

---

# On sensitivity of meta-learning to support data

---

Mayank Agarwal<sup>1</sup>  
mayank.agarwal@ibm.com

Mikhail Yurochkin<sup>1,2</sup>  
mikhail.yurochkin@ibm.com

Yuekai Sun<sup>3</sup>  
yuekai@umich.edu

IBM Research,<sup>1</sup> MIT-IBM Watson AI Lab,<sup>2</sup> University of Michigan<sup>3</sup>.

## Abstract

Meta-learning algorithms are widely used for few-shot learning. For example, image recognition systems that readily adapt to unseen classes after seeing only a few labeled examples. Despite their success, we show that modern meta-learning algorithms are extremely sensitive to the data used for adaptation, i.e. support data. In particular, we demonstrate the existence of (unaltered, in-distribution, natural) images that, when used for adaptation, yield accuracy as low as 4% or as high as 95% on standard few-shot image classification benchmarks. We explain our empirical findings in terms of class margins, which in turn suggests that robust and safe meta-learning requires larger margins than supervised learning.

## 1 Introduction

Meta-learning, or learning to learn [29], is the problem of training models that can adapt to new tasks quickly, using only a handful of examples. The problem is inspired by humans' ability to learn new skills or concepts at a rapid pace (*e.g.* recognizing previously unknown objects after seeing only a single example [20]). Meta-learning has found applications in many domains, including safety-critical medical image analysis [23], autonomous driving [35], visual navigation [43] and legged robots control [40]. In this paper, we investigate the vulnerabilities of modern meta-learning algorithms in the context of few-shot image classification problem [25], where a meta-learner needs to solve a classification task on classes unseen during training using a small number (typically 1 or 5) of labeled samples from each of these classes to adapt. Specifically, we demonstrate that the performance of modern meta-learning algorithms on few-shot image recognition benchmarks varies drastically depending on the examples provided for adaptation, typically called *support data*, raising concerns about its safety in deployment.

Despite the many empirical successes of artificial intelligence, its vulnerabilities are important to explore and mitigate in our pursuit of safe AI that is suitable for critical applications. Sensitivity to small (possibly adversarial) perturbations to the inputs [13], backdoor attacks allowing malicious model manipulations [6], algorithmic biases [2] and poor generalization outside of the training domain [18] are some of the prominent examples of AI safety failures. Some of these issues have also been studied in the context of meta-learning, *e.g.* adversarial robustness [45, 11, 44] and fairness [37]. Most of the aforementioned AI-safety dangers are associated with adversarial manipulations of the train or test data, or significant distribution shifts at test time. In meta-learning, prior works demonstrated that an adversary can create visually imperceptible changes of the test inputs [11] or the support data [44, 30] causing meta-learning algorithms to fail on various few-shot image recognition benchmarks. In this work we demonstrate *adversary-free* and *in-distribution* failures specific to meta-learning. Sohn et al. [39] studied a similar problem in the context of semi-supervised learning.

A distinct feature of meta-learning is the adaptation stage where the model is updated using a scarce amount of labeled examples from a new task. In practice, these examples need to be selected and presented to a human for labeling. Then a meta-learner adapts and can be used to classify the

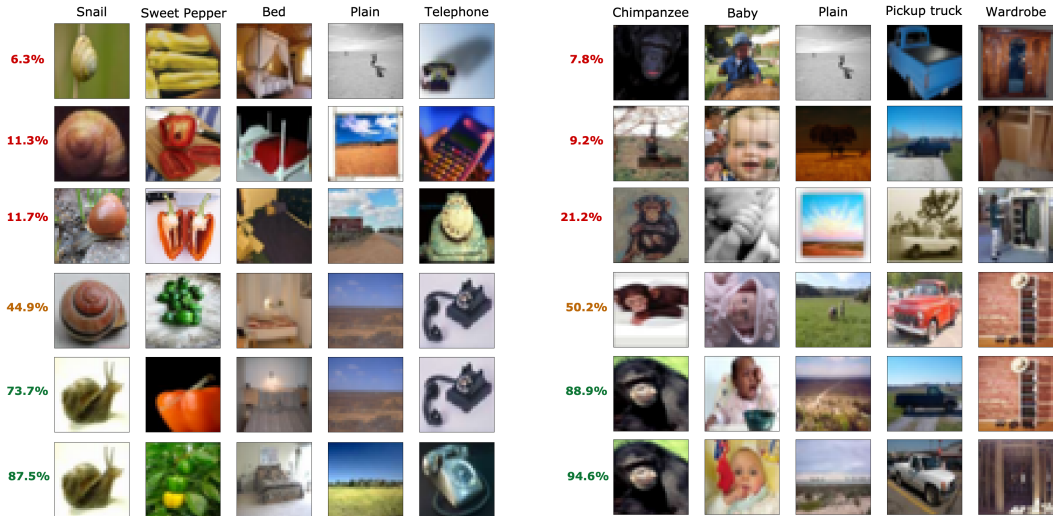


Figure 1: Examples of CIFAR-FS support images (unaltered, i.e. not modified adversarially or in any other way) from two test tasks that yield vastly different 1-shot post-adaptation performances of a popular meta-learning algorithm (MetaOptNet-SVM). Accuracy ranges from 6.3% to 94.6% suggesting extreme sensitivity of the meta-learner to the support data. Images mostly appear representative of the corresponding classes and would be hard to recognize as potentially problematic without significant expertise of the training dataset from the human labeling the support data.

remaining images without a human. Currently, meta-learning algorithms choose support examples for labeling at random. Dhillon et al. [8] noted that the standard deviation of accuracies computed with random support examples may be large. We demonstrate that if the support data is not carefully curated, the performance of the meta-learner can be unreliable. In Figure 1 we present examples of unaltered support images (i.e. they are not modified adversarially or in any other way) representative of the corresponding task (in-distribution) that lead to vastly different performance of the meta-learner after adaptation. Support examples causing poor performance are not prevalent, however, they are also not data artifacts as there are multiple of them. The existence of such examples might be concerning when deploying meta-learning in practice, even when adversarial intervention is out of scope.

Our main contributions are as follows:

1. We present a simple algorithm for finding the best and the worst support examples and empirically demonstrate the sensitivity of popular meta-learning algorithms to support data.
2. We demonstrate that a popular strategy for achieving robustness [22] adapted to our setting *fails* to solve the support data sensitivity problem.
3. We explain the existence of the worst case examples from the margin perspective suggesting that robust meta-learning requires class margins more stringent than classical supervised learning.

## 2 Meta-learning approaches

Meta-learning approaches are typically categorized into model-based [34, 26, 33], metric-based [17, 41, 38] and optimization-based methods [10, 28, 14]. We refer to Hospedales et al. [15] for a recent survey of the area. In this paper we mostly focus on the optimization-based method popularized by the Model Agnostic Meta Learning (MAML) [10] bi-level formulation (metric-based prototypical networks [38] are also considered in the experiments). Let  $\{\mathcal{T}_n\}_{i=1}^n$  be a collection of  $n$  tasks  $\mathcal{T}_i = \{\mathcal{A}_i, \mathcal{D}_i\}$  each consisting of a support  $\mathcal{A}_i$  and a query (or validation)  $\mathcal{D}_i$  datasets. MAML

bi-level optimization problem is as follows [10]:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^n \ell(\theta_i, \theta; \mathcal{D}_i), \tag{2.1}$$

such that  $\theta_i = \arg \min_{\theta_i} \ell(\theta_i, \theta; \mathcal{A}_i), i = 1, \dots, n.$

The first line of equation 2.1 is typically minimized using gradient-based optimization of  $\theta$  and is called *meta-update*. The second line is the *adaptation* step and its implementation differs depending on whether the fine-tuned  $\theta_i$  are treated as parameters of the end-to-end neural network or as the “head” parameters, i.e. the last linear classification layer. We refer to Goldblum et al. [12] for an empirical study of the differences between the two adaptation perspectives.

MAML [10] and Reptile [28] are examples of algorithms that fine-tune all network parameters during the adaptation. Instead of solving the argmin of  $\theta_i$  exactly they approximate it with a small number of gradient steps. For one gradient step approximation equation 2.1 can be written as

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^n \ell(\theta - \alpha \frac{\partial \ell(\theta; \mathcal{A}_i)}{\partial \theta}; \mathcal{D}_i) \tag{2.2}$$

for some step size  $\alpha$ . MAML directly optimizes equation 2.2 differentiating through the inner gradient step, which requires expensive Hessian computations. Reptile introduces an approximation by-passing the Hessian computations and often performs better.

Another family of meta-learning algorithms considers  $\theta$  as the neural network feature extractor parameters shared across tasks and adapts only the linear classifier parametrized with  $\theta_i$ , that takes the features extracted with  $\theta$  as inputs. The advantage of this perspective is that the adaptation minimization problem is convex and, for many linear classifiers, can be solved fast and exactly. Let  $a(\theta, \mathcal{A})$  denote a procedure that takes data in  $\mathcal{A}$ , passes it through a feature extractor parametrized with  $\theta$ , and returns  $\theta_i$ , i.e. optimal parameters of a linear classifier using the obtained features to predict the corresponding labels. Then equation 2.1 can be written as

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^n \ell(\theta, a(\theta, \mathcal{A}_i); \mathcal{D}_i). \tag{2.3}$$

This approach requires  $a(\theta, \mathcal{A})$  to be differentiable. R2D2 [4] casts classification as a multi-target ridge-regression problem and utilizes the corresponding closed-form solution as  $a(\theta, \mathcal{A})$ . Lee et al. [21] propose MetaOptNet, where  $a(\theta, \mathcal{A})$  is a differentiable quadratic programming solver [1], and implement it with linear support vector machines (SVM) and ridge-regression. Prototypical networks [38] is a metric-based approach that can also be viewed from the perspective of equation 2.3. Here  $a(\theta, \mathcal{A})$  outputs class centroids in the neural-feature space and the predictions are based on the closest class centroids. Last-layer adaptation approaches, especially MetaOptNet, outperform full-network approaches on most benchmarks [21, 12].

Both adaptation perspectives have a weakness underlying our study: their test performance is based on an optimization problem with large number of parameters and as little as 5 data points (1-shot 5-way setting). To understand the problem, consider the last-layer adaptation approach: even when the feature extractor produces linearly separable representations, the dimension is large (for example, Lee et al. [21] utilize a ResNet-12 architecture with 2560-dimensional last layer features that we adopt in our experiments) making the corresponding linear classifier extremely sensitive to the support data. In Section 3 we demonstrate the problem empirically and provide theoretical insights in Section 4.

## 2.1 Meta-learning benchmarks

Before presenting our findings, we discuss the meta-learning benchmarks we consider. Meta-learning algorithms are often compared in a few-shot image recognition setting. Each task typically has five unique classes, i.e. 5-way, and one or five examples per class for adaptation, i.e. 1-shot or 5-shot. Classes that appear at test time are not seen during training. Few-shot learning datasets are typically the derivatives of the existing supervised learning benchmarks, e.g. CIFAR-100 [19] and ImageNet [7]. Few-shot problem is setup by disjoint partitioning of the available classes into train, validation and test. Tasks are obtained by sampling 5 distinct classes from the corresponding pool and assigning

a subset of examples for adaptation and a different subset for meta-update during training or accuracy evaluation for reporting the performance.

CIFAR-FS [4] is a dataset of 60000  $32 \times 32$  RGB images from CIFAR-100 partitioned into 64, 16 and 20 classes for training, validation and testing, respectively. FC-100 [31] is also a derivative of CIFAR-100 with a different partition aimed to reduce semantic overlap between 60 classes assigned for training, 20 for validation, and 20 for testing. MiniImageNet [41] is a subsampled, downsized version of ImageNet. It consists of 60000  $84 \times 84$  RGB images from 100 classes split into 64 for training, 16 for validation, and 20 for testing.

### 3 Finding adaptation vulnerabilities

In this section we study the performance range of meta-learners trained with a variety of algorithms: MAML [10], Meta-Curvature (MC) [32], Prototypical networks [38], R2D2 [4], and MetaOptNet with Ridge and SVM heads [21]. At test time a meta-learner receives support data and its performance is recorded after it adapts. Typical evaluation protocols sample support data randomly and report average performance across tasks. While this is a useful measure for comparing algorithms, safety-critical applications demand understanding of potential performance ranges. We demonstrate that there are support examples the lead to vastly different test performances. To identify the worst and the best case support examples for a given task we perform an iterative greedy search.

Let  $\mathcal{X} = \{x_m^k\}_{m \in [M]}^{k \in [K]}$  be a set of potential support examples for a given task.  $K$  is the number of classes,  $M$  is the number of examples per class (we assume same  $M$  for each class for simplicity), and  $[M] = \{0, \dots, M - 1\}$ . Let  $\mathcal{D}$  be the evaluation set for this task. In a  $J$ -shot setting, let  $\mathcal{Z} = \{z_j^k\}_{j \in [J]}^{k \in [K]}$  be a set of *indices* of the support examples ( $z_j^k \neq z_{j'}^k$  for any  $k, j \neq j'$  and  $z_j^k \in [M]$  for any  $k, j$ ). Denote  $R(\mathcal{Z}, \mathcal{X}, \mathcal{D})$  a function computing accuracy of a meta learner on the query set  $\mathcal{D}$  after adapting on  $\mathcal{A} = \{x_{z_j^k}^k\}_{j \in [J]}^{k \in [K]}$ . Finding the worst/best case accuracy amounts to finding indices  $\mathcal{Z}$  minimizing/maximizing  $R(\mathcal{Z}, \mathcal{X}, \mathcal{D})$ . We solve this greedily by updating a single index  $z_j^k$  at a time, holding the rest of  $\mathcal{Z}$  fixed, iterating over  $[K]$  and  $[J]$  multiple times. We summarize the procedure for finding the worst case accuracy in Algorithm 1 (the best case accuracy is analogous).<sup>1</sup> Due to the greedy nature, this algorithm finds a local optima, but it is sufficiently effective as we see in the following section.

---

#### Algorithm 1 Finding the worst case support examples

---

**Input:** trained meta-learner, potential support examples  $\mathcal{X}$  to search over, query data  $\mathcal{D}$ .  
**repeat**  
  Initialize indices  $\mathcal{Z} = \{z_j^k\}_{j \in [J]}^{k \in [K]}$  randomly.  
  **for**  $j \in [J], k \in [K]$  **do**  
     $z_j^k \leftarrow \arg \min_{z_j^k} R(\mathcal{Z}, \mathcal{X}, \mathcal{D}), z_j^k \in [M] \setminus \{z_{j'}^k\}_{j' \neq j}$   
  **end for**  
**until**  $R(\mathcal{Z}, \mathcal{X}, \mathcal{D})$  stops decreasing  
**Output:** support examples indices  $\mathcal{Z} = \{z_j^k\}_{j \in [J]}^{k \in [K]}$

---

**Visualizing the worst case support search** In Figure 2 we visualize a single iteration of Algorithm 1 on a 1-shot task from the CIFAR-FS dataset. This is a 5-way task with “snail”, “red pepper”, “bed”, “plain”, and “telephone” as classes. In round 1, i.e.  $k = 0$ , of iteration 1 we start with a random example per class and evaluate post-adaptation accuracy on the query data  $\mathcal{D}$  for each potential support “snail”, i.e.  $\{x_m^0\}_{m \in [M]}$ . Here the first line corresponds to “snail” indexed with  $m = 0$ , i.e.  $x_0^0$ , and post-adaptation accuracy of 73.5%. We select a “snail” support image corresponding to the worst accuracy of 53.8% and proceed to round 2, i.e.  $k = 1$ , of iteration 1. In round 2 we repeat the same procedure for the corresponding class “red pepper” (again the first line corresponds to “red pepper” indexed with  $m = 0$ , i.e.  $x_0^1$ ), holding the support “snail” image selected previously and

<sup>1</sup>This is a simple approach, however we found it faster and more efficient than a more sophisticated continuous relaxation using weights of the potential support examples with sparsity constraints.

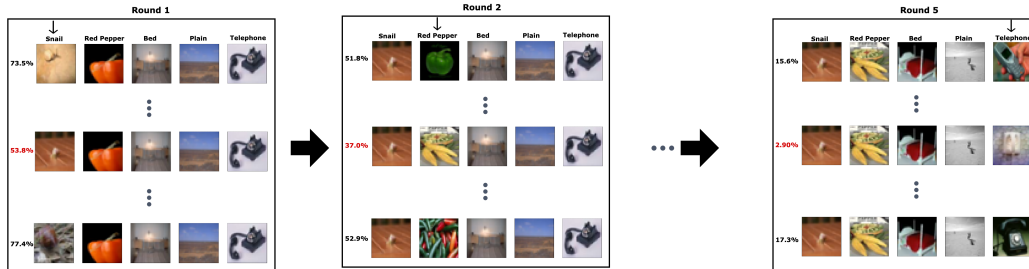


Figure 2: Visualization of Algorithm 1 on a random 1-shot task for the MetaOptNet-SVM method on the CIFAR-FS dataset. We show 1 iteration of the algorithm here, where the algorithm starts by randomly sampling a support image per class and then iterates over all data samples for all classes.

support images for other classes fixed. The algorithm finds the worst case support “red pepper” image corresponding to the post-adaptation accuracy of 37%. Then the algorithm proceeds analogously to rounds 3, 4, and 5 of iteration 1, resulting in a combination of support examples corresponding to 2.9% post-adaptation accuracy. This is already sufficiently low, however on some tasks and higher-shot settings it may be beneficial to run additional iterations. On iteration 2, the algorithm will again go through all the classes starting from the support examples found on iteration 1 instead of random ones. In our experiments we always run Algorithm 1 for 3 iterations. In Appendix B we empirically study the convergence of the algorithm justifying this choice.

Table 1: Accuracies for different meta-learning methods on the CIFAR-FS dataset

		Worst acc	Avg acc	Best acc
MAML	1-shot	$5.91 \pm 1.94\%$	$56.52 \pm 10.85\%$	$80.04 \pm 6.83\%$
	5-shot	$13.88 \pm 7.28\%$	$70.45 \pm 8.42\%$	$85.15 \pm 6.12\%$
	10-shot	$26.26 \pm 7.85\%$	$70.88 \pm 7.82\%$	$85.81 \pm 6.47\%$
MC	1-shot	$4.73 \pm 1.51\%$	$47.76 \pm 11.28\%$	$69.81 \pm 8.37\%$
	5-shot	$9.29 \pm 6.13\%$	$70.23 \pm 8.82\%$	$85.22 \pm 6.53\%$
	10-shot	$16.95 \pm 10.78\%$	$70.62 \pm 7.89\%$	$87.81 \pm 6.91\%$
ProtoNets	1-shot	$5.06 \pm 2.37\%$	$63.80 \pm 0.71\%$	$84.88 \pm 6.29\%$
	5-shot	$18.28 \pm 9.85\%$	$80.06 \pm 0.46\%$	$90.41 \pm 4.72\%$
	10-shot	$27.32 \pm 11.17\%$	$82.95 \pm 0.44\%$	$91.44 \pm 4.35\%$
R2D2	1-shot	$6.14 \pm 2.89\%$	$68.86 \pm 0.68\%$	$86.63 \pm 5.97\%$
	5-shot	$14.73 \pm 8.21\%$	$82.29 \pm 0.44\%$	$92.34 \pm 4.21\%$
	10-shot	$29.30 \pm 12.16\%$	$85.74 \pm 0.42\%$	$93.41 \pm 3.81\%$
MetaOptNet-Ridge	1-shot	$5.17 \pm 2.97\%$	$71.21 \pm 0.67\%$	$87.40 \pm 5.95\%$
	5-shot	$16.81 \pm 11.57\%$	$84.18 \pm 0.45\%$	$93.15 \pm 4.43\%$
	10-shot	$32.10 \pm 16.26\%$	$86.82 \pm 0.42\%$	$93.89 \pm 3.59\%$
MetaOptNet-SVM	1-shot	$5.27 \pm 2.82\%$	$70.79 \pm 0.69\%$	$87.65 \pm 5.76\%$
	5-shot	$14.92 \pm 8.60\%$	$83.98 \pm 0.44\%$	$93.36 \pm 4.60\%$
	10-shot	$22.24 \pm 10.13\%$	$87.11 \pm 0.40\%$	$93.56 \pm 3.98\%$

### 3.1 Performance range results

We summarize the worst, average and best accuracies of six meta-learning algorithms on three benchmark datasets (see Section 2.1 for data descriptions) in 1-shot, 5-shot, and 10-shot setting in Tables 1, 2, and 3. All meta-learners are trained using code from the authors or more modern meta-learning libraries [3] (see Appendix A for implementation and additional experimental details). For evaluation we randomly partition each class in each task into 400 potential support examples composing  $\mathcal{X}$  and 200 query examples composing  $\mathcal{D}$  (all datasets have 600 examples per class). To compute average accuracy we randomly sample corresponding number of support examples per class; for the best and the worst case accuracies we use Algorithm 1 to search over support examples in  $\mathcal{X}$ .

Table 2: Accuracies for different meta-learning methods on the FC100 dataset

		Worst acc	Avg acc	Best acc
MAML	1-shot	7.32 ± 1.49%	31.89 ± 6.75%	50.13 ± 6.75%
	5-shot	7.51 ± 3.59%	43.58 ± 7.61%	61.64 ± 8.50%
	10-shot	10.51 ± 2.86%	44.30 ± 6.89%	64.98 ± 7.27%
MC	1-shot	6.53 ± 1.56%	36.56 ± 8.05%	56.75 ± 5.57%
	5-shot	5.17 ± 1.34%	47.12 ± 7.02%	66.33 ± 5.11%
	10-shot	8.35 ± 3.85%	49.12 ± 6.67%	65.27 ± 5.58%
ProtoNets	1-shot	5.25 ± 1.69%	37.21 ± 0.50%	59.75 ± 6.39%
	5-shot	5.57 ± 2.99%	50.49 ± 0.48%	70.31 ± 6.41%
	10-shot	9.93 ± 4.39%	56.15 ± 0.47%	72.94 ± 6.23%
R2D2	1-shot	6.13 ± 1.63%	37.91 ± 0.48%	59.67 ± 6.17%
	5-shot	6.72 ± 2.85%	54.35 ± 0.49%	74.34 ± 6.59%
	10-shot	12.00 ± 7.07%	61.72 ± 0.47%	77.94 ± 3.56%
MetaOptNet-Ridge	1-shot	5.44 ± 1.57%	39.13 ± 0.51%	61.28 ± 6.18%
	5-shot	5.97 ± 3.29%	53.20 ± 0.47%	72.65 ± 6.43%
	10-shot	11.56 ± 2.78%	59.52 ± 0.48%	75.30 ± 1.56%
MetaOptNet-SVM	1-shot	5.29 ± 1.57%	38.19 ± 0.48%	60.14 ± 6.12%
	5-shot	5.75 ± 2.83%	54.45 ± 0.49%	74.01 ± 6.64%
	10-shot	9.54 ± 3.86%	60.52 ± 0.48%	77.03 ± 6.27%

Table 3: Accuracies for different meta-learning methods on the miniImageNet dataset

		Worst acc	Avg acc	Best acc
MAML	1-shot	6.08 ± 1.77%	47.13 ± 8.78%	71.39 ± 6.74%
	5-shot	10.15 ± 8.40%	57.69 ± 7.92%	79.60 ± 5.43%
	10-shot	20.88 ± 7.22%	59.52 ± 8.34%	79.94 ± 3.41%
MC	1-shot	4.46 ± 2.05%	45.03 ± 8.79%	65.98 ± 6.23%
	5-shot	5.79 ± 3.45%	60.47 ± 7.57%	75.09 ± 4.72%
	10-shot	9.67 ± 3.84%	60.54 ± 7.45%	73.23 ± 6.24%
ProtoNets	1-shot	4.69 ± 2.16%	53.42 ± 0.59%	76.46 ± 5.64%
	5-shot	9.53 ± 4.91%	70.60 ± 0.43%	85.33 ± 3.60%
	10-shot	15.39 ± 5.17%	75.91 ± 0.38%	87.31 ± 3.24%
R2D2	1-shot	6.10 ± 3.27%	56.09 ± 0.58%	78.17 ± 5.33%
	5-shot	12.03 ± 5.51%	72.04 ± 0.43%	86.64 ± 3.34%
	10-shot	15.78 ± 6.10%	77.32 ± 0.36%	86.70 ± 1.99%
MetaOptNet-Ridge	1-shot	5.19 ± 3.21%	57.94 ± 0.62%	79.15 ± 5.05%
	5-shot	9.83 ± 4.05%	74.80 ± 0.43%	84.81 ± 4.12%
	10-shot	19.16 ± 10.07%	80.31 ± 0.36%	89.72 ± 2.99%
MetaOptNet-SVM	1-shot	4.82 ± 3.03%	59.03 ± 0.62%	80.38 ± 5.40%
	5-shot	9.52 ± 4.88%	75.54 ± 0.40%	85.41 ± 3.86%
	10-shot	15.93 ± 7.64%	80.16 ± 0.37%	90.63 ± 2.71%

Our key finding is the large range of performances of *all* meta-learning algorithms considered in 1-, 5-, and 10-shot settings. Prototypical networks have slightly better worst-case 5-shot accuracy on CIFAR-FS, but it has large variance and is likely due to our algorithm finding poor local optima on one of the tasks. We also see no differences between meta-learners adapting end-to-end, i.e. MAML and MC, and those adapting only the last linear classification layer, i.e. R2D2 and MetaOptNet. Goldblum et al. [12] showed that the last-layer adaptation methods produce good quality linear-separable embeddings. One could expect such methods to be less sensitive to support data, however, as we discuss in Section 4, linear separability is not sufficient in the few-shot learning setting. 10-shot worst-case accuracies are not as poor (despite mostly remaining worse than random guessing), but

we expect that with more support data available, the gap would narrow.<sup>2</sup> Finally, we note that the best-case accuracy is significantly better, especially in the 1-shot setting.

### 3.2 Worst case support examples are not artifacts

We have demonstrated that it is possible to find support examples yielding poor post-adaptation performance of a variety of meta-learners. It is also important to understand the nature of these worst-case examples: are they data artifacts, i.e. outliers or miss-labeled examples, or realistic images that could cause failures in practice? We argue that the latter is the case.

1. In Figure 1 we presented several examples of the worst-case support images on CIFAR-FS: they are correctly labeled and appear representative of the respective classes. Inspecting the images on the left closer we note that it is often not easy to notice visually that such support examples could result in a poor performance without significant expert knowledge of the dataset. Only the cellphone image labelled “telephone” and grey-scale image labeled “plain” seem potentially problematic. On the other hand, “snail” and “bed” images all appear normal. On the right figure, majority of the images also appear reasonable.
2. 10-shot setting should be a lot more resilient to outliers, however Algorithm 1 continues to be successful in 10-shot setting, finding ten different examples per class leading to accuracy slightly better than a random predictor as shown in Tables 1, 2, and 3.
3. In Figure 3 we present histograms of accuracies visualizing the first iteration over classes of Algorithm 1 in 1-shot learning on CIFAR-FS. The right most histogram corresponds to post-adaptation accuracies for different choices of support image for class 0 and random choices for classes 1-4. The subsequent histogram is for different choices of support images for class 1, where image for class 0 is chosen with Algorithm 1 and classes 2-4 are random, and analogously for the remaining three histograms. We see that the lower accuracy tails of the first two histograms contain multiple worst-case support examples in the corresponding classes. By the third histogram, the range of accuracies is below 50% for *all* possible support examples in the corresponding classes.
4. In Figure 4 we present histogram of accuracies of 3991 unique combinations from CIFAR-FS of 1-shot support examples evaluated by Algorithm 1 throughout 3 iterations. There are 3335 distinct sets of 5 examples each with less than 50% post-adaptation accuracy.

We present analogous analysis for other meta-learners and datasets in Appendix C, where we also conclude that worst-case adaptation examples are realistic and can cause malfunctions in practice.

### 3.3 Improving support data robustness with adversarial training

The issue of robustness has been studied in many contexts in machine learning, and adversarial vulnerability of deep learning based models has been explored extensively in the recent literature [13, 5, 22]. While the support data sensitivity of meta learners presented in this work is a new type of non-robustness, it is possible to approach the problem borrowing ideas from adversarial training [22]. The high-level idea of adversarial training is to use a mechanism exposing non-robustness to guide the data used for model updates. For example, if a model is sensitive to perturbations in the inputs, for an incoming batch of data we find the worst-case (by maximizing the loss) perturbations to the inputs and use the perturbed inputs to updated model parameters. To converge, adversarial training needs to find model parameters such that the mechanism exposing non-robustness can no-longer damage the performance. Adversarial training has theoretical guarantees for convex models [42] and has been shown empirically to be successful in defending against adversarial attacks on deep learning models [22].

We use adversarial training scheme in an attempt to achieve robustness to support data in meta learning. Specifically, we use Algorithm 1 to find worst-case adaptation examples during training instead of using random ones as in standard training. The result is quite intriguing: Tables 4 and 5 summarize performance of adversarially (in a sense of support data) trained meta-learners on train and test tasks. In most cases, adversarial training converged, i.e. we are no longer able to find detrimental worst-case support examples with Algorithm 1 on the *training* tasks, however we observe

---

<sup>2</sup>With enough support data, last-layer methods should succeed as their embeddings are linearly separable.

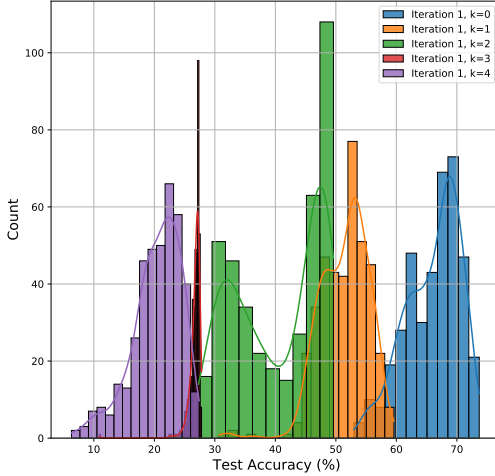


Figure 3: Histogram of test accuracies on the first iteration of Algorithm 1 as it progresses through classes evaluating 1-shot combinations of images for adaptation on a CIFAR-FS test task with MetaOptNet-SVM meta-learner.

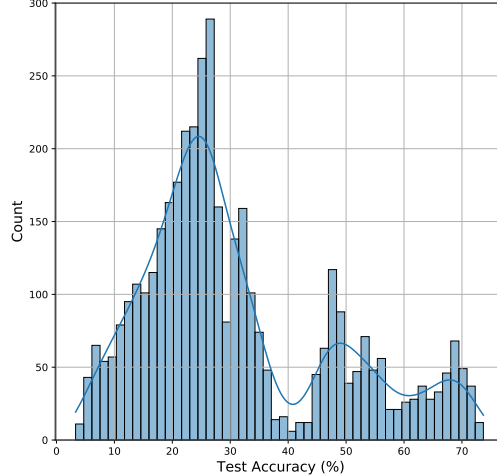


Figure 4: Histogram of test accuracies computed during 3 iterations of Algorithm 1 evaluating different unique 1-shot combinations of images for adaptation on a CIFAR-FS test task with MetaOptNet-SVM meta-learner.

no improvements of the worst-case accuracy on the *test* tasks. Our experiment demonstrates that support data sensitivity in meta-learning is not easily addressed with existing methods and requires exploring new solution paths.

Table 4: Accuracies for different meta-learning methods trained in the standard manner and adversarially on the CIFAR-FS dataset

Method	Dataset	Training	Worst acc	Avg acc	Best acc
R2D2	Train	Standard	$13.83 \pm 9.35\%$	$87.68 \pm 0.56\%$	$96.91 \pm 3.15\%$
		Adversarial	$46.34 \pm 14.93\%$	$88.19 \pm 0.59\%$	$97.94 \pm 2.89\%$
	Test	Standard	$6.14 \pm 2.89\%$	$68.86 \pm 0.68\%$	$86.63 \pm 5.97\%$
		Adversarial	$6.76 \pm 2.69\%$	$68.62 \pm 0.66\%$	$87.46 \pm 5.86\%$
MetaOptNet-Ridge	Train	Standard	$77.75 \pm 17.80\%$	$99.23 \pm 0.15\%$	$99.81 \pm 0.99\%$
		Adversarial	$90.08 \pm 14.22\%$	$98.87 \pm 0.21\%$	$99.84 \pm 0.66\%$
	Test	Standard	$5.17 \pm 2.96\%$	$71.21 \pm 0.67\%$	$87.41 \pm 5.95\%$
		Adversarial	$5.38 \pm 2.79\%$	$71.81 \pm 0.67\%$	$88.42 \pm 5.60\%$
MetaOptNet-SVM	Train	Standard	$9.74 \pm 9.00\%$	$91.93 \pm 0.47\%$	$97.58 \pm 2.51\%$
		Adversarial	$93.47 \pm 11.84\%$	$99.08 \pm 0.19\%$	$99.81 \pm 0.89\%$
	Test	Standard	$5.27 \pm 2.82\%$	$70.79 \pm 0.69\%$	$87.66 \pm 5.76\%$
		Adversarial	$4.97 \pm 2.58\%$	$71.11 \pm 0.70\%$	$87.84 \pm 5.93\%$

#### 4 Meta-learning margin analysis

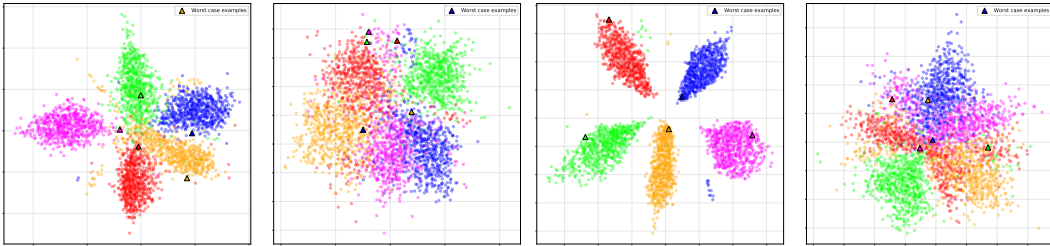
Meta-learners, specifically those only adapting the last layer linear classifier, produce embeddings that appear (approximately) linearly separable even when projected into two dimensions as shown in Figure 5 (we used Multidimensional Scaling [24] to preserve the relative cluster sizes and distances between clusters). In the supervised learning context this would be considered an easy problem for, e.g., linear SVM as in the MetaOptNet-SVM meta-learner. The problem, however, is that in the supervised learning context we typically have sufficient data, while in the few-shot setting meta-learners are restricted to as little as a single example per class in a high-dimensional space.

Lets take another look at Figure 5. These are Multidimensional Scaling [24] projections of the embeddings obtained with the MetaOptNet trained on CIFAR-FS. Embeddings in (a) and (c) correspond



Table 5: Accuracies for different meta-learning methods trained in the standard manner and adversarially on the FC100 dataset

Method	Dataset	Training	Worst acc	Avg acc	Best acc
R2D2	Train	Standard	$8.95 \pm 6.62\%$	$84.24 \pm 0.60\%$	$95.48 \pm 3.45\%$
		Adversarial	$45.01 \pm 12.55\%$	$87.50 \pm 0.57\%$	$97.11 \pm 2.97\%$
	Test	Standard	$6.13 \pm 1.63\%$	$37.91 \pm 0.48\%$	$59.67 \pm 6.17\%$
		Adversarial	$6.44 \pm 1.68\%$	$38.70 \pm 0.46\%$	$60.87 \pm 6.30\%$
MetaOptNet-Ridge	Train	Standard	$14.27 \pm 11.32\%$	$92.54 \pm 0.45\%$	$97.62 \pm 2.51\%$
		Adversarial	$79.67 \pm 14.69\%$	$96.83 \pm 0.38\%$	$98.76 \pm 2.26\%$
	Test	Standard	$5.44 \pm 1.57\%$	$39.13 \pm 0.51\%$	$61.29 \pm 6.18\%$
		Adversarial	$5.47 \pm 1.89\%$	$36.87 \pm 0.48\%$	$59.07 \pm 6.63\%$
MetaOptNet-SVM	Train	Standard	$18.05 \pm 12.64\%$	$94.13 \pm 0.38\%$	$98.28 \pm 2.09\%$
		Adversarial	$90.37 \pm 12.18\%$	$97.12 \pm 0.39\%$	$99.06 \pm 1.98\%$
	Test	Standard	$5.29 \pm 1.57\%$	$38.19 \pm 0.48\%$	$60.14 \pm 6.11\%$
		Adversarial	$6.08 \pm 1.78\%$	$35.92 \pm 0.45\%$	$60.30 \pm 6.26\%$



(a) Train task embeddings (standard training) (b) Test task embeddings (standard training) (c) Train task embeddings (adversarial training) (d) Test task embeddings (adversarial training)

Figure 5: Projected embeddings of MetaOptNet-SVM for a train and a test task query data from CIFAR-FS. We compare standard and adversarial training discussed in Section 3.3. Points are colored with their labels. Highlighted points are the worst-case support examples selected with Algorithm 1.

to a query data from a train task for standard and adversarially trained models, and embeddings in (b) and (d) to a query data from a test task for the corresponding models. All embeddings appear well-clustered, however embeddings in (c) have the largest separation and the smallest within-class variance. This wouldn't make a significant difference in supervised learning with enough data, but makes a big difference for meta-learning: when using Algorithm 1 to find the worst-case support examples (highlighted in the figures), the corresponding accuracies are 1.8% for (a), 2.90% for (b), 99.4% for (c), and 5.70% for (d). We see that although all embeddings are well-separated<sup>3</sup>, only (c) is robust to support data selection. We present the projected embeddings for a variety of meta-learners on the FC100 dataset in Appendix D. As we discuss next, robust meta-learning, in addition to vanilla linear-separability, requires features with bigger class separation and lower intra-class variances.

In the following theorem, we show that as long as the class embeddings are sufficiently separated, the probability of any two points sampled from the classes leading to a max-margin classifier with large misclassification rate is exponentially small. We note the high degree of separation (linear in embedding dimension) necessary to guarantee robustness to the choice of support data. This suggests unless the class embeddings are well-separated, the resulting meta-learning algorithm will be sensitive to the choice of support data.

**Theorem 4.1.** Consider a (binary) Gaussian discriminant analysis model:

$$\begin{aligned}
 X \mid Y = 1 &\sim N(\mu, \sigma^2 I_d), \\
 X \mid Y = 0 &\sim N(-\mu, \sigma^2 I_d).
 \end{aligned}$$

<sup>3</sup>For comparison, the accuracies of the corresponding supervised learning problems (i.e. linear classifiers trained using embeddings of all 400 per class potential support examples  $\mathcal{X}$ , rather than a single example per class) are 99.6% for (a) and 94.3% for (b).

As long as  $\mu = (d + \sqrt{2dt} + 2t)\sigma$  for some  $t > 0$ , then the max-margin classifier between two points sampled independently from each cluster has misclassification rate at most  $\Phi(-d - \sqrt{2dt} - 2t)$  with probability at least  $(1 - e^{-t})^2$ .

*Proof.* Consider a (binary) Gaussian discriminant analysis model:

$$\begin{aligned} X | Y = 1 &\sim N(\mu, \sigma^2 I_d), \\ X | Y = 0 &\sim N(-\mu, \sigma^2 I_d). \end{aligned}$$

Define the core of the clusters as the sets

$$\begin{aligned} \mathcal{C}_1 &\triangleq \{x \in \mathbb{R}^d \mid \|x - \mu\|_2 \leq r\sigma\}, \\ \mathcal{C}_0 &\triangleq \{x \in \mathbb{R}^d \mid \|x + \mu\|_2 \leq r\sigma\}. \end{aligned}$$

It is a tedious geometric exercise to show that as long as  $\mu > 2r\sigma$ , then the midpoint of any pair of points  $(x_1, x_0) \in \mathcal{C}_1 \times \mathcal{C}_0$  falls outside  $\mathcal{C}_1 \cup \mathcal{C}_0$ . Further, the hyperplane

$$\mathcal{H} \triangleq \{x \in \mathbb{R}^d \mid (x_1 - x_0)^T x = \frac{1}{2}(\|x_1\|_2^2 - \|x_0\|_2^2)\}$$

bisects  $\mathcal{C}_0$  and  $\mathcal{C}_1$  for any choice of  $x_1, x_0$ . This implies the risk of any max-margin classifier constructed from  $x_1$  and  $x_0$  has misclassification error rate at most  $2\Phi(-r)$ , where  $\Phi$  is the  $N(0, 1)$  CDF. We note that the probability of a pair of independently sampled points  $x_1 \sim N(\mu, \sigma^2 I_d)$  and  $x_0 \sim N(-\mu, \sigma^2 I_d)$  falling in  $\mathcal{C}_1$  and  $\mathcal{C}_0$  respectively is  $F_{\chi_d^2}(r)$ , where  $F_{\chi_d^2}$  is the CDF of a  $\chi_d^2$  random scalar. By picking  $r = d + \sqrt{2dt} + 2t$  for some  $t > 0$ , the probability of  $x_1 \in \mathcal{C}_1$  and  $x_0 \in \mathcal{C}_0$  is at least  $(1 - e^{-t})^2$ .  $\square$

## 5 Conclusion

We studied the problem of support data sensitivity in meta-learning: the performance of existing algorithms is extremely sensitive to the examples used for adaptation. Our findings suggest that when deploying meta-learning, especially in safety-critical applications such as autonomous driving or medical imaging, practitioners should carefully check the support examples they label for the meta-learner. However, even when the data is interpretable for a human, e.g. images, recognizing potentially detrimental examples could be hard as we have seen in our experiments.

In our experiments, we considered popular few-shot image classification benchmarks. We note that meta-learning has also been applied to data in other modalities such as language understanding [9] and speech recognition [16]. We expect our conclusions and Algorithm 1 for finding the worst-case support data to apply in other meta-learning applications, however, an empirical study is needed to verify this.

Going forward, our results suggest that robustness in meta-learning could be achieved by explicitly encouraging separation and tighter intra-class embeddings (at least in the context of last-layer adaptation meta-learners). Unfortunately, the adversarial training approach, while successful in promoting robustness in many applications, fails to achieve robustness of meta-learners to support data. In our experiments, adversarial training achieved well-separated and tight intra-class embeddings resulting in robustness on the train tasks (i.e., tasks composed of classes seen during training), but failed to improve on the test tasks. Our findings demonstrate that new approaches are needed to achieve robustness in meta-learning.

Finally, we note that our results provide a new perspective on a different meta-learning phenomenon studied in prior work. Setlur et al. [36] and Ni et al. [27] studied the importance of the support data during training. Setlur et al. [36] quantified the impact of the support data diversity, while Ni et al. [27] considered the effectiveness of the support data augmentation, and both concluded that meta-learning is insensitive to the support data quality and diversity. In our experiments with a variation of adversarial training in Section 3.3, where support data during training is selected based on Algorithm 1 to find worst-case support examples, we achieved significant improvements in terms of the worst-case accuracies on the train tasks. Thus, the conclusion is different from [36, 27], i.e. the support data used during training has an impact on the resulting meta-learner from the perspective of sensitivity studied in our work.

## Acknowledgments and Disclosure of Funding

This note is based upon work supported by the National Science Foundation (NSF) under grants no. 1916271, 2027737, and 2113373. Any opinions, findings, and conclusions or recommendations expressed in this note are those of the authors and do not necessarily reflect the views of the NSF.

## References

- [1] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.
- [2] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine Bias. [www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing](http://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing), May 2016.
- [3] Sébastien MR Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. learn2learn: A library for meta-learning research. *arXiv preprint arXiv:2008.12284*, 2020.
- [4] Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.
- [5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.
- [6] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.
- [7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [8] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019.
- [9] Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. Investigating meta-learning algorithms for low-resource natural language understanding tasks. *arXiv preprint arXiv:1908.10423*, 2019.
- [10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.
- [11] Micah Goldblum, Liam Fowl, and Tom Goldstein. Adversarially robust few-shot learning: A meta-learning approach. *arXiv preprint arXiv:1910.00982*, 2019.
- [12] Micah Goldblum, Steven Reich, Liam Fowl, Renkun Ni, Valeriia Cherepanova, and Tom Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *International Conference on Machine Learning*, pages 3607–3616. PMLR, 2020.
- [13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [14] Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.
- [15] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.
- [16] Jui-Yang Hsu, Yuan-Jui Chen, and Hung-yi Lee. Meta learning for end-to-end low-resource speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7844–7848. IEEE, 2020.
- [17] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.
- [18] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanus Phillips, Sara Beery, et al. Wilds: A benchmark of in-the-wild distribution shifts. *arXiv preprint arXiv:2012.07421*, 2020.

- [19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [20] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.
- [21] Kwonjoon Lee, Subhansu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.
- [22] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- [23] Gabriel Maicas, Andrew P Bradley, Jacinto C Nascimento, Ian Reid, and Gustavo Carneiro. Training medical image analysis systems like radiologists. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 546–554. Springer, 2018.
- [24] Al Mead. Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(1):27–39, 1992.
- [25] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471. IEEE, 2000.
- [26] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.
- [27] Renkun Ni, Micah Goldblum, Amr Sharaf, Kezhi Kong, and Tom Goldstein. Data augmentation for meta-learning. In *International Conference on Machine Learning*, pages 8152–8161. PMLR, 2021.
- [28] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.
- [29] Joseph D Novak, D Bob Gowin, and Gowin D Bob. *Learning how to learn*. cambridge University press, 1984.
- [30] Elre Talea Oldewage, John F Bronskill, and Richard E Turner. Attacking few-shot classifiers with adversarial support sets. 2020.
- [31] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 719–729, 2018.
- [32] Eunbyung Park and Junier B Oliva. Meta-curvature. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL <https://proceedings.neurips.cc/paper/2019/file/57c0531e13f40b91b3b0f1a30b529a1d-Paper.pdf>.
- [33] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018.
- [34] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.
- [35] Ahmad El Sallab, Mahmoud Saeed, Omar Abdel Tawab, and Mohammed Abdou. Meta learning framework for automated driving. *arXiv preprint arXiv:1706.04038*, 2017.
- [36] Amrith Setlur, Oscar Li, and Virginia Smith. Is support set diversity necessary for meta-learning? *arXiv preprint arXiv:2011.14048*, 2020.
- [37] Dylan Slack, Sorelle A Friedler, and Emile Givental. Fairness warnings and fair-maml: learning fairly with minimal data. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 200–209, 2020.
- [38] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.
- [39] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

- [40] Xingyou Song, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn, and Jie Tan. Rapidly adaptable legged robots via evolutionary meta-learning. *arXiv preprint arXiv:2003.01239*, 2020.
- [41] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.
- [42] Abraham Wald. Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, pages 265–280, 1945.
- [43] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6750–6759, 2019.
- [44] Han Xu, Yaxin Li, Xiaorui Liu, Hui Liu, and Jiliang Tang. Yet meta learning can adapt fast, it can also break easily. *arXiv preprint arXiv:2009.01672*, 2020.
- [45] Chengxiang Yin, Jian Tang, Zhiyuan Xu, and Yanzhi Wang. Adversarial meta-learning. *arXiv preprint arXiv:1806.03316*, 2018.

## Checklist

1. For all authors...
  - (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [\[Yes\]](#)
  - (b) Did you describe the limitations of your work? [\[Yes\]](#) See 3 for limitations on finding adaptation vulnerabilities and 3.3 for adversarial training limitations
  - (c) Did you discuss any potential negative societal impacts of your work? [\[N/A\]](#)
  - (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [\[Yes\]](#)
2. If you are including theoretical results...
  - (a) Did you state the full set of assumptions of all theoretical results? [\[Yes\]](#)
  - (b) Did you include complete proofs of all theoretical results? [\[Yes\]](#)
3. If you ran experiments...
  - (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [\[Yes\]](#) Material for reproducibility of experiments is currently included in the supplement, and will later be released online
  - (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [\[Yes\]](#) See sections 3.1 and 3.3
  - (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [\[Yes\]](#) All experiments are run multiple times with different seeds and standard deviations are reported for each experiment
  - (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [\[Yes\]](#) We use machines on internal cluster with a single V-100 GPU to run all our experiments
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
  - (a) If your work uses existing assets, did you cite the creators? [\[Yes\]](#)
  - (b) Did you mention the license of the assets? [\[Yes\]](#) External existing assets used in this work are either MIT or Apache licensed
  - (c) Did you include any new assets either in the supplemental material or as a URL? [\[Yes\]](#) Additional code for methods proposed in this work are included in the supplemental material
  - (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [\[N/A\]](#)

- (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Standardized image classification datasets are used with no known personally identifiable information available
5. If you used crowdsourcing or conducted research with human subjects...
- (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
  - (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
  - (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

## A Experiment details

**Network architectures:** For MAML and Meta-Curvature experiments, we use a 4-layer CNN network, where each convolutional block in the network is a sequential composition of a  $[2 \times 2$  max-pooling layer, batch normalization, and a  $3 \times 3$  convolution layer]. The final classification layer of the network is fully-connected layer mapping the input features to 5-way output.

For ProtoNet and R2D2 experiments, we use the same architectures as are used in the original papers. The ProtoNet feature extractor is a combination of four convolutional blocks. Each block consists of a 64-filter  $3 \times 3$  convolution, batch normalization layer, a ReLU nonlinearity, and a  $2 \times 2$  max-pooling layer. The R2D2 feature extractor is a combination of 4 convolutional layers with [96, 192, 384, 512] filters. Each convolutional layer consists of a  $3 \times 3$  convolution, batch normalization,  $2 \times 2$  max pooling, and a leaky ReLU with a factor of 0.1

For MetaOptNet experiments, we use the same implementation and setting as described in the original paper. MetaOptNet networks consist of a ResNet-12 network as feature extractors, and either a support vector machine or ridge regression based head for classification.

**Meta-learning setup:** MetaOptNet networks utilize SGD with Nesterov momentum of 0.9 and a weight decay of  $5 \times 10^{-4}$  for optimization. The learning rate for this set of experiments was initially set to 1.0 and then modified to 0.06 for epochs 20 to 40, 0.012 for epochs 40 to 50, and 0.024 thereafter. MAML and Meta-Curvature networks are trained using Adam optimizer with an initial learning rate of  $3 \times 10^{-4}$  and 0.01 respectively.

All networks are trained for 60000 iterations – 60 epochs of 1000 episodes each, with the batch sizes for MAML experiments set to 32 tasks in each batch, batch size for Meta-Curvature set to 16 tasks in each batch, and for MetaOptNet experiments it’s set to 8 tasks in each batch.

During the meta-training phase, we apply the random crop, color jitter, and random horizontal flip transformations for MetaOptNet networks. Additionally, we match the meta-training shot with the meta-testing shot for all networks. While meta-training, we compute the accuracy on a 5-shot 5-way validation dataset, and select the model with the best accuracy on this validation dataset for sensitivity analysis.

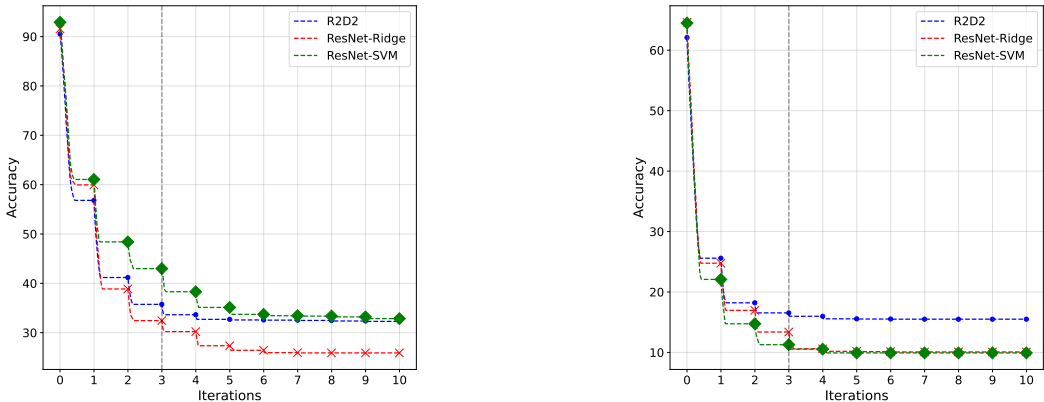
During evaluation, i.e. results in Tables 1, 2, and 3, to compute best and worst accuracies we randomly partition each class in each task into 400 potential adaptation examples composing  $\mathcal{X}$  and 200 evaluation examples composing  $\mathcal{D}$  (all datasets have 600 examples per class). We use the corresponding algorithm to find the adaptation examples in  $\mathcal{X}$  and report mean and standard deviations of evaluation data  $\mathcal{D}$  post-adaptation accuracies over 500 random tasks for 1-shot setting, and 100 random tasks for 5-shot and 10-shot settings. To compute average accuracies we follow the setup of previous meta-learning papers, i.e. sample corresponding number of adaptation examples randomly and choose a random subset of 50 examples per class for evaluation, and report mean and standard deviation of accuracies over 1000 random tasks.

**Adversarial training setup:** To adversarially train the models as described in Section 3.3, we initialize with models trained in the standard fashion. We then train these models in an adversarial manner for 60 epochs of 1000 episodes each. We use the same hyperparameters and experimental setup as we did for the standard training, except that we reduce the learning rate by a factor of 10. Thus, the learning rate is initially set to 0.1 for the first 20 epochs, then modified to 0.006 for epochs 20 to 40, 0.0012 for epochs 40 to 50, and 0.0024 thereafter. To find the adversarial examples, we find the worst-case examples using algorithm 1 run for 3 iterations. We then use these worst-case examples as the query data to update the model parameters.

**Algorithm runtimes:** We run all our experiments on a 12 CPU core, 32 GB RAM, and 1 V100 GPU machine. The run times for a single iteration (in our experiments we ran 3 iterations to find the worst/best case examples) for MetaOptNet-SVM method on CIFAR-FS are approximately 3 minutes for 1-shot setting, approximately 18 minutes for 5-shot setting, and approximately 43 minutes for 10-shot setting. The corresponding run times for a single iteration for R2D2 method on FC100 are approximately 1 minute for 1-shot setting, approximately 6 minutes for 5-shot setting, and approximately 20 minutes for 10-shot setting.

## B Convergence of the algorithm to find adaptation vulnerabilities

To find the worst-case support examples (Section 3) and to find the adversarial examples for adversarial training (Section 3.3), we use Algorithm 1 with 3 rounds of attack or iterations. In this section, we show the convergence of Algorithm 1 and the rationale behind choosing 3 iterations. We execute algorithm 1 to find the worst-case 5-way 10-shot support examples on CIFAR-FS and FC100 datasets for R2D2, ResNet-Ridge, and the ResNet-SVM algorithms. We track the worst-case accuracy as it updates through 10 iterations, and show the mean worst-case accuracy through the iterations averaged over 5 different randomly-sampled tasks in figure 6. We see that while the worst-case accuracy drops significantly in the first few iterations, it stabilizes after 3 iterations and does not show significant change after the 3 iterations for all datasets and algorithms. Additionally, running for longer iterations can reduce accuracy slightly more for the 10-shot setting as compared to the 1-shot and 5-shot settings; however, the 10-shot setting is more robust to this algorithm than the 5-shot setting and thus the drop in accuracy in later iterations is understandable.



(a) Worst-case accuracy over 10 iterations of algorithm 1 for different algorithms and on the CIFAR-FS dataset.

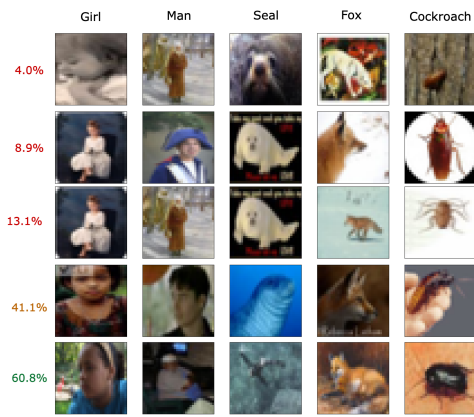
(b) Worst-case accuracy over 10 iterations of Algorithm 1 for different algorithms and on the FC100 dataset.

Figure 6: Convergence plots for Algorithm 1 for different meta-learning algorithms on the CIFAR-FS and FC100 datasets.

## C Performance range results

1. In Figure 7 we present several examples of the worst-case support images on the FC100 and the miniImageNet datasets for the MetaOptNet-SVM method. Additionally, in Figure 8 we present several examples of the worst-case support images on the CIFAR-FS, FC100, and the miniImageNet datasets for the ProtoNets method. All the support images are correctly labeled and appear representative of the respective classes. Closely inspecting the images, we note that it is often not easy to notice visually that such support examples could result in a poor performance without significant expert knowledge of the dataset. Barring a very small portion of the images (e.g., gray-scale “Girl” image and a truck-size “Nematode”), images appear reasonable and adequately representative of their respective classes.
2. In Figure 9 we present histograms of accuracies visualizing the first iteration over classes of Algorithm 1 in 1-shot learning on miniImageNet dataset for MAML, R2D2, MetaOptNet-Ridge, and the MetaOptNet-SVM algorithms. The rightmost histogram (in each sub-figure) corresponds to post-adaptation accuracies for different choices of support image for class 0 and random choices for classes 1-4. The subsequent histogram (in each sub-figure) is for different choices of support images for class 1, where image for class 0 is chosen with Algorithm 1 and classes 2-4 are random, and analogously for the remaining three histograms. We see that the lower accuracy tails of the first two histograms contain multiple worst-case support examples in the corresponding classes. By the third histogram, the range of accuracies is well below the average accuracies for each of the algorithms for *all* possible support examples in the corresponding classes.



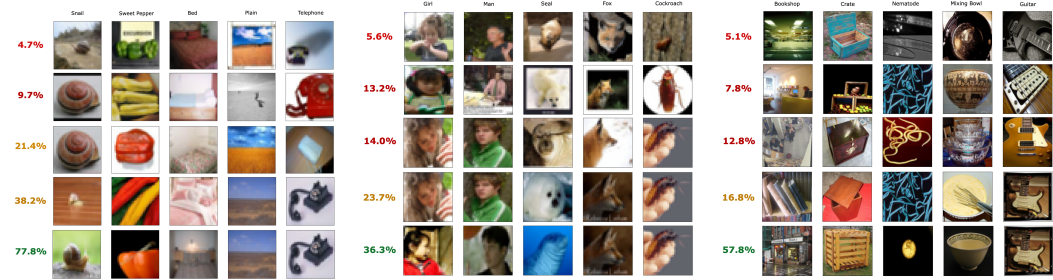


(a) Examples of FC100 adaptation images from a random 1-shot test task and the corresponding post-adaptation accuracies of MetaOptNet-SVM. Accuracy ranges from 4.0% to 60.8% illustrating the sensitivity of the meta-learner to the adaptation data.



(b) Examples of MiniImageNet adaptation images from a random 1-shot test task and the corresponding post-adaptation accuracies of MetaOptNet-SVM. Accuracy ranges from 5.8% to 75.9% illustrating the sensitivity of the meta-learner to the adaptation data.

Figure 7: Examples of unaltered support images from the FC100 and miniImageNet datasets for a random 1-shot task, depicting the post-adaptation performance of a popular meta-learning algorithm (MetaOptNet-SVM).



(a) Examples of CIFAR-FS adaptation images from a random 1-shot test task and the corresponding post-adaptation accuracies of ProtoNet. (b) Examples of FC100 adaptation images from a random 1-shot test task and the corresponding post-adaptation accuracies of ProtoNet. (c) Examples of MiniImageNet adaptation images from a random 1-shot test task and the corresponding post-adaptation accuracies of ProtoNet.

Figure 8: Examples of unaltered support images from the CIFAR-FS, FC100, and miniImageNet datasets for a random 1-shot task, depicting the post-adaptation performance of a popular meta-learning algorithm (ProtoNets).

3. In Figure 10 we present histogram of accuracies of unique combinations from the miniImageNet dataset of 1-shot support examples evaluated by Algorithm 1 throughout 3 iterations.

- (a) For the MAML algorithm, out of the 4390 distinct sets represented in the histogram, 3054 distinct sets of 5 examples each have less than 30% post-adaptation accuracy.
- (b) For the R2D2 algorithm, out of the 5986 distinct sets represented in the histogram, 5274 distinct sets of 5 examples each have less than 40% post-adaptation accuracy.
- (c) For the MetaOptNet-Ridge algorithm, out of the 3592 distinct sets represented in the histogram, 2795 distinct sets of 5 examples each have less than 40% post-adaptation accuracy.

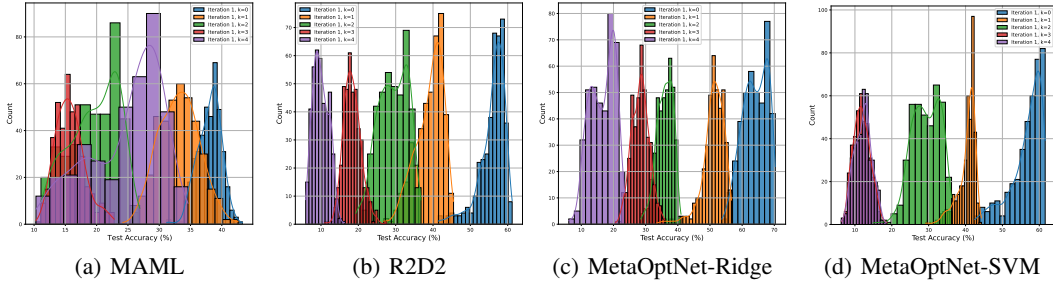


Figure 9: Histogram of accuracies visualizing progression of the first iteration of Algorithm 1 in 1-shot learning over the miniImageNet dataset for different meta-learning algorithms.

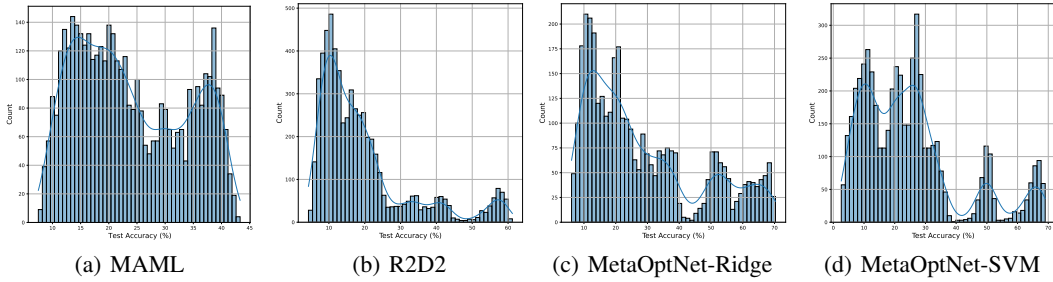


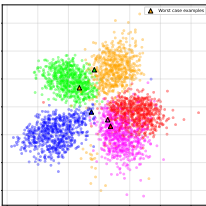
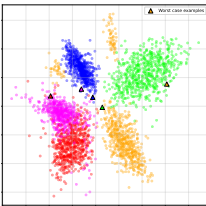
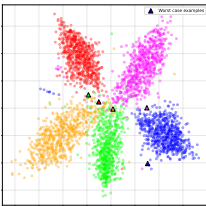
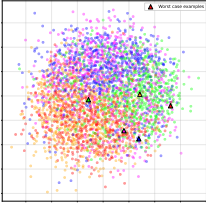


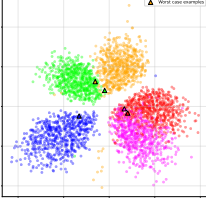
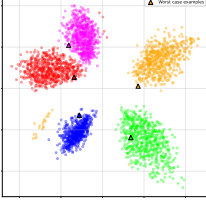
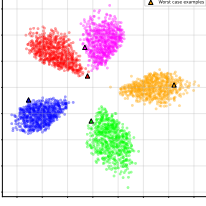
Figure 10: Histogram of accuracies of unique combinations from the miniImageNet dataset of 1-shot support examples evaluated by Algorithm 1 for 3 iterations,

- (d) For the MetaOptNet-SVM algorithm, out of the 5188 distinct sets represented in the histogram, 4403 distinct sets of 5 examples each have less than 40% post-adaptation accuracy.

## D Improving support data robustness with adversarial training

In Table 6, we show the projected embeddings for the R2D2, MetaOptNet-Ridge, and the MetaOptNet-SVM algorithms on the training and the test dataset, when trained in a standard manner vs when trained adversarially. As we note in Section 3.3 and as results depict in Tables 4 and 5, the adversarial training converges and the worst-case accuracy improves drastically on the training tasks while no improvement is observed on the test tasks.

Table 6: Projected embeddings of R2D2, MetaOptNet-Ridge, and MetaOptNet-SVM methods for a train and a test task query data from the FC100 dataset. We compare standard training and adversarial training discussed in Section 3.3. Points are colored with their labels. Highlighted points are the worst-case support examples selected with Algorithm 1.

Training (↓)	Dataset (↓)	R2D2	MetaOptNet-Ridge	MetaOptNet-SVM
Standard	Train			
	Test			
Adversarial	Train			
	Test	