CS330 Autumn 2020 Homework 1 - Data Processing and Black-Box Meta-Learning

- SUNet ID: shubhams
- Name: Shubham Shrivastava
- Collaborators: None

- Google Colab: https://colab.research.google.com/drive/1pdyJS9iB2k9rGoBRHPHN_3C7T8qOWioV?usp=sharing

Overview

Goals: In this assignment, we will look at meta-learing for few shot classification.

1. Learn how to process and partition data for meta learning problems, where training is done over a distribution of training tasks p(T).
2. Implement and train memory augmented neural networks, a black-box meta-learner that uses a recurrent neural network.
3. Analyze the learning performance for different size problems.
4. Experiment with model parameters and explore how they improve performance.

We will be working with Omniglot [1], a dataset with 1623 characters from 50 different languages. Each character has 20 28x28 images. We are interested in training models for K-shot, N-way classification, i.e. training a classifier to distinguish between N previously unseen characters, given only K labeled examples of each character.

Problem 1 - Data Processing for Few-Shot Classification

The `sample batch` function within `DataGenerator` class is responsible of sampling `K` samples from `N` classes randomly. During training of the Memory-Augmented Neural Networks for Machine Translation (MANN)[2], `K+1` samples are obtained using this function, labels for last (`[K+1]th`) samples are replaced with zeros. These are the test samples that are concatenated with `K` train samples, and for which the network is trained to output correct labels. The input images of size 28x28 is flattened to a 1-dimensional vector of size 784.

Samples for K=2, N=4, are shown below.

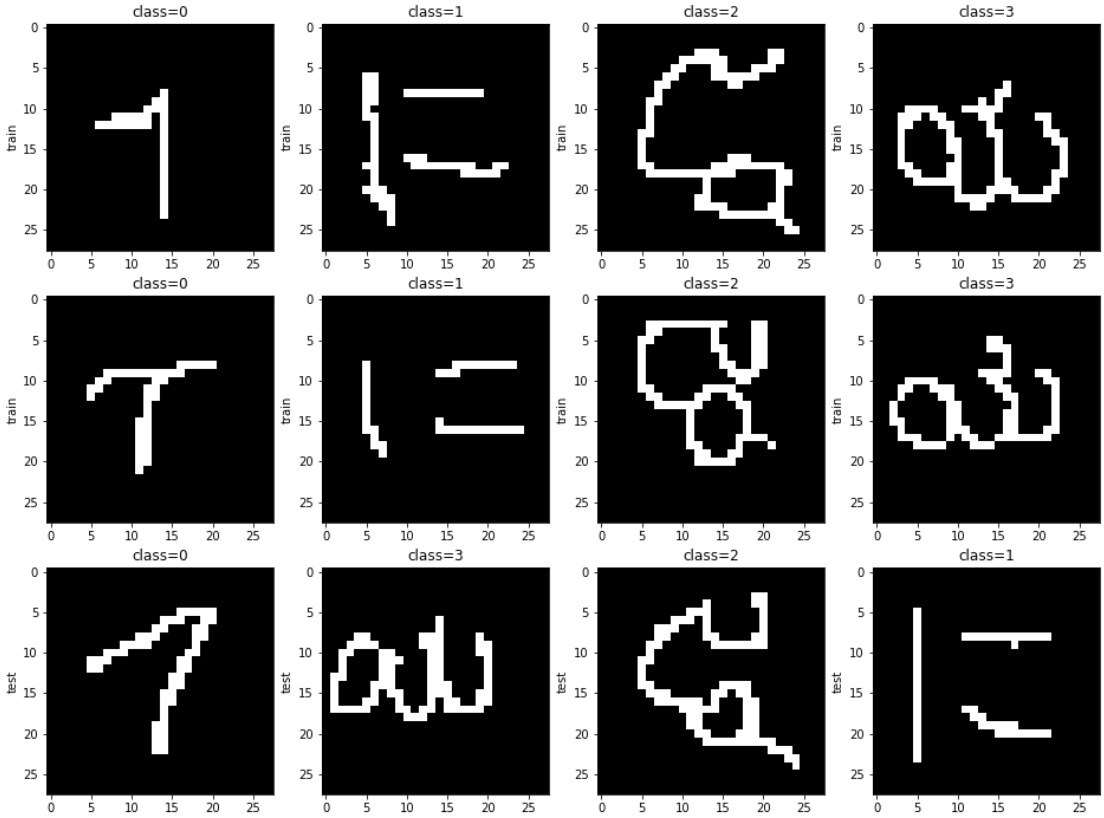


Figure - Visualization of the output of `sample_batch` function of `DataGenerator` class

Problem 2: Memory Augmented Neural Networks

Here, few shot classification using memory augmented neural networks is attempted. The idea of memory augmented networks is to use a classifier with recurrent memory, such that information from the K training examples of each class informs classification through the hidden state of the network.

The data processing is done as in SNAIL [3]. Specifically, during training, I sample batches of N classes with K + 1 samples per class. Each set of labels and images are concatenated together, and then $N * K$ of these concatenated pairs are sequentially passed through the network (as the task training set). Then the final test example of each class is fed through the network, concatenated with 0 instead of the true label. The loss is computed between these final outputs and the ground truth label, which is then backpropagated through the network as shown in the figure below.

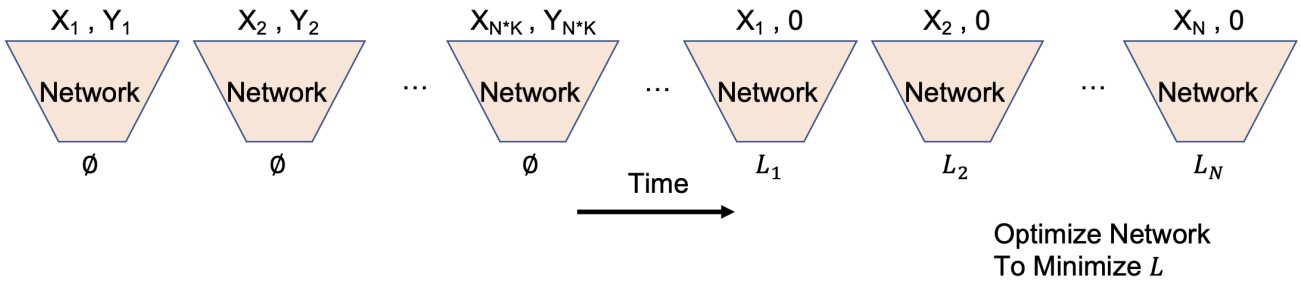


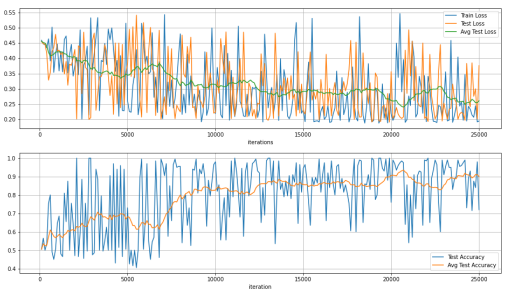
Figure 1: Feed K labeled examples of each of N classes through network with memory. Then feed final set of N examples and optimize to minimize loss.

I use two LSTM layers, first with a 128 units and the second with number of units equal to the number of classes. A batch of data of size [B,K+1,N,784] is sampled from the data generator and fed to the MANN model, which is then resized to [B,(K+1)xN,784] which then corresponds to dimensions [batch, timesteps, feature]. The model then learns how to encode the first K examples of each class into memory such that it can be used to enable accurate classification on the (K + 1)th example. Cross-Entropy loss is then computed for (K+1)th sample and backpropagated through the network.

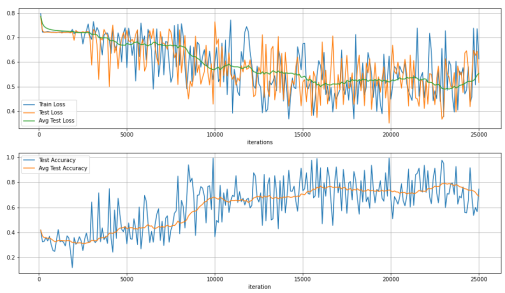
Problem 3: Analysis

This few shot classification model is then trained on a few combinations of K and N. Plots of `train loss`, `test loss`, and `accuracy` for these combination are shown below. Losses and Accuracy tend to be very noisy and hence I plot moving average as well for better visualization of the loss and accuracy trend. All the losses and accuracy reported are for moving average with a length of 20 samples, for a total number of iteration of 25000.

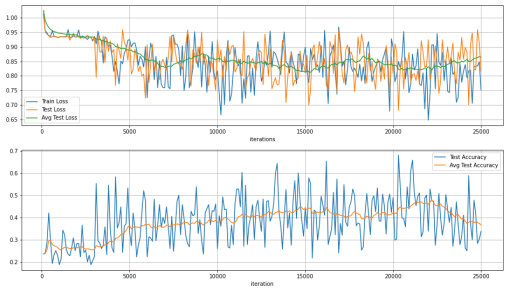
K	N	Test Accuracy (%)
1	2	91.05
1	3	73.63
1	4	37.89
5	4	74.10

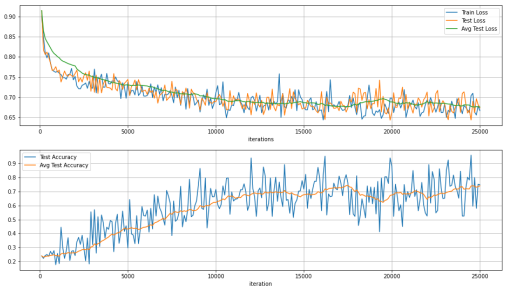


K = 1, N = 2



K = 1, N = 3





K = 1, N = 4 K = 5, N = 4

Comparision of above 4 cases are shown below.

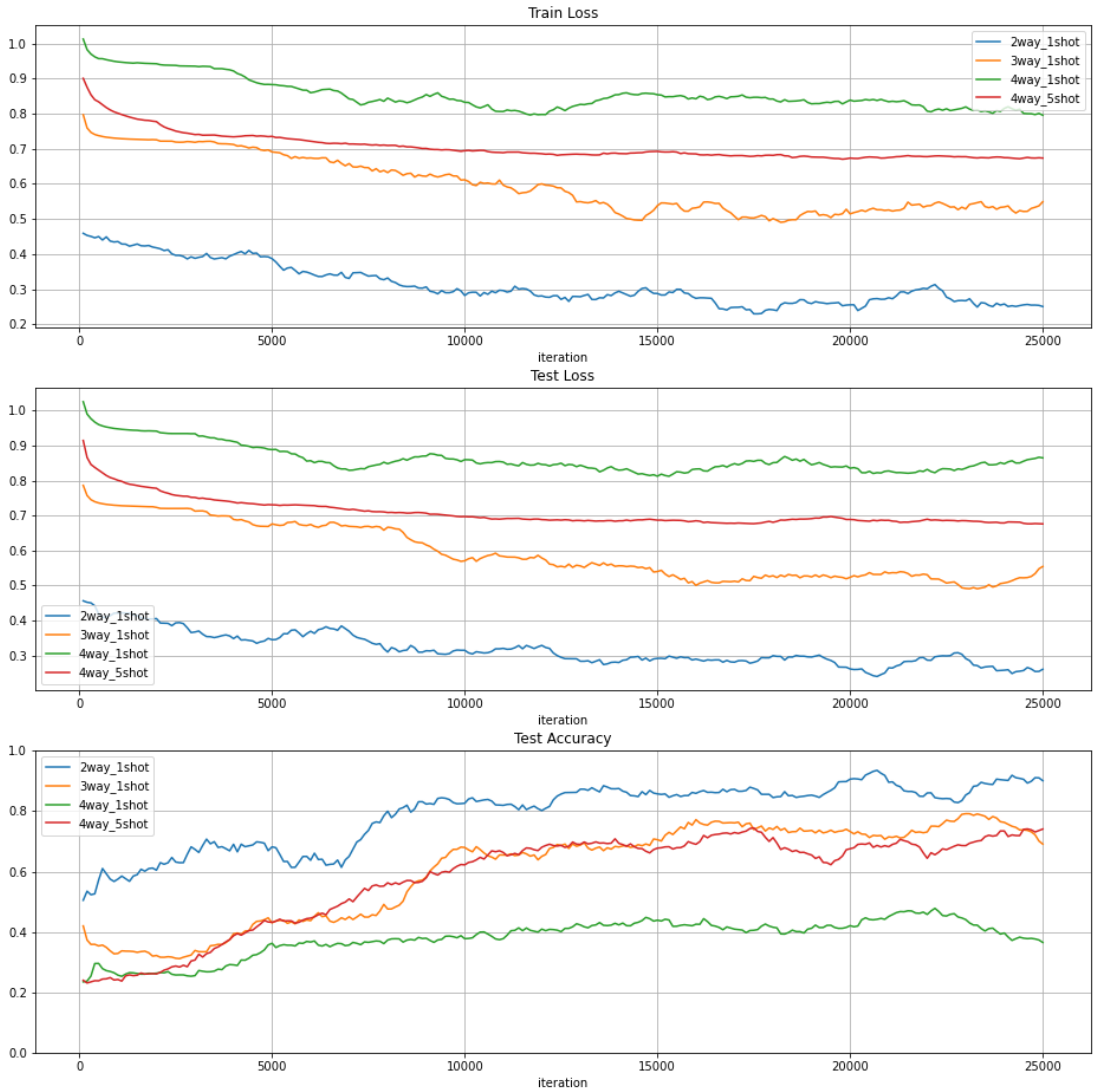


Figure - Plots comparing the model performance on 2-way 1-shot, 3-way 1-shot, 4-way 1-shot, and 4-way 5-shot classification problems

From the plots shown above, we observe that, for the same number of samples per class (K), as the number of classes (N) increase, loss takes longer to converge and accuracy takes longer to improve. This happens because the network now needs to put in more efforts and needs to learn how to differentiate between multiple classes as well as find similarities between the test sample and example classes provided. Memorizing more of the previous encoded relationships between example samples and labels are required as well. In order to improve the network performance as number of classes increase, we would need to encode more information within the network (i.e. add more RNN layers), look at more examples (i.e. increase batch size), as well as try different RNN layers.

Problem 4: Experimentation

Several parameters have been experimented with, including batch size, learning rate, RNN layers, as well as adding more layers.

a. In order to examine the effect of one of the hyperparameters that affects the performance of the model, I chose to work with the type of RNN layers. I hypothesize that, since LSTM layers have more gate units, it can take longer to converge, however they can learn more from complex data sequences. Replacing LSTM with a more simpler layer such as GRU might improve the convergence time since the data sequences we are dealing with is not too complex. As a result of using GRU unit, I see faster convergence as shown in the plot below.

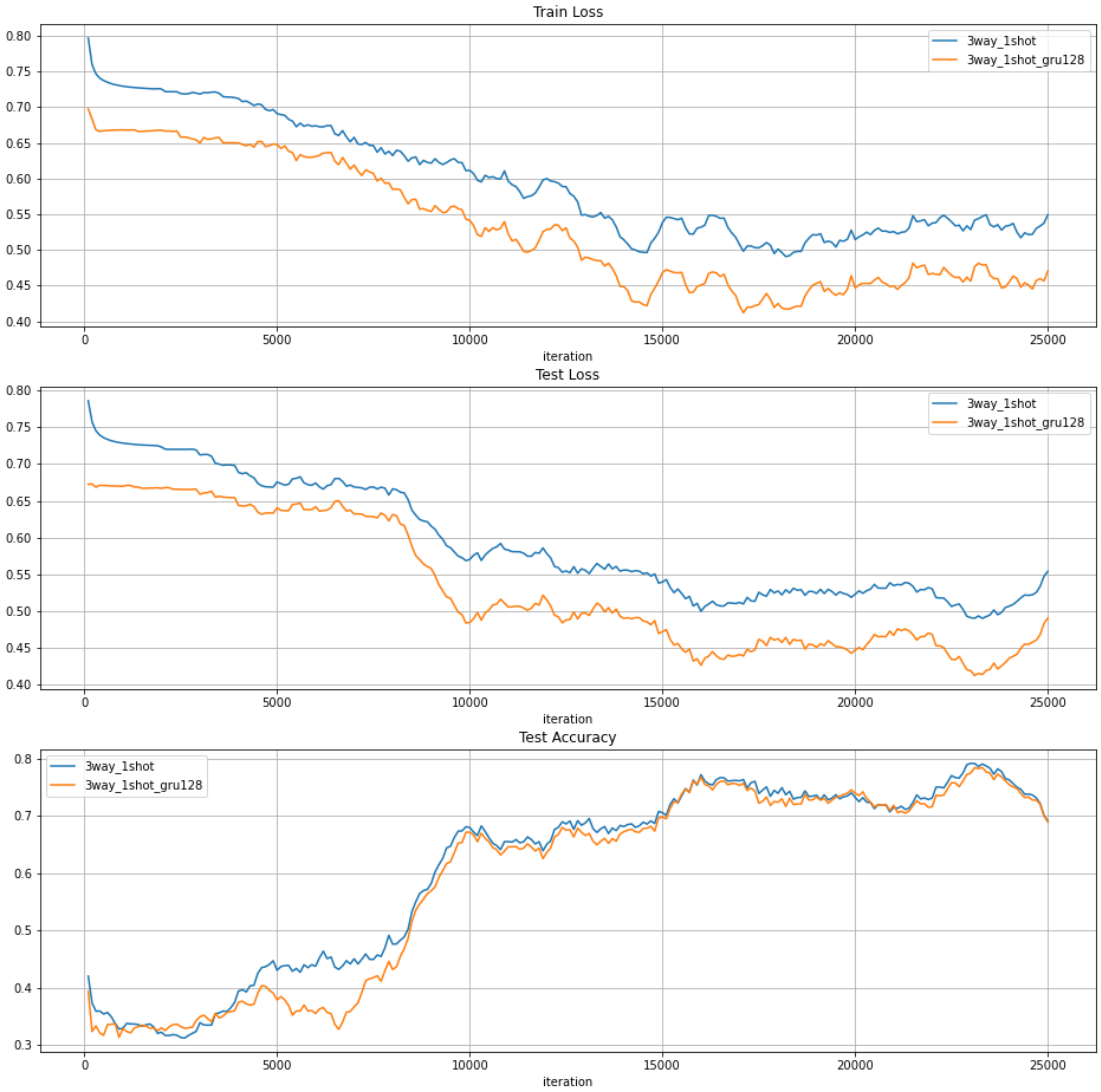


Figure - Plots showing the losses and accuracy for LSTM layers (blue line) and GRU layer (orange line). It can be seen that the loss converges very quickly with GRU layers

Furthermore, I also examine the effect of increasing batch sizes from 16 to 64 and 128. This follows my intuition from some of the works [5,6] that suggests increasing batch size adaptively, as well as by using a higher batch size in general [4]. It is evident from the plots shown below that the network seems to do slightly better as we increase the batch size (blue - batch_size=16; orange - batch_size=64; green - batch_size=128).

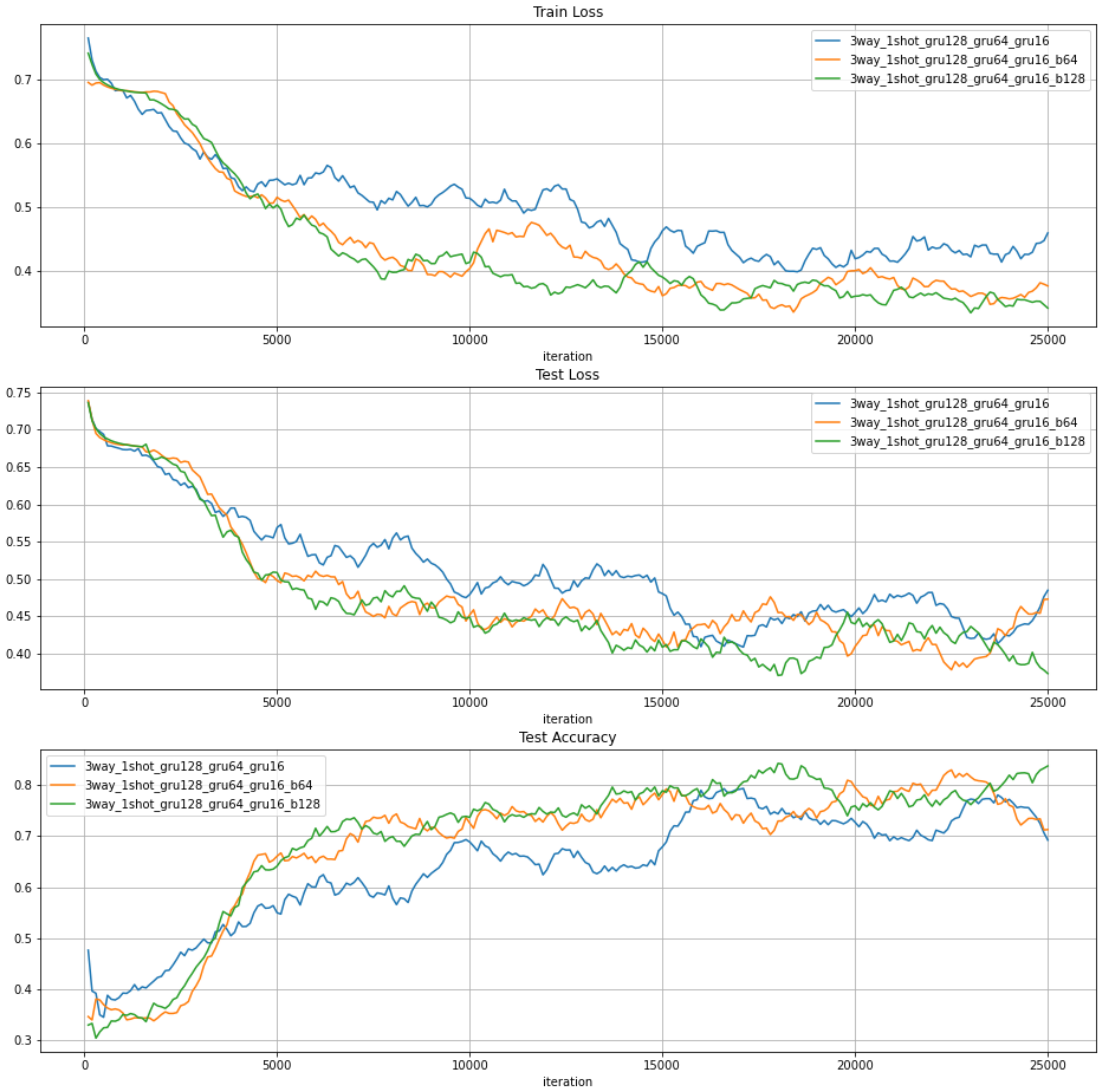


Figure - Plots showing the losses and accuracy for a model with GRU layers and increasing batch size. Network seems to do better as we increase the batch size

b. I modify the MANN architecture by replacing the LSTM layers by GRU layers and further adding more layers with decreasing number of cell states. I find that a network with 4 GRU layers with number of cell: 128, 64, 16, num_classes, provides the best performance. The plot shown below shows the trend in increased performance as I add these GRU layers.

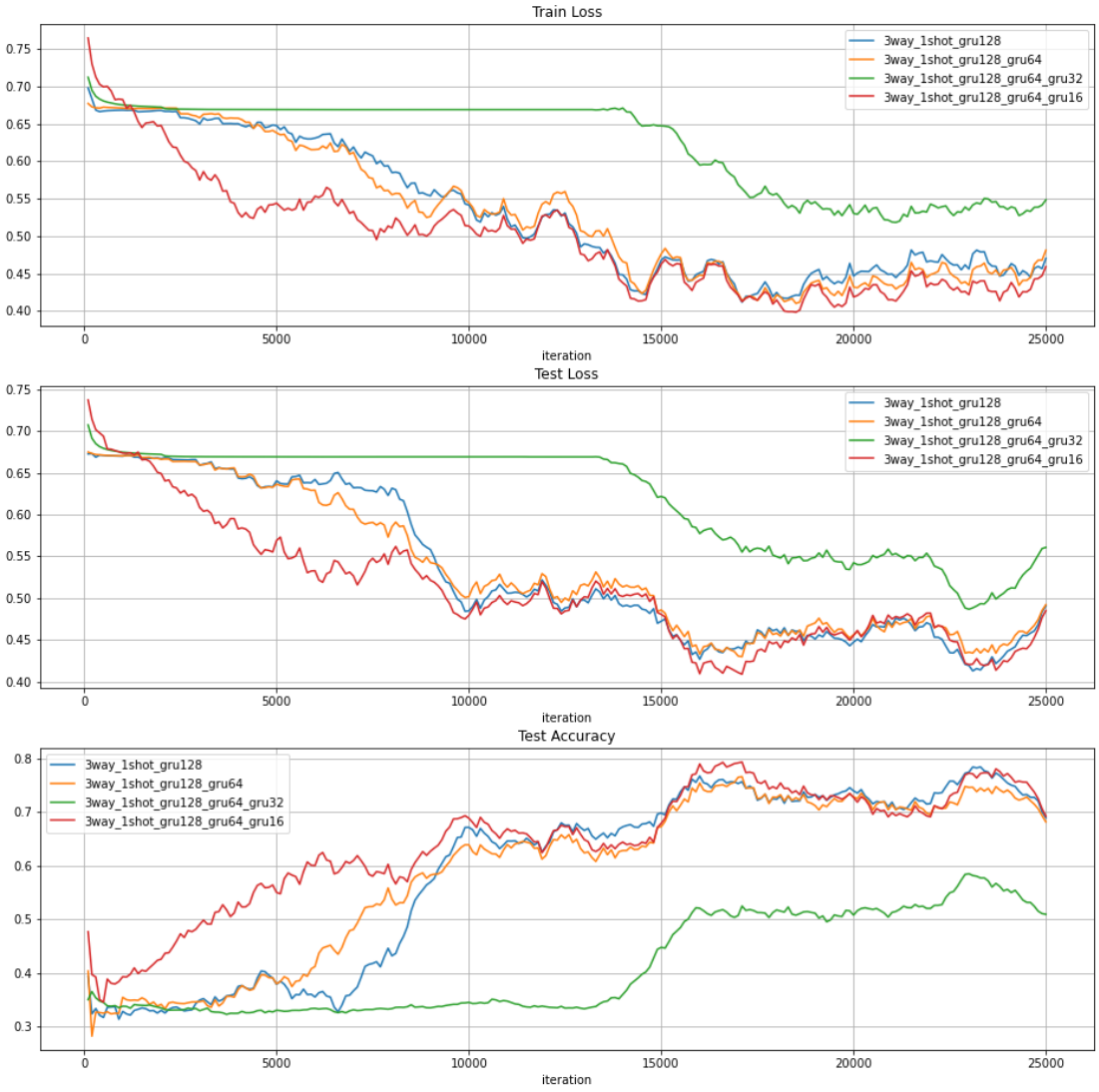
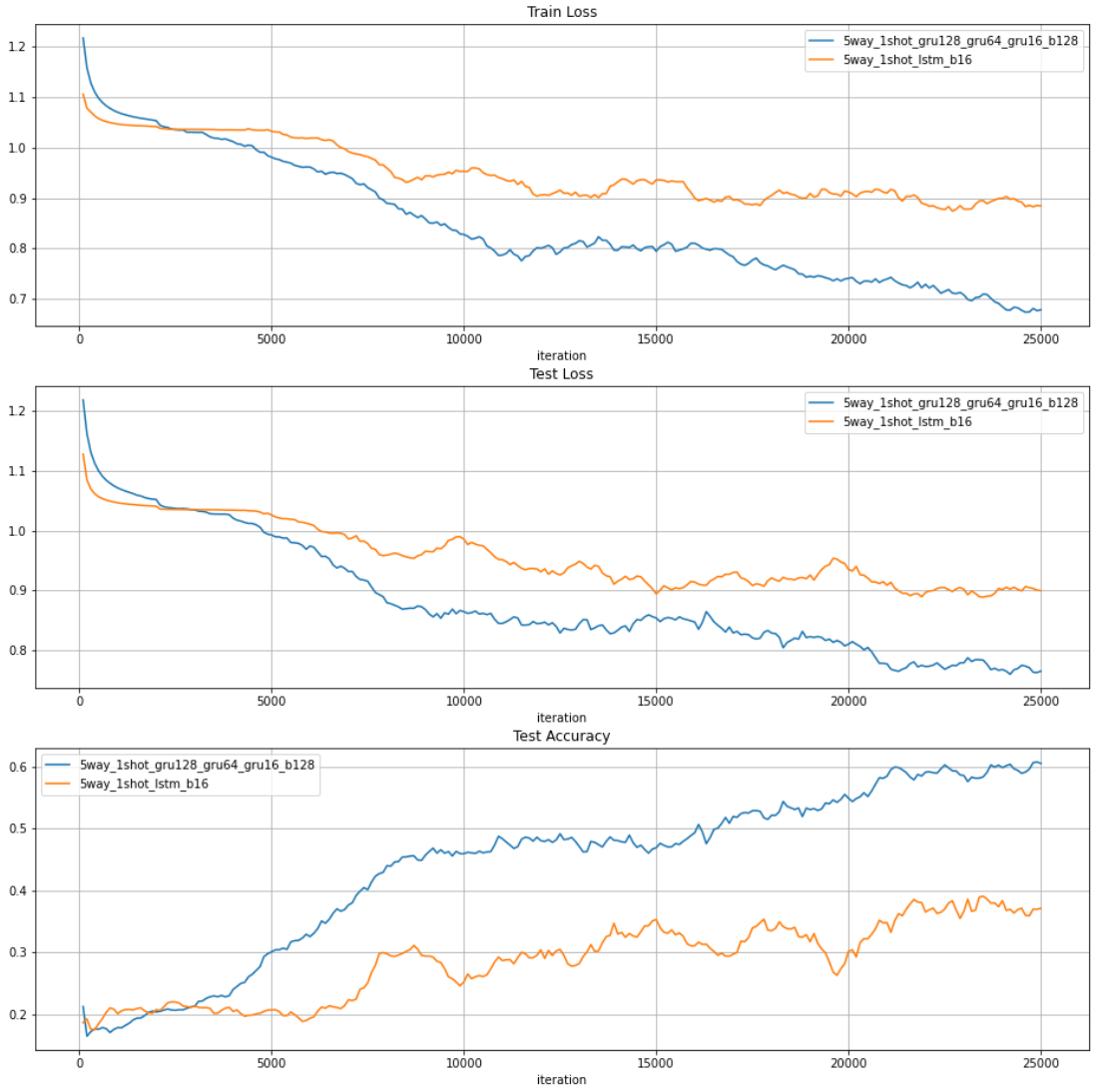


Figure - Performance increases by using more layers. In the plot: `3way_1shot_gru128` *- 2 GRU layers with number of cells (128, num_classes);* `3way_1shot_gru128_gru64` *- 3 GRU layers with number of cells (128, 64, num_classes);* `3way_1shot_gru128_gru64_gru32` *- 4 GRU layers with number of cells (128, 64, 32, num_classes);* `3way_1shot_gru128_gru64_gru16` *- 4 GRU layers with number of cells (128, 64, 16, num_classes)*

I then use this architecture with batch size of 128 for a 5-way 1-shot classification problem and compare the plots with intial model (2 LSTM layers with number of cells - 128, num_classes) with a batch size of 16. As seen in the plot below, this model outperforms the model with 2 LSTM layers by as much as 25% in terms of accuracy and is able to achieve 60% accuracy in less than 25000 iterations.



References

[1] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level con- cept learning through probabilistic program induction. Science, 350(6266):1332–1338, 2015.
[2] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lill- icrap. Meta-learning with memory-augmented neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pages 1842–1850, New York, New York, USA, 20–22 Jun 2016. PMLR.
[3] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. CoRR, abs/1707.03141, 2017.
[4] Andrew Brock, Jeff Donahue, Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. https://arxiv.org/abs/1809.11096
[5] Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, Quoc V. Le. Don't Decay the Learning Rate, Increase the Batch Size. https://arxiv.org/abs/1711.00489
[6] Zhewei Yao, Amir Gholami, Daiyaan Arfeen, Richard Liaw, Joseph Gonzalez, Kurt Keutzer, Michael

Mahoney. Large batch size training of neural networks with adversarial training and second-order information. https://arxiv.org/abs/1810.01021