This handout includes space for every question that requires a written response. Please feel free to use it to handwrite your solutions (legibly, please). If you choose to typeset your solutions, the `README.md` for this assignment includes instructions to regenerate this handout with your typeset LaTeX solutions.

1.a

The black-box meta-learners architecture in the previous homework used LSTM, where ordering was important. The hidden states from support set fed into the query set, which contained 1 sample for each characters. The output hidden state from the first sample was fed as input to the second sample and so on and all the characters are predicted serially one after the other. If same order (E.g., 0-¿1-¿2, where 0, 1 and 2 are the 3 classes) is used across each of the tasks, then the LSTM model would just learn this order instead of the task. So, regardless of the character used in the test set of the test task, it would classify the first character as a 0, the second as a 1 and so on. Thus, shuffling the query set was important so that the model to learns identifying the character instead of the order

However in protonets, decoding of characters in the test set is independent of the other. The loss computed on the test set is independent for each character which is later summed up and back propagated. Thus, the ordering doesn't mater and hence shuffling is not required

1.c.i

The model is placing support examples of the same class close together in the feature space.

If the support examples are kept close to each other, then the distance of these examples to their class prototypes would be the lowest. Hence those would be predicted correctly. From the train and validation support accuracy figure, we see that the accuracy on the support images are 100 %, which indicates that the support examples of the same class a placed close together by the model.

1.c.ii

The model does seem to generalize reasonably well to new tasks.

The validation tasks are not seen during training, and hence the validation accuracy on the query examples reflects how well the model generalizes to new tasks. The validation accuracy on query examples $> 99\%$ for 5-way 5-shot scenario as seen in the figure.

1.d.i

From the below table, baring at step 100, (num_class, num_shot) = (5, 1) has a higher mean accuracy than (5, 5) at 95% confidence interval.

Num classes	Num shots	Step	Mean accuracy	95% confidence interval
5	1	100	0.938	0.005
5	5	100	**0.941**	0.005
5	1	500	**0.969**	0.004
5	5	500	0.962	0.004
5	1	1000	**0.977**	0.003
5	5	1000	0.968	0.003
5	1	2000	**0.982**	0.003
5	5	2000	0.98	0.003
5	1	4900	**0.988**	0.003
5	5	4900	0.976	0.003

1.d.ii

The first checkpoint chosen was at the very beginning - after 100 steps. This would give an indication on which of the two learnt the test task quickly. The next two checkpoints were selected in between (during the learning phase) - after steps 500 and 1000. Here the model should be able to generalize reasonably well to the new tasks. The last 2 checkpoints - after steps 2000 and 4900 - were when the model should have converged.

1.d.iii

The accuracy of 5-way 1-shot model is higher than the 5-way 5-shot model on the 5-way 1-shot test task. Intuitively, one would think otherwise, since the 5-way 5-shot model was trained on using more data which would result in superior performance.

However, 5-way 5-shot model is trained for a 5-way 5-shot task. And the task that is being tested is a 5-way 1-shot task - which is a more difficult task compared to the 5-way 5-shot task. The test task (1-shot) is different from the train task (5-shot) and have a different optimal class prototypes. 5-way 5-shot should now adapt itself with the limited support examples provided in the new test task. As a results, its performance is poorer when compared to the 5-way 1-shot model.

Another way to look at it - 5-way 1-shot model may have learned more transferable features or representations that generalize better to new examples, compared to the 5-way 5-shot model. This is because, the 1-shot model has to learn to generalize from very few examples, which requires it to extract more abstract and invariant features that are applicable to a wider range of examples. In contrast, the 5-shot model has access to more training examples, which may allow it to overfit to the specific characteristics of the training data and result in less generalizable features.

2.c.i

As seen in the below figures, both the train_pre_adapt_support, and val_pre_adapt_support accuracies converges to 20% (which is a equivalent to a random prediction). Intuitively one would expect both these accuracies to start off at 20%, but it to increase as the meta parameters are being learnt during the outer loop. The pre-adapt accuracy is calculated using the meta parameters, which is learnt during the outer loop. So, though they are not optimized for the specific task, these are still being learnt and should be relatively close to the task-specific optimal parameters (so that it can quickly learn this optimal parameters using minimum number of gradient descent steps)

However in the below figure, the accuracy remains 20% throughout. This is because

- the task being sampled for training is random and never used previously
- the labels given to the characters are randomized. So, even if the model has faint idea on what the image would correspond to, the label could be anything

2.c.ii

Though the train_pre_adapt_support = 20%, the train_post_adapt_support accuracies quickly converges to 100% with num_inner_step = 1. Same trend is observed for val_pre_adapt_support, the val_post_adapt_support accuracies

This indicates few things about the model

- The model knows nothing about the task prior to adaptation (pre-adapt accuracy = 20% corresponds to a random prediction)
- Model learns the task quickly (1-shot) by adapting the meta parameters using the support images

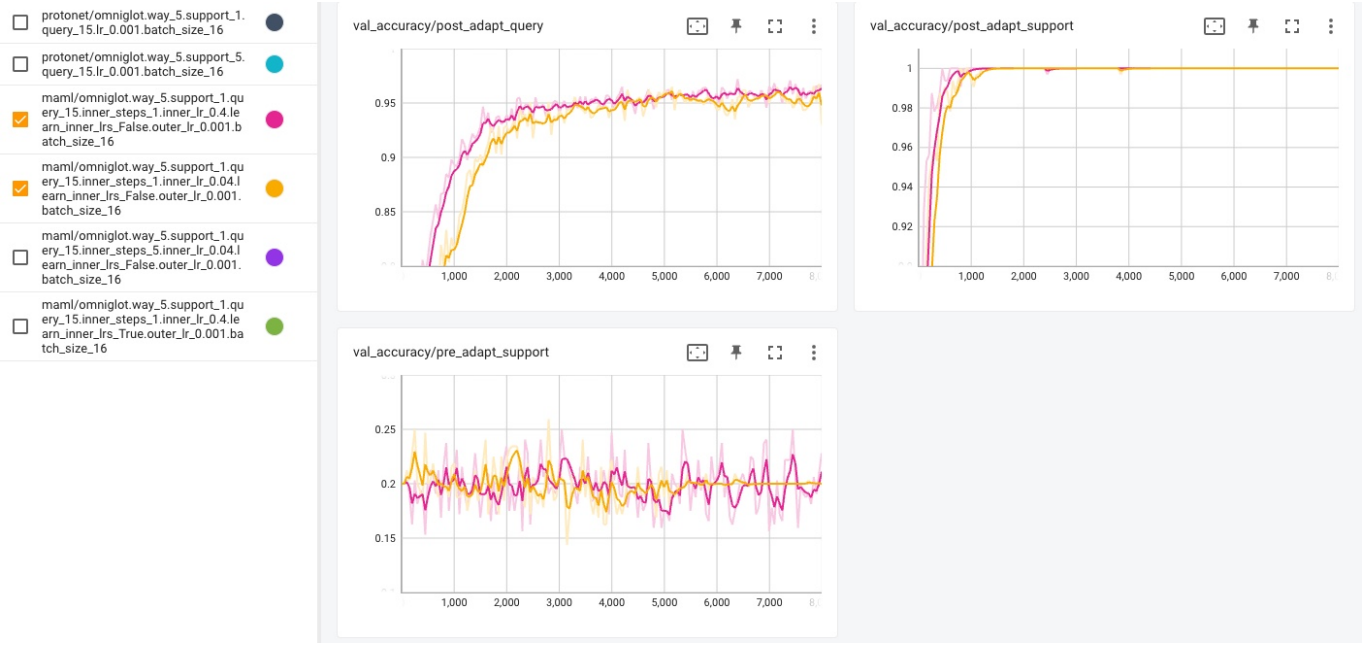
2.c.iii

While the train_post_adapt_support converges to 100% accuracy, the train_post_adapt_query accuracy converge to ¿ 95%. Comparing these two would give an idea on whether the model is over-fitting to the support images for the task.

- Both the support and query accuracies are relatively close to each other. This gives an indication that the model is not overfitting to the task during the inner-loop
- The query accuracy ¿ 95% indicates that the model generalizes very well for new tasks
- The model meta parameters are reasonably close to task-specific optimal parameters. As a result, with a single inner loop step, the model is able to achieve accuracies greater than 95%

2.d.i

From the below plot, the models validation query accuracy is lower with learning rate of 0.04 as compared to 0.4. This indicates that lowering the inner learning rate reduces the generalization of the model

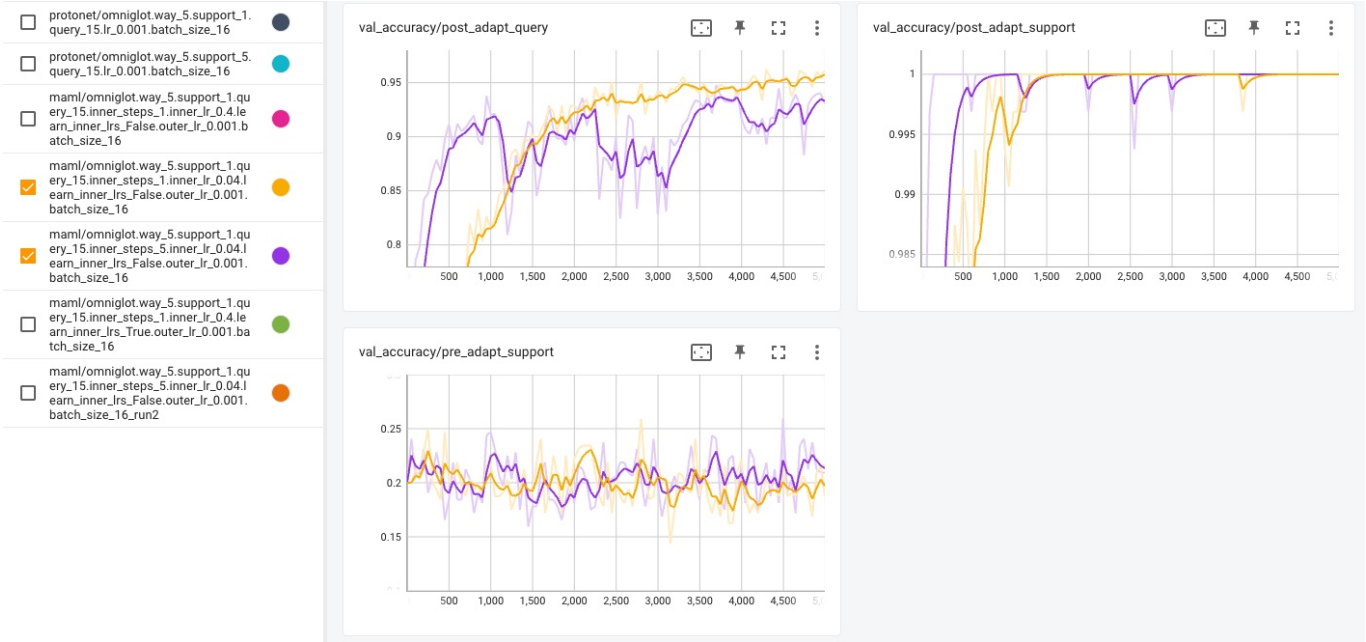


2.d.ii

Using larger number of inner loop steps should help the model generalize better. As the number of inner loop steps increases, the meta parameter can lie farther away from a particular task-specific optimal parameter ϕ_1 while trying to get closer to another task-specific optimal parameter ϕ_2

Another way to look at it - number of inner loop steps correspond to number of shots. Higher the number of inner loop steps, we have more images of each class. And hence, it should improve the models capability to generalize on unseen tasks.

In the below plot, though the validation query accuracy for 5 inner steps ¡ 1 inner step, it should eventually get higher as the outer loop is trained beyond 20k steps



2.d.iii

From the below plot, learning the inner learning rates improves the generalization capability of the model, i.e., the validation query accuracy improves