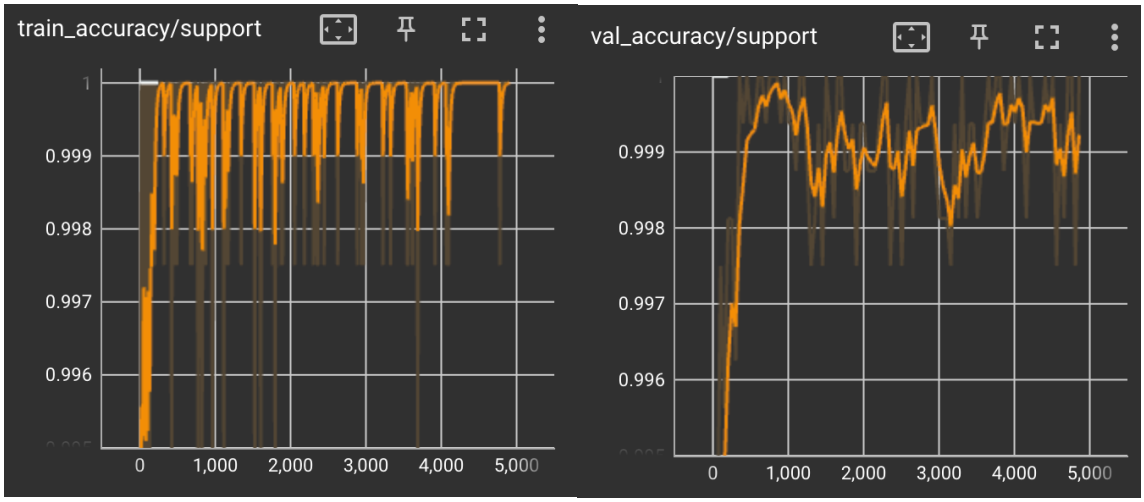
1. a

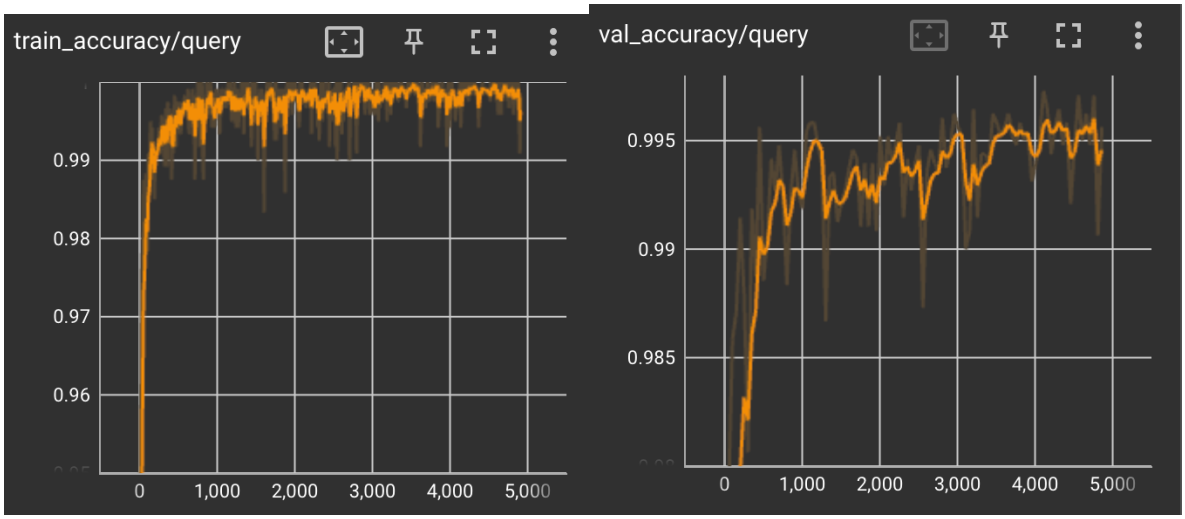
When training black-box meta-learners like the one in prior assignment we need to shuffle the order of examples in the query set, as otherwise the network can learn to output the same sequence of classes and achieve 100% accuracy, without actually learning to predict the classes. This isn't necessary when training protonets, because the class prediction is done through calculating the distance to each prototypical example of a class, and the model doesn't leverage the order of examples.

1. c(i)



Above two plots show the training/validation accuracy on support examples from a 5-way 5-shot model training. The model achieved >99% accuracy after early iterations, which means it's placing support examples of the same class close together and making accurate predictions.

1. c(ii)



Above two plots show the training/validation accuracy on query examples from a 5-way 5-shot model training. The model achieved a fairly high accuracy of above 99% for both train and valid groups after early iterations. This indicates it can generalize well to new tasks.

1. d(i)

Model Training	Checkpoint	Mean Accuracy	95% CI
5-way 1-shot	4400	0.98	0.002
5-way 5-shot	4700	0.978	0.003

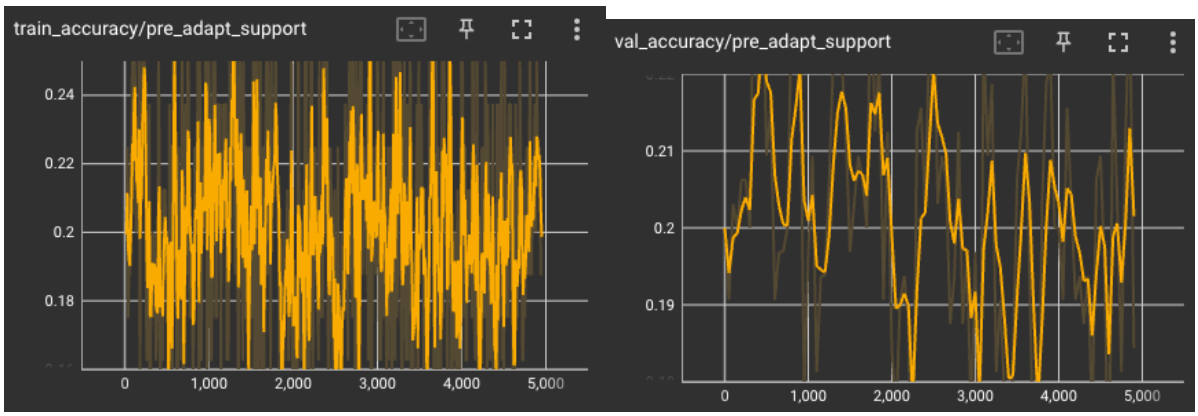
1. d(ii)

Checkpoint are chosen based on the time steps corresponding to the highest val accuracy achieved on query set.

1. d()iii)

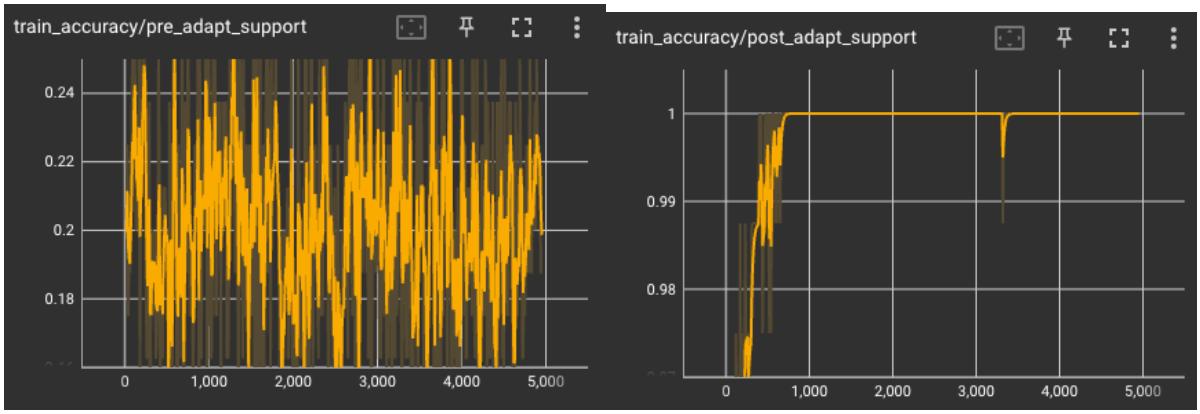
Between training on 5-way 5-shot tasks and 5-way 1-shot tasks, we do not observe a statistically significant difference in the test performance on 5-way 1-shot tasks. The model that was trained on a harder task (5-way 1-shot) performs better on 5-way 1-shot tasks than the model that was trained on an easier task (5-way 5-shot).

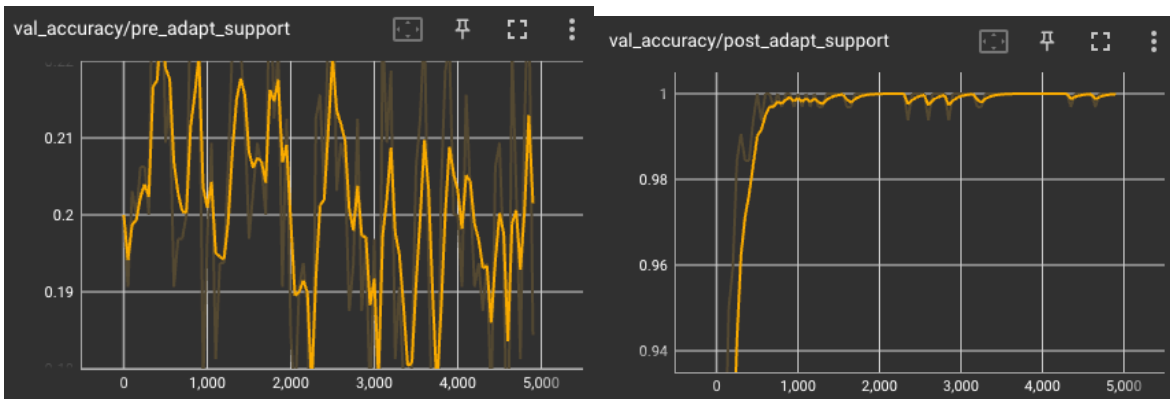
2. c (i)



Plots above show the train pre-adaptation support and val pre-adaptation support accuracies. They are computed over the stage of inner loop, where the model randomly initializes the metaparameters and performs gradient updates over randomly sampled support examples for each class. At this stage the model hasn't learned to differentiate the 5 classes and hence the class predictions are random. This explains why the accuracies for both train and val centered around 0.2 (1/5).

2. c (ii)

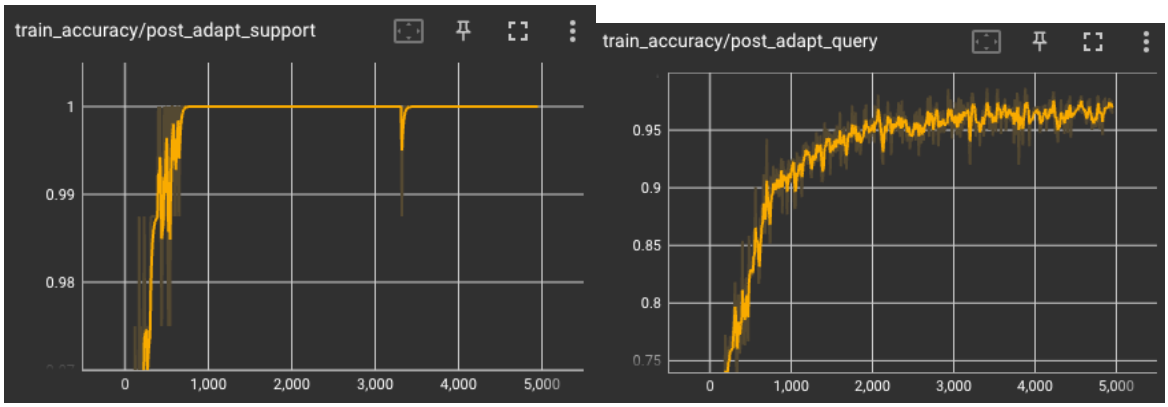


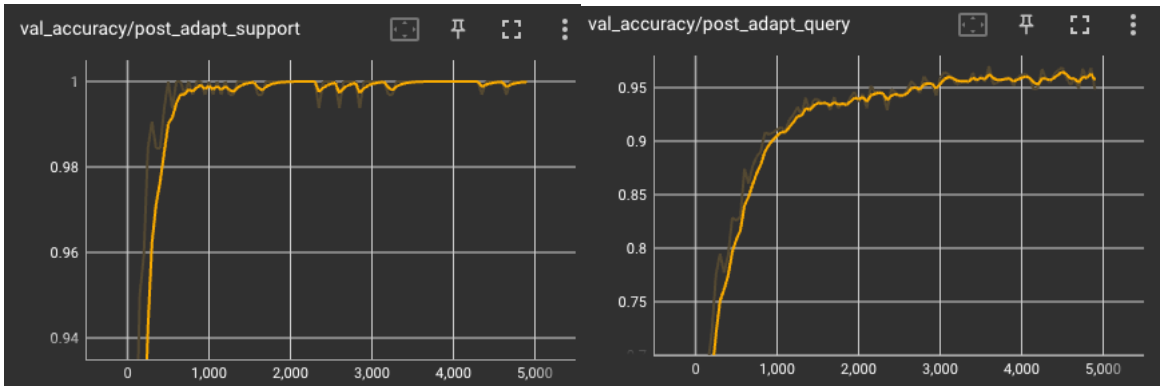


train_pre_adapt_support and val_pre_adapt_support accuracies are computed when meta parameters are randomly initialized, whereas train_post_adapt_support and val_post_adapt_support are computed with updated meta parameters. Hence we could observe an improvement of the post-adaptation accuracies over the course of training.

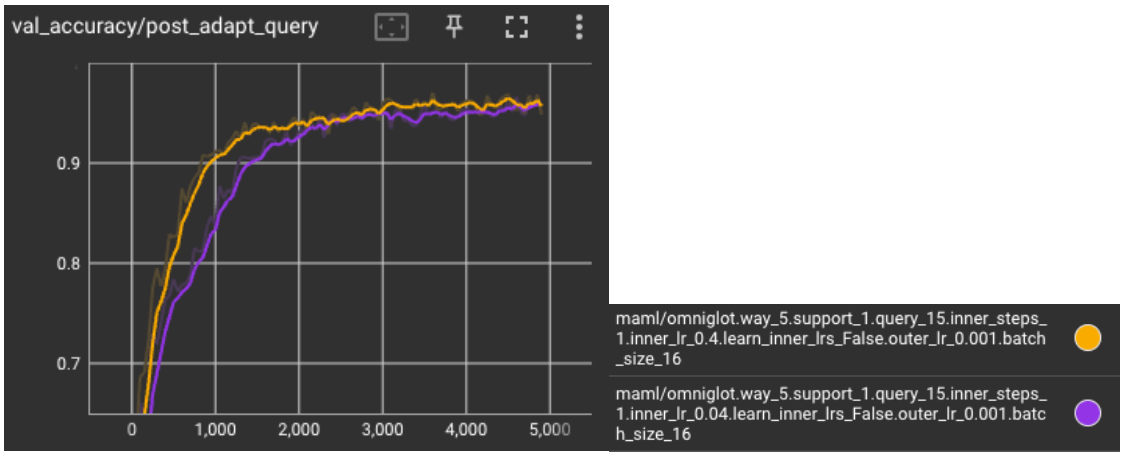
2.c (iii)

We can observe high post-adaptation accuracies on both support and query sets. Due to learning with 1 shot, train_post_adapt_support and val_post_adpt_support easily achieved 1, but the MAML network still generalizes well in the query set as inidicated by the >95% accuracies.



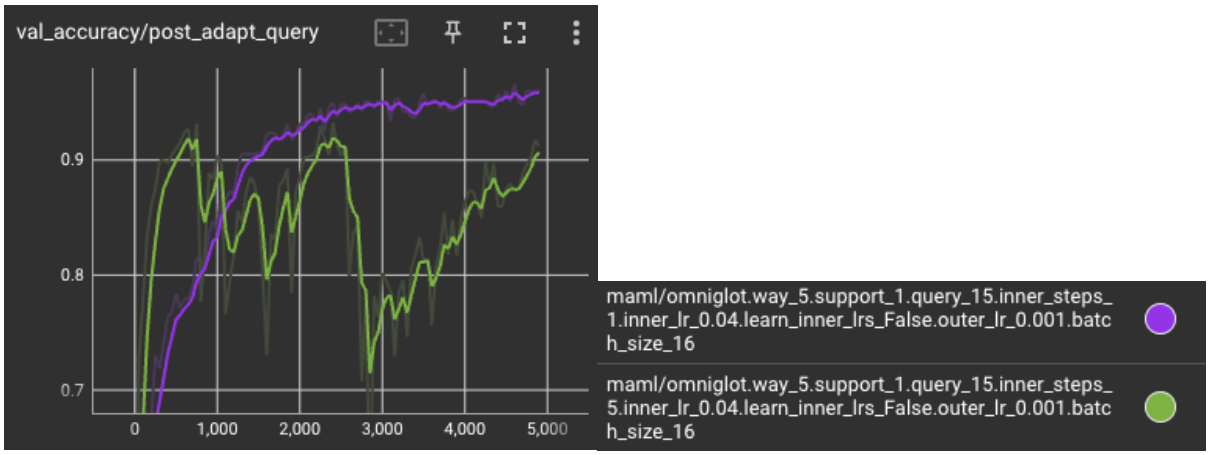


2.d (i)



Plot above compares the val post-adaptation query accuracy over the course of training with two different inner learning rates (0.4, 0.04). Reducing the inner learning rate from 0.4 to 0.04 led to a slower improvement on query accuracy over training iterations and lower query accuracies overall. Hence reducing the inner learning rate seems having a negative impact on outer-loop optimization and generalization.

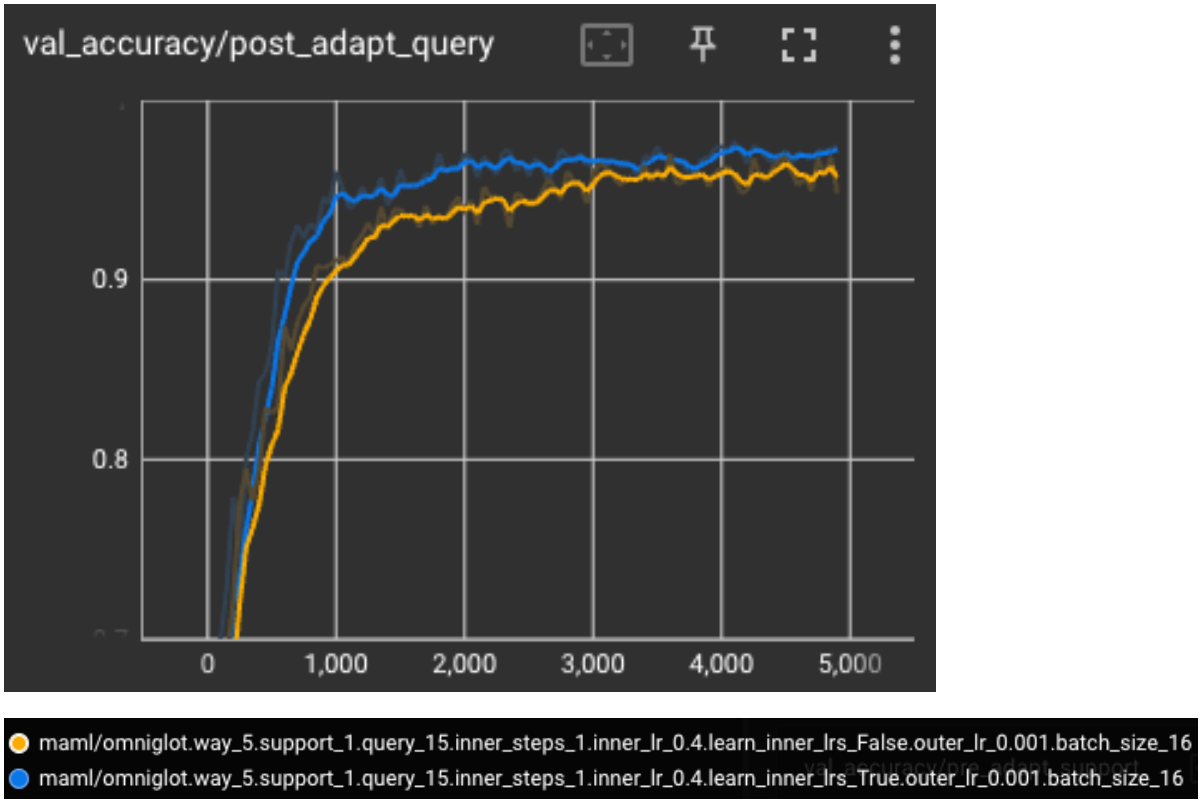
2.d (ii)



Plot above compares the val post-adaptation query accuracy over the course of training with two different inner loop steps (1, 5). It seems increasing the number of inner loop

steps can lead to more performance fluctuations and a negative impact on the outer loop optimization over the observed 5K training iterations.

2.d (iii)



The plot above shows the val post-adaptation query accuracies over the course of training for adapting and not adapting the inner learning rates with an initialization at 0.4. We can see that compared to using a fixed inner learning rate, updating the inner learning rates outperformed throughout the training iterations, leading to a better generalization to query examples.