On sensitivity of meta-learning to support data

Anonymous Author(s)
Affiliation
Address
email

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 [24], 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 [17]). Meta-learning has found applications in many domains, including safety-critical medical image analysis [20], autonomous driving [29], visual navigation [35] and legged robots control [32]. In this paper, we investigate the vulnerabilities of modern meta-learning algorithms in the context of few-shot image classification problem [21], 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 [11], backdoor attacks allowing malicious model manipulations [6], algorithmic biases [2] and poor generalization outside of the training domain [15] 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 [37, 9, 36] and fairness [30]. 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 [9] or the support data [36] 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.

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 remaining images without a human. Currently, meta-learning algorithms choose support examples for

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

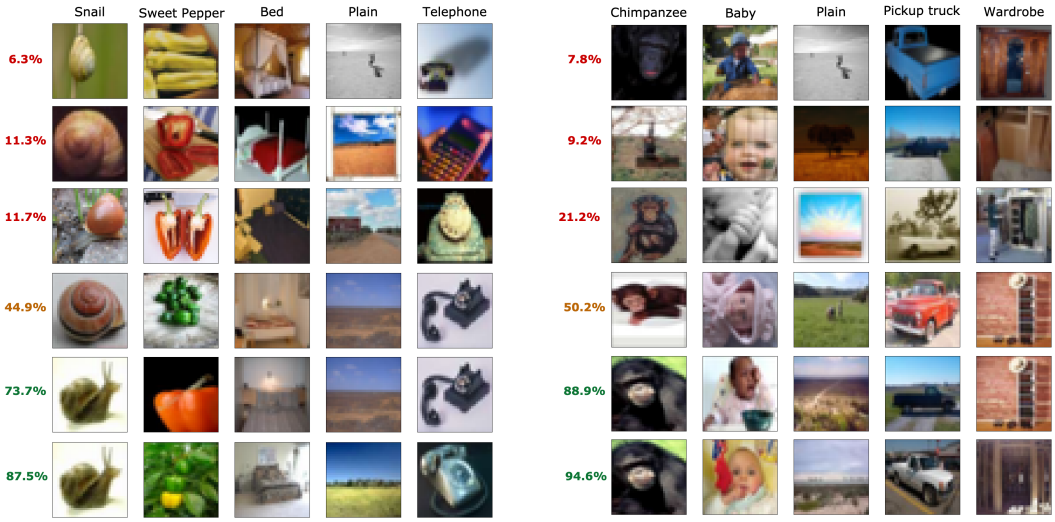


Figure 1: Examples of CIFAR-FS support images (unaltered, i.e. not modified adversarially or in any other way) from two test tasks that yield vastly different 1-shot post-adaptation performances of a popular meta-learning algorithm (MetaOptNet-SVM). Accuracy ranges from 6.3% to 94.6% suggesting extreme 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.

labeling at random. 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. 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 [19] 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 the classical supervised learning.

2 Meta-learning approaches

Meta-learning approaches are typically categorized into model-based [28, 22, 27], metric-based [14, 33, 31] and optimization-based methods [8, 23, 12]. We refer to Hospedales et al. [13] 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) [8] bi-level formulation (metric-based prototypical networks [31] 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 [8]:

$$\begin{aligned} &\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta_i, \theta; \mathcal{D}_i), \\ &\text{such that } \theta_i = \arg\min_{\theta_i} \ell(\theta_i, \theta; \mathcal{A}_i),\ i = 1, \ldots, n. \end{aligned} \tag{2.1}$$

The first line of equation 2.1 is typically minimized using gradient-based optimization of θ and is called *meta-update*. The second line is the *adaptation* step and its implementation differs depending on whether the fine-tuned θ_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. [10] for an empirical study of the differences between the two adaptation perspectives.

MAML [8] and Reptile [23] are examples of algorithms that fine-tune all network parameters during the adaptation. Instead of solving the argmin of θ_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 α. 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 θ as the neural network feature extractor parameters shared across tasks and adapts only the linear classifier parametrized with θ_i, that takes the features extracted with θ 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 θ, and returns θ_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. [18] 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 [31] 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 [18, 10].

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. [18] 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 [16] 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 32×32 RGB images from CIFAR-100 partitioned into 64, 16 and 20 classes for training, validation and testing, respectively. FC-100 [25] 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 [33] is a subsampled, downsized version of ImageNet. It consists of 60000 84×84 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 [8], Meta-Curvature (MC) [26], Prototypical networks [31], R2D2 [4], and MetaOptNet with Ridge and SVM heads [18]. 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 $[M] = \{1, \ldots, M\}$. 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).[1] 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 $\mathcal{Z} = \{z_j^k\}_{j\in[J]}^{k\in[K]}$

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 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

[1] This is a simple approach, however we found it faster and more efficient then a more sophisticated continuous relaxation using weights of the potential support examples with sparsity constraints.

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

		Worst acc	Avg acc	Best acc
MAML	1-shot	5.91 ± 1.94%	60.76 ± 25.98%	80.04 ± 6.83%
	5-shot	13.88 ± 7.28%	72.59 ± 11.34%	85.15 ± 6.12%
	10-shot	26.26 ± 7.85%	72.94 ± 9.29%	85.81 ± 6.47%
MC	1-shot	4.73 ± 1.51%	50.36 ± 23.88%	69.81 ± 8.37%
	5-shot	9.29 ± 6.13%	71.62 ± 12.50%	85.22 ± 6.53%
	10-shot	16.95 ± 10.78%	72.17 ± 9.38%	87.81 ± 6.91%
ProtoNets	1-shot	5.06 ± 2.37%	63.80 ± 0.71%	84.88 ± 6.29%
	5-shot	18.28 ± 9.85%	80.06 ± 0.46%	90.41 ± 4.72%
	10-shot	27.32 ± 11.17%	82.95 ± 0.44%	91.44 ± 4.35%
R2D2	1-shot	6.14 ± 2.89%	68.86 ± 0.68%	86.63 ± 5.97%
	5-shot	14.73 ± 8.21%	82.29 ± 0.44%	92.34 ± 4.21%
	10-shot	29.30 ± 12.16%	85.74 ± 0.42%	93.41 ± 3.81%
MetaOptNet-Ridge	1-shot	5.17 ± 2.97%	71.21 ± 0.67%	87.40 ± 5.95%
	5-shot	16.81 ± 11.57%	84.18 ± 0.45%	93.15 ± 4.43%
	10-shot	32.10 ± 16.26%	86.82 ± 0.42%	93.89 ± 3.59%
MetaOptNet-SVM	1-shot	5.27 ± 2.82%	70.79 ± 0.69%	87.65 ± 5.76%
	5-shot	14.92 ± 8.60%	83.98 ± 0.44%	93.36 ± 4.60%
	10-shot	22.24 ± 10.13%	87.11 ± 0.40%	93.56 ± 3.98%

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%	31.88 ± 21.33%	50.13 ± 6.75%
	5-shot	7.51 ± 3.59%	44.18 ± 11.70%	61.64 ± 8.50%
	10-shot	10.51 ± 2.86%	45.39 ± 9.52%	64.98 ± 7.27%
MC	1-shot	6.53 ± 1.56%	40.24 ± 24.36%	56.75 ± 5.57%
	5-shot	5.17 ± 1.34%	48.22 ± 11.11%	66.33 ± 5.11%
	10-shot	8.35 ± 3.85%	49.96 ± 9.36%	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%

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 $\mathcal{X}$.

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. [10] 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

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

		Worst acc	Avg acc	Best acc
MAML	1-shot	$6.08 \pm 1.77\%$	$49.68 \pm 21.67\%$	$71.39 \pm 6.74\%$
	5-shot	$10.15 \pm 8.40\%$	$59.68 \pm 10.84\%$	$79.60 \pm 5.43\%$
	10-shot	$20.88 \pm 7.22\%$	$63.82 \pm 9.09\%$	$79.94 \pm 3.41\%$
MC	1-shot	$4.46 \pm 2.05\%$	$45.84 \pm 22.79\%$	$65.98 \pm 6.23\%$
	5-shot	$5.79 \pm 3.45\%$	$60.18 \pm 11.26\%$	$75.09 \pm 4.72\%$
ProtoNets	1-shot	$4.69 \pm 2.16\%$	$53.42 \pm 0.59\%$	$76.46 \pm 5.64\%$
	5-shot	$9.53 \pm 4.91\%$	$70.60 \pm 0.43\%$	$85.33 \pm 3.60\%$
	10-shot	$15.39 \pm 5.17\%$	$75.91 \pm 0.38\%$	$87.31 \pm 3.24\%$
R2D2	1-shot	$6.10 \pm 3.27\%$	$56.09 \pm 0.58\%$	$78.17 \pm 5.33\%$
	5-shot	$12.03 \pm 5.51\%$	$72.04 \pm 0.43\%$	$86.64 \pm 3.34\%$
	10-shot	$15.78 \pm 6.10\%$	$77.32 \pm 0.36\%$	$86.70 \pm 1.99\%$
MetaOptNet-Ridge	1-shot	$5.19 \pm 3.21\%$	$57.94 \pm 0.62\%$	$79.15 \pm 5.05\%$
	5-shot	$32.42 \pm 7.46\%$	$74.80 \pm 0.43\%$	$84.81 \pm 4.12\%$
MetaOptNet-SVM	1-shot	$4.82 \pm 3.03\%$	$59.03 \pm 0.62\%$	$80.11 \pm 5.09\%$
	5-shot	$34.77 \pm 7.98\%$	$75.54 \pm 0.40\%$	$85.41 \pm 3.86\%$

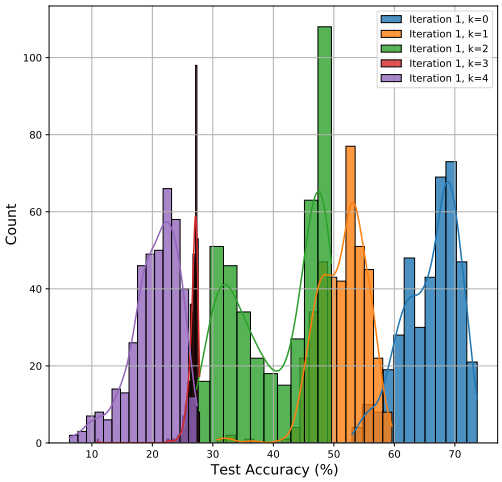


Figure 2: Histogram of test accuracies on the first iteration of Algorithm 1 as it progresses through classes evaluating 1-shot combinations of images for adaptation on a CIFAR-FS test task with MetaOptNet-SVM meta-learner.

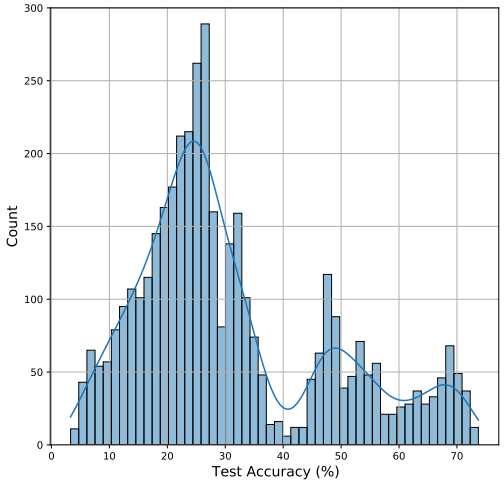


Figure 3: Histogram of test accuracies computed during 3 iterations of Algorithm 1 evaluating different unique 1-shot combinations of images for adaptation on a CIFAR-FS test task with MetaOptNet-SVM meta-learner.

we expect that with more support data available, the gap would narrow.[2] 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

[2] With enough support data, last-layer methods should succeed as their embeddings are linearly separable.

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 2 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 3 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 the appendix, 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 [11, 5, 19]. 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 [19]. 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 [34] and has been shown empirically to be successful in defending against adversarial attacks on deep learning models [19].

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 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.

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 4. 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.

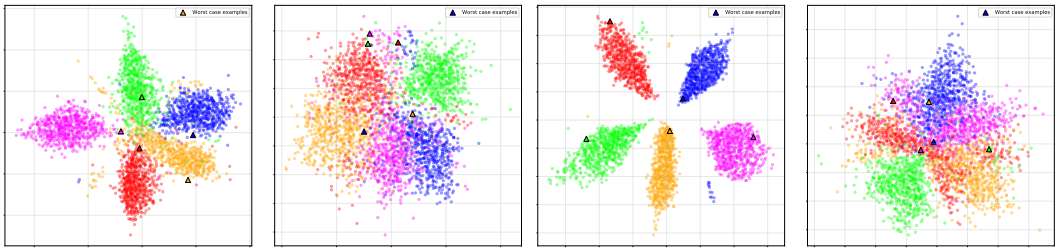
Lets take another look at Figure 4. These are multidimensional scaling projections of the embeddings obtained with the MetaOptNet trained on CIFAR-FS. Embeddings in (a) and (c) correspond to a

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

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 ± 6.62%	84.24 ± 0.60%	95.48 ± 3.45%
		Adversarial	45.01 ± 12.55%	87.50 ± 0.57%	97.11 ± 2.97%
	Test	Standard	6.13 ± 1.63%	37.91 ± 0.48%	59.67 ± 6.17%
		Adversarial	6.44 ± 1.68%	38.70 ± 0.46%	60.87 ± 6.30%
MetaOptNet-Ridge	Train	Standard	14.27 ± 11.32 %	92.54 ± 0.45%	97.62 ± 2.51%
		Adversarial	79.67 ± 14.69%	96.83 ± 0.38%	98.76 ± 2.26%
	Test	Standard	5.44 ± 1.57%	39.13 ± 0.51%	61.29 ± 6.18%
		Adversarial	5.47 ± 1.89%	36.87 ± 0.48%	59.07 ± 6.63%
MetaOptNet-SVM	Train	Standard	18.05 ± 12.64%	94.13 ± 0.38%	98.28 ± 2.09%
		Adversarial	90.37 ± 12.18%	97.12 ± 0.39%	99.06 ± 1.98%
	Test	Standard	5.29 ± 1.57%	38.19 ± 0.48%	60.14 ± 6.11%
		Adversarial	6.08 ± 1.78%	35.92 ± 0.45%	60.30 ± 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 4: Projected embeddings of MetaOptNet-SVM for a train and a test task query data from CIFAR-FS. 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.

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[3], only (c) is robust to support data selection. 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:*

$$X \mid Y = 1 \sim N(\mu, \sigma^2 I_d),$$
$$X \mid Y = 0 \sim N(-\mu, \sigma^2 I_d).$$

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:

$$X \mid Y = 1 \sim N(\mu, \sigma^2 I_d),$$
$$X \mid Y = 0 \sim N(-\mu, \sigma^2 I_d).$$

Define the core of the clusters as the sets

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

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

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

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 Φ 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 χ_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 $(1 - e^{-t})^2$. □

5 Conclusion

We studied the problem of support data sensitivity in meta-learning: performance of existing algorithms is extremely sensitive to the examples used for adaptation. Our findings suggest that when deploying meta-learning, especially in the 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.

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, 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, but failed to improve on the test tasks. Our findings demonstrate that new approaches are needed to achieve robustness in meta learning.

[3]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).

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, 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] 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.

[9] Micah Goldblum, Liam Fowl, and Tom Goldstein. Adversarially robust few-shot learning: A meta-learning approach. *arXiv preprint arXiv:1910.00982*, 2019.

[10] 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.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[12] 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.

[13] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[14] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[15] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, et al. Wilds: A benchmark of in-the-wild distribution shifts. *arXiv preprint arXiv:2012.07421*, 2020.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[17] 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.

[18] Kwonjoon Lee, Subhransu 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.

[19] 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.

[20] 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.

[21] 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.

[22] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[23] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[24] Joseph D Novak, D Bob Gowin, and Gowin D Bob. *Learning how to learn*. cambridge University press, 1984.

[25] 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.

[26] 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`.

[27] 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.

[28] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[29] Ahmad El Sallab, Mahmoud Saeed, Omar Abdel Tawab, and Mohammed Abdou. Meta learning framework for automated driving. *arXiv preprint arXiv:1706.04038*, 2017.

[30] 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.

[31] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

[32] 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.

[33] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

[34] Abraham Wald. Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, pages 265–280, 1945.

[35] 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.

[36] 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.

[37] 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 reproducbility 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]