMNIST dataset, we see massive improvements over the base FedMD approach by applying drift correction
- However, for the CIFAR dataset, there is much less improvement
- For both datasets, however, we converge faster or at a similar rate
- Obviously, the sample size here is just one, so no true conclusions can be drawn
- Anecdotally, the base FedMD has tended to have much larger variations in the accuracy

Conclusion

- In this project, I integrated the approach taken by FedMD to control local parameter drift to FedMD, a state of the art approach for Federated Learning across clients with non-specific architectures
- I showed that by utilising FedDC's approach, model convergence and accuracy was increased

Limitations and Future Research

- Was not able to recreate the findings from FedMD's paper
- Generate more samples
- Testing was not done on the drift correction
- Why does the CIFAR dataset see less improvement than FEMNIST
- I used FedMD's approach towards non-iid data generation, could be greatly enhanced by using Dirichlet distribution
- I am unsure of the effect that some layers have on the drift correction approach taken e.g. normalisation layers
- Evaluation could be greatly improved by comparing results against a greater number of models (FL-Bench) and Datasets