CS 330 Autumn 2021/2022 Homework 2

Prototypical Networks and Model-Agnostic Meta-Learning Due Wednesday October 18, 11:59 PM PST

SUNetID: tminh
Name: Minh Tran
Collaborators: N/A

October 19, 2021

Abstract

The document contains solutions of implementation of the Prototypical Networks and the Model-Agnostic Meta-Learning. It also includes the results of different experiment settings on the Omniglot dataset as well as the explanation for performance.

1 Prototypical Networks

1.1 Implementation

The implementation for protonet step for each task in a batch task is in the function _step in class ProtoNet, in the file "protonet.py". For each task, the data is divided to support set to create prototypes using a embedding network and query set to classify the data by assign it to the prototype with the shortest distance from itself. The function return list of accuracy of support and query set, and the loss which is optimized to find the best parameters of the embedding network.

1.2 Performance on 5-way 1-shot Omniglot ProtoNet

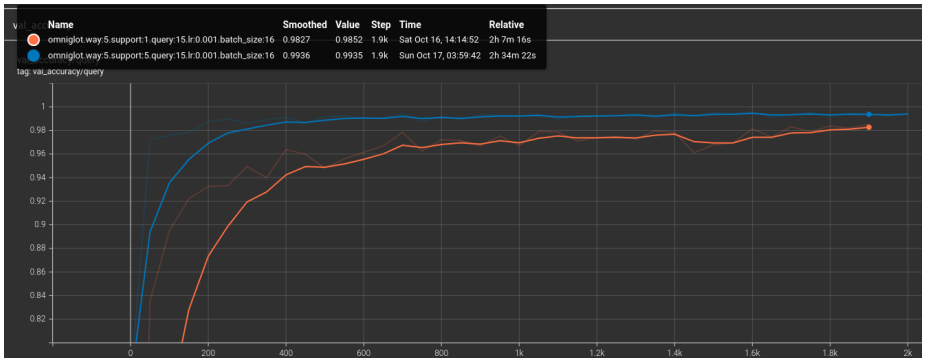

Figure 1: Query accuracy over the course of training 5-way 1-shot (red chart)

From the query accuracy 5-way 1-shot over the training course, we can see that the model reach to a stable state after around 1800 steps with the accuracy at around 98

1.3 Metric explanation

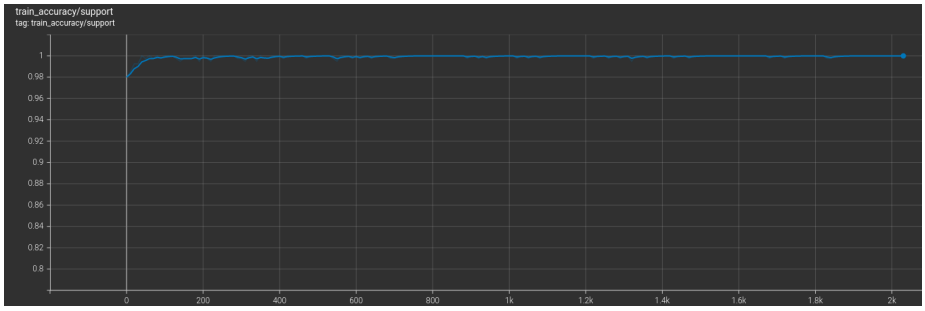

Figure 2: Support accuracy over the course of training 5-way 1-shot (red chart)

a. For the support set, the accuracy in both train and validation are always almost 100%, this makes sense since the support set is used to calculate the prototype so validating on those should return a almost perfect result. It suggests that the protonet did some clustering with the support examples of the same class.

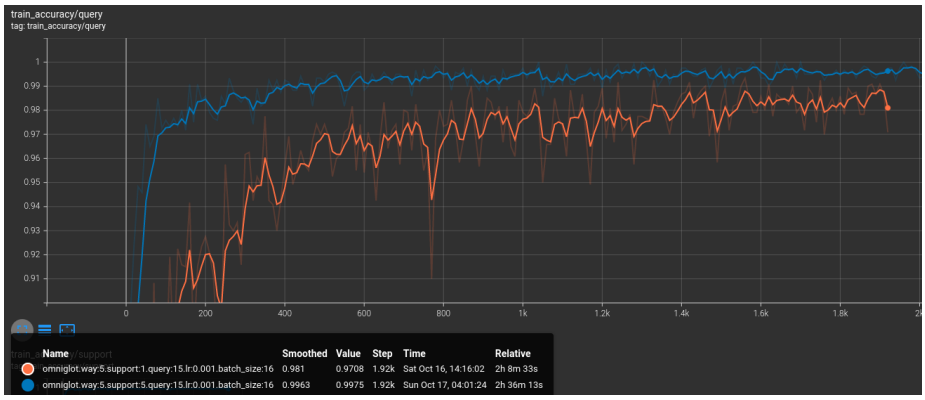

Figure 3: Query accuracy over the course of training 5-way 1-shot (red chart)

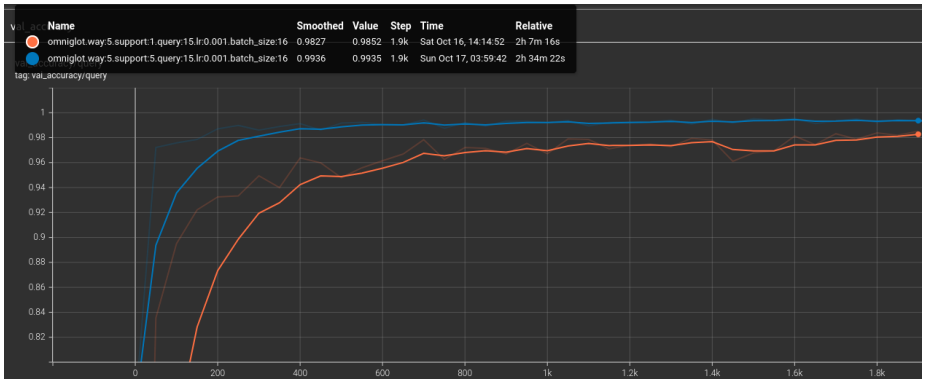

Figure 4: Query accuracy over the course of testing 5-way 1-shot (red chart)

b. For the query set, the training is more fluctuate but eventually reaches and stays at around 98%. The validation is smoother but ended up reaches slightly lower than the train accuracy but still at 97%, at the step 2000, the performance is good and the model does not have any sight of over fitting yet.

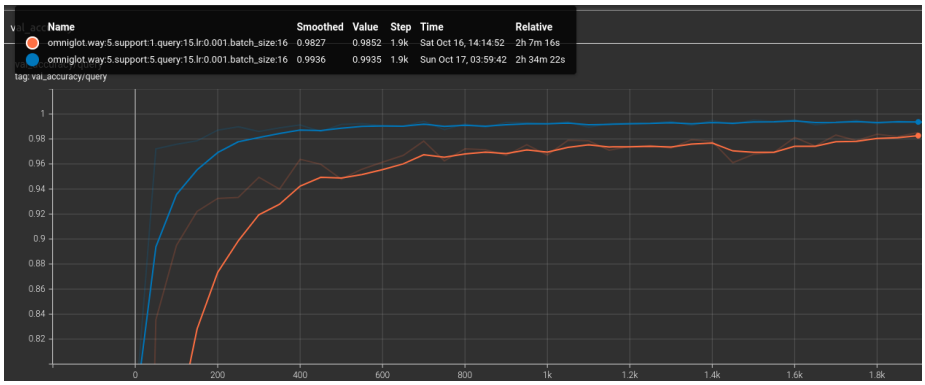

Figure 5: Query accuracy over the course of evaluating for 1-shot and 5-shot

1.4 Benchmark setting

Looking at Figure 5, we can see the query accuracy validation of 5-way 1-shot task (red chart) and 5-way 5-shot task (blue chart). I choose the checkpoint at the step 1900 since that is when both model became stable and the accuracy is high and there is no sight of over fitting. The 5-shot model ()blue) has the better performance since the beginning, however, over the training, the accuracy of both models tend to converge at 97-98 percent after step 1900. This makes sense since with 5-shot, with more data the model will have better initial embedded prototype and thus have a better generalization, more shot also increased the convergence speed during training.

2 Model-Agnostic Meta-Learning

2.1 Implementation

The implementation for MAML training step functions is in "_outer_step" and "_inner_loop" inside class MAML, details comments are in "maml.py". In the outer step, the function parse task in each task batch to support and query set, the support set was passed to the inner loop adaptation and the support set was later for testing with new parameters returned from the adaptation loop. Inside the inner loop, for each inner step, the initial loss is used to calculate the gradient of each parameter which is updated each step with the according learning rate.

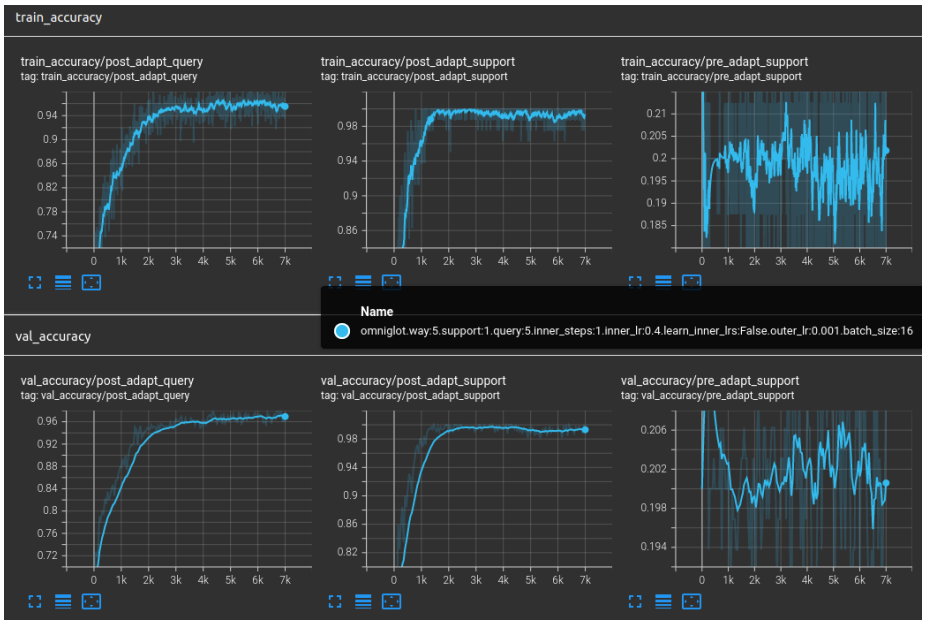

Figure 6: Train/Test Accuracy of Support/Query Set Pre/Post Adaptation

2.2 Performance on 5-way 1-shot Omniglot MAML

From the query accuracy 5-way 1-shot over the training course, we can see that the model reach to a stable state after around 2000 steps with the accuracy around 95% for both training and evaluating, without any sign of over fitting, after 2000 steps, the performance can increase but slowly and mostly is a flat line. Due to GPU memory problem, all the experiments have batch size of 4 while the other settings stay the same as given.

2.3 Metric explanation

Please use Figure 6 as a reference for answers below.
a. Without the adaptation, the model performed really bad in both training and evaluating for the support set, it reached 20% which is near perfect random for a 5-way classification. It makes sense since for the test sampling, the process is random while the model can not learn anything to get optimal parameter for new tasks.
b. For the support set training, unlike the pre-adapt model, we can see that with the adaptation, the model performed really well with the accuracy around 99% just after 2000 steps. We also observed the same pattern with the evaluation of the support set. This result indicate clearly that the model with adapted parameters is robust to new task in both training and testing process.

c. For adapted training process, both the support set and query set returned stable performance. The support set reach 99% accuracy after 2000 steps while at the same step the query set reached 94% and kept increasing stably for later steps. The same pattern appears in the validation, as well as the performance of the support set always higher than the query set which makes sense since we used the support set to calibrate the parameters.

2.4 Benchmark learning rate

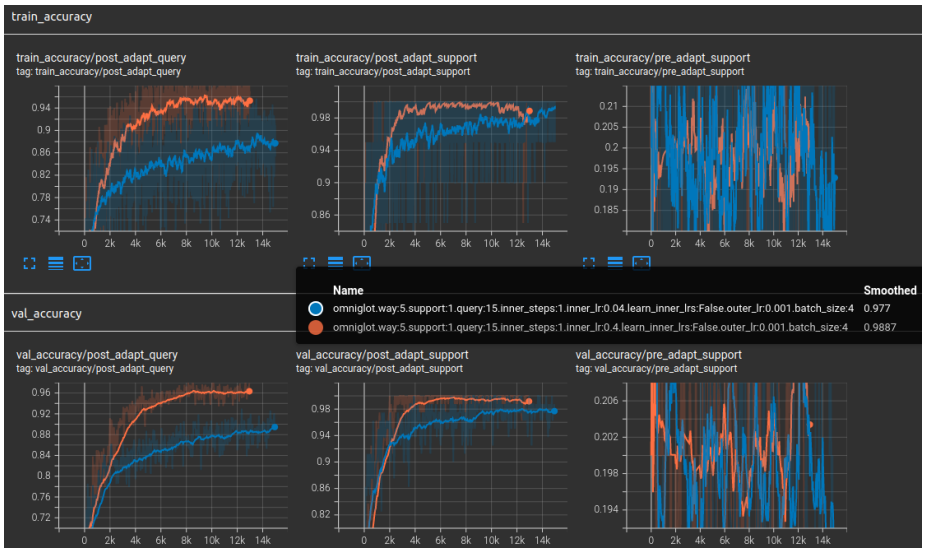

Figure 7: Performance with different learning rates 0.4 (red) and 0.04 (blue)

From Figure 7, we can observe clearly that for this model settings, smaller learning makes the model's performance on query set lower than the ones with higher learning rate. After 15000 steps, the performance of support set on two settings almost matched while there is still a gap between those for the query set. This pattern makes sense since if every gradient step is smaller so the learning phase takes more time thus the query set will have a lower accuracy but this gap will be narrow down toward the training process. Eventually, the setting with smaller learning rate can match or even slightly outperform the larger learning rate since it could optimize the loss a little bit better, one of the best solution for this case is adaptive learning rate or scheduled learning rate.

2.5 Learn the learning rate

Figure 8: Performance with normal rate (orange) and learned learning rate (red)

From the Figure 8, we can see that for query accuracy, the learned learning rate clearly helped the model to learn more efficiently from the data, but toward the end, it slowed down and eventually the original learning rate setting bested the learned one, I suspect that could be the learned rate stuck at regional optimal which could happen when the learning rate is too small and the loss surface is not smooth.

3 More Support Data at Test Time

3.1 ProtoNet vs MAML

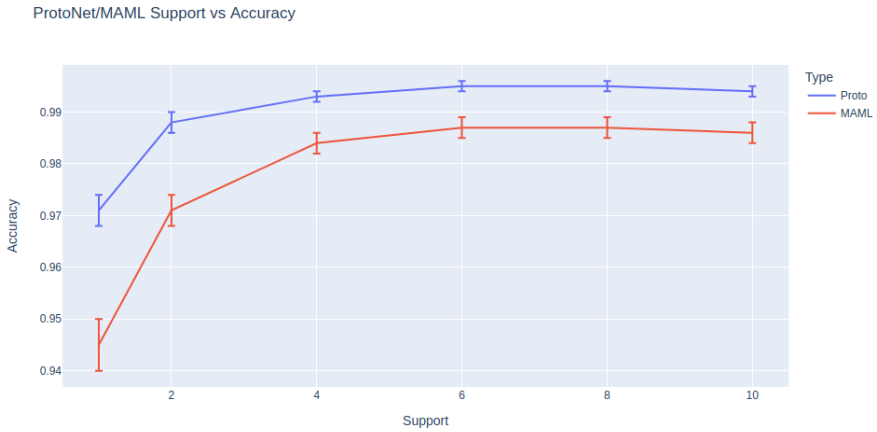

Figure 9: ProtoNet/MAML Support versus Accuracy

From Figure 9, we can see that both model performed well using additional data. I think it makes sense since for MAML is it trained to adapt to new ask while for ProtoNet, the clustering of prototype did well in some degree so that it still can distinguish new tasks, however I suspect it could be a problem for ProtoNet when the number of class is very large.