CS330 Homework 1

Matheus Medeiros Centa

May 2020

1 Preliminaries

This report contains results and discussion of problems 3 and 4 of the first homework of Stanford's CS330: Deep Multi-Task and Meta-Learning course. The code used to generate these plots can be found on the file `plot.py`. All plots are smoothed using a sliding window average with a window size of 11. All experiments use the ADAM [1] optimizer with a learning rate of 1×10^{-3} unless otherwise specified. Additional plots of the learning curves of each experiment can be found on the `images` directory.

Memory-Augmented Neural Networks [2] are used to perform few-shot classification on the Omniglot dataset [3]. Long Short-Term Memory [4] (LSTM) layers are the building blocks of these networks.

2 Problem 3: Analysis

The results are summarized in Figure 1. As expected, test accuracy decreases as we increase N and keep K constant. Interestingly, more complex tasks (bigger N and K) seem to struggle more to make progress at the earlier stages of training. Given that this is a black-box approach, this may be since learning how to use memory to solve more complex tasks is difficult. This hypothesis is supported by comparing the results of 1-shot, 4-way and the 5-shot, 5-way experiments, in which even though the latter provides more data for each class it still initially lags in performance when compared to the former. This may be because the examples are shuffled before being fed to the network, which requires learning how to combine information from several examples in no specific order.

3 Problem 4: Experimentation

For this problem, two hyperparameters controlling the network architecture were chosen: the number of hidden layers and the number of units per layer. The values used were 64, 128, and 256 for the number of units and 1 or 2 for the number of hidden layers. In this setting, all hidden layers have the same amount of units. The results of these experiments can be found in Figure 2.

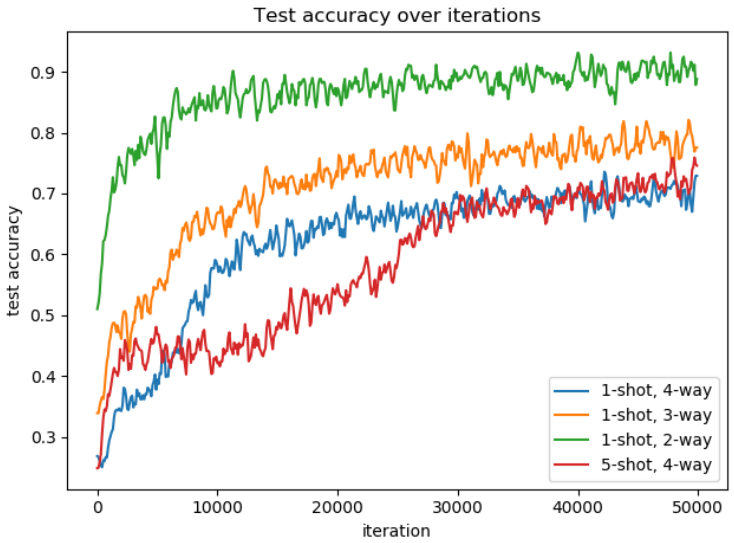


Figure 1: Results of few-shot image classification on the Omniglot dataset with an LSTM network with a hidden layer of size 128 and an output layer of size N (the number of classes). Not surprisingly, as N increases, the few-shot classification problem becomes harder, as demonstrated by the decreasing performance. An unexpected result was that increasing the number of example images from 1-shot to 5-shot in the 4-way classification runs did not improve test accuracy.

Interestingly, for experiments using a single hidden layer, the performance was best when using fewer hidden units. Conversely, when using two hidden layers, performance increased with the number of hidden units.

In the context of the bonus item, experiments were made using time-distributed convolutional layers, and convolutional LSTM [5] layers were not promising. Though the learning rate was tuned and dropout regularization applied, the models struggled to make progress and stagnated around 40% test accuracy. However, a simple model with three hidden layers with 256 hidden units each achieved 71% test accuracy. The only adjustment made was tuning the learning rate to half of the default value. Figure 3 plots test accuracy as well as train and test loss throughout the run. After the sharp rise in accuracy that finishes at around 20 thousand iterations, it is possible to see that the gap between train and test losses widens - an indication of overfitting. Perhaps regularization can help improve the model's performance.

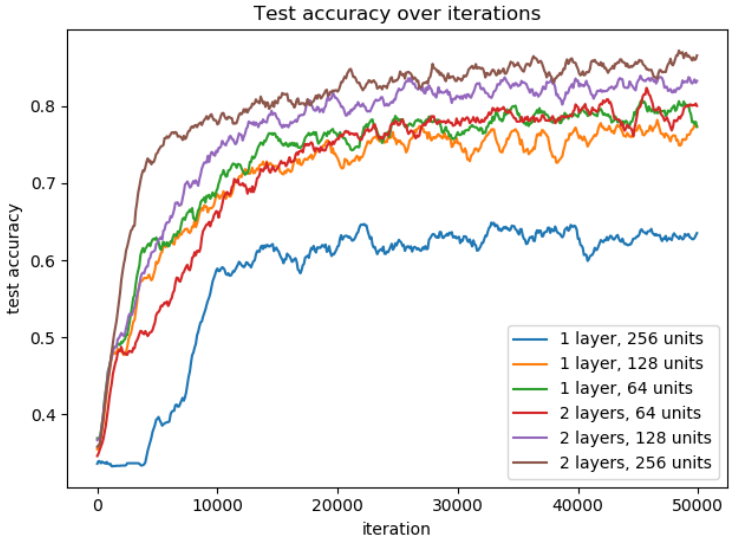
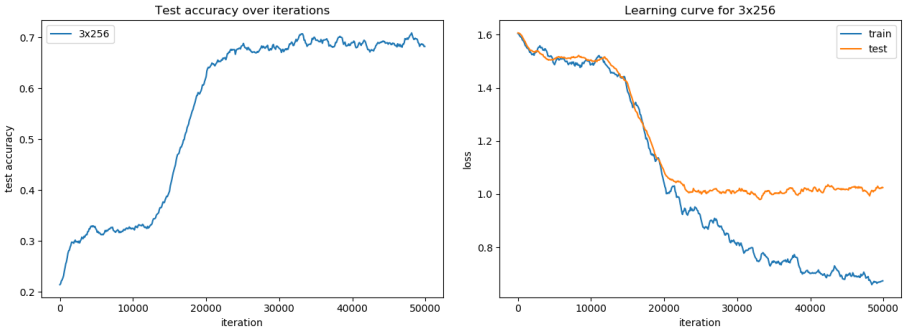


Figure 2: Results of 1-shot, 5-way image classification on the Omniglot dataset with different LSTM network architectures. The legend specifies the number of hidden layers and the number of units per layer (which is the same for all hidden layers). For the experiments with a single hidden layer, it was best to reduce the number of hidden units, with the architecture with 256 units as the worse performing. Using two hidden layers instead of one resulted in a significant performance gain for 128 and 256 hidden units per layer.



(a) Test accuracy versus iteration (b) Train and test loss versus iteration

Figure 3: Results of 1-shot, 5-way image classification on the Omniglot dataset with a LSTM architecture consisting of three hidden layers with 256 units each. After initial difficulties making progress, the model's test accuracy improves sharply at around 20 thousand iteration to around 70%. After this sharp rise, we see the gap between train and test losses widen.

References

[1] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[2] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 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. URL `http://proceedings.mlr.press/v48/santoro16.html`.

[3] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL `https://science.sciencemag.org/content/350/6266/1332`.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting, 2015.