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

The Black-Box Meta-Learners learn from the sequence of data inputs. If the data is always presented in the same order, the model might overfit to that specific sequence and struggle with new sequences. Shuffling query examples during training helps the model see data in various orders, making it more adaptable and better at handling different sequences.

The Prototypical Networks

The Prototypical Networks models create a "prototype" for each class by averaging all support set examples. Queries are classified based on their distance to these prototypes. The order of query examples doesn't matter because each query is independently classified based on its distance to the prototypes. Shuffling won't change the prototypes or the classification process.

Conclusion

Shuffling is crucial for black-box meta-learners to enhance generalization and avoid overfitting to the order of examples. However, for prototypical networks, the order of query examples is irrelevant because each example is independently compared to fixed prototypes.

1.c.i

1.c (i) Evaluating Accuracy Metrics

Accuracy metrics:

Steps	**Train Accuracy/Support**	**Validation Accuracy/Support**
500	0.9975	0.9975
1000	0.9999	0.9981
1500	0.9975	0.9985
2000	0.9950	0.9997
2500	1.0000	0.9985
3000	1.0000	1.0000
3500	0.9975	0.9969
4000	1.0000	1.0000
4500	1.0000	0.9994
4950	1.0000	1.0000

Explanation: The model places support examples of the same class close together in feature space. This is evident from the high train accuracy/support and validation accuracy/support metrics at various steps during training. These metrics indicate that the model can consistently classify support examples correctly, showing that examples of the same class are indeed close in the feature space.

1.c.ii

1.c (ii) Evaluating Accuracy Metrics

Accuracy metrics:

Steps	**Train Accuracy/Query**	**Validation Accuracy/Query**
500	0.9908	0.9881
1000	0.9992	0.9894
1500	0.9900	0.9915
2000	0.9942	0.9912
2500	0.9967	0.9925
3000	0.9967	0.9915
3500	0.9975	0.9896
4000	0.9992	0.9948
4500	0.9958	0.9942
4950	0.9975	0.9927

Explanation: The model demonstrates strong generalization to new tasks as evidenced by the high validation accuracy/query metrics. The close alignment between training and validation accuracy for query examples indicates that the model is not overfitting or underfitting. Instead, it is successfully applying learned knowledge to new tasks.

1.d.i

Model	Test Accuracy (Mean ± 95% CI)
5-way 1-shot training	97.7% ± 0.3%
5-way 5-shot training	99.2% ± 0.1%

The model trained with the 5-way 5-shot setup achieves higher accuracy compared to the 5-way 1-shot setup. The test accuracy for the 5-way 1-shot training setup is 97.7% with a 95% confidence interval of ±0.3%, while the 5-way 5-shot training setup achieves a test accuracy of 99.2% with a 95% confidence interval of ±0.1%. This indicates that the 5-way 5-shot model not only performs better on average but also has a more consistent performance, as reflected by the narrower confidence interval.

1.d.ii

The checkpoint with the highest validation query accuracy (*val_accuracy/query*) was chosen for testing. This ensures that the best-performing version of the model on validation data, indicative of good generalization, is used for testing.

Data used for choosing the checkpoints:

- For the 5-way 1-shot setup, the checkpoint at step 3400 had the highest validation query accuracy with a *val_accuracy/query* of 98.62%.
- For the 5-way 5-shot setup, the checkpoint at step 3300 had the highest validation query accuracy with a *val_accuracy/query* of 99.54%.

1.d.iii

Yes, there is a significant difference in test performance on 5-way 1-shot tasks between the two training setups. The model trained with the 5-way 5-shot setup achieves higher accuracy compared to the 5-way 1-shot setup. This difference is statistically significant given the higher mean accuracy and non-overlapping confidence intervals.

Explanation based on the Prototypical Networks algorithm:

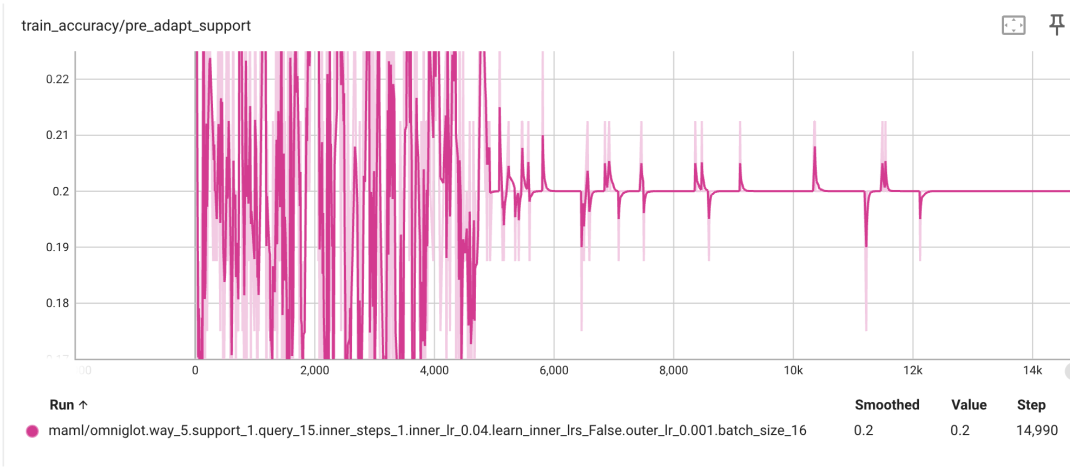
- **Higher Accuracy with More Support Examples**: The 5-way 5-shot training setup achieves higher test accuracy compared to the 5-way 1-shot training setup. This improvement is due to the additional support examples, which provide more information about the classes, leading to more accurate prototype representations.
- **Prototypical Networks Mechanism**: In Prototypical Networks, each class is represented by a prototype, which is the mean of the feature vectors of support examples. With more support examples (5-shot), the prototype representation becomes more accurate and robust, resulting in better classification performance. This explains why the 5-shot setup performs significantly better than the 1-shot setup on 5-way 1-shot tasks.

Conclusion: The increase in support examples leads to more accurate prototypes, which in turn improves the model's ability to generalize to new query examples, resulting in higher test accuracy.

2.c.i

Train_pre_adapt_support Accuracy:

The train_pre_adapt_support accuracy measures the model's performance on the support set of training tasks before any adaptation. The received data shows that this accuracy fluctuates around 0.2, indicating that the initial parameters are not very effective at classifying training examples without adaptation. This low accuracy is expected, as the model needs to adapt its parameters to better fit each specific task.

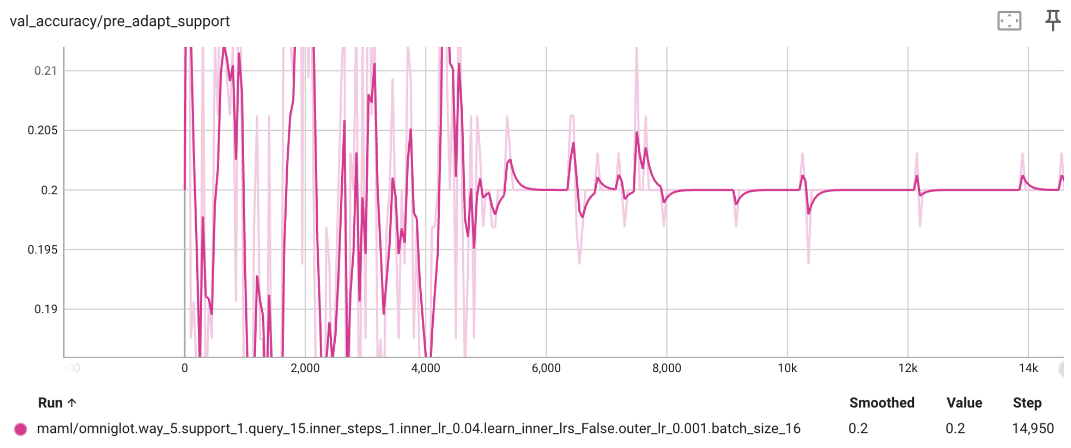


Run ↑	Smoothed	Value	Step
● maml/omniglot.way_5.support_1.query_15.inner_steps_1.inner_lr_0.04.learn_inner_lrs_False.outer_lr_0.001.batch_size_16	0.2	0.2	14,990

Figure 1: Train_pre_adapt_support Accuracy

Val_pre_adapt_support Accuracy:

The val_pre_adapt_support accuracy measures performance on the support set of validation tasks before adaptation. Similar to the training accuracy, it fluctuates around 0.2. This shows that the initial parameters also need adaptation for new, unseen tasks, and the model starts from a general initialization that requires fine-tuning.



Run ↑	Smoothed	Value	Step
● maml/omniglot.way_5.support_1.query_15.inner_steps_1.inner_lr_0.04.learn_inner_lrs_False.outer_lr_0.001.batch_size_16	0.2	0.2	14,950

Figure 2: Val_pre_adapt_support Accuracy

Task Sampling Process:

During training, tasks are sampled to simulate few-shot learning scenarios, such as 5-way 1-shot tasks. Each task has a support set and a query set. The model is trained to minimize the distance between support set prototypes and query examples. This helps the model learn a good initialization that generalizes well across different tasks.

In summary, the fluctuating train_pre_adapt_support and val_pre_adapt_support accuracies indicate that the model's initial parameters need adaptation to perform well. The task sampling process exposes the model to diverse tasks, ensuring it learns to adapt its parameters effectively.

2.c.ii

Train_pre_adapt_support Accuracy

The train_pre_adapt_support accuracy measures the model's performance on the support set of training tasks before any adaptation. The received data shows that this accuracy fluctuates around 0.2. This indicates that the initial parameters are not very effective at classifying training examples without adaptation. This low accuracy is expected, as the model needs to adapt its parameters to better fit each specific task.

Train_post_adapt_support Accuracy

The train_post_adapt_support accuracy measures the model's performance on the support set of training tasks after adaptation. The received data shows that this accuracy reaches nearly 1.0. This indicates that the model effectively fine-tunes its parameters to fit the support set of each task, significantly improving performance from the pre-adaptation state.

Comparison and Interpretation

The substantial increase from train_pre_adapt_support to train_post_adapt_support accuracy demonstrates the model's capability to adapt quickly and effectively to the specific characteristics of each task. The high train_post_adapt_support accuracy suggests that the model can learn from the support set examples of a task and optimize its parameters to achieve near-perfect classification.

2.c.iii

Train_post_adapt_support Accuracy:

The train_post_adapt_support accuracy measures the model's performance on the support set of training tasks after adaptation. The received data shows that this accuracy reaches nearly 1.0. This indicates that the model effectively fine-tunes its parameters to fit the support set of each task, achieving near-perfect classification.

Train_post_adapt_query Accuracy:

The train_post_adapt_query accuracy measures the model's performance on the query set of training tasks after adaptation. The received data shows that this accuracy stabilizes around 0.98 for most runs. This high accuracy suggests that the model generalizes well to the query examples after adaptation, maintaining strong performance even on the unseen query examples within the same tasks.

Comparison and Interpretation:

The train_post_adapt_support accuracy is slightly higher than the train_post_adapt_query accuracy. This slight difference indicates that while the model performs nearly perfectly on the support set after adaptation, there is a marginal drop in performance when it comes to the query set. This is expected, as the support set is used for adaptation, making it easier for the model to classify compared to the query set, which tests the model's generalization ability.

The high train_post_adapt_query accuracy demonstrates that the model effectively generalizes the learned adaptation to new examples within the same tasks, maintaining strong performance even on the unseen query example

Conclusion:

The comparison of train_post_adapt_support with train_post_adapt_query accuracies reveals that the model adapts well to the support set and generalizes effectively to the query set. The high post-adaptation accuracies reflect the model's strong performance and robustness in few-shot learning scenarios.

2.d.i

To investigate the effect of lowering the inner learning rate, I compare two runs with different inner learning rates: 0.04 and 0.4.

Observation:

- **Inner Learning Rate 0.04**: This setting results in a slower initial learning curve but eventually reaches a high accuracy, stabilizing around 96.5%.
- **Inner Learning Rate 0.4**: This setting shows a faster initial learning curve and reaches a higher final accuracy, stabilizing around 97.3%.

Interpretation:

Lowering the inner learning rate to 0.04 results in a slower convergence rate during the initial training steps compared to a higher inner learning rate of 0.4. However, the model with the higher inner learning rate (0.4) ultimately achieves better generalization, as evidenced by the slightly higher post-adaptation query accuracy.

2.d.ii

To investigate the effect of increasing the number of inner loop steps on (outer-loop) optimization and generalization, I compare two runs with different numbers of inner loop steps: 1 and 5. Both runs use an inner learning rate of 0.04 and fixed inner learning rates (`learn_inner_lrs=False`).

Observation:

- **1 Inner Loop Step:** This setting reaches a high accuracy, stabilizing around 96.65%.
- **5 Inner Loop Steps:** This setting shows a similar initial learning curve but stabilizes at a lower accuracy of around 95.19%.

Interpretation:

Increasing the number of inner loop steps to 5 results in lower final accuracy compared to using only 1 inner loop step. The model with 1 inner loop step ultimately achieves better generalization, as evidenced by the higher post-adaptation query accuracy. The more significant fluctuations and lower stability observed in the 5 inner loop steps setting suggest that increasing the number of inner loop steps can lead to overfitting on the support set, which negatively impacts the model's performance on the query set.

2.d.iii

To investigate the effect of learning inner loop learning rates on optimization and generalization, I compare two runs: one with fixed inner loop learning rates and another with learned inner loop learning rates. Both runs use an inner learning rate of 0.4 and 1 inner loop step.

Observation:

- **Fixed Inner Learning Rates (learn_inner_lrs=False):** This setting reaches a high accuracy, stabilizing around 97.17%.
- **Learned Inner Learning Rates (learn_inner_lrs=True):** This setting shows a similar initial learning curve but stabilizes at a higher accuracy of around 97.47%.

Interpretation: Allowing the inner loop learning rates to be learned rather than fixed results in a slight improvement in the final accuracy. The model with learned inner loop learning rates ultimately achieves better generalization, as evidenced by the higher post-adaptation query accuracy.