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

Harichandana Vejendla
(50478049)

University at Buffalo
School of Engineering and Applied Sciences

Definitions

- **Meta-Learning**: Meta-learning describes machine learning algorithms that acquire knowledge and understanding from the outcome of other machine learning algorithms. They learn how to best combine the predictions from other machine-learning algorithms.
- **Few-shot Learning**: Few-Shot Learning is a Machine Learning framework that enables a pre-trained model to generalize over new categories of data using only a few labeled samples per class.
- **Feature Extraction**: Feature extraction is a process of dimensionality reduction that involves transforming raw data into numerical features that can be processed.
- **Feature clustering**: Feature clustering aggregates point features into groups whose members are similar to each other and not similar to members of other groups.
- **Feature Representation**: Representation Learning or feature learning is the subdiscipline of the machine learning space that deals with extracting features or understanding the representation of a dataset.

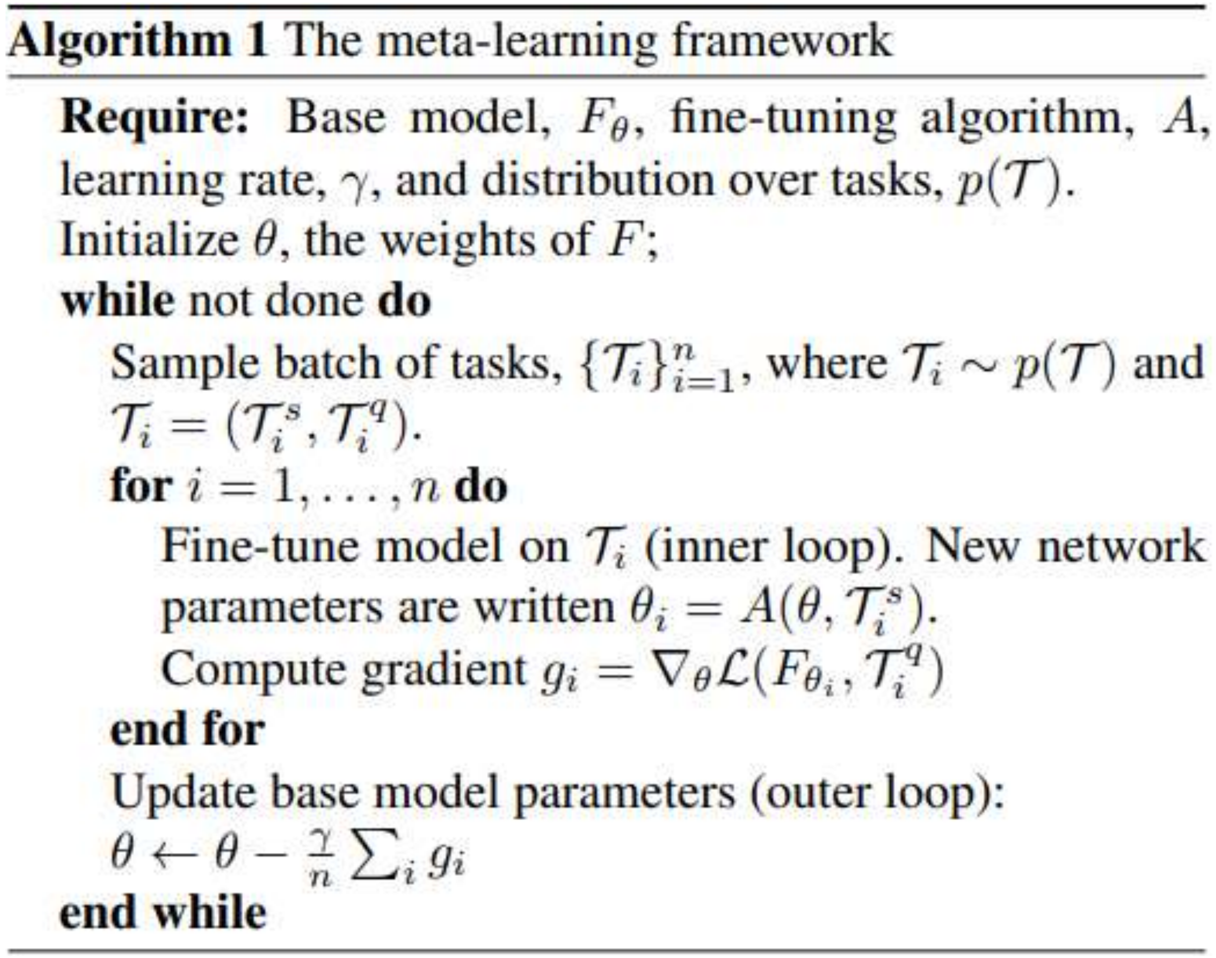
Introduction

- **Transfer Learning**: Pre-training a model on large auxiliary datasets and then fine-tuning the resulting models on the target task. This is used for few-shot learning since only a few data samples are available in the target domain.
- Transfer learning from classically trained models yields poor performance for few-shot learning. Recently, few-shot learning has been rapidly improved using meta-learning methods.
- This suggests that the feature representations learned by meta-learning must be fundamentally different from feature representations learned through conventional training.
- This paper understands the differences between features learned by meta-learning and classical training.
- Based on this, the paper proposes simple regularizers that boost few-shot performance appreciably.

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.
- Meta-learning algorithms find parameters that can be fine-tuned in a few optimization steps and on a few data points in order to achieve good generalization.
- The task is characterized as n-way, k-shot if the meta-learning algorithm must adapt to classify data from Ti after seeing k examples from each of the n classes in Ti.

Algorithm



Algorithm 1 The meta-learning framework

```
Require: Base model, $F_\theta$, fine-tuning algorithm, $A$, learning rate, $\gamma$, and distribution over tasks, $p(\mathcal{T})$.
Initialize $\theta$, 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 $i = 1, \dots, n$ 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
```

Algorithm Description

- 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, T_i^s, and the query data, T_i^q.
- In the inner loop, the model being trained is fine-tuned using the support data.
- Fine-tuning produces new parameters θ_i , that are a function of the original parameters and support data.
- We evaluate the loss on the query data and compute the gradients w.r.t the original parameters θ. We need to unroll the fine-tuning steps and backpropagate through them to compute the gradients.
- 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. Base model parameters are updated using the gradients.

Meta-Learning Algorithms

A variety of meta-learning algorithms exist, mostly differing in how they are fine-tuned using the support data during the inner loop:

- **MAML**: Updates all network parameters using gradient descent during fine-tuning.

- **R2-D2 and MetaOptNet**: Last-layer meta-learning methods (only train the last layer). They freeze the feature extraction layers (feature extractor’s parameters are frozen) during the inner loop. Only the linear classifier layer is trained during fine-tuning.

- **ProtoNet**: Last-layer meta-learning method. It classifies examples by the proximity of their features to those of class centroids. The extracted features are used to create class centroids which then determine the network’s class boundaries.

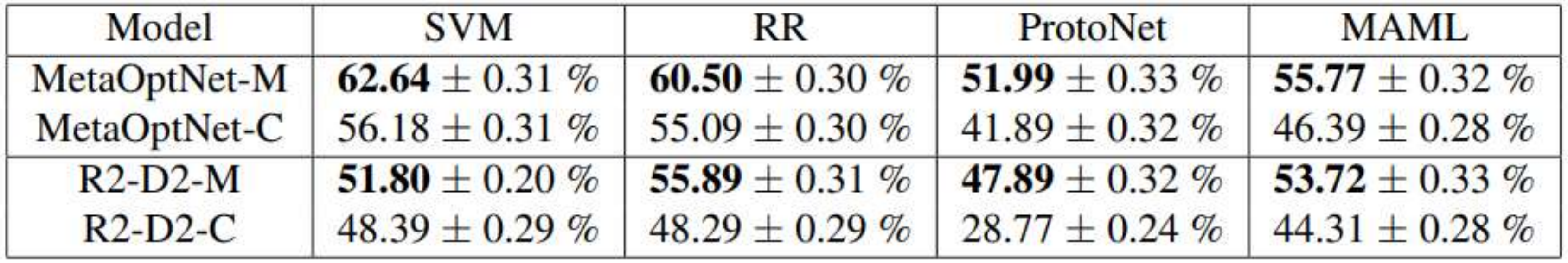
Few-Shot Datasets

- **Mini-ImageNet**: It is a pruned and downsized version of the ImageNet classification dataset, consisting of 60,000, 84×84 RGB color images from 100. These 100 classes are split into 64, 16, and 20 classes for training, validation, and testing sets, respectively.

- **CIFAR-FS dataset**: samples images from CIFAR-100. 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.

Comparison between Meta-Learning and Classical Training Models

- Dataset Used: 1-shot mini-ImageNet
- Classically trained models are trained using cross-entropy loss and SGD.
- Common fine-tuning procedures are used for both meta-learned and classically-trained models for a fair comparison
- Results show that meta-learning models perform better than classical training models on few-shot classification.
- This performance advantage across the board suggests that meta-learned features are qualitatively different from conventional features and fundamentally superior for few-shot learning.



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.

Class Clustering in Feature Space

Measuring Clustering in Feature Space:
To measure feature clustering (FC), we consider the intra-class to inter-class variance ratio:

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

$\phi_{i,j}$	- feature vector corresponding to data point in class i in training data
μ_i	- mean of feature vectors in class i
μ	- mean across all feature vectors
C	- number of classes
N	- number of data points per class

Where, $f_\theta\,(x_{i,j}) = \phi_{i,j}$

f_θ	- feature extractor
$x_{i,j}$	- training data in class i

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.

Why Clustering is important?

- 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.
- 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 the dependence of the class boundary on the choice of one-shot samples becomes weaker.

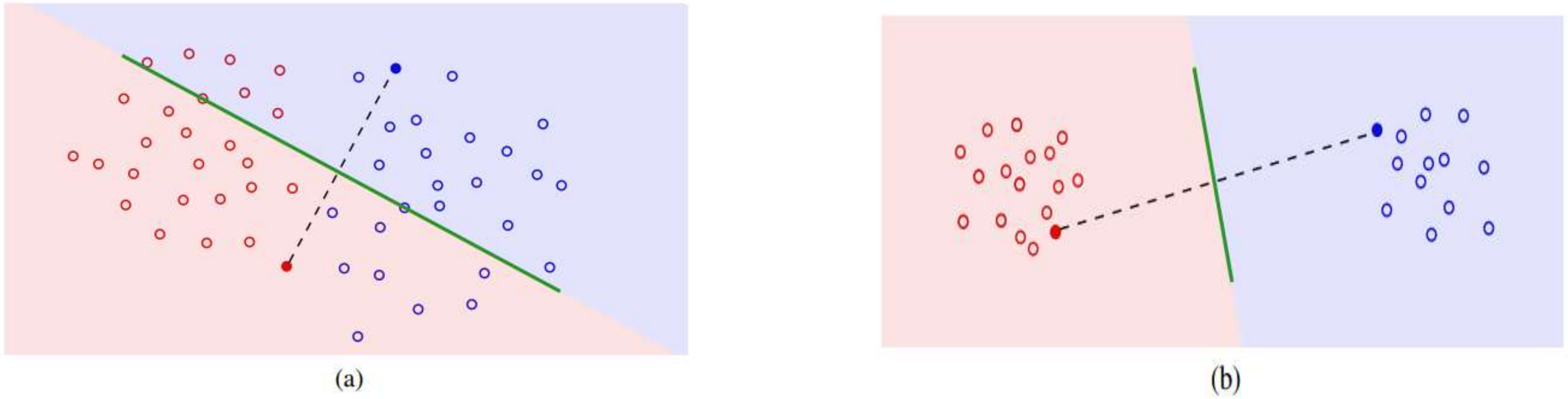


Figure 1. a) When class variation is high relative to the variation between classes, decision boundaries formed by one-shot learning are inaccurate, even though classes are linearly separable. b) As classes move farther apart relative to the class variation, one-shot learning yields better decision boundaries.

Comparing Feature Representations of Meta-Learning and Classically Trained Models

- 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 plotted.
- Because feature space is high-dimensional, we perform a linear projection 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.
- The classically trained model mashes features together, while the meta-learned models draw the classes farther apart.

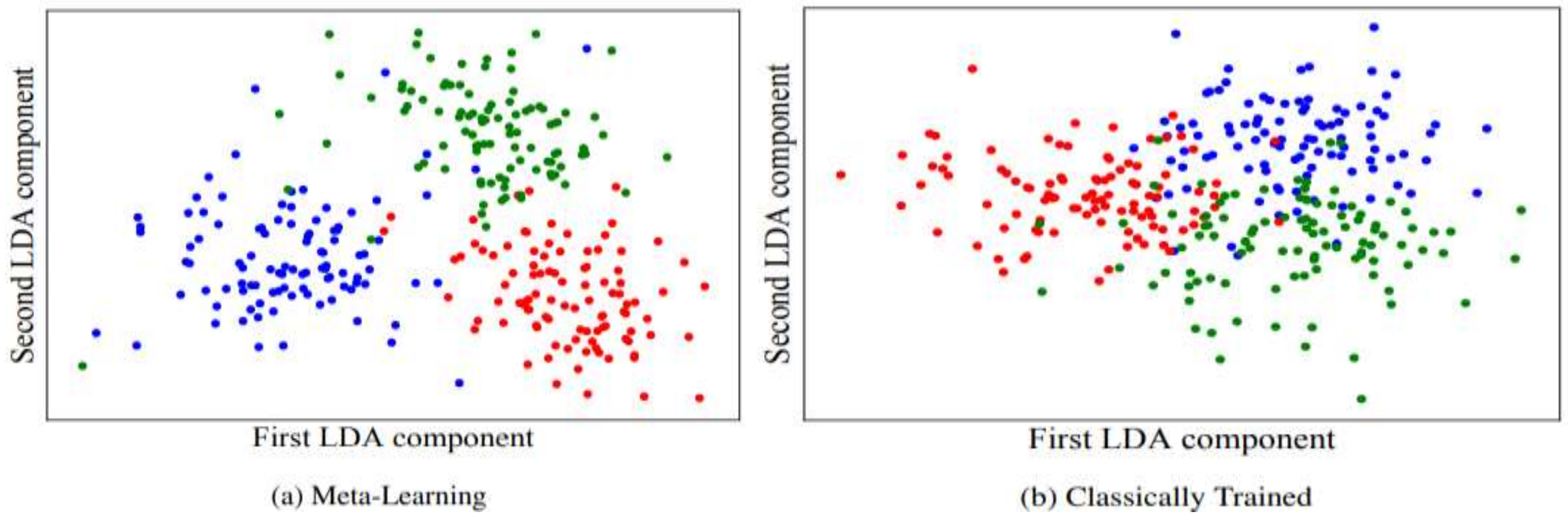


Figure 2. Features extracted from mini-ImageNet test data by a) ProtoNet and b) classically trained models with identical architectures (4 convolutional layers). The meta-learned network produces better class separation.

University at Buffalo
School of Engineering and Applied Sciences

Hyperplane Invariance

This regularizer with one that penalizes variations in the maximum-margin hyperplane separating feature vectors in opposite classes
Hyperplane Variation Regularizer:

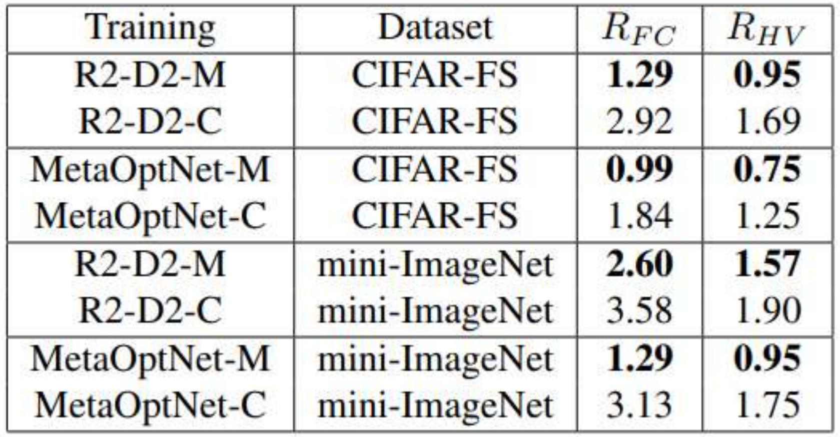
$$R_{HV}(f_\theta(x_1), f_\theta(x_2), f_\theta(y_1), f_\theta(y_2)) = \frac{\|(f_\theta(x_1) - f_\theta(y_1)) - (f_\theta(x_2) - f_\theta(y_2))\|_2}{\|(f_\theta(x_1) - f_\theta(y_1)\|_2 + \|f_\theta(x_2) - f_\theta(y_2)\|_2}.$$

Dat points in class A : x1, x2
Data points in class B: y1, y2
f_θ - feature extractor
f_θ (x1) - f_θ (y1) : determines the direction of the maximum margin hyperplane separating the two points in the feature space

- This function measures the distance between distance vectors x1 − y1 and x2 − y2 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 RHV on all class pairs and add these terms to the cross-entropy loss.
- We find that this regularizer performs almost as well as Feature Clustering Regularizer and conclusively outperforms non-regularized classical training.

University at Buffalo
School of Engineering and Applied Sciences

Experiments

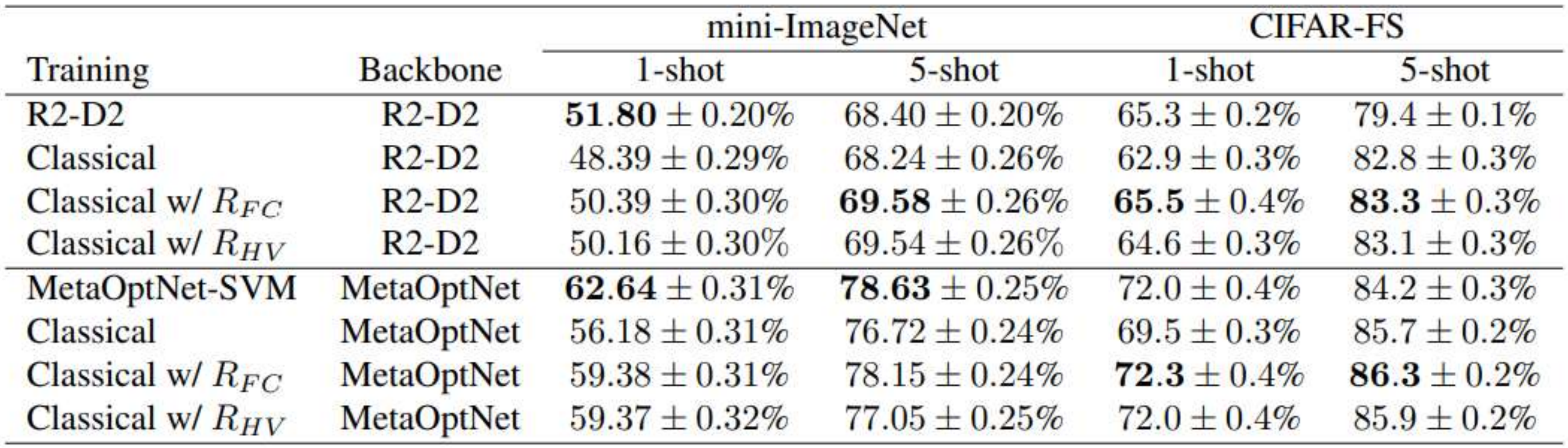


Training	Dataset	R_{FC}	R_{HV}
R2-D2-M	CIFAR-FS	**1.29**	**0.95**
R2-D2-C	CIFAR-FS	2.92	1.69
MetaOptNet-M	CIFAR-FS	**0.99**	**0.75**
MetaOptNet-C	CIFAR-FS	1.84	1.25
R2-D2-M	mini-ImageNet	**2.60**	**1.57**
R2-D2-C	mini-ImageNet	3.58	1.90
MetaOptNet-M	mini-ImageNet	**1.29**	**0.95**
MetaOptNet-C	mini-ImageNet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

- Feature clustering and Hyperplane variation values are computed.
- These two quantities measure the intra-class to inter-class variance ratio and invariance of separating hyperplanes.
- 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.

Experiments



Training	Backbone	mini-ImageNet 1-shot	mini-ImageNet 5-shot	CIFAR-FS 1-shot	CIFAR-FS 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.

- We incorporate these regularizers into a standard training routine of the classical training model.
- In all experiments, feature clustering improves the performance of transfer learning and sometimes even achieves higher performance than meta-learning

Weight Clustering: Finding Clusters of Local Minima for Task Losses in Parameter Space

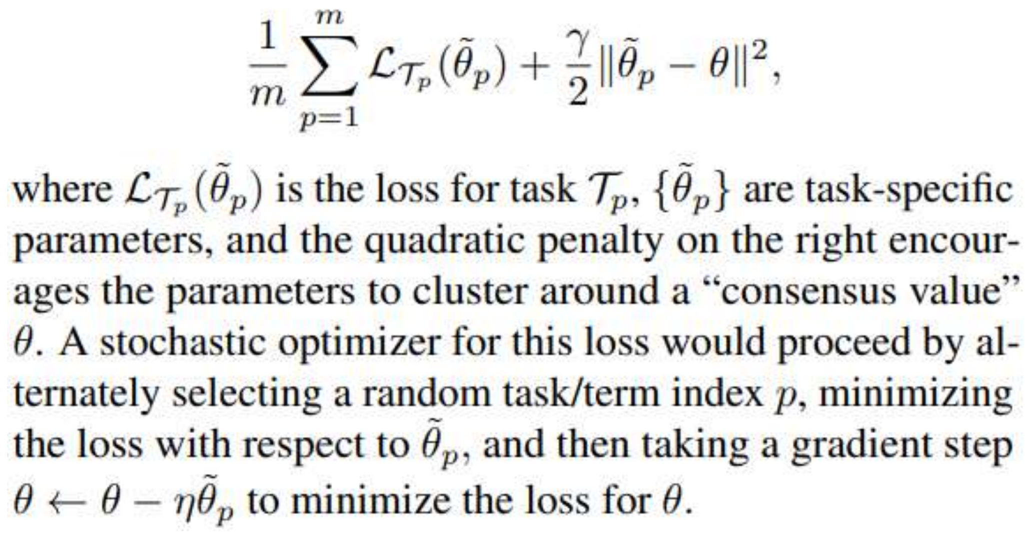
- Since Reptile does not fix the feature extractor during fine-tuning, it must find parameters that adapt easily to new tasks.

- 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.

- 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.

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 can be interpreted as approximately minimizing the consensus formulation
- Reptile diverges from a traditional consensus optimizer only in that it does not explicitly consider the quadratic penalty term when minimizing for ˜θp.

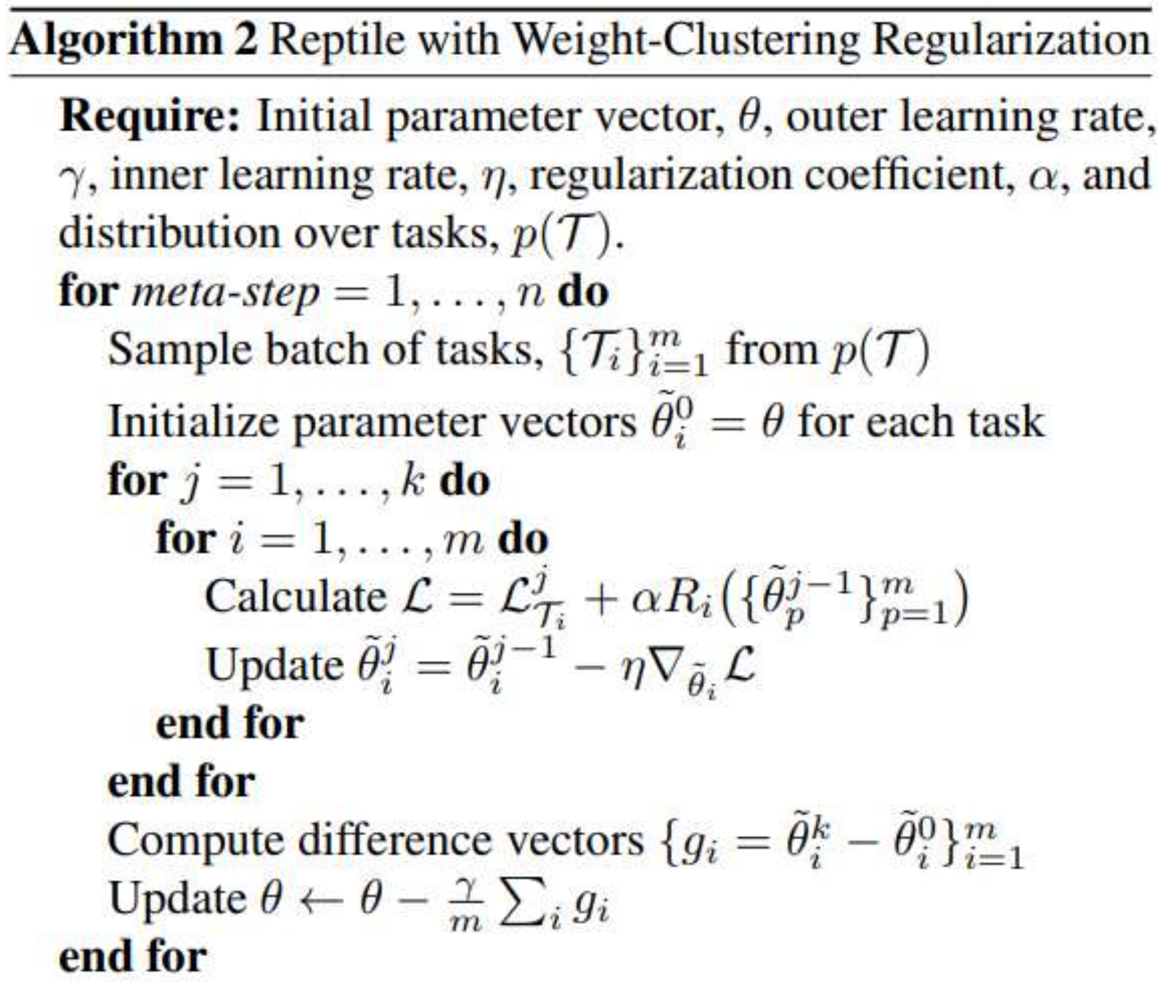
Consensus Optimization Improves Reptile

- We modify Reptile to explicitly enforce parameter clustering around a consensus value.
- We find that directly optimizing the consensus formulation leads to improved performance.
- during each inner loop update step in Reptile, we penalize the squared distance from the parameters for the current task to the average of the parameters across all tasks in the current batch.
- This is equivalent to the original Reptile when α = 0. We call this method “Weight-Clustering.

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

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

Reptile with Weight Clustering Regularizer



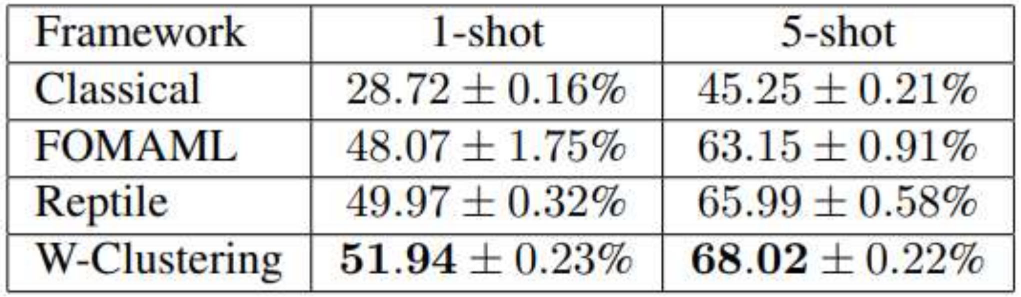
Algorithm 2 Reptile with Weight-Clustering Regularization

```
Require: Initial parameter vector, $\theta$, outer learning rate, $\gamma$, inner learning rate, $\eta$, regularization coefficient, $\alpha$, 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 $\mathcal{L} = \mathcal{L}_{\mathcal{T}_i}^j + \alpha R_i(\{\tilde{\theta}_p^{j-1}\}_{p=1}^m)$
            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
```

n - number of meta-training steps
k - number of iterations or steps to perform within each meta-training step

Results of weight clustering

- 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
- Reptile with Weight-Clustering achieves higher performance.

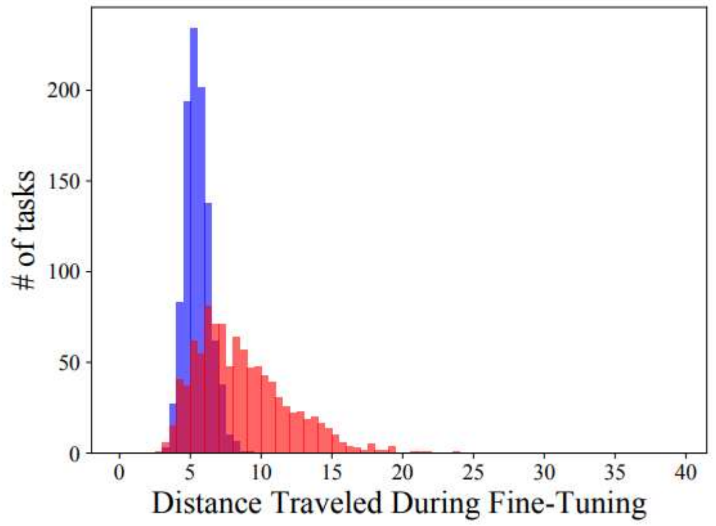


Framework	1-shot	5-shot
Classical	$28.72 \pm 0.16\%$	$45.25 \pm 0.21\%$
FOMAML	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
Reptile	$49.97 \pm 0.32\%$	$65.99 \pm 0.58\%$
W-Clustering	$\mathbf{51.94} \pm 0.23\%$	$\mathbf{68.02} \pm 0.22\%$

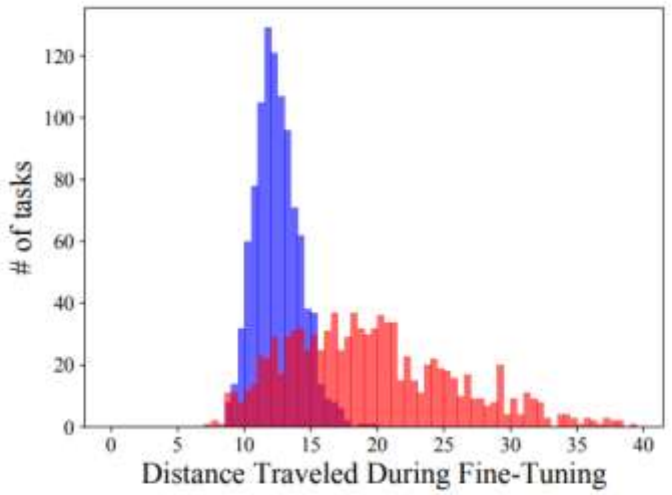
Table 6. Comparison of methods on 1-shot and 5-shot mini-ImageNet 5-way classification. The top accuracy for each task is in bold. Confidence intervals have width equal to one standard error. W-Clustering denotes the Weight-Clustering regularizer.

Results of weight clustering

- 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
- From these, we conclude that our regularizer does indeed move model parameters toward a consensus which is near good minima for many tasks



(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).

Conclusion

- 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 principle, and our regularizers consistently improve few-shot performance.
- The success of this method helps to confirm the hypothesis that minimizing within-class feature variation is critical for few-shot performance.

References

Meta-Learning:
- https://machinelearningmastery.com/meta-learning-in-machine-learning/
- https://www.analyticsvidhya.com/blog/2022/09/meta-learning-structure-advantages-examples/#:~:text=Meta%2Dlearning%20describes%20machine%20learning,combine%20predictions%20from%20ensemble%20members.

Few-shot Learning:
- https://blog.paperspace.com/few-shot-learning/

Feature Extraction:
- https://www.mathworks.com/discovery/feature-extraction.html#:~:text=Feature%20extraction%20refers%20to%20the,directly%20to%20the%20raw%20data

Feature Clustering
- https://www.ra.ethz.ch/CDstore/www6/Technical/Paper118/node8.html

Representation Learning
- https://pub.towardsai.net/a-gentle-introduction-to-representation-learning-e79e5ed6964

Thank You