Studying Extrapolation of Black-Box, Optimization-Based, and Non-Parametric Few-Shot Learners

Jessica Tawade
Stanford University
Stanford, CA 94305, USA
`jt658@stanford.edu`

Extended Abstract

Meta-learning algorithms have been shown to be quite successful at learning to perform new few-shot classification tasks when sampling data from the same distribution during meta-train and meta-test time. However, can the same be said when these meta-learning algorithms are meta-trained on a certain distribution of tasks in order to perform similar, but extrapolated tasks during meta-test time? The most relevant prior work (Finn & Levine, 2018) which began exploring this question compared how well optimization-based and black-box meta-learners perform on out-of-distribution (OOD) tasks and found that the inductive bias of gradient descent in MAML enabled better few-shot learning performance on tasks outside of the training distribution when compared to black-box methods.

In this paper, we further expand upon this topic by investigating how non-parametric few-shot learning methods perform on out-of-domain data when compared with MAML and black-box meta-learners. Comparing MAML to a non-parametric few-shot learner, such as a prototypical network, allows us to see whether the inductive bias of gradient descent present in MAML is necessary for good performance on OOD tasks, or if prototypical networks result in accuracy comparable to MAML without the optimization complexity. Furthermore, testing additional forms of extrapolation provides more information about the limits each type of meta-learner has when it comes to its ability to effectively adapt to new data that further strays from the data that it was trained on.

We compared three different meta-learners for few-shot classification: a black-box meta-learner (SNAIL, Mishra et al. (2018)), an optimization-based meta-learner (MAML, Finn et al. (2017)), and a non-parametric meta-learner (prototypical networks, Snell et al. (2017)). All three meta-learners used data from the Omniglot dataset during the meta-train process to perform N-way, K-shot classification. During meta-test time, the learners were provided with "close" and "far" OOD tasks. These "close" OOD tasks still involved performing N-way, K-shot classification, but used a modified Omniglot dataset that includes scaled and sheared images as in Finn & Levine (2018). The "far" OOD tasks involved performing N-way, K-shot classification using data from the "Quick, Draw!" dataset, a collection of crude hand-drawn images spanning 345 different classes.

From these experiments, we found that the inductive bias of gradient descent seems to help on a case-by-case basis, where some extrapolated tasks benefit from the use of MAML and others yield higher accuracy simply through the use of non-parametric methods. These results highlight an existing trade-off, in need of further exploration, between optimization simplicity and accuracy. Furthermore, our findings also demonstrate that SNAIL, MAML, and ProtoNet all adapt fairly well to "close" OOD tasks, but start to quickly deteriorate when presented with "far" OOD tasks. All three-meta-learners require more shared structure than provided by "far" OOD data.

1 Introduction

Few-shot learning refers to the process of learning to perform new tasks by witnessing only a few examples (Chen et al., 2020). Meta-learning algorithms such as MAML (Finn et al., 2017), SNAIL (Mishra et al., 2018), and many others approach N-way, K-shot learning by first performing meta-training which involves sampling N classes from a task-training set and learning to execute classification when given K examples from each of the N classes. The process of classifying K examples from each of the N classes is known as a task and each task differs by the N classes that are sampled. Meta-training is then followed by meta-testing, where the model is evaluated on new tasks. The meta-test tasks are constructed by sampling K examples from N classes belonging to a separate meta-test dataset. Although these classes are separate from the meta-train classes, traditionally, meta-learners draw meta-test tasks from the same distribution as the meta-train tasks. Overall, few-shot meta-learning algorithms aim to find the common structure among all the meta-train tasks $T_1, T_2, ..., T_n$ and leverage the learned structure to quickly learn the new meta-test task.

Meta-learning algorithms have made significant progress in successfully performing few-shot classification with high accuracy when the meta-train and meta-test tasks are drawn from the same distribution (Finn & Levine, 2018). However, there exist practical applications where one may want to leverage a model trained on a certain distribution of tasks in order to perform similar, but extrapolated tasks. One such example is mentioned in Chen et al. (2020) where it may be easier to train a model using images from general classes due to the high availability of these types of images, but the ultimate goal of the model may be to classify fine-grained images, which are harder to obtain in large quantities.

With the growing push for more generalizable few-shot learning algorithms, many have embarked upon the journey to compare how well various methods can classify out-of-distribution (OOD) data during meta-test time. In Finn & Levine (2018), the optimization-based algorithm, MAML, and various black-box meta-learning algorithms were compared in order to determine whether the inductive bias of gradient descent present in MAML is necessary for good performance on OOD meta-test tasks. Their findings (Finn & Levine, 2018) demonstrated that deep gradient-based meta-learners, such as MAML, provide substantially improved performance in scenarios where there is a significant domain shift between meta-training and meta-testing tasks due to MAML's strong inductive bias for reasonable learning strategies.

Our work: In this paper, we expand upon the work done in Finn & Levine (2018) by investigating how non-parametric few-shot learning methods perform on OOD meta-test tasks when compared with optimization-based and black-box meta-learners. Non-parametric learners are known to achieve competitive performance with respect to other sophisticated algorithms (Chen et al., 2020). Adding the comparison of non-parametric learners against meta-learning methods, such as MAML, allowed us to see whether non-parametric methods result in accuracy comparable to MAML without the optimization complexity. Furthermore, our work aims to evaluate each explored model's flexibility in adapting to what we define as "close" and "far" OOD tasks. "Close" OOD few-shot classification tasks make use of a slightly modified version of held-out classes which belong to the same distribution of data used during meta-training. On the other hand, "far" OOD tasks utilize data from a similar, yet separate dataset. Testing additional forms of extrapolation provides more information about the limits each type of meta-learner has when it comes to its ability to effectively adapt to new data that further strays from the data that it was trained on.

Our contributions:

1. We provide insight on how meta-learners perform against non-parametric learners for OOD few-shot classification. Our results show that the inductive bias of gradient descent seems to help on a case-by-case basis. While MAML was able to achieve higher performance for certain "close" OOD meta-test tasks, in general ProtoNet was close or better in performance to MAML.

2. We highlight the existence of a trade-off between optimization simplicity and accuracy. In cases where the accuracy of MAML and ProtoNet is close, the decision on which one to utilize comes down to whether an application can sacrifice accuracy in order to reduce optimization complexity.

3. We investigate the generalizability of three different meta-learners (SNAIL, MAML, and ProtoNet) by sampling the meta-train and meta-test time classes from different domains. The meta-test tasks are constructed using both "close" and "far" OOD data and our findings demonstrate that while all three meta-learners adjust fairly well to "close" OOD data, they require more shared structure than provided by "far" OOD data for few-shot classification.

2 Related Work

In this work we focus on evaluating three different few-shot learners on "close" and "far" meta-test tasks in order to find baseline comparisons between the three methods and to learn each method's extrapolation capabilities. Evaluating other few-shot learners is out of the scope of this paper, but in this section we will discuss the findings of others who have looked into various learners and comparisons.

Dataset Diversity: In one of the most recent works (Triantafillou et al., 2020), a new few-shot classification benchmark called Meta-Dataset was created in an effort to provide an environment for evaluating few-shot classification using diverse sources of data. Meta-dataset is a combination of 10 different datasets: ILSVRC-2012 (Russakovsky et al., 2015), Omniglot (Lake et al., 2015), Aircraft (Maji et al., 2013), CUB-200-2011 (Wah et al., 2011), Describable Textures (Cimpoi et al., 2013),Quick Draw (Jongejan et al., 2016), Fungi (Schroeder & Cui, 2018), VGG Flower (Nilsback & Zisserman, 2008), Traffic Signs (Stallkamp et al., 2011) and MSCOCO (Lin et al., 2015). Meta-dataset aims to distinguish itself from other benchmarks by moving away from measuring only within-dataset generalization. With the push towards more generalizable models, Meta-Dataset allows models to be evaluated using entirely new distributions.

Domain Adaption and Algorithm Limitations: Two other works also focused on highlighting the limitations of existing few-shot classification algorithms in handling task extrapolations. In Dong & Xing (2019), the concept of domain adaption is defined as learning how to map from a train domain to a test domain with the existence of a shift in the data distribution from train to test (Zhang et al., 2013; Ganin et al., 2016). Dong & Xing (2019) also aimed to approach domain adaption in the context of one-shot learning by proposing an adversarial framework and utilizing techniques such as reinforced sample selection. Similarly, Chen et al. (2020) explored the idea of applying simple adaptation schemes to methods such as ProtoNet, MatchingNet, and MAML. The adaptation processes for ProtoNet and MatchingNet involved fixing the features and training a new softmax classifier during meta-test time. The adaptation for MAML involved updating the model for the sufficient number of iterations required to train a new classification layer. These meta-learners (both with and without adaptation schemes) were compared to a novel Baseline method. This Baseline method first trained a feature extractor and classifier from scratch to minimize cross-entropy classification loss during the meta-train stage. It then fixed the pre-trained parameters of the feature extractor and trained a new classifier for the OOD data during the meta-test stage. All algorithms were meta-trained to perform object recognition tasks using mini-ImageNet and then meta-tested to perform fine-grained image classification using the CUB-200-2011 dataset. Overall, the results in Dong & Xing (2019) demonstrated that the lack of in-built adaptation techniques in methods like MAML and ProtoNet cause them to fall behind the Baseline method when dealing with domain shifts, which highlighted the need for adaptation learning in the meta-training stage.

Inductive Bias of Gradient Descent Lastly, our work directly expands upon the findings of Finn & Levine (2018). Finn & Levine (2018) aimed to compare MAML to state-of-the-art recurrent meta-learners on tasks that were extrapolated from the distribution of meta-training tasks. The experiments involved meta-training SNAIL, MAML, and MetaNet to perform image classification using the Omniglot dataset. The extrapolation was introduced during the meta-test stage where a meta-test set of Omniglot images (separate from the meta-train set) were either sheared or scaled. Each experiment sheared or scaled the images by a different amount and the results demonstrated that as the shearing and scaling became more severe, black-box methods such as SNAIL and MetaNet started to degrade in performance, while MAML was still able to perform extrapolated tasks with high accuracy due to the strong inductive bias of gradient descent.

Our work aims to expand upon the experiments presented in Finn & Levine (2018) by including the comparison of non-parametric few-shot classification algorithms on extrapolated tasks such as the shearing and scaling of images while also testing “far” OOD meta-test tasks similar to those implemented using Meta-Dataset (Triantafillou et al., 2020).

3 Overview of Few-Shot Classification Algorithms

In this section, we review the few-shot classification problem details. We also briefly overview the SNAIL, MAML, and Prototypical Network (ProtoNet) algorithm details that we will be utilizing in our experiments.

3.1 Few-Shot Classification Problem

The few-shot classification problem is described in Section 1 above, but omits a few details for the sake of simplicity. These details will be briefly covered in this section.

The main goal of few-shot classification is *learning to learn* how to perform new classification tasks by witnessing a few examples of tasks. In the meta-learning case, a model first performs meta-training on a certain number of tasks. Each task i has a sampled dataset $\mathcal{D}_i$ which is sampled into two disjoint datasets: $\mathcal{D}_i^{tr}$ and $\mathcal{D}_i^{test}$. The model learns from the data points in $\mathcal{D}_i^{tr}$, which is also known as the support set and can be represented as: $\mathcal{D}_i^{tr} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_M, y_M)\}$ for a N-way, K-shot classification problem, where $M = N * K$. The model is then evaluated on held-out data points in $\mathcal{D}_i^{test}$, which is also known as the query set and can be represented as $\mathcal{D}_i^{test} = \{(\mathbf{x}_1^{ts}, y_1^{ts}), (\mathbf{x}_2^{ts}, y_2^{ts}), ..., (\mathbf{x}_Q^{ts}, y_Q^{ts})\}$, where Q is an arbitrary number of chosen query points. In the support and query set definitions above, $\mathbf{x}$ is a flattened vector of the embedding of each image and y is the associated label. These two stages within meta-training which correspond to the support and query set are known as simple *training* and *testing* respectively.

The model's ability to *learn how to learn* new, but similar (in-distribution) classification tasks is then tested during the meta-test stage using completely new classes.

3.2 Black-Box Meta-Learners: SNAIL

The black-box meta-learner (also known as a recurrent-meta learner) that we will be focusing on in this paper is a state-of-the-art model called SNAIL (Mishra et al., 2018). The SNAIL algorithm combines the use of temporal convolution (TC) and attention layers. The combination of both layers allows SNAIL to avoid constraints on the amount of experience it can effectively use while still having high-bandwidth access over its past experiences (Mishra et al., 2018). The use of multiple inter-weaved attention layers within a model that is trained end-to-end also allows SNAIL to learn the important information that it should select from the gathered experience.

The SNAIL architecture is composed of a few primary building blocks: dense block, TC block, and attention block. The dense block utilizes a single causal 1D-convolution with D filters and dilation rate R (Mishra et al., 2018). The result is then concatenated with the input and a gated activation function is used.

The TC block is generated using a series of dense blocks. The dilation rate of the dense blocks increases exponentially until the receptive field of the dilation rate reaches the sequence length T which is defined as $T = NK + 1$.

Finally, the attention block executes a single causal key-value look-up and concatenates the input to the output.

3.3 Optimization-Based Meta-Learners: MAML

MAML differs from black-box meta-learners because it focuses on optimizing for a set of pre-trained parameters θ in order to generalize effectively when fine-tuning with a small number of examples (Finn et al., 2017). The key idea of MAML is to learn a parameter vector θ, over many tasks, that transfers via fine-tuning. More concretely, MAML (Finn et al., 2017) utilizes two loops during the meta-training stage, an inner loop and an outer loop. Using data points $\mathcal{D}_i^{tr}$ from each task i, the inner loop executes a number of gradient updates on the task-specific parameters ϕ_i and then calculates the loss on test data points $\mathcal{D}_i^{ts}$ from task i using the updated parameters ϕ_i for the model. The outer loop then sums the post-update losses from all tasks and updates the original model parameters θ via a meta-gradient update in order to learn an initialization that can quickly adapt to new tasks during meta-test time and to minimize the aggregated loss. Both of these steps can be represented using the following expressions:

$$\text{Fine-tuning: } \phi_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{tr}) \tag{1}$$

$$\text{Meta-learning: } \min_\theta \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{tr}), \mathcal{D}_i^{ts}) \tag{2}$$

During the meta-testing stage, MAML uses data points from unseen classes to compute new model parameters. The new model parameters are then used to classify test examples from the same unseen classes.

As stated in (Finn et al., 2017), MAML has the benefit of the inductive bias of the inner-loop gradient descent without losing expressive power, making it an effective tool for OOD few-shot classification during the meta-test stage.

3.4 Non-Parametric Few-Shot Learners: Prototypical Networks

Prototypical Networks is a distance-based algorithm that resembles nearest neighbors for class prototypes (Snell et al., 2017). Class prototypes are created using data points from the support set $\mathcal{D}_i^{tr}$, i.e. the per-task training data. To be more specific, we take the embedding of each image within the same class ($f_\theta(x)$) and find the mean of the embeddings to create a prototype $\mathbf{c}_k$.

$$\mathbf{c}_k = \frac{1}{|\mathcal{D}_i^{tr}|} \sum_{(x,y) \in \mathcal{D}_i^{tr}} f_\theta(x) \tag{3}$$

Once a prototype is created for each class in the support set, the Euclidean distance is calculated between a query example (the per-task test data points) and each prototype. The query example is given the label of the prototype that it is closest to. More concretely, the model makes the prediction using the conditional probability which is based on the softmax function over the distance between the embedding of the query example and each prototype (Snell et al., 2017).

$$p_\theta(y = k|x) = \frac{\exp(-d(f_\theta(x), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\theta(x), \mathbf{c}_{k'}))} \tag{4}$$

4 Experiments

4.1 Experimental Setup

The experiment setup involved implementing three different learners for few-shot classification: a black-box meta-learner (SNAIL), an optimization-based meta-learner (MAML), and a non-parametric learner (prototypical networks - ProtoNet). All three types of few-shot learners were evaluated using two datasets, one of which was considered to be the standard distribution for the meta-train tasks and the other was considered to be the OOD data for the meta-test tasks. In this work, all three learners used data from the Omniglot dataset during the meta-train process to perform 5-way, 1-shot classification. During the meta-test stage, the learners were evaluated using held-out data samples from unseen classes in the Omniglot dataset. These results provided a baseline for how well the learners do when provided with data within the distribution. The learners were also provided with "close" and "far" OOD tasks during meta-test time. These "close" OOD tasks still involved performing 5-way, 1-shot classification, but were constructed using a modified Omniglot dataset that includes scaled and sheared images as in Finn & Levine (2018). The "far" OOD tasks involved performing 5-way, 1-shot classification using data from the "Quick, Draw!" dataset. The "Quick, Draw!" dataset is a collection of crude hand-drawn images spanning 345 different classes. The Omniglot and the "Quick, Draw!" datasets do share some structure since they both include hand-drawn images, but simultaneously are different enough to be considered as two separate distributions. A visualization of the experimental setup can be seen in Figure 1.

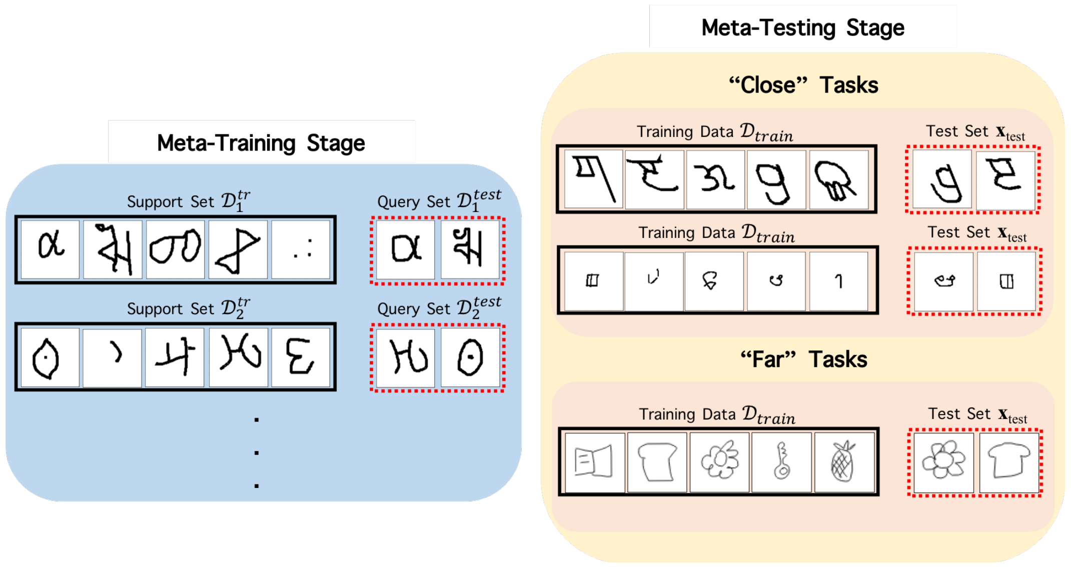

Meta-Training Stage
Support Set $\mathcal{D}_1^{tr}$
Query Set $\mathcal{D}_1^{test}$
Support Set $\mathcal{D}_2^{tr}$
Query Set $\mathcal{D}_2^{test}$
Meta-Testing Stage
"Close" Tasks
Training Data $\mathcal{D}_{train}$
Test Set $\mathbf{x}_{test}$
"Far" Tasks

Figure 1: **Experiment Setup for 5-way 1-shot classification.** During meta-training, a model learns how to perform a fixed amount of tasks. During each task, 5 Omniglot characters are sampled and 1 example of each character is randomly selected to create the support set. A few held out test examples are used to construct the query set, which is used to evaluate the model. During meta-testing, the model's ability to perform OOD tasks is tested. These OOD tasks are broken down into two categories: "close" and "far." "Close" OOD tasks are extrapolated from the meta-training data distribution (sheared and scaled Omniglot images), while "far" OOD tasks use a separate dataset ("Quick, Draw!").

4.1.1 Dataset Formulation and Sampling

The Omniglot dataset used for our experiments consisted of 1623 different handwritten characters from 50 different alphabets. Each character class had 20 examples of the character. The original images had dimensions 105×105, but were downsampled to be 28×28 following prior work Santoro et al. (2016). During each experiment, the 1623 character folders are shuffled and 1100 character folders are chosen for the meta-training stage, 100 are chosen for the meta-validation stage, and the remaining 423 folders are for the meta-test stage. For the baseline experiments, we did not modify the meta-test character images. However, for the "close" OOD tasks, we focused on two types of modifications: scaling and shearing. For each shearing experiment, we iterated through all the meta-test character folders and sheared each image by the same amount. We ran 8 different shearing experiments for each of our 3 learners and the shearing amounts we focused on were: -0.8, -0.6, -0.4, -0.2, 0.2, 0.4, 0.6, and 0.8 (radians). Similarly, we ran 7 different scaling experiments for each of our 3 learners and the scaling amounts were: 0.25, 0.5, 0.75, 1.25, 1.5, 1.75, and 2. All image shearing and scaling was performed on the original 105×105 images and then downsampled to be 28×28. Each sheared image was also re-centered after shearing to avoid meta-testing with cut-off character images.

The "far" OOD tasks were generated using data from the "Quick, Draw!" dataset which consisted of 345 unique image classes. Each class contains hundreds of images, but in order to mimic the structure of the Omniglot meta-test task dataset, I randomly selected 20 images for each class when constructing the dataset. These images were 28×28 and in grayscale bitmap format, similar to the Omniglot images.

During each iteration of the meta-train, meta-test, and meta-validation stages, a batch of images was sampled. The batch sampling process for a N-way K-shot classification problem involved randomly sampling N classes from the associated meta-train, meta-validation, or meta-test dataset and then randomly sampling K images from each of the N classes. This process was repeated for the specified number of batches (B). The image data was structured as an array with dimensions $[B, N, K, D]$ where D is the dimension of the flattened images (784 in the case of a 28×28 Omniglot image). The corresponding labels for the images were generated sequentially where images in class n, where $n \in [0, N-1]$, were given the label n in the one-hot encoding format. The labels corresponding to the image data were also structured as an array with dimensions $[B, N, K, N]$.

We also followed the embedding architecture detailed in Mishra et al. (2018) and Finn & Levine (2018) which repeated the following block 4 times: {3x3 conv (64 filters), batch norm, ReLU 2x2 max pool}, and then ended with a single fully-connected layer to output a 64-dimensional feature vector. Finally, all three models were evaluated using cross-entropy loss.

Links to all code segments can be found in the Appendix.

4.1.2 SNAIL Experiment Details

The meta-training and meta-validation stages of our SNAIL implementation runs for 7 epochs with 10,000 iterations (episodes) per epoch. The model utilizes a batch size of 32 and a learning rate of 0.0001 with the Adam optimizer. Most of the same specifications hold true for the meta-testing stage as well. The only difference is that the meta-testing stage runs for 10 epochs.

We follow the general structure of the SNAIL network for few-shot classification as described in the Appendix of Mishra et al. (2018). For our 5-way, 1-shot experiments, the sequence length is $T = NK + 1 = 6$ where $N = 5$ and $K = 1$. We used the following blocks: AttentionBlock(64,32), TCBlock(T,128), AttentionBlock(256,128), TCBlock(T,128), AttentionBlock(512,256), and finally a 1×1 convolution with N filters.

4.1.3 MAML Experiment Details

The MAML algorithm structure for our experiments followed the description from Section 3.3. The model was trained for 65,000 meta-iterations with 2 inner gradient steps and an initial inner loop update learning rate of 0.04. Similar to SNAIL, the model used the Adam optimizer with a learning rate of 0.0001 for the outer loop update. The batch size for both the meta-train and meta-test stages was 25 and the meta-test stage ran for 600 iterations.

In addition to the vanilla-MAML implementation described in Section 3.3, we also implemented a modification to MAML from Antoniou et al. (2019) which automatically learns the inner update learning rate during meta-training. This modification attempts to use backpropagation to learn a separate inner update learning rate for each gradient update for each weight variable stored in the dictionary of weights in the model. Since the updated weights in the inner loop and the learning rate used to apply gradients in the inner loop update are trainable parameters, the outer loop will optimize both the weights and the inner loop learning rate. This modification helps address the possible gradient instabilities that can be associated with bi-level optimization.

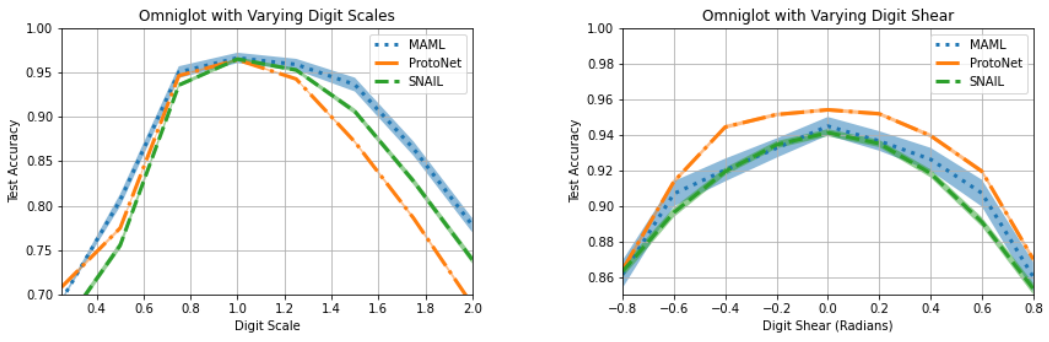
4.1.4 ProtoNet Experiment Details

The meta-training stage of the ProtoNet experiments were run for 40 epochs, each of which contained 500 episodes, while the meta-testing stage was run for 4000 episodes. Each task used 5 query data points during meta-training and 4 query data points during meta-testing. The model also utilized a learning rate of 0.0001 with the Adam optimizer. Although ProtoNets can train for N-way classification and test for $> N$-way classification, we kept N consistent during meta-training and meta-testing.

4.2 Experiment Results

4.2.1 "Close" OOD Experiments

As described in Sections 4.1 and 4.1.1, the "close" OOD meta-test tasks were performed by scaling and shearing Omniglot images that were kept aside from the meta-training data. Our results for MAML and SNAIL seem to follow the general trend as described in Finn & Levine (2018), where MAML seems to recover more generalizable learning strategies when compared to SNAIL. However, overall, the results from Figure 2 show mixed results in terms of which learner was able to perform the best at "close" OOD tasks. Interestingly, the digit scale experiment results in Figure 2a show two different trends based on the direction of extrapolation. To be more specific, we see that as the images are expanded, MAML and SNAIL seem to out-perform ProtoNet in few-shot clas-



(a) Digit Scale (b) Digit Shear

Figure 2: **"Close" Out-of-Distribution Task Results.** The inductive bias of gradient descent seems to help on a case-by-case basis. MAML was able to out-perform SNAIL and ProtoNet for all scaling experiments, but not for any shearing experiments. This suggests that gradient-based meta-learning has an advantage only is certain scenarios.

sification abilities. The margin between the meta-test accuracies isn't as large for the 1.25 scaled experiments, but as the scaling factor increases, there seems to be a 3-4% gap between the meta-test accuracy of ProtoNet and SNAIL as well as between SNAIL and MAML. On the other hand, as we shrink the images, MAML still out-performs ProtoNet and SNAIL, but by a much slimmer margin. Furthermore, ProtoNet is able to achieve higher accuracy than SNAIL as the scaling factor decreases.

The results from the digit shear experiments, shown in Figure 2b, are more consistent across all shearing values. ProtoNet achieves the highest accuracy for every shearing experiment. The margin between the accuracies of all three learners is much larger between -0.4 radians and 0.4 radians than it is at values less than -0.4 radians and greater than 0.4 radians.

When comparing the accuracy between Figures 2a and 2b, we see that the accuracy of all three learners decreased quite gradually as the shearing tasks became more extrapolated. In contrast, the accuracy of all three learners decreased at a faster rate as the scaling tasks became more extrapolated. A possible explanation for this behavior may be that the scaling experiments result in more of a domain shift than the shearing experiments. As the scaling of Omniglot images becomes more severe, the classification task becomes more challenging because the image is either expanded outside of the image window or the image loses valuable features as it is shrunken down and downscaled. The sheared images seem to remain within the image window and don't lose many features during the downscaling process.

These results indicate that the inductive bias of gradient descent seems to help on a case-by-case basis. There are certain tasks, such as the scaling tasks, that benefit from the the fact that MAML's in-built optimization allows the algorithm to recover more generalizable learning strategies. However, there are other tasks, such as the shearing tasks, which don't require the optimization complexity of MAML to obtain good performance and instead can utilize algorithms such as ProtoNet which are computationally fast and easy to optimize.

These results also highlight the trade-off between optimization simplicity and accuracy. In cases where the meta-test accuracy of ProtoNet is slightly below the accuracy of MAML, the choice of which algorithm to use becomes more application-dependent. If the application of one's classification algorithm highly values energy-efficiency and fast-computation, it may be more beneficial to use ProtoNet even if the accuracy is slightly lower than when utilizing MAML. The opposite can also be said if the application values accuracy more than computation speed and efficiency.

4.2.2 "Far" OOD Experiments

As described in Sections 4.1 and 4.1.1, the "far" OOD meta-test tasks were performed using images from the "Quick, Draw!" dataset. The results in Table 1 show that ProtoNet out-performed SNAIL and MAML when given "far" OOD tasks during meta-test time. However, the more interesting

Algorithm	Meta-test Accuracy (In-Distrubution)	Meta-test Accuracy (OOD)
SNAIL	96.47 ± 0.043	48.78 ± 0.021
MAML	95.34 ± 0.011	41.57 ± 0.016
ProtoNet	96.39 ± 0.055	50.51 ± 0.0043

Table 1: **"Far" Out-of-Distribution Task Results**: While all three learners were able to adjust fairly well to "close" OOD tasks, the "Quick, Draw" results demonstrate that these algorithms require more shared structure between meta-train and meta-test tasks than provided by "far" OOD data.

observation is the comparison between the in-distribution meta-test accuracy and the OOD meta-test accuracy for each learner. To be more concrete, the accuracy dropped 40-50% by evaluating with "Quick, Draw!" images. This indicates that while these three learners may adjust fairly well to "close" OOD data, they all require more shared structure between the meta-train and meta-test data sets than what was provided by the "far" OOD tasks in our experiments. These results align with the findings in Chen et al. (2020) which conclude that algorithms such as SNAIL, MAML, and ProtoNet fail to perform well when given a completely different data set at meta-test time due to the lack of adaptation.

Another possible reason for the low meta-test accuracy could be the structure of the "Quick, Draw!" dataset. Unlike the Omniglot dataset where all 20 examples of any character are incredibly similar, the "Quick, Draw!" dataset is more vague at times. For example, one of the classes in the "Quick, Draw!" dataset is "book" which has many different interpretations. Furthermore, there are classes that look very similar to one another such as "book" and "passport." As a result, it's possible that the chosen example of "book" in the support set is quite different from the example of "book" in the query set or if the "book" and "passport" classes are chosen in the same task, the model could easily miss-classify the examples. A possible future extension to this work could try to select a subset of the "Quick, Draw!" dataset to avoid these possible inconsistencies.

5 Conclusion and Future Work

In this paper, we have investigated three different few-shot classification learning algorithms and their performance on OOD tasks during meta-test time. By studying the ability of each few-shot classification learner to accurately execute "close" and "far" OOD tasks, we have not only gained insight on the performance comparison of ProtoNet against MAML and SNAIL, but have also learned more about the limit to which each individual model can be generalized. While, the results did not indicate a clear algorithm as consistently out-performing the others, we did observe that the inductive bias of gradient descent isn't always necessary to maintain high accuracy on OOD tasks.

An interesting extension to this project would be to apply all three algorithms to other various types of datasets in order to better test the "far" OOD task capabilities. Furthermore, other extensions could evaluate modified versions of these learners, such as Proto-MAML (Triantafillou et al., 2020) in order to see whether newer algorithms have been able to incorporate domain adaptation as specified in Chen et al. (2020). Any of these extensions would help to provide more insight on the topic of OOD tasks and create advances towards better handling of domain shifts between the meta-train and meta-test stages.

6 Acknowledgments

I would like to thank Dr. Chelsea Finn, my project mentor Rafael Rafailov, and the CS 330 teaching staff for providing code templates and guidance for MAML and ProtoNet. I would also like to thank GitHub user "eambutu" for providing code for SNAIL.

References

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml, 2019.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification, 2020.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild, 2013.

Nanqing Dong and Eric Xing. Domain adaption in one-shot learning. pp. 573–588, 01 2019. ISBN 978-3-030-10924-0. doi: 10.1007/978-3-030-10925-7_35.

Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks, 2016.

Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. The quick,draw! – a.i. experiment, 2016. URL https://quickdraw.withgoogle.com.

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.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft, 2013.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner, 2018.

M. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, pp. 722–729, 2008. doi: 10.1109/ICVGIP.2008.47.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

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 (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1842–1850, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/santoro16.html.

Brigit Schroeder and Yin Cui. Fgvcx fungi classification challenge 2018, 2018. URL github.com/visipedia/fgvcx_fungi_comp.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning, 2017.

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pp. 1453–1460, 2011. doi: 10.1109/IJCNN.2011.6033395.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples, 2020.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

K. Zhang, B. Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *ICML*, 2013.

A Appendix

This section contains links to code used in Google Colab Documents.

SNAIL Implementation for Sheared Omniglot: https://colab.research.google.com/drive/1VNkd7UWsl91udZ8FV7CNkubG6k4E7Sup?usp=sharing

SNAIL Implementation for Scaled Omniglot: https://colab.research.google.com/drive/1uxDg7Kjqgq9t9oFo5K_vzlINKrtKAK-R?usp=sharing

SNAIL Implementation for Quick Draw: https://colab.research.google.com/drive/1_Vu-DQvOR2cpuK7SwCnvFiQJeDa0jbdN?usp=sharinghttps://colab.research.google.com/drive/1_Vu-DQvOR2cpuK7SwCnvFiQJeDa0jbdN?usp=sharing

MAML and ProtoNet Code: https://colab.research.google.com/drive/1Zp2Mr49i7Z1tpZWMdYwqw90ByZ83Tyxb?usp=sharing

The MAML and ProtoNet Code was modified from the Homework 2 code. The same Colab file was used for all ProtoNet and MAML experiments. This was possible because in order to change the meta-test OOD tasks, only one code segment needed to be changed in the DataGenerator class. This code segment assigned the folders of the meta-test data to a variable. For each experiment, I re-assigned the variable to point to the corresponding meta-test data for the experiment.

The SNAIL files needed to be separated for each type of OOD task because the data generation structure of the code provided by GitHub user "eambutu" needed more involved changes for each OOD meta-test experiment.