Unraveling Meta-Learning: Understanding Feature Representations for Few-Shot Tasks

Micah Goldblum [1] **Steven Reich** [* 1] **Liam Fowl** [* 1] **Renkun Ni** [* 1] **Valeriia Cherepanova** [* 1] **Tom Goldstein** [1]

Abstract

Meta-learning algorithms produce feature extractors which achieve state-of-the-art performance on few-shot classification. While the literature is rich with meta-learning methods, little is known about why the resulting feature extractors perform so well. We develop a better understanding of the underlying mechanics of meta-learning and the difference between models trained using meta-learning and models which are trained classically. In doing so, we introduce and verify several hypotheses for why meta-learned models perform better. Furthermore, we develop a regularizer which boosts the performance of standard training routines for few-shot classification. In many cases, our routine outperforms meta-learning while simultaneously running an order of magnitude faster.

1. Introduction

Training neural networks from scratch requires large amounts of labeled data, making it impractical in many settings. When data is expensive or time consuming to obtain, training from scratch may be cost prohibitive (Altae-Tran et al., 2017). In other scenarios, models must adapt efficiently to changing environments before enough time has passed to amass a large and diverse data corpus (Nagabandi et al., 2018). In both of these cases, massive state-of-the-art networks would overfit to the tiny training sets available. To overcome this problem, practitioners pre-train on large auxiliary datasets and then fine-tune the resulting models on the target task. For example, ImageNet pre-training of large ResNets has become an industry standard for transfer learning (Kornblith et al., 2019b). Unfortunately, transfer learning from classically trained models often yields sub-par performance in the extremely data-scarce regime or breaks down entirely when only a few data samples are available in the target domain.

Recently, a number of few-shot benchmarks have been rapidly improved using *meta-learning* methods (Lee et al., 2019; Song et al., 2019). Unlike classical transfer learning, which uses a base model pre-trained on a different task, meta-learning algorithms produce a base network that is specifically designed for quick adaptation to new tasks using few-shot data. Furthermore, meta-learning is still effective when applied to small, lightweight base models that can be fine-tuned with relatively few computations.

The ability of meta-learned networks to rapidly adapt to new domains suggests that *the feature representations learned by meta-learning must be fundamentally different than feature representations learned through conventional training*. Because of the good performance that meta-learning offers in various settings, many researchers have been content to use these features without considering how or why they differ from conventional representations. As a result, little is known about the fundamental differences between meta-learned feature extractors and those which result from classical training. Training routines are often treated like a black box in which high performance is celebrated, but a deeper understanding of the phenomenon remains elusive. To further complicate matters, a myriad of meta-learning strategies exist that may exploit different mechanisms.

In this paper, we delve into the differences between features learned by meta-learning and classical training. We explore and visualize the behaviors of different methods and identify two different mechanisms by which meta-learned representations can improve few-shot learning. In the case of meta-learning strategies that fix the feature extractor and only update the last (classification) layer of a network during the inner-loop, such as MetaOptNet (Lee et al., 2019) and R2-D2 (Bertinetto et al., 2018), we find that meta-learning tends to cluster object classes more tightly in feature space. As a result, the classification boundaries learned during fine-tuning are less sensitive to the choice of few-shot samples. In the second case, we hypothesize that meta-learning strategies that use end-to-end fine-tuning, such as Reptile (Nichol & Schulman, 2018), search for meta-parameters that lie close in weight space to a wide range of task-specific

---

[*]Equal contribution [1]University of Maryland, College Park. Correspondence to: Micah Goldblum <goldblum@umd.edu>.

minima. In this case, a small number of SGD steps can transport the parameters to a good minimum for a specific task.

Inspired by these observations, we propose simple regularizers that improve feature space clustering and parameter-space proximity. These regularizers boost few-shot performance appreciably, and improving feature clustering does so without the dramatic increase in optimization cost that comes from conventional meta-learning.

2. Problem Setting

2.1. The Meta-Learning Framework

In the context of few-shot learning, the objective of meta-learning algorithms is to produce a network that quickly adapts to new classes using little data. Concretely stated, meta-learning algorithms find parameters that can be fine-tuned in few optimization steps and on few data points in order to achieve good generalization on a task $\mathcal{T}_i$, consisting of a small number of data samples from a distribution and label space that was not seen during training. The task is characterized as *n-way*, *k-shot* if the meta-learning algorithm must adapt to classify data from $\mathcal{T}_i$ after seeing k examples from each of the n classes in $\mathcal{T}_i$.

Meta-learning schemes typically rely on bi-level optimization problems with an *inner loop* and an *outer loop*. An iteration of the outer loop involves first sampling a "task," which comprises two sets of labeled data: the support data, $\mathcal{T}_i^s$, and the query data, $\mathcal{T}_i^q$. Then, in the inner loop, the model being trained is fine-tuned using the support data. Finally, the routine moves back to the outer loop, where the meta-learning algorithm minimizes loss on the query data with respect to the pre-fine-tuned weights. This minimization is executed by differentiating through the inner loop computation and updating the network parameters to make the inner loop fine-tuning as effective as possible. Note that, in contrast to standard transfer learning (which uses classical training and simple first-order gradient information to update parameters), meta-learning algorithms differentiate through the entire fine-tuning loop. A formal description of this process can be found in Algorithm 1, as seen in (Goldblum et al., 2019a).

2.2. Meta-Learning Algorithms

A variety of meta-learning algorithms exist, mostly differing in how they fine-tune on support data during the inner loop. Some meta-learning approaches, such as MAML, update all network parameters using gradient descent during fine-tuning (Finn et al., 2017). Because differentiating through the inner loop is memory and computationally intensive, the fine-tuning process consists of only a few (sometimes just 1) SGD steps.

Algorithm 1 The meta-learning framework

Require: Base model, F_θ, fine-tuning algorithm, A, learning rate, γ, and distribution over tasks, $p(\mathcal{T})$.

Initialize θ, the weights of F;

while not done **do**

    Sample batch of tasks, $\{\mathcal{T}_i\}_{i=1}^n$, where $\mathcal{T}_i \sim p(\mathcal{T})$ and $\mathcal{T}_i = (\mathcal{T}_i^s, \mathcal{T}_i^q)$.

    for $i = 1, \ldots, n$ **do**

        Fine-tune model on $\mathcal{T}_i$ (inner loop). New network parameters are written $\theta_i = A(\theta, \mathcal{T}_i^s)$.

        Compute gradient $g_i = \nabla_\theta \mathcal{L}(F_{\theta_i}, \mathcal{T}_i^q)$

    end for

    Update base model parameters (outer loop):

    $\theta \leftarrow \theta - \frac{\gamma}{n} \sum_i g_i$

end while

Reptile, which functions as a zero'th-order approximation to MAML, avoids unrolling the inner loop and differentiating through the SGD steps. Instead, after fine-tuning on support data, Reptile moves the central parameter vector in the direction of the fine-tuned parameters during the outer loop (Nichol & Schulman, 2018). In many cases, Reptile achieves better performance than MAML without having to differentiate through the fine-tuning process.

Another class of algorithms freezes the feature extraction layers during the inner loop; only the linear classifier layer is trained during fine-tuning. Such methods include R2-D2 and MetaOptNet (Bertinetto et al., 2018; Lee et al., 2019). The advantage of this approach is that the fine-tuning problem is now a convex optimization problem. Unlike MAML, which simulates the fine-tuning process using only a few gradient updates, last-layer meta-learning methods can use differentiable optimizers to exactly minimize the fine-tuning objective and then differentiate the solution with respect to feature inputs. Moreover, differentiating through these solvers is computationally cheap compared to MAML's differentiation through SGD steps on the whole network. While MetaOptNet relies on an SVM loss, R2-D2 simplifies the process even further by using a quadratic objective with a closed-form solution. R2-D2 and MetaOptNet achieve stronger performance than MAML and are able to harness larger architectures without overfitting.

Another last-layer method, ProtoNet, classifies examples by the proximity of their features to those of class centroids - a metric learning approach - in its inner loop (Snell et al., 2017). Again, the feature extractor's parameters are frozen in the inner loop, and the extracted features are used to create class centroids which then determine the network's class boundaries. Because calculating class centroids is mathematically simple, this algorithm is able to efficiently backpropagate through this calculation to adjust the feature extractor.

Model	SVM	RR	ProtoNet	MAML
MetaOptNet-M	**62.64** ± 0.31 %	**60.50** ± 0.30 %	**51.99** ± 0.33 %	**55.77** ± 0.32 %
MetaOptNet-C	56.18 ± 0.31 %	55.09 ± 0.30 %	41.89 ± 0.32 %	46.39 ± 0.28 %
R2-D2-M	**51.80** ± 0.20 %	**55.89** ± 0.31 %	**47.89** ± 0.32 %	**53.72** ± 0.33 %
R2-D2-C	48.39 ± 0.29 %	48.29 ± 0.29 %	28.77 ± 0.24 %	44.31 ± 0.28 %

Table 1. Comparison of meta-learning and classical transfer learning models with various fine-tuning algorithms on 1-shot mini-ImageNet. "MetaOptNet-M" and "MetaOptNet-C" denote models with MetaOptNet backbone trained with MetaOptNet-SVM and classical training. Similarly, "R2-D2-M" and "R2-D2-C" denote models with R2-D2 backbone trained with ridge regression (RR) and classical training. Column headers denote the fine-tuning algorithm used for evaluation, and the radius of confidence intervals is one standard error.

In this work, "classically trained" models are trained, using cross-entropy loss and SGD, on all classes simultaneously, and the feature extractors are adapted to new tasks using the same fine-tuning procedures as the meta-learned models for fair comparison. This approach represents the industry-standard method of transfer learning using pre-trained feature extractors.

2.3. Few-Shot Datasets

Several datasets have been developed for few-shot learning. We focus our attention on two datasets: mini-ImageNet and CIFAR-FS. Mini-ImageNet is a pruned and downsized version of the ImageNet classification dataset, consisting of 60,000, 84×84 RGB color images from 100 classes (Vinyals et al., 2016). These 100 classes are split into 64, 16, and 20 classes for training, validation, and testing sets, respectively. The CIFAR-FS dataset samples images from CIFAR-100 (Bertinetto et al., 2018). CIFAR-FS is split in the same way as mini-ImageNet with 60,000 32×32 RGB color images from 100 classes divided into $64, 16$, and 20 classes for training, validation, and testing sets, respectively.

2.4. Related Work

In addition to introducing new methods for few-shot learning, recent work has increased our understanding of why some models perform better than others at few-shot tasks. One such exploration performs baseline testing and discovers that network size has a large effect on the success of meta-learning algorithms (Chen et al., 2019). Specifically, on some very large architectures, the performance of transfer learning approaches that of some meta-learning algorithms. We thus focus on architectures common in the meta-learning literature. Methods for improving transfer learning in the few-shot classification setting focus on much larger backbone networks (Chen et al., 2019; Dhillon et al., 2019).

Other work on transfer learning has found that feature extractors trained on large complex tasks can be more effectively deployed in a transfer learning setting by distilling knowledge about only important features for the transfer task (Wang et al., 2020). Yet other work finds that features generated by a pre-trained model on data from classes absent from training are entangled, but the logits of the unseen data tend to be clustered (Frosst et al., 2019). Meta-learners without supervision in the outer loop have been found to perform well when equipped with a clustering-based penalty in the meta-objective (Huang et al., 2019a). Work on standard supervised learning has alternatively studied low-dimensional structures via rank (Goldblum et al., 2019b; Sainath et al., 2013).

While improvements have been made to meta-learning algorithms and transfer learning approaches to few-shot learning, little work has been done on understanding the underlying mechanisms that cause meta-learning routines to perform better than classically trained models in data scarce settings.

3. Are Meta-Learned Features Fundamentally Better for Few-Shot Learning?

It has been said that meta-learned models "learn to learn" (Finn et al., 2017), but one might ask if they instead learn to optimize; their features could simply be well-adapted for the specific fine-tuning optimizers on which they are trained. We dispel the latter notion in this section.

In Table 1, we test the performance of meta-learned feature extractors not only with their own fine-tuning algorithm, but with a variety of fine-tuning algorithms. We find that in all cases, the meta-learned feature extractors outperform classically trained models of the same architecture. See Appendix A.1 for results from additional experiments.

This performance advantage across the board suggests that meta-learned features are qualitatively different than conventional features and fundamentally superior for few-shot learning. The remainder of this work will explore the characteristics of meta-learned models.

4. Class Clustering in Feature Space

Methods such as ProtoNet, MetaOptNet, and R2-D2 fix their feature extractor during fine-tuning. For this reason, they must learn to embed features in a way that enables few-

shot classification. For example, MetaOptNet and R2-D2 require that classes are linearly separable in feature space, but mere linear separability is not a sufficient condition for good few-shot performance. The feature representations of randomly sampled few-shot data from a given class must not vary so much as to cause classification performance to be sample-dependent. In this section, we examine clustering in feature space, and we find that meta-learned models separate features differently than classically trained networks.

4.1. Measuring Clustering in Feature Space

We begin by measuring how well different training methods cluster feature representations. To measure feature clustering (FC), we consider the intra-class to inter-class variance ratio

$$\frac{\sigma^2_{within}}{\sigma^2_{between}} = \frac{C}{N} \frac{\sum_{i,j} \|\phi_{i,j} - \mu_i\|_2^2}{\sum_i \|\mu_i - \mu\|_2^2},$$

where $\phi_{i,j}$ is a feature vector in class i, μ_i is the mean of feature vectors in class i, μ is the mean across all feature vectors, C is the number of classes, and N is the number of data points per class. Low values of this fraction correspond to collections of features such that classes are well-separated and a hyperplane formed by choosing a point from each of two classes does not vary dramatically with the choice of samples.

In Table 2, we highlight the superior class separation of meta-learning methods. We compute two quantities, R_{FC} and R_{HV}, for MetaOptNet and R2-D2 as well as classical transfer learning baselines of the same architectures. These two quantities measure the intra-class to inter-class variance ratio and invariance of separating hyperplanes to data sampling. Mathematical formulations of R_{FC} and R_{HV} can be found in Sections 4.4 and 4.5, respectively. Lower values of each measurement correspond to better class separation. On both CIFAR-FS and mini-ImageNet, the meta-learned models attain lower values, indicating that feature space clustering plays a role in the effectiveness of meta-learning.

4.2. Why is Clustering Important?

To demonstrate why linear separability is insufficient for few-shot learning, consider Figure 1. As features in a class become spread out and the classes are brought closer together, the classification boundaries formed by sampling one-shot data often misclassify large regions. In contrast, as features in a class are compacted and classes move far apart from each other, the intra-class to inter-class variance ratio drops, and dependence of the class boundary on the choice of one-shot samples becomes weaker.

This intuitive argument is formalized in the following result.

Theorem 1 *Consider two random variables, X representing class 1, and Y representing class 2. Let U be the random*

Training	Dataset	R_{FC}	R_{HV}
R2-D2-M	CIFAR-FS	**1.29**	**0.95**
R2-D2-C	CIFAR-FS	2.92	1.69
MetaOptNet-M	CIFAR-FS	**0.99**	**0.75**
MetaOptNet-C	CIFAR-FS	1.84	1.25
R2-D2-M	mini-ImageNet	**2.60**	**1.57**
R2-D2-C	mini-ImageNet	3.58	1.90
MetaOptNet-M	mini-ImageNet	**1.29**	**0.95**
MetaOptNet-C	mini-ImageNet	3.13	1.75

Table 2. Comparison of class separation metrics for feature extractors trained by classical and meta-learning routines. R_{FC} and R_{HV} are measurements of feature clustering and hyperplane variation, respectively, and we formalize these measurements below. In both cases, lower values correspond to better class separation. We pair together models according to dataset and backbone architecture. "-C" and "-M" respectively denote classical training and meta-learning. See Sections 4.4 and 4.5 for more details.

variable equal to X with probability $1/2$, and Y with probability $1/2$. Assume the variance ratio bound

$$\frac{Var[X] + Var[Y]}{Var[U]} < \epsilon$$

holds for sufficiently small $\epsilon \geq 0$.

Draw random one-shot data, $x \sim X$ and $y \sim Y$, and a test point $z \sim X$. Consider the linear classifier

$$c(z) = \begin{cases} 1, & \text{if } z^T(x - y) - \frac{1}{2}\|x\|^2 + \frac{1}{2}\|y\|^2 \geq 0 \\ 2, & \text{otherwise.} \end{cases}$$

This classifier assigns the correct label to z with probability at least

$$1 - \frac{32\epsilon}{1 - \epsilon}.$$

Note that the linear classifier in the theorem is simply the maximum-margin linear classifier that separates the two training points. In plain words, Theorem 1 guarantees that one-shot learning performance is effective when the variance ratio is small, with classification becoming asymptotically perfect as the ratio approaches zero. A proof is provided in Appendix B.

4.3. Comparing Feature Representations of Meta-Learning and Classically Trained Models

We begin our investigation into the feature space of meta-learned models by visualizing features. Figure 2 contains a visual comparison of ProtoNet and a classically trained model of the same architecture on mini-ImageNet. Three classes are randomly chosen from the test set, and 100 samples are taken from each class. The samples are then passed through the feature extractor, and the resulting vectors are
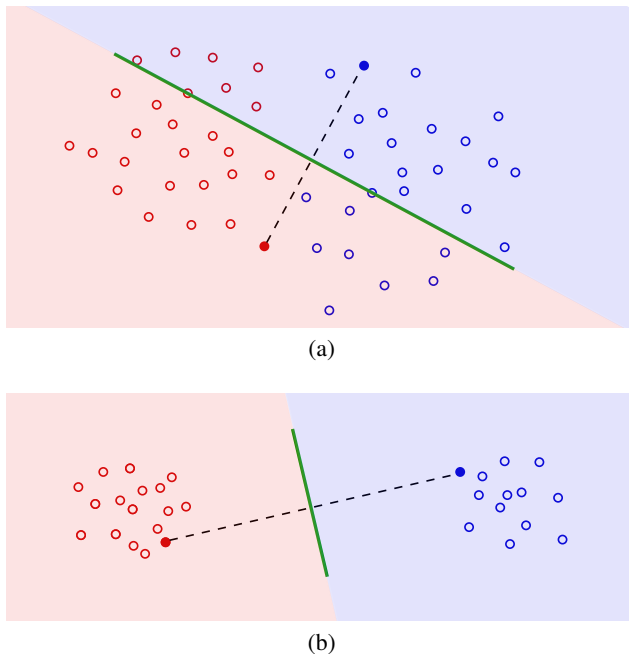
(a)

(b)

Figure 1. a) When class variation is high relative to the variation between classes, decision boundaries formed by one-shot learning are inaccurate, even though classes are linearly separable. b) As classes move farther apart relative to the class variation, one-shot learning yields better decision boundaries.

(a) Meta-Learning

(b) Classically Trained

Figure 2. Features extracted from mini-ImageNet test data by a) ProtoNet and b) classically trained models with identical architectures (4 convolutional layers). The meta-learned network produces better class separation.

plotted. Because feature space is high-dimensional, we perform a linear projection into $\mathbb{R}^2$. We project onto the first two component vectors determined by LDA. Linear discriminant analysis (LDA) projects data onto directions that minimize the intra-class to inter-class variance ratio (Mika et al., 1999), and LDA is therefore ideal for visualizing the class separation phenomenon.

In the plots, we see that relative to the size of the point clusters, the classically trained model mashes features together, while the meta-learned models draws the classes farther apart. While visually separate class features may be neither a necessary nor sufficient condition for few-shot performance, we take these plots as inspiration for our regularizer in the following section.

4.4. Feature Space Clustering Improves the Few-Shot Performance of Transfer Learning

We now further test the feature clustering hypothesis by promoting the same behavior in classically trained models. Consider a network with feature extractor f_θ and fully-connected layer g_w. Then, denoting training data in class i by $\{x_{i,j}\}$, we formulate the feature clustering regularizer by

$$R_{FC}(\theta, \{x_{i,j}\}) = \frac{C}{N} \frac{\sum_{i,j} \|f_\theta(x_{i,j}) - \mu_i\|_2^2}{\sum_i \|\mu_i - \mu\|_2^2},$$
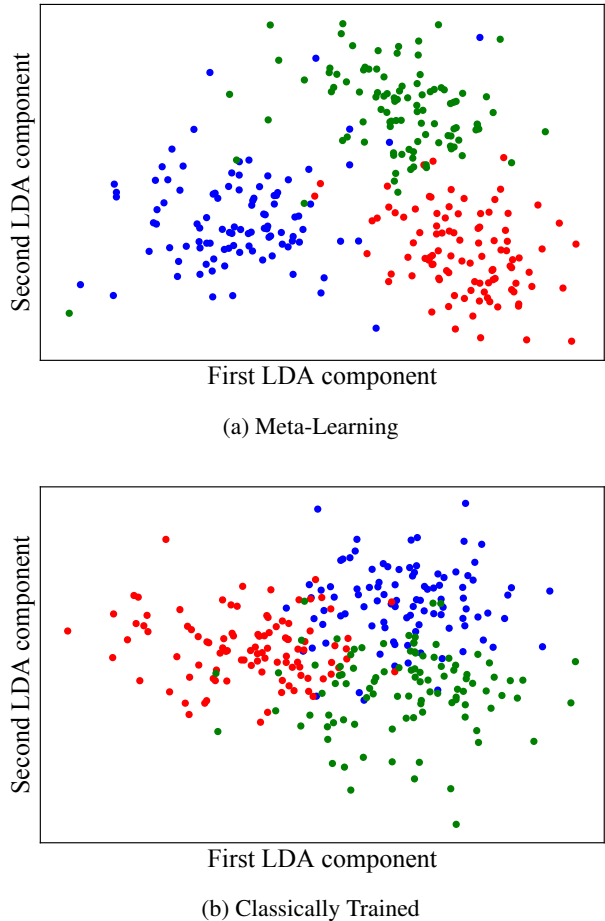
where $f_\theta(x_{i,j})$ is a feature vector corresponding to a data point in class i, μ_i is the mean of feature vectors in class i, and μ is the mean across all feature vectors. When this regularizer has value zero, classes are represented by distinct point masses in feature space, and thus the class boundary is invariant to the choice of few-shot data.

We incorporate this regularizer into a standard training routine by sampling two images per class in each mini-batch so that we can compute a within-class variance estimate. Then, the total loss function becomes the sum of cross-entropy and R_{FC}. We train the R2-D2 and MetaOptNet backbones in this fashion on the mini-ImageNet and CIFAR-FS datasets, and we test these networks on both 1-shot and 5-shot tasks. In all experiments, feature clustering improves the performance of transfer learning and sometimes even achieves higher performance than meta-learning. Furthermore, the regularizer does not appreciably slow down classical training, which, without the expense of differentiating through

an inner loop, runs as much as 13 times faster than the corresponding meta-learning routine. See Table 3 for numerical results, and see Appendix A.2 for experimental details including training times.

In addition to performance evaluations, we calculate the similarity between feature representations yielded by a feature extractor produced by meta-learning and that of one produced by the classical routine with and without R_{FC}. To this end, we use centered kernel alignment (CKA) (Kornblith et al., 2019a). Using both R2-D2 and MetaOptNet backbones on both mini-ImageNet and CIFAR-FS datasets, networks trained with R_{FC} exhibit higher similarity scores to meta-learned networks than networks trained classically but without R_{FC}. These measurements provide further evidence that feature clustering makes feature representations closer to those trained by meta-learning and thus, that meta-learners perform feature clustering. See Table 4 for more details.

4.5. Connecting Feature Clustering with Hyperplane Invariance

For further validation of the connection between feature clustering and invariance of separating hyperplanes to data sampling, we replace the feature clustering regularizer with one that penalizes variations in the maximum-margin hyperplane separating feature vectors in opposite classes. Consider data points x_1, x_2 in class A, data points y_1, y_2 in class B, and feature extractor f_θ. The difference vector $f_\theta(x_1) - f_\theta(y_1)$ determines the direction of the maximum margin hyperplane separating the two points in feature space. To penalize the variation in hyperplanes, we introduce the hyperplane variation regularizer,

$$
\begin{aligned}
&R_{HV}(f_\theta(x_1), f_\theta(x_2), f_\theta(y_1), f_\theta(y_2)) \\
&= \frac{\|(f_\theta(x_1) - f_\theta(y_1)) - (f_\theta(x_2) - f_\theta(y_2))\|_2}{\|(f_\theta(x_1) - f_\theta(y_1)\|_2 + \|f_\theta(x_2) - f_\theta(y_2)\|_2}.
\end{aligned}
$$

This function measures the distance between distance vectors $x_1 - y_1$ and $x_2 - y_2$ relative to their size. In practice, during a batch of training, we sample many pairs of classes and two samples from each class. Then, we compute R_{HV} on all class pairs and add these terms to the cross-entropy loss. We find that this regularizer performs almost as well as R_{FC} and conclusively outperforms non-regularized classical training. We include these results in Table 3. See Appendix A.2 for more details on these experiments, including training times (which, as indicated in Section 4.4, are significantly lower than those needed for meta-learning).

4.6. MAML Does Not Have the Same Feature Separation Properties

Remember that the previous measurements and experiments examined meta-learning methods which fix the feature extractor during the inner loop. MAML is a popular example of a method which does not fix the feature extractor in the inner loop. We now quantify MAML's class separation compared to transfer learning by computing our regularizer values for a pre-trained MAML model as well as a classically trained model of the same architecture. We find that, in fact, MAML exhibits even worse feature separation than a classically trained model of the same architecture. See Table 5 for numerical results. These results confirm our suspicion that the feature clustering phenomenon is specific to meta-learners which fix the feature extractor during the inner loop of training.

5. Finding Clusters of Local Minima for Task Losses in Parameter Space

Since Reptile does not fix the feature extractor during fine-tuning, it must find parameters that adapt easily to new tasks. One way Reptile might achieve this is by finding parameters that can reach a task-specific minimum by traversing a smooth, nearly linear region of the loss landscape. In this case, even a single SGD update would move parameters in a useful direction. Unlike MAML, however, Reptile does not backpropagate through optimization steps and thus lacks information about the loss surface geometry when performing parameter updates. Instead, we hypothesize that Reptile finds parameters that lie very close to good minima for many tasks and is therefore able to perform well on these tasks after very little fine-tuning.

This hypothesis is further motivated by the close relationship between Reptile and *consensus optimization* (Boyd et al., 2011). In a consensus method, a number of models are independently optimized with their own task-specific parameters, and the tasks communicate via a penalty that encourages all the individual solutions to converge around a common value. Reptile can be interpreted as approximately minimizing the consensus formulation

$$
\frac{1}{m} \sum_{p=1}^{m} \mathcal{L}_{\mathcal{T}_p}(\tilde{\theta}_p) + \frac{\gamma}{2} \|\tilde{\theta}_p - \theta\|^2,
$$

where $\mathcal{L}_{\mathcal{T}_p}(\tilde{\theta}_p)$ is the loss for task $\mathcal{T}_p$, $\{\tilde{\theta}_p\}$ are task-specific parameters, and the quadratic penalty on the right encourages the parameters to cluster around a "consensus value" θ. A stochastic optimizer for this loss would proceed by alternately selecting a random task/term index p, minimizing the loss with respect to $\tilde{\theta}_p$, and then taking a gradient step $\theta \leftarrow \theta - \eta \tilde{\theta}_p$ to minimize the loss for θ.

Reptile diverges from a traditional consensus optimizer only

Training	Backbone	mini-ImageNet		CIFAR-FS	
		1-shot	5-shot	1-shot	5-shot
R2-D2	R2-D2	**51.80** $\pm$ 0.20%	68.40 $\pm$ 0.20%	65.3 $\pm$ 0.2%	79.4 $\pm$ 0.1%
Classical	R2-D2	48.39 $\pm$ 0.29%	68.24 $\pm$ 0.26%	62.9 $\pm$ 0.3%	82.8 $\pm$ 0.3%
Classical w/ R_{FC}	R2-D2	50.39 $\pm$ 0.30%	**69.58** $\pm$ 0.26%	**65.5** $\pm$ 0.4%	**83.3** $\pm$ 0.3%
Classical w/ R_{HV}	R2-D2	50.16 $\pm$ 0.30%	69.54 $\pm$ 0.26%	64.6 $\pm$ 0.3%	83.1 $\pm$ 0.3%
MetaOptNet-SVM	MetaOptNet	**62.64** $\pm$ 0.31%	**78.63** $\pm$ 0.25%	72.0 $\pm$ 0.4%	84.2 $\pm$ 0.3%
Classical	MetaOptNet	56.18 $\pm$ 0.31%	76.72 $\pm$ 0.24%	69.5 $\pm$ 0.3%	85.7 $\pm$ 0.2%
Classical w/ R_{FC}	MetaOptNet	59.38 $\pm$ 0.31%	78.15 $\pm$ 0.24%	**72.3** $\pm$ 0.4%	**86.3** $\pm$ 0.2%
Classical w/ R_{HV}	MetaOptNet	59.37 $\pm$ 0.32%	77.05 $\pm$ 0.25%	72.0 $\pm$ 0.4%	85.9 $\pm$ 0.2%

Table 3. Comparison of methods on 1-shot and 5-shot CIFAR-FS and mini-ImageNet 5-way classification. The top accuracy for each backbone/task is in bold. Confidence intervals have radius equal to one standard error. Few-shot fine-tuning is performed with SVM except for R2-D2, for which we report numbers from the original paper.

Backbone	Dataset	C	R_{FC}	R_{HV}
R2-D2	CIFAR-FS	0.71	**0.77**	0.73
MetaOptNet	CIFAR-FS	0.77	**0.89**	0.87
R2-D2	mini-ImageNet	0.69	**0.72**	0.70
MetaOptNet	mini-ImageNet	0.70	**0.82**	0.79

Table 4. Similarity (CKA) representations trained via meta-learning and via transefer learning with/without the two proposed regularizers for various backbones and both CIFAR-FS and mini-ImageNet datasets. "C" denotes the classical transfer learning without regularizers. The highest score for each dataset/backbone combination is in bold.

Model	R_{FC}	R_{HV}
MAML-1	3.9406	1.9434
MAML-5	3.7044	1.8901
MAML-C	**3.3487**	**1.8113**

Table 5. Comparison of regularizer values for 1-shot and 5-shot MAML models (MAML-1 and MAML-5) as well as MAML-C, a classically trained model of the same architecture on mini-ImageNet training data. The lowest value of each regularizer is in bold.

in that it does not explicitly consider the quadratic penalty term when minimizing for $\tilde{\theta}_p$. However, it implicitly considers this penalty by initializing the optimizer for the task-specific loss using the current value of the consensus variables θ, which encourages the task-specific parameters to stay near the consensus parameters. In the next section, we replace the standard Reptile algorithm with one that explicitly minimizes a consensus formulation.

5.1. Consensus Optimization Improves Reptile

To validate the weight-space clustering hypothesis, we modify Reptile to explicitly enforce parameter clustering around a consensus value. We find that directly optimizing the consensus formulation leads to improved performance. To

this end, during each inner loop update step in Reptile, we penalize the squared ℓ_2 distance from the parameters for the current task to the average of the parameters across all tasks in the current batch. Namely, we let:

$$R_i\big(\{\tilde{\theta}_p\}_{p=1}^m\big) = d\big(\tilde{\theta}_i, \frac{1}{m}\sum_{p=1}^m \tilde{\theta}_p\big)^2,$$

where $\tilde{\theta}_p$ are the network parameters on task p and d is the filter normalized ℓ_2 distance (see Note 1). Note that as parameters shrink towards the origin, the distances between minima shrink as well. Thus, we employ filter normalization to ensure that our calculation is invariant to scaling (Li et al., 2018). See below for a description of filter normalization. This regularizer guides optimization to a location where many task-specific minima lie in close proximity. A detailed description is given in Algorithm 2, which is equivalent to the original Reptile when $\alpha = 0$. We call this method "Weight-Clustering."

Note 1 *Consider that a perturbation to the parameters of a network is more impactful when the network has small parameters. While previous work has used layer normalization or even more coarse normalization schemes, the authors of Li et al. (2018) note that since the output of networks with batch normalization is invariant to filter scaling as long as the batch statistics are updated accordingly, we can normalize every filter of such a network independently. The latter work suggests that this scheme, "filter normalization", correlates better with properties of the optimization landscape. Thus, we measure distance in our regularizer using filter normalization, and we find that this technique prevents parameters from shrinking towards the origin.*

We compare the performance of our regularized Reptile algorithm to that of the original Reptile method as well as first-order MAML (FOMAML) and a classically trained model of the same architecture. We test these methods on a sample of 100,000 5-way 1-shot and 5-shot mini-ImageNet tasks

Algorithm 2 Reptile with Weight-Clustering Regularization

> **Require:** Initial parameter vector, θ, outer learning rate, γ, inner learning rate, η, regularization coefficient, α, and distribution over tasks, $p(\mathcal{T})$.
> **for** *meta-step* $= 1, \ldots, n$ **do**
>   Sample batch of tasks, $\{\mathcal{T}_i\}_{i=1}^{m}$ from $p(\mathcal{T})$
>   Initialize parameter vectors $\tilde{\theta}_i^0 = \theta$ for each task
>   **for** $j = 1, \ldots, k$ **do**
>   **for** $i = 1, \ldots, m$ **do**
>   Calculate $\mathcal{L} = \mathcal{L}_{\mathcal{T}_i}^j + \alpha R_i\left(\{\tilde{\theta}_p^{j-1}\}_{p=1}^{m}\right)$
>   Update $\tilde{\theta}_i^j = \tilde{\theta}_i^{j-1} - \eta \nabla_{\tilde{\theta}_i} \mathcal{L}$
>   **end for**
>   **end for**
>   Compute difference vectors $\{g_i = \tilde{\theta}_i^k - \tilde{\theta}_i^0\}_{i=1}^{m}$
>   Update $\theta \leftarrow \theta - \frac{\gamma}{m} \sum_i g_i$
> **end for**

and find that in both cases, Reptile with Weight-Clustering achieves higher performance than the original algorithm and significantly better performance than FOMAML and the classically trained models. These results are summarized in Table 6.

Framework	1-shot	5-shot
Classical	$28.72 \pm 0.16\%$	$45.25 \pm 0.21\%$
FOMAML	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
Reptile	$49.97 \pm 0.32\%$	$65.99 \pm 0.58\%$
W-Clustering	$\mathbf{51.94 \pm 0.23\%}$	$\mathbf{68.02 \pm 0.22\%}$

Table 6. Comparison of methods on 1-shot and 5-shot mini-ImageNet 5-way classification. The top accuracy for each task is in bold. Confidence intervals have width equal to one standard error. W-Clustering denotes the Weight-Clustering regularizer.

We note that the best-performing result was attained when the product of the constant term collected from the gradient of the regularizer R_i and the regularization coefficient α was 5.0×10^{-5}, but a range of values up to ten times larger and smaller also produced improvements over the original algorithm. Experimental details, as well as results for other values of this coefficient, can be found in Appendix A.3.

In addition to these performance gains, we found that the parameters of networks trained using our regularized version of Reptile do not travel as far during fine-tuning at inference as those trained using vanilla Reptile. Figure 3 depicts histograms of filter normalized distance traveled by both networks fine-tuning on samples of 1,000 1-shot and 5-shot mini-ImageNet tasks. From these, we conclude that our regularizer does indeed move model parameters toward a consensus which is near good minima for many tasks. Interestingly, we applied these same measurements to networks trained using MetaOptNet and R2-D2, and we found

that these feature extractors lie in wide and flat minimizers across many task losses. Thus, when the whole network is fine-tuned, the parameters move a lot without substantially decreasing loss. Previous work has associated flat minimizers with good generalization (Huang et al., 2019b).
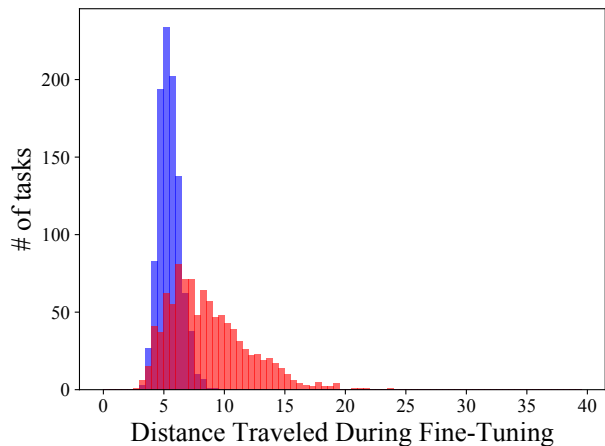
(a)
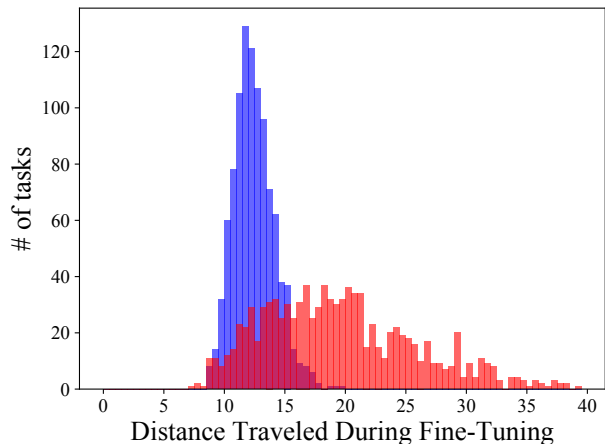
(b)

Figure 3. Histogram of filter normalized distance traveled during fine-tuning on a) 1-shot and b) 5-shot mini-ImageNet tasks by models trained using vanilla Reptile (red) and weight-clustered Reptile (blue).

6. Discussion

In this work, we shed light on two key differences between meta-learned networks and their classically trained counterparts. We find evidence that meta-learning algorithms minimize the variation between feature vectors within a class relative to the variation between classes. Moreover, we design two regularizers for transfer learning inspired by this principal, and our regularizers consistently improve few-shot performance. The success of our method helps to

confirm the hypothesis that minimizing within-class feature variation is critical for few-shot performance.

We further notice that Reptile resembles a consensus optimization algorithm, and we enhance the method by designing yet another regularizer, which we apply to Reptile, in order to find clusters of local minima in the loss landscapes of tasks. We find in our experiments that this regularizer improves both one-shot and five-shot performance of Reptile on mini-ImageNet.

A PyTorch implementation of the feature clustering and hyperplane variation regularizers can be found at: https://github.com/goldblum/FeatureClustering

Acknowledgements

References

Altae-Tran, H., Ramsundar, B., Pappu, A. S., and Pande, V. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.

Bertinetto, L., Henriques, J. F., Torr, P. H., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Frosst, N., Papernot, N., and Hinton, G. Analyzing and improving representations with the soft nearest neighbor loss. *arXiv preprint arXiv:1902.01889*, 2019.

Goldblum, M., Fowl, L., and Goldstein, T. Robust few-shot learning with adversarially queried meta-learners. *arXiv preprint arXiv:1910.00982*, 2019a.

Goldblum, M., Geiping, J., Schwarzschild, A., Moeller, M., and Goldstein, T. Truth or backpropaganda? an empirical investigation of deep learning theory. In *International Conference on Learning Representations*, 2019b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huang, G., Larochelle, H., and Lacoste-Julien, S. Centroid networks for few-shot clustering and unsupervised few-shot classification. *CoRR*, abs/1902.08605, 2019a. URL http://arxiv.org/abs/1902.08605.

Huang, W. R., Emam, Z., Goldblum, M., Fowl, L., Terry, J. K., Huang, F., and Goldstein, T. Understanding generalization through visualizations. *arXiv preprint arXiv:1906.03291*, 2019b.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019a.

Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2661–2671, 2019b.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pp. 6389–6399, 2018.

Mika, S., Ratsch, G., Weston, J., Scholkopf, B., and Mullers, K.-R. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pp. 41–48. Ieee, 1999.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Nichol, A. and Schulman, J. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2:2, 2018.

Oreshkin, B., López, P. R., and Lacoste, A. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 721–731, 2018.

Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E., and Ramabhadran, B. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6655–6659. IEEE, 2013.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Song, L., Liu, J., and Qin, Y. Fast and generalized adaptation for few-shot learning. *arXiv preprint arXiv:1911.10807*, 2019.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Wang, K., Gao, X., Zhao, Y., Li, X., Dou, D., and Xu, C.-Z. Pay attention to features, transfer learn faster CNNs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxyCeHtPB.