On Data Efficiency of Meta-learning

Maruan Al-Shedivat CMU
Liam Li Determined AI
Eric Xing CMU, Petuum Inc.
Ameet Talwalkar CMU, Determined AI

Abstract

Meta-learning has enabled learning statistical models that can be quickly adapted to new prediction tasks. Motivated by use-cases in personalized federated learning, we study the often overlooked aspect of the modern meta-learning algorithms—their data efficiency. To shed more light on which methods are more efficient, we use techniques from algorithmic stability to derive bounds on the transfer risk that have important practical implications, indicating how much supervision is needed and how it must be allocated for each method to attain the desired level of generalization. Further, we introduce a new simple framework for evaluating meta-learning methods under a limit on the available supervision, conduct an empirical study of MAML, Reptile, and ProtoNets, and demonstrate the differences in the behavior of these methods on few-shot and federated learning benchmarks. Finally, we propose *active meta-learning*, which incorporates active data selection into learning-to-learn, leading to better performance of all methods in the limited supervision regime.

1 Introduction

One of the emerging applications of meta-learning [1, 2, 3] is the problem of personalization in federated learning settings [4, 5, 6]. Multiple recent works have explored the parallels between personalizing models to different users in a federated context and adapting models to different tasks in a multitask context [7, 8, 9, 10]. While the initial results from these efforts are promising, there are still many open questions when it comes to applying meta-learning to personalization in federated settings. In this work, we aim to understand which of the modern meta-learning algorithms, and under which conditions, are best suited for personalization.

It is tempting to extrapolate the performance of meta-learning methods on standard few-shot learning benchmarks [11, 12, 13] to the federated learning setting. Unfortunately, the training and evaluation routines used for benchmarking violate some of the modeling assumptions of federated learning and potentially other real-world scenarios. Specifically, the current practice is to train meta-learning methods on a large number of programmatically constructed supervised few-shot tasks sampled from an underlying labeled meta-dataset and then evaluate them on a small set of test tasks [14]. This approach implicitly assumes a full control over and an unrestricted access to the training data, allows to train on combinatorially many tasks that reuse the underlying labeled data and, as a result, ignores the associated labeling costs. In personalization, however, when tasks correspond to different users and labels correspond to user-specific preferences, ratings, etc., the data is private and cannot be reused across multiple tasks and labeling user data can be quite costly, which makes existing evaluation practices often ill-suited.

For a learning-to-learn method to work well in federated settings, it must be data efficient and able to generalize to new tasks under a limit on the available supervision. Our work is motivated by the current lack of basic understanding of generalization properties of modern meta-learning algorithms. To this end, we analyze theoretically two major families of modern algorithms—*gradient-based* (MAML [15] and Reptile [16]) and *metric-based* (ProtoNets [17])—and characterize how the number of training tasks and the number of labeled data points per task affect performance of each method.

To validate our theory and understand data efficiency of different methods in practice, we further introduce an alternative evaluation framework for meta-learning, where we measure performance as a function of the *supervision budget* or the total amount of labeled data across training tasks. Despite the conceptual simplicity, our framework allows to compare meta-learning algorithms under more realistic assumptions and reveals in-

Preprint. Preliminary version.
An updated version is to appear in AISTATS 2021.

teresting and practically relevant tradeoffs. Finally, to improve data efficiency a step further, we introduce *active meta-learning*—a method-agnostic approach that extends meta-learning with active data selection at training time and yields improved empirical performance on the benchmarks under limited supervision.

Contributions.

1. We characterize data-efficiency of modern meta-learning methods theoretically using techniques from algorithmic stability [18, 19, 20] and provide generalization bounds that indicate how much supervision is needed and how it must be allocated for each method to attain the desired performance.
2. We analyze MAML, Reptile, and ProtoNets experimentally both on the standard Omniglot and *mini*-ImageNet meta-datasets as well as on federated learning benchmarks [21]. Our results support predictions of the stability theory, reveal the relative differences between the methods in the limited supervision regime, and provide insights into how to best allocate the available supervision.
3. Finally, we benchmark meta-learning methods with and without active data selection at training time and demonstrate improved performance of active meta-learning under limited supervision.

2 Related Work

Meta-learning theory. Our analysis builds on the classical notion of algorithmic stability of Bousquet and Elisseeff [18] and extends the bounds of Maurer [19] to modern gradient-based and metric-based methods. Recent theoretical work on meta-learning has largely focused on gradient-based methods in online settings [8, 22, 23, 24, 25], providing sharper bounds but under stronger assumptions on smoothness and convexity than those required by our stability theory. Several other works have studied convergence properties of gradient-based meta-learning from the optimization standpoint rather than generalization [*e.g.*, 26, 27]. To the best of our knowledge, none of the previous work provides sufficient insight into the data efficiency aspects of modern meta-learning algorithms.

Federated learning. While the classical problem of federated learning involves estimation of a single, global model from heterogeneous data [28], many recent works have pointed out the growing importance of tailoring models to each individual user [4, 10]. Gradient-based meta-learning has been explored empirically [7, 9] and analyzed theoretically [29] as a natural choice for this problem. However, personalization in federated settings is still in a fairly nascent state [5] and our work aims to make a step toward better understanding of meta-learning in this new context.

Active learning. Combinations of active and few-shot learning have been explored in prior work in multiple settings: Woodward and Finn [30] analyzed active-learning in a streaming setting. Boney and Ilin [31] showed that active learning can improve performance at test time on new tasks, but did not consider active learning at meta-training time. The works of Bachman et al. [32] and Ravi and Larochelle [33] most closely resemble our setup, but neither approach was evaluated in the limited supervision regime or considered the problem of personalization in federated learning.

3 Background

We start by introducing the preliminaries necessary to state our theoretical results as well as overview the key elements of modern meta-learning algorithms.[1]

3.1 Meta-learning Formulation

Meta-learning operates in a multi-task setting, where the goal is to design a meta-algorithm $\mathbb{A}$ that can process data from multiple tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots\}$ and output a learning algorithm A. Given a task $\mathcal{T}_i$, the latter must be able to produce an accurate model for that task. In the context of federate learning (FL), tasks correspond to users and are represented by their personal datasets.

A meta-algorithm is data-efficient if a small number of training tasks (with only few data points per task) is sufficient for it to produce a good learning algorithm. Assuming that all tasks originate from a common underlying distribution, $\mathcal{T} \sim \mathbb{P}$, we are interested in *meta-generalization* of $\mathbb{A}$, *i.e.*, how many training tasks and how much data per task is necessary to ensure a certain level of performance on future tasks sampled from $\mathbb{P}$.

In this paper, we focus on *few-shot classification* [12], where a learning task $\mathcal{T}_i$ is defined by a small i.i.d. sample of size m (called the *support set* [34]) that consists of (x, y)-pairs, $S_i := \{(x_j, y_j)\}_{j=1}^m \sim \mathcal{D}_i^m$, where $\mathcal{D}_i \sim \mathbb{P}$. Formally, meta-learning can be formulated as a search problem over some family of algorithms $\mathcal{A}$:

$$\min_{A \in \mathcal{A}} \left\{ \mathcal{R}(A, \mathbb{P}) := \mathbb{E}_{\mathcal{D} \sim \mathbb{P}} \left[\mathbb{E}_{S \sim \mathcal{D}^m} \left[R(A(S), \mathcal{D}) \right] \right] \right\}, \quad (1)$$

$$\text{where } R(f, D) := \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\ell(f(x), y) \right]. \quad (2)$$

The objective $\mathcal{R}(A, \mathbb{P})$ given in Eq. 1 is called the *transfer risk* [35] and is defined as the expected error encountered by models $f_i(\cdot) := A(S_i)$ produced by the learning algorithm A on new tasks $\mathcal{T}_i$ sampled from $\mathbb{P}$. Transfer risk characterizes how well an algorithm A meta-generalizes over a task distribution $\mathbb{P}$.

[1]We assume that the reader is generally familiar with gradient-based and metric-based meta-learning [15, 16, 17]. Our overview is mainly focused on establishing the notation.

The algorithms from family $\mathcal{A}$ are typically designed to be able to learn from the limited support data. Meta-learning methods vary in how they define $\mathcal{A}$, which may consist of iterative optimization procedures [13, 15, 16], approximate inference [36, 37, 38, 39], or nearest neighbor approaches [12, 17], among many other hybrid methods proposed in recent years [*e.g.*, 40, 41, 42]. The methods also differ in terms of the objective functions they optimize to minimize the transfer risk Eq. 1, as the latter cannot be computed or optimized directly.

Notation. Throughout the paper, we denote data samples with S (or Q), learning algorithms with A, models that these algorithms produce after processing data samples with $f(\cdot)$ or $A(S)(\cdot)$; subscripts next to A indicate the variables that parametrize algorithms. Similarly, meta-samples (*i.e.*, sets of data samples for multiple tasks) and meta-algorithms (*i.e.*, procedures that return algorithms) are denoted with $\mathbb{S}$ and $\mathbb{A}$, respectively. The number of training tasks is denoted by n and the number of data points per task by m.

3.2 Generalization in Meta-learning

The formulation of meta-learning given in Eq. 1 was originally introduced by Baxter [35], who derived the first bounds on the excess transfer risk (*i.e.*, *meta-generalization error bounds*). For meta-algorithms $\mathbb{A}$ that produce learning algorithms A by optimizing a loss $\mathcal{L}$, we can define meta-generalization as follows.

Definition 1 (Meta-generalization Error Bound) *Let $\mathbb{A}$ be a meta-algorithm which, given a meta-sample $\mathbb{S} := \{S_1, \dots, S_n\}$ from n tasks, outputs an algorithm, $\mathbb{A}(\mathbb{S}) := \arg\min_{A \in \mathcal{A}} \mathcal{L}(A, \mathbb{S})$. A two-argument function $B(\delta, \mathbb{S})$ is called a meta-generalization error bound for $\mathbb{A}$ if for any task distribution $\mathbb{P}$ and $\delta \in (0, 1]$, the following inequality holds with probability at least $1 - \delta$:*

$$\mathcal{R}(\mathbb{A}(\mathbb{S}), \mathbb{P}) - \mathcal{L}(\mathbb{A}(\mathbb{S}), \mathbb{S}) \leq B(\delta, \mathbb{S}) \tag{3}$$

Maurer [19] developed a general technique for obtaining such bounds $B(\delta, \mathbb{S})$ using algorithmic stability [18]. In Section 4, we will specialize Maurer's bounds to modern meta-learning algorithms.

3.3 Modern Meta-learning Algorithms

In this paper, we study three popular meta-learning methods that represent two broad categories: *gradient-based* (MAML [15] and Reptile [16]) and *metric-based* (ProtoNets [17]). We have selected these methods as many recent algorithmic developments in meta-learning, few-shot learning, and their applications are variations of those three [*e.g.*, 28, 43, 44, 40, 42, 7]. However, conclusions of our study are broadly applicable to the majority of modern meta-learning.

Gradient-based meta-learning defines the family of algorithms $\mathcal{A}$ as iterative optimization procedures: $A(\mathcal{S}) := \arg\min_{\theta \in \Theta} L(f_\theta; S)$. MAML and Reptile are gradient-based methods that approximately solve this minimization problem using an *inner loop* of T (stochastic) gradient steps with a learning rate α starting from a common initialization θ_0 shared across tasks:

$$A_{\theta_0}(S) := f_{\theta_T}, \text{ where } \theta_{t+1} := \theta_t - \alpha \nabla_{\theta_t} L(f_{\theta_t}; S). \tag{4}$$

In this case, meta-learning amounts to search for an optimal initialization $\theta_0^\star$, which is done via the *outer-loop* optimization of another objective function $\mathcal{L}(A_{\theta_0}; \mathbb{S})$. The key difference between MAML and Reptile is the loss $\mathcal{L}$ they optimize in the outer loop. Reptile optimizes the empirical estimator of the transfer risk:[2]

$$\mathcal{L}_{\text{emp}}(A_{\theta_0}; \mathbb{S}) := \frac{1}{n} \sum_{i=1}^{n} \hat{R}(A_{\theta_0}, S_i), \tag{5}$$

$$\hat{R}(A_{\theta_0}, S_i) := \frac{1}{|S_i|} \sum_{(x,y) \in S_i} \ell(A_{\theta_0}(S_i)(x), y), \tag{6}$$

while MAML holds out a sub-sample of S—called the *query set* [15], which we denote Q—and optimizes an estimator of the transfer risk based on the held out set:

$$\mathcal{L}_Q(A_{\theta_0}; \mathbb{S}) := \frac{1}{n} \sum_{i=1}^{n} \hat{R}_Q(A_{\theta_0}, S_i), \tag{7}$$

$$\hat{R}_Q(A_{\theta_0}, S_i) := \frac{1}{|Q_i|} \sum_{(x,y) \in Q_i} \ell(A_{\theta_0}(S_i \setminus Q_i)(x), y). \tag{8}$$

As we will see, this difference in the meta-objectives will result in different meta-generalization bounds as well as different empirical behavior and practical implications. We note that the most commonly used algorithm in federated learning [FedAvg, 28] is identical to Reptile without fine-tuning at test time [7, 8, 9].

Metric-based meta-learning methods define the family of algorithms $\mathcal{A}$ that return non-parametric soft-nearest-neighbor models. ProtoNets is one of such methods that computes *prototype vectors* for each class in the inner loop and returns the following models:

$$A_\theta(S)(x) := \arg\max_{y \in \mathcal{Y}} \frac{\exp(-d(\mathbf{g}_\theta(x), \mathbf{c}_y)}{\sum_{y' \in \mathcal{Y}} \exp(-d(\mathbf{g}_\theta(x), \mathbf{c}_{y'})}, \tag{9}$$

$$\mathbf{c}_y := \frac{1}{|\mathcal{S}_y|} \sum_{(x, \cdot) \in \mathcal{S}_y} \mathbf{g}_\theta(x), \quad \forall y \in \mathcal{Y} \tag{10}$$

where the distance $d(\cdot, \cdot)$ is computed in the embedding space of $\mathbf{g}_\theta(\cdot)$ and the so called *class prototypes* $\mathbf{c}_y$ are computed by averaging embeddings of the corresponding support points. In the outer loop, ProtoNets optimize the same $\mathcal{L}_Q(A_\theta; \mathbb{S})$ objective as MAML.

[2]More precisely, Reptile updates θ_0 iteratively: $\theta_0 \leftarrow \theta_0 + \varepsilon \frac{1}{n} \sum_{i=1}^{n} (\theta_i - \theta_0)$, where $\theta_i = A_{\theta_0}(S_i)$, $\varepsilon > 0$. These updates approximately minimize $\mathcal{L}_{\text{emp}}$ (see Appendix A).

4 Analysis

We adapt results from the stability theory of stochastic gradient methods [20] and extend the classical bounds provided by Maurer [19] to MAML, Reptile, and ProtoNets. We also make a few key observations about their expected behavior of these methods when the number of tasks and data points per task is limited, which is of practical importance to federated settings. All proofs are provided in Appendix B.

4.1 Understanding Meta-generalization via Algorithmic Stability

As Definition 1 suggests, meta-generalization error is the discrepancy between the objective $\mathcal{L}(A;\mathbb{S})$ optimized by a meta-learning method and the true transfer risk $\mathcal{R}(A,\mathbb{P})$. If the objective function is the empirical estimator $\mathcal{L}_{\text{emp}}(A;\mathbb{S})$, then following bound holds [19]:

$$\begin{aligned}&\mathcal{R}(\mathbb{A}(\mathbb{S}),\mathbb{P}) - \mathcal{L}_{\text{emp}}(\mathbb{A}(\mathbb{S});\mathbb{S})\\ &\qquad \le 2\beta' + (4n\beta' + M)\sqrt{\frac{\ln(1/\delta)}{2n}} + 2\beta\end{aligned} \tag{11}$$

with probability at least $1-\delta$; β' and β define stability[3] of the meta-learning algorithm $\mathbb{A}$ and of any learning algorithm A it produces, respectively. Generally, β' and β are functions of the number of training tasks n and data points per task m and depend on the specific algorithms. Maurer's bound becomes non-trivial when $\beta' = o(1/n^a), a \ge 1/2$ and $\beta = o(1/m^b), b \ge 0$.

To derive bounds for modern meta-learning algorithms, we need two additional results. First, for algorithms that optimize the Q-estimator $\mathcal{L}_Q(\mathbb{A};\mathbb{S})$ instead of the $\mathcal{L}_{\text{emp}}(\mathbb{A};\mathbb{S})$ of the transfer risk, we need to bound on the difference $\mathcal{R}(\mathbb{A}(\mathbb{S}),\mathbb{P}) - \mathcal{L}_Q(\mathbb{A}(\mathbb{S});\mathbb{S})$. We prove the following theorem that provides such a bound.

Theorem 2 *Let $\mathbb{A}$ be β'_Q-uniformly stable with respect to $\hat{R}_Q$. Then, the following indequality holds for any task distribution $\mathbb{P}$ with probability at least $1-\delta$:*

$$\begin{aligned}&\mathcal{R}(\mathbb{A}(\mathbb{S}),\mathbb{P}) - \frac{1}{n}\sum_{i=1}^{n}\hat{R}_Q(\mathbb{A}(\mathbb{S}),S_i)\\ &\qquad \le 2\beta'_Q + (4n\beta'_Q + M)\sqrt{\frac{\ln(1/\delta)}{2n}}\end{aligned} \tag{12}$$

where $\hat{R}_Q(A,S_i) := \frac{1}{|Q_i|}\sum_{(x,y)\in Q_i}\ell(A(S_i\setminus Q_i)(x),y)$ with $\ell(\cdot,\cdot)$ bounded by M.

Note that the bound in Eq. 12 lacks the term 2β which depends on the stability of the inner loop learning algorithm and is present in Eq. 11.

To be able to compare the generalization of Reptile, MAML, and ProtoNets, we need expressions for β, β', β'_Q, as functions of the number of training tasks n and the number of data points per task m. Using results from stability theory of stochastic gradient method (SGM) due to Hardt et al. [20] and bounds in Eqs. 11 and 12, we arrive at the following theorem.

Theorem 3 *Let the meta-algorithm $\mathbb{A}$ be an SGM that optimizes an L'-Lipschitz and γ'-smooth loss $\mathcal{L}(A,\mathbb{S})$ by taking T' steps with non-increasing step sizes $\alpha'_t \le c'/t$. With probability at least $1-\delta$, we have the following:*

1. *If $\mathcal{L}(A;\mathbb{S})$ is Q-estimator of the transfer risk, then the following bound holds:*

$$\mathcal{R}(A,\mathbb{P}) - \mathcal{L}(A;\mathbb{S}) \le O\left(L'^2 T'\sqrt{\frac{\ln(1/\delta)}{n}}\right) \tag{13}$$

2. *If $\mathcal{L}(A;\mathbb{S})$ is the empirical estimator of the transfer risk, the inner loop learning algorithm A is an SGM that optimizes L-Lipschitz and γ-smooth loss $\ell(f(x),y)$ by taking T steps with step sizes $\alpha_t \le c/t$:*

$$\begin{aligned}&\mathcal{R}(A,\mathbb{P}(\mathcal{T})) - \mathcal{L}(A;\mathbb{S})\\ &\qquad \le O\left(L'^2 T'\sqrt{\frac{\ln(1/\delta)}{n}} + L^2 T\frac{1}{m}\right)\end{aligned} \tag{14}$$

The bound in Eq. 13 is applicable to ProtoNets and MAML; the one given in Eq. 14 applies to Reptile.

Observations. We can make the following observations by comparing expression in Eq. 13 and Eq. 14:

O1. When $n \to \infty$, the generalization error of any meta-learning method to which bound in Eq. 13 applies (*e.g.*, MAML, ProtoNets) goes to 0.

O2. The bound for Reptile has an additive term $O(L^2T/m)$ compared to MAML or ProtoNets. This implies that while we can reduce the generalization gap for MAML/ProtoNets by training on more tasks, Reptile always has a non-zero gap due to within-task sample complexity.

O3. The bound in Eq. 13 may seem to be independent of the support set size m. This is unlikely, as the Lipschitz and smoothness constants of the $\mathcal{L}(A,\mathbb{S})$ objective must depend on the properties of the support sets in $\mathbb{S}$, with larger sets more likely to results in better-behaved meta-objective.[4] Our analysis suggests that the dependence of Eq. 13 on m and n is multiplicative rather than additive in Eq. 14, which means that a large enough n can perhaps compensate for a small m in Eq. 13.

[3]Intuitively, an algorithm (or meta-algorithm) is called stable if removing a single point from S (or $\mathbb{S}$) would not affect its output by much. Precise definitions of algorithmic stability are provided in Appendix B.

[4]In practice, we observe that training ProtoNets and MAML on the same number of tasks but with larger support sets leads to better meta-test performance consistently, suggesting that larger support sets are generally better.

4.2 Implications for Meta-learning in the Limited Supervision Regime

The observations we have made in the previous section have important practical implications.

Improving evaluation. To measure data-efficiency of meta-learning methods, as **O1** suggests, we should control for the number of tasks at meta-training time, since otherwise the observed differences in performance will *not* be indicative of generalization. However, all popular few-shot classification benchmarks based on Omniglot, *mini*-ImageNet, and other datasets [14] train on tasks generated programmatically by randomly sampling support sets from a large underlying data pool. Such a construction provides access to combinatorially many training tasks, virtually setting $n \to \infty$. Instead of training on an endless stream of tasks, we propose an alternative evaluation scheme, where we strictly limit the number of unique tasks available at training time. Not only our evaluation scheme corresponds to the standard FL setting, where we have a limited number of users for meta-training, but also is compatible with the popular few-shot learning benchmarking datasets.

Understanding tradeoffs. In federated settings, acquisition of supervised data has an associated cost: in some cases, the number of users might be large, but tasks may require manual data labeling (*e.g.*, prompt users about their preferences); in other cases, manual data labeling may not be necessary or expensive, but the number of unique users might be limited. To select the best meta-learning method for a given problem, we need understand the tradeoffs. **O2** suggests that using Reptile might be suboptimal if tasks have very small support sets; at the same time, MAML and ProtoNets are likely to work better when trained on more tasks with fewer labels allocated to each task.

Optimally allocating the labeling budget. Even when the labeling budget, the number of tasks, and the support sets are all fixed, we often have additional flexibility in terms of which support points to label in each task. The standard approach is to select these points uniformly at random. Based on **O3**, we hypothesize that carefully selected support sets may lead to *better-behaved* meta-losses and improve meta-generalization.

5 Active Meta-learning Algorithms

We conjecture that actively selecting labeled support points can potentially improve performance of meta-learning methods in the limited supervision regime. Assuming that we are given a hard labeling budget B and a set of training tasks with *unlabeled* support sets, we propose an algorithm that allocates this budget among the tasks (m points per task) during meta-training by adaptively selecting which points to label.

Algorithm 1 Active Meta-learning

```
input $\mathbb{P}$: task distribution, $B$: labeling budget, $L$: # labels
    per task, $A_\theta(\cdot)$: learning algorithm, select_labeled($\cdot$):
    active labeling, meta_update($\cdot$): meta-learning update.
 1: repeat
 2:   Initialize: $\theta \leftarrow \theta_0$, $D \leftarrow \varnothing$.
 3:   if $B > 0$ and time to get new tasks then
 4:     Sample a new unlabeled set: $S_u \sim \mathbb{P}$.
 5:     Initialize labeled support set: $S_l \leftarrow \varnothing$.
 6:     for l in $1 \dots L$ do
 7:       $S_l \leftarrow S_l \cup$ select_labeled($D, A_\theta(S_l)$).
 8:     end for
 9:     Update training tasks: $D \leftarrow D \cup \{S_L\}$.
10:     Reduce available budget: $B \leftarrow B - L$.
11:   end if
12:   Sample a batch of tasks: $\{S_i\} \sim D$.
13:   Meta-learn: $\theta \leftarrow$ meta_update($A_\theta, \{S_i\}$).
14: until Convergence
output Meta-learned algorithm: $A_{\theta^\star}$.
```

Algorithm 2 Active Label Selection

```
input $S_u$: unlabeled set, $\theta_0$: initial parameters,
    $\mathbf{g}_\theta(\cdot)$: embedding function, $f_\theta(\cdot)$: predictive model,
    $A_\theta(\cdot)$: learning algorithm, $L$: # labels to sample.
 1: Initialize: $\theta \leftarrow \theta_0$.
 2: Compute representations: $\mathbf{r}_i \leftarrow \mathbf{g}_\theta(x_i), x_i \in S_u$
 3: Compute predictive entropy: $h_i \leftarrow \mathcal{H}(f_\theta(x_i)), x_i \in S_u$
 4: Get clusters: $C \leftarrow$ k-means++($\{\mathbf{h}_i\}$).
 5: Initialize the labeled set: $S_l \leftarrow \varnothing$.
 6: for all $c_j \in C$ do
 7:   Sample $L/|C|$ points from cluster $c_j$:
      $\{i_1, \dots, i_{L/|C|}\} \sim$ Categorical $\left(\{h_i\}_{i \in c_j}\right)$
 8:   Request labels: $S_l \leftarrow S_l \cup \{(x_{i_k}, y_{i_k})\}_{k=1}^{l}$.
 9:   Update models: $\mathbf{g}_\theta, f_\theta \leftarrow A_\theta(S_l)$.
10: end for
output Labeled set: $\{(x_{i_1}, y_{i_1}), \dots, (x_{i_L}, y_{i_L})\}$
```

Active meta-learning. We propose Algorithm 1 for *active meta-learning*, which gradually acquires labels for selected support points at meta-training time and proceeds in 3 steps: 1) start with a fully unlabeled support set (lines 4-5), 2) run an active label selection sub-routine that selects and labels a few points from the support set (line 6-8), 3) add the task with the labeled support set to the growing collection of training tasks (lines 9-10). Importantly, the active sampling procedure is integrated into the inner loop, where our approach interleaves active labeling with model adaptation. Put differently, instead of selecting support points that are labeled all at once, we sample them in mini-batches and re-adapt the model on already sampled points before requesting the next batch (Algorithm 2, lines 6-10).

Active labeling. Algorithm 1 relies on active labeling as a subroutine. We designed a very simple active labeling method (Algorithm 2), which uses model uncertainty and data diversity as the selection criteria, which are most common in the active learning literature [45]. Given an unlabeled support set for a new task, first, we compute representations for each point by extracting them form a hidden layer of the current model

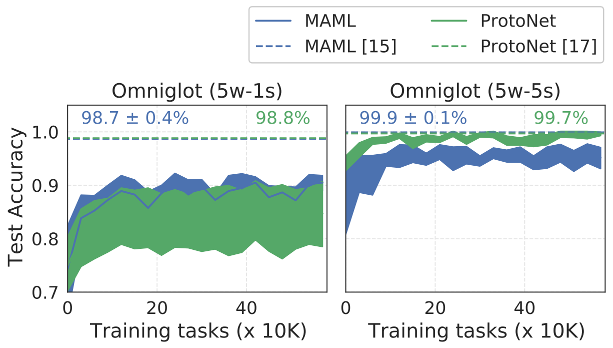

Figure 1: Test accuracy of MAML and ProtoNets trained under different constraints on the number of training tasks. The difference between the method is only visible when we control for the number of training tasks. Shaded regions are 95% CI (based on 3 runs with different seeds).

and clusters them using `k-means++` [46]. Next, we compute predictive probabilities of the current model for each unlabeled point and select points that will be labeled proportionally to the model's uncertainty with stratification by cluster. We use the entropy of the model's predictive distribution as the measure of model's uncertainty. Our approach is simple, works well, and is essentially a combination of uncertainty- and diversity-based active label acquisition [47, 48].

6 Experiments

Our experimental analysis is organized into two parts. In the first part, we validate the predictions of our theory by analyzing behavior of MAML, Reptile, and ProtoNets under different supervision tradeoffs on the standard few-shot learning datasets. In the second part, we benchmark these methods on few-shot and federate learning datasets with a fixed total labeling budget and random versus active data labeling.

6.1 The Setup

Datasets. We conduct our study on Omniglot [11] and *mini*-ImageNet [12] with the standard data splits into train, validation, and test, and federated EMNIST dataset [21]. We consider 5-way and 20-way classification tasks with 1-5 support shots and 1 query shot for Omniglot and *mini*-ImageNet. For EMNIST, each task is a 62-way classification with the data corresponding to a unique user (all 3400 users were split into train/validation/test sets as 3000/200/200), where we similarly limit the support data to 1-5 shots and use the full tests sets of each user as query sets. Note that our personalized federated learning setup slightly differs from the standard FL benchmarks where the amount of support data per user is not restricted to 1-5 points. No data augmentation is used in any of our settings.

Models and methods. We consider 3 meta-learning methods: MAML, Reptile, and ProtoNets, using the standard hyperparameters and small convolutional backbone networks [15, 16, 17].

Labeling budgets and strategies. In Omniglot and *mini*-ImageNet, each k-way task is constructed by first selecting k handwritten characters as classes, then selecting a 1-shot query set and a small support set from the corresponding data. In EMNIST, the classes are fixed for all tasks, each task corresponds to a user, the support sets are selected from the users' training data and test data is used as query sets. The labeled points in the support sets are either selected uniformly at *random* or *actively* using Algorithm 2.

Evaluation. All methods are trained either (a) in the *classical regime*, where we do not control for the the number of training tasks or the total amount of labeled data, or (b) in the *limited supervision regime*, where the total amount training labeled data is limited. We report performance in terms of accuracy in each setting, denoted `[dataset] (Xw-Ys) (@ Z)`, where `X` is the number of ways, `Y` is the number of shots, and `Z` is the labeling budget.[5] For each labeling budget, we report the accuracy on the test tasks for the method with hyperparameters selected based on the accuracy on the validation tasks. Each of our experiments was repeated 3 times with different random seeds.

Reproducibility. To support reproducibility, we have developed a software framework that allows users benchmark arbitrary meta-learning methods under different supervision tradeoffs. The code and configurations for all our experiments will be released.

All additional details on the setup are in Appendix C.

6.2 Understanding Supervision Tradeoffs

To validate the implications of our theory (**O1-3**), we conduct the following study. First, we analyze the differences in behavior of meta-learning methods when the number of training tasks is limited. Next, we analyze how the tradeoff between the number of tasks and the number of data points per task affect performance of different methods. Finally, we discuss the effect of active data labeling on meta-learning.

Classical evaluation vs. limited supervision. We ask whether different meta-learning methods behave differently when trained and evaluated in the classical vs. limited supervision regime. To answer this question, we trained MAML and ProtoNets on 5-way, 1-shot and 5-shot Omniglot benchmark under different limits on the number of training tasks (ranging between 1-500K). Fig. 1 shows that the performance of

[5]The `Z` is not specified for the classical training regime.

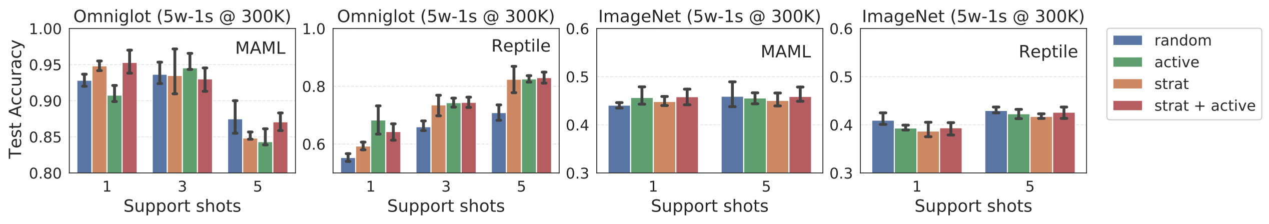

Figure 2: Test accuracy of MAML and Reptile trained on Omniglot and *mini*-ImageNet under a limit on the available supervision (the total labeling budget was fixed at 300K) as a function of the number of support shots. MAML and Reptile exhibit visibly different behaviors. Error bars indicate 95% CI (based on 3 runs with different random seeds).

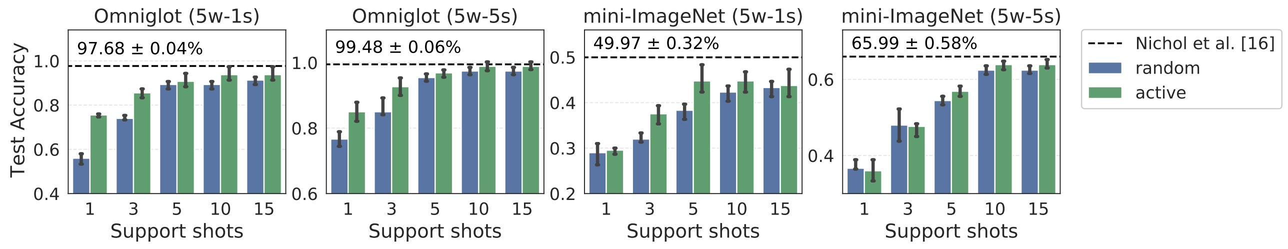

Figure 3: Test accuracy of Reptile meta-trained for 100K steps without controlling for the number of training tasks (*i.e.*, classical evaluation regime) as a function of the support set size of the training tasks. The dashed line corresponds to test accuracy of Reptile reported by Nichol et al. [16] (trained in the same regime with 10 support shots and data stratification by class). Error bars indicate 95% CI (based on 3 runs with different random seeds).

MAML and ProtoNets is virtually indistinguishable in the classical evaluation regime.[6] However, once the number of training tasks is limited, we observe quite a distinct behavior—MAML works better when trained on 1-shot tasks, while ProtoNets dominate the 5-shot benchmark. Moreover, as the number of tasks increases ($n \to \infty$ in the limit), the generalization error reduces at a similar rate for both methods and the gap between them shrinks, as suggested by our theory.

Exploring tradeoffs under limited supervision. In our next set of experiments, given a fixed labeling budget, we would like to find out how different ways of allocating supervision across tasks affects performance. To this end, we fixed the labeling budget at 300K, and constructed meta-datasets based on Omniglot and *mini*-ImageNet that satisfied the limit on the labeling budget. Each meta-dataset consisted of 5-way tasks with 1-shot, 3-shot, or 5-shot support sets that were selected using different labeling strategies. Due to fixed labeling budget, settings with larger support sets had fewer training tasks. In addition to selecting labels uniformly at random and actively, we also conducted experiments with selection stratified by class.[7] The results for MAML and Reptile are presented in Fig. 2.

First, we notices that MAML is able to attain much better performance when trained on more tasks with fewer labels per task, suggesting that the number of tasks is indeed the dominant factor that determines how well the method generalizes. Interestingly, Reptile exhibits quite the opposite behavior and yields strictly better performance when trained on overall fewer tasks with more data points each. This is consistent with the meta-generalization error bounds given in Theorem 3 which has an additional $O(1/m)$ term for Reptile that turns out to be dominant in this particular case.

Second, active selection of labeled data is beneficial for both methods, although Reptile benefits from it more. Theoretically, we hypothesise (and conjecture) that the effect might be due to improved constants in each terms of the meta-generalization bounds (*i.e.*, due to "better-behaved" inner- and outer-loop objectives, when the labeled points are selected to minimize the uncertainty of the adapted model). Note that the effects of active label selection and more data per task on Reptile are also visible in the classical evaluation regime (see Fig. 3, where the number of training tasks was not controlled).

Taken together, our observations suggest that there are multiple factors that significantly affect performance of meta-learning in *different ways*, when the availability of training data is limited. While some of the effects of limited supervision can be reasonably explained by the stability theory, from a practical point of view, having benchmarks that can capture such effects is essential.

[6]When the number of training tasks is not limited, 10M+ unique tasks are typically generated during training.

[7]Stratification ensured that the selected support sets where class-balanced. Although typical in meta-learning, such an approach requires knowing all labels in the support set *a priori*, narrowing the scope of possible applications.

Table 1: Results on the suite of bounded supervision benchmarks for Reptile, MAML, and ProtoNets trained using random (R) or active (A) selection of the labeled support. Each row in the table corresponds to a benchmark.

Benchmark			Reptile		MAML		Proto	
	Dataset	Budget	R	A	R	A	R	A
Omniglot	5w-1s	30K	73.3	77.2	83.3	79.6	**84.8**	84.5
		300K	76.8	80.1	91.8	93.6	92.2	**94.3**
	5w-5s	30K	89.3	88.0	91.5	90.2	95.1	**96.8**
		300K	92.4	93.0	96.3	96.9	97.0	**98.1**
	20w-1s	60K	68.4	67.9	79.0	79.8	84.9	**85.4**
		600K	76.3	77.8	84.8	83.9	86.1	**87.9**
	20w-5s	60K	89.8	92.4	93.3	94.1	95.2	**96.0**
		600K	92.0	94.1	95.5	95.9	**96.8**	**96.8**
ImageNet	5w-1s	30K	38.5	39.4	**44.2**	44.0	42.9	40.7
		300K	42.0	41.9	**45.4**	45.2	39.9	41.0
	5w-5s	30K	55.4	55.2	56.8	**57.1**	54.0	53.4
		300K	56.7	59.4	60.8	**61.0**	57.4	55.8
EMNIST	62w-1s	10K	69.6	67.9	73.9	70.7	**74.0**	73.3
		50K	77.5	75.9	**80.3**	79.1	78.2	78.0
		100K	82.0	84.5	86.7	87.7	88.8	**90.1**
	62w-5s	10K	78.1	**80.1**	68.3	75.0	71.2	72.9
		50K	84.3	**85.8**	76.0	79.4	77.1	76.8
		100K	87.1	**89.0**	82.5	86.3	86.1	88.5

Table 2: FedAvg vs. Reptile on EMNIST with random or active labeling. The labeling budget was fixed to 100K. Error bars indicate 95% CI (3 runs, different random seeds).

	FedAvg		Reptile	
# shots	Random	Active	Random	Active
1	77.1 ± 2.0	77.8 ± 2.1	82.2 ± 1.6	**84.2**±1.9
5	80.7 ± 2.4	79.6 ± 1.8	87.0 ± 2.1	**89.1**±1.5

6.3 Benchmarking @ Fixed Labeling Budgets

There are many tradeoffs that we need to consider and balance when selecting a meta-learning algorithm for a setting with limited labeling budget. To enable fair and reproducible comparison of meta-learning methods in different regimes, we introduce a suite of new benchmarks that test performance at different fixed labeling budgets. Our benchmarks are compatible with both few-shot learning and federated learning datasets.

Table 1 presents results for each pair of meta-learning method and labeling strategy (random vs. active). For each pair, we selected the best performance across settings with 1, 3, and 5 support shots at training time, and report results on both 1-shot and 5-shot test tasks.

Omniglot and *mini*-ImageNet. Comparing random versus active data labeling, we observe that active selection almost always consistently improves performance. Overall, ProtoNets dominate other methods on Omniglot; MAML does better on *mini*-ImageNet. Reptile performs significantly worse than other methods when a strict limit on the labeled data is enforced.

Federated EMNIST. We observe overall similar trends for all methods evaluated in our personalized federated learning setting. Interestingly, however, Reptile (with active data selection) tends to dominate other methods on 5-shot version of this benchmark. This is likely due to the fact that the query sets in EMNIST larger than for Omniglot and *mini*-ImageNet, which benefits Reptile since it does not distinguish between support and query sets and uses all available labeled data for inner loop updates.

For completeness purposes, we also compare Reptile and FedAvg [28] (the most popular FL algorithm used in many practical settings) and present results in Table 2. These algorithms differ only in whether or not they fine-tune the model on the support data at test time. Reptile uses fine-tuning and clearly outperforms FedAvg on our personalized FL benchmarks.

7 Conclusion

Motivated by use-cases of meta-learning for personalization in federated learning, we have analyzed theoretically and experimentally the data efficiency of two major families of modern meta-learning algorithms.

Using stability theory, we derived bounds on the transfer risk (or meta-generalization error). Our bounds revealed that: 1) Reptile and other methods designed to meta-learn by optimizing the empirical estimator of the transfer risk do not work well unless each training task contains a sufficient number of labeled points; 2) ProtoNets, MAML, and other methods designed to meta-learn by optimizing held out set losses can effectively learn from data-scarce tasks but require a large number of such tasks to meta-generalize.

Further, through multiple experiments, we confirmed predictions of our theory as well as studied behavior of popular meta-learning algorithms under different supervision tradeoffs that have important practical implications. To that end, we introduced a new approach to benchmarking meta-learning methods in the limited supervision regime, which is compatible with arbitrary few-shot and federated learning datasets.

Finally, we conjectured that selecting labeled support sets at meta-training actively can improve performance of meta-learning methods in the limited supervision regime. To test that hypothesis, we proposed active meta-learning—a simple approach that incorporated active labeling into the inner-loop. Our method turned out to be quite effective, leading to improved performance of multiple methods under limited supervision.

We hope that this study will further accelerate research and enable wider adoption of meta-learning in personalized federated learning and other practical settings.

Acknowledgments

The authors thank Mikhail Khodak, Harri Edwards, Afshin Rostamizadeh, Jenny Gillenwater for helpful comments on early versions of this manuscript. This work was supported in part by DARPA FA875017C0141, the National Science Foundation grants IIS1705121 and IIS1838017, an Amazon Web Services Award, a JP Morgan A.I. Research Faculty Award, a Carnegie Bosch Institute Research Award, a Facebook Faculty Research Award, and a Block Center Grant. MA was supported by Google PhD Fellowship. LL contributed to this work while at CMU. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA, the National Science Foundation, or any other funding agency.

References

[1] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche opérationnelle, 1990.

[2] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Learning*, 4(1), 1992.

[3] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

[4] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[5] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[6] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[7] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*, 2018.

[8] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems 32*, 2019.

[9] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[10] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.

[11] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[12] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

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

[14] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1126–1135, 2017.

[16] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR, abs/1803.02999*, 2, 2018.

[17] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[18] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[19] Andreas Maurer. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 6(Jun):967–994, 2005.

[20] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

[21] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[22] Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, 2019.

[23] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, 2019.

[24] Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In *International Conference on Machine Learning*, 2019.

[25] Giulia Denevi, Dimitris Stamos, Carlo Ciliberto, and Massimiliano Pontil. Online-within-online meta-learning. In *Advances in Neural Information Processing Systems*, 2019.

[26] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, 2018.

[27] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092, 2020.

[28] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[29] Zachary Charles and Jakub Konečný. On the outsized importance of learning rates in local update methods. *arXiv preprint arXiv:2007.00878*, 2020.

[30] Mark Woodward and Chelsea Finn. Active one-shot learning. *arXiv preprint arXiv:1702.06559*, 2017.

[31] Rinu Boney and Alexander Ilin. Semi-supervised few-shot learning with prototypical networks. *arXiv preprint arXiv:1711.10856*, 2017.

[32] Philip Bachman, Alessandro Sordoni, and Adam Trischler. Learning algorithms for active learning. In *International Conference on Machine Learning*, pages 301–310, 2017.

[33] Sachin Ravi and Hugo Larochelle. Meta-learning for batch mode active learning, 2018. URL https://openreview.net/forum?id=r1PsGFJPz.

[34] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[35] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12: 149–198, 2000.

[36] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in non-stationary and competitive environments. In *International Conference on Learning Represenations*, 2018.

[37] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

[38] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1690–1699, 2018.

[39] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

[40] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[41] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[42] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702, 2019.

[43] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017.

[44] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[45] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[46] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[47] Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. In *Advances in neural information processing systems*, pages 892–900, 2010.

[48] Fedor Zhdanov. Diverse mini-batch active learning. *arXiv preprint arXiv:1901.05954*, 2019.

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Reptile Optimizes $\mathcal{L}_{\text{emp}}(A; \mathbb{S})$

The Reptile meta-learning algorithm [16] is defined as follows. Given a model f_θ parametrized by θ, it defines the inner loop algorithm A_{θ_0} as a T-step stochastic gradient optimization:

$$A_{\theta_0}(S_i) := \theta_T^i, \quad \text{where} \quad \theta_{t+1}^i := \theta_t^i - \alpha_t \nabla_{\theta_t^i}\left(\frac{1}{|S_i|}\sum_{(x,y)\in S_i} \ell(f_{\theta_t}(x), y)\right) \tag{15}$$

In the outer loop, it updates θ_0, which is shared across all tasks, as follows:

$$\theta_0 \leftarrow \theta_0 - \varepsilon \frac{1}{n}\sum_{i=1}^{n}(\theta_0 - A_{\theta_0}(S_i)), \varepsilon > 0 \tag{16}$$

We argue that Reptile outer loop updates (16) approximate gradient descent on the empirical estimator of the transfer risk:

$$\mathcal{L}_{\text{emp}}(A_{\theta_0}; \mathbb{S}) := \frac{1}{n}\sum_{i=1}^{n} \hat{R}(A_{\theta_0}, S_i), \quad \hat{R}(A_{\theta_0}, S_i) := \frac{1}{|S_i|}\sum_{(x,y)\in S_i} \ell(A_{\theta_0}(S_i)(x), y) \tag{17}$$

To understand why this is the case, first, consider the gradient of $\mathcal{L}_{\text{emp}}(A_{\theta_0}; \mathbb{S})$ with respect to θ_0:

$$\nabla_{\theta_0}\mathcal{L}_{\text{emp}}(A_{\theta_0}; \mathbb{S}) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\theta_0}\hat{R}(A_{\theta_0}, S_i) \tag{18}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{|S_i|}\sum_{(x,y)\in S_i} \nabla_{\theta_0}\ell(A_{\theta_0}(S_i)(x), y) \tag{19}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{|S_i|}\sum_{(x,y)\in S_i} \nabla_{\theta_T^i}\ell(f_{\theta_T^i}(x), y)[\nabla_{\theta_0}A_{\theta_0}(S_i)] \tag{20}$$

Now, we can compute the difference between $\nabla_{\theta_0}\hat{R}(A_{\theta_0}, S_i)$ and the Reptile update $A_{\theta_0}(S_i) - \theta_0$:

$$(\theta_0 - A_{\theta_0}(S_i)) - \nabla_{\theta_0}\hat{R}(A_{\theta_0}, S_i) = \frac{1}{|S_i|}\sum_{(x,y)\in S_i}\left[\left(\sum_{t=1}^{T}\alpha_t \nabla_{\theta_t^i}\ell(f_{\theta_t^i}(x), y)\right) - \nabla_{\theta_T^i}\ell(f_{\theta_T^i}(x), y)[\nabla_{\theta_0}A_{\theta_0}(S_i)]\right] \tag{21}$$

Expression in the square brackets is the difference between T inner loop gradient steps on $\ell(f_\theta(x), y)$ and the gradient at the final T-th step transformed by the Jacobian $\nabla_{\theta_0}A_{\theta_0}(S_i)$. This expression was analyzed by Nichol et al. [16] using perturbation theory and Taylor approximation, where it was shown that this difference is equal to the following:

$$\left(\sum_{t=1}^{T}\alpha_t \nabla_{\theta_t^i}\ell(f_{\theta_t^i}(x), y)\right) - \nabla_{\theta_T^i}\ell(f_{\theta_T^i}(x), y)[\nabla_{\theta_0}A_{\theta_0}(S_i)] = (I - \alpha H_{\theta_0}^T)\sum_{t=1}^{T-1}\nabla_{\theta_t^i}\ell(f_{\theta_t^i}(x), y) + O(\alpha^2) \tag{22}$$

where $H_{\theta_0}^T$ is the Hessian of $\ell(f_{\theta_T}(x), y)$ at θ_0, $\alpha := \max_{t\in[1,T]} \alpha_t$.

Assuming that α (*i.e.*, the inner loop step size) is sufficiently small and the norm of $\nabla_\theta \ell(f_\theta(x), y)$ is bounded by some constant G, the difference in (22) is bounded by $(1 - \alpha\lambda_{\max}(H_{\theta_0}^T))G(T-1) + O(\alpha^2)$, which implies:

$$(\theta_0 - A_{\theta_0}(S_i)) - \nabla_{\theta_0}\hat{R}(A_{\theta_0}, S_i) \leq (1 - \alpha\lambda_{\max}(H_{\theta_0}^T))G(T-1) + O(\alpha^2) \tag{23}$$

The deviation between $\nabla_{\theta_0}\mathcal{L}_{\text{emp}}$ and Reptile updates would be small when the inner loop objective is well behaved (has a small G), the number of inner loops steps T is not too large and the step sizes α are small, which would ensure convergence of Reptile to a stationary point of $\mathcal{L}_{\text{emp}}$. We leave sharper analysis of the convergence rates in the case of non-convex and convex $\ell(\cdot, \cdot)$ to future work.

B Proofs

In this section, we provide detailed expressions for meta-generalization bounds, proofs for Theorems 2 and 3, statements (and proof sketches where necessary) for classical auxiliary results, and further discuss the implications and limitations of our analysis.

B.1 Classical Bounds on Meta-generalization Error

The meta-generalization bounds provided in Section 4 directly extend of the following classical result by Maurer [19] (which in turn uses meta-learning formulation of Baxter [35] and is a direct adaptation of the algorithmic stability bounds of Bousquet and Elisseeff [18]).

Theorem 4 (Theorem 1 from [19]) *Let the meta-algorithm $\mathbb{A}$ satisfy the following two conditions:*

C1. For every pair of meta-samples $\mathbb{S} = \{S_1, \dots, S_n\}$, $\mathbb{S}^{-i} := \mathbb{S} \backslash \{S_i\}$, and for any sample S, we have $|\hat{R}(\mathbb{A}(\mathbb{S}), S) - \hat{R}(\mathbb{A}(\mathbb{S}^{-i}), S)| \leq \beta'$.

C2. For any pair of samples $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$, $S^{-j} := S \setminus \{(x_j, y_j)\}$, any algorithm A produced by $\mathbb{A}$, and any (x, y), we have $|\ell(A(S)(x), y) - \ell(A(S^{-j})(x), y)| \leq \beta$.

Then for any task distribution $\mathbb{P}(\mathcal{T})$, with probability at least $1 - \delta$ the following inequality holds:

$$\mathcal{R}(\mathbb{A}(\mathbb{S}), \mathbb{P}(\mathcal{T})) - \mathcal{L}_{\text{emp}}(\mathbb{A}(\mathbb{S}); \mathbb{S}) \leq 2\beta' + (4n\beta' + M)\sqrt{\frac{\ln(1/\delta)}{2n}} + 2\beta, \tag{24}$$

where $\mathcal{L}_{\text{emp}}(\mathbb{A}(\mathbb{S}); \mathbb{S}) := \frac{1}{n}\sum_{i=1}^{n} \hat{R}(\mathbb{A}(\mathbb{S}), S_i)$, $\hat{R}(A, S_i) := \frac{1}{|S_i|}\sum_{(x,y)\in S_i} \ell(A(S_i)(x), y)$ with the loss function $\ell(\cdot, \cdot)$ bounded by M.

Conditions C1 and C2 in Theorem 4 define uniform stability (*i.e.*, sensitivity of the algorithm to removal of an arbitrary point from the training sample [18]) and state that the bound holds if the meta-algorithm $\mathbb{A}$ and every algorithm A it produces are uniformly β'- and β-stable with respect to the empirical risk $\hat{R}$ and a loss function ℓ, respectively. The bound becomes non-trivial when $\beta' = o(1/n^a), a \geq 1/2$ and $\beta = o(1/m^b), b \geq 0$.

Theorem 4 provides a bound on the difference between the transfer risk $\mathcal{R}[\mathbb{A}(\mathbb{S}), \mathbb{P}(\mathcal{T})]$ and its empirical estimator $\mathcal{L}_{\text{emp}}(\mathbb{A}(\mathbb{S}); \mathbb{S})$ based on meta-sample $\mathbb{S}$, implying that a small $\mathcal{L}_{\text{emp}}(\mathbb{A}(\mathbb{S}); \mathbb{S})$ guarantees meta-generalization within the bound. Denoting $A \equiv \mathbb{A}(\mathbb{S})$ to simplify our notation, the bound is obtained as follows:

$$\mathcal{R}(A, \mathbb{P}(\mathcal{T})) - \mathcal{L}_{\text{emp}}(A; \mathbb{S}) = \mathbb{E}_{\mathcal{D}\sim\mathbb{P}(\mathcal{T})}\left[\mathbb{E}_{S\sim\mathcal{D}^m}\left[\hat{R}(A, S)\right]\right] - \frac{1}{n}\sum_{i=1}^{n}\hat{R}(A, S_i) + \tag{25}$$

$$\mathbb{E}_{\mathcal{D}\sim\mathbb{P}(\mathcal{T})}\left[\mathbb{E}_{S\sim\mathcal{D}^m}\left[R(A(S), \mathcal{D}) - \hat{R}(A, S)\right]\right] \tag{26}$$

The term (25) is the difference between the expected empirical risk over the true distribution of tasks and its estimate $\mathcal{L}_{\text{emp}}(A; \mathbb{S})$ based on the meta-sample $\mathbb{S}$. As long as $\mathbb{A}$ is β'-uniformly stable with respect to $\hat{R}(A, S)$ (C1, Theorem 4), this term is bounded by $2\beta' + (4n\beta' + M)\sqrt{\ln(1/\delta)/2n}$, which follows directly from the classical result of Bousquet and Elisseeff [18].

The term (26) is the estimation error of a model $f(\cdot) = A(S)$ learned by A from S with respect to the data distribution $\mathcal{D}$, computed in expectation over the distribution of tasks $\mathbb{P}(\mathcal{T})$. Stability of the inner-loop (C2, Theorem 4) directly implies a bound of 2β on this term (see Theorem 6 in [19]). Putting together bounds of terms (25) and (26), we arrive at (24).

B.2 Bounding Meta-generalization of Reptile, MAML, and ProtoNets

The bound given in (24) is on the generalization error, *i.e.*, the deviation of the true transfer risk $\mathcal{R}$ from the empirical estimator $\mathcal{L}_{\text{emp}}$, and has meaningful practical implications only when the meta-algorithm $\mathbb{A}$ minimizes $\mathcal{L}_{\text{emp}}$. As we have shown in A, $\mathcal{L}_{\text{emp}}(A; \mathbb{S})$ is the meta-training objective function optimized by Reptile, and thus the bound from Theorem 4 applies directly. However, MAML and ProtoNets optimize $\mathcal{L}_Q(A; \mathbb{S})$, so we have to bound $\mathcal{R}(A,\mathbb{P}) - \mathcal{L}_Q(A;\mathbb{S})$ instead, which can be decomposed into two terms similar to (25) and (26), where $\hat{R}$ is replaced by $\hat{R}_Q$ and S is replaced by $S \setminus Q$ (since samples from the query set Q are not used in the inner-loop). The bound on the first term will not change much as we can still directly apply results from stability theory with the only caveat that we would require β'_Q-uniform stability of the meta-algorithm with respect to $\hat{R}_Q$. The second term, however, vanishes:

$$\begin{aligned}&\mathbb{E}_{S\sim\mathcal{D}^m}\left[R(A(S\setminus Q),\mathcal{D}) - \hat{R}_Q(A,S)\right]\\ &= \mathbb{E}_{S\setminus Q\sim\mathcal{D}^{m-k}}\left[R(A(S\setminus Q),\mathcal{D}) - \mathbb{E}_{Q\sim\mathcal{D}^{m-k}}\left[\frac{1}{|Q|}\sum_{(x,y)\in Q}\ell(A(S\setminus Q)(x),y)\right]\right] \equiv 0 \end{aligned} \tag{27}$$

This allows us to reformulate Theorem 4 and obtain the following generalization bound applicable to any meta-learning method that optimizes $\hat{R}_Q$ in the outer loop, including MAML and ProtoNets.

Theorem 5 *Let the meta-algorithm $\mathbb{A}$ satisfy the following two conditions:*

C1. For every pair of meta-samples $\mathbb{S} = \{S_1,\dots,S_n\}$, $\mathbb{S}^{-i} := \mathbb{S}\setminus\{S_i\}$, and for any sample S, we have $|\hat{R}_Q(\mathbb{A}(\mathbb{S}),S) - \hat{R}_Q(\mathbb{A}(\mathbb{S}^{-i}),S)| \le \beta'_Q$.

C2. For any pair of samples $S = \{(x_1,y_1),\dots,(x_m,y_m)\}$, $S^{-j} := S\setminus\{(x_j,y_j)\}$, any algorithm A produced by $\mathbb{A}$, and any (x,y), we have $|\ell(A(S)(x),y) - \ell(A(S^{-j})(x),y)| \le \beta$.

Then for any task distribution $\mathbb{P}(\mathcal{T})$, with probability at least $1-\delta$ the following inequality holds:

$$\mathcal{R}(\mathbb{A}(\mathbb{S}),\mathbb{P}) - \mathcal{L}_Q(\mathbb{A}(\mathbb{S});\mathbb{S}) \le 2\beta'_Q + (4n\beta'_Q + M)\sqrt{\frac{\ln(1/\delta)}{2n}}, \tag{28}$$

where $\mathcal{L}_Q(\mathbb{A}(\mathbb{S});\mathbb{S}) := \frac{1}{n}\sum_{i=1}^n \hat{R}_Q(\mathbb{A}(\mathbb{S}),S_i)$, $\hat{R}_Q(A,S_i) := \frac{1}{|Q_i|}\sum_{(x,y)\in Q_i}\ell(A(S_i\ Q_i)(x),y)$ with the loss function $\ell(\cdot,\cdot)$ bounded by M.

Since MAML, Reptile, and ProtoNets use stochastic gradient method (SGM) for solving the outer loop optimization problem, and Reptile additionally uses SGM in the inner loop as well, we further adopt the following general result from stability theory of SGM due to Hardt et al. [20].

Lemma 6 (Theorem 3.12 in [20]) *Let $\ell(\cdot,z)\in[0,1]$ be L-Lipschitz and γ-smooth loss function for every z. Suppose that we optimize $\frac{1}{n}\sum_{i=1}^n \ell(\theta,z_i)$ by running SGM for T steps with monotonically non-increasing step sizes $\alpha_t \le c/t$. Then, SGM is β-uniformly stable with*

$$\beta \le \frac{1+1/(\gamma c)}{n-1}(2cL^2)^{1/(\gamma c+1)}T^{1-1/(\gamma c+1)} \tag{29}$$

Combining Theorems 4, 5, and 6 we finally arrive at the meta-generalization error bounds for modern meta-learning algorithms.

Theorem 7 *Let the meta-algorithm $\mathbb{A}$ be an SGM that optimizes an L'-Lipschitz and γ'-smooth loss $\mathcal{L}(A;\mathbb{S})$ by taking T' steps with non-increasing step sizes $\alpha'_t \le c'/t$. With probability at least $1-\delta$, we have the following:*

1. *If $\mathcal{L}(A;\mathbb{S})$ is Q-estimator of the transfer risk, then the following bound holds:*

$$\mathcal{R}[A,\mathbb{P}(\mathcal{T})] - \mathcal{L}(A;\mathbb{S}) \le B'(n,T',L',\gamma',c') \approx O\left(L'^2T'\sqrt{\frac{\ln(1/\delta)}{n}}\right) \tag{30}$$

2. *If $\mathcal{L}(A;\mathbb{S})$ is the empirical estimator of the transfer risk and the inner loop learning algorithm A is an SGM that optimizes L-Lipschitz and γ-smooth loss $\ell(f(x),y)$ by taking T steps with non-increasing step sizes $\alpha_t \le c/t$, then:*

$$\mathcal{R}[A,\mathbb{P}(\mathcal{T})] - \mathcal{L}(A;\mathbb{S}) \le B'(n,T',L',\gamma',c') + B(m,T,L,\gamma,c) \approx O\left(L'^2T'\sqrt{\frac{\ln(1/\delta)}{n}} + L^2T\frac{1}{m}\right) \tag{31}$$

Proof Conditions of the theorem and Lemma 6 imply that $\mathbb{A}$ is β'-(or β'_Q-)uniformly stable and the coefficient can be expressed through the Lipschitz and smoothness constants of $\mathcal{L}_{\text{emp}}$ (or $\mathcal{L}_Q$). This leads to the following expression for $B'(n, T', L', \gamma', c')$:

$$B'(n, T', L', \gamma', c') = \frac{2C}{n}\left(1 + \frac{1}{n-1}\right) + 2C\sqrt{\frac{2\ln(1/\delta)}{n}}\left(1 + \frac{1}{n-1} + \frac{M}{4C}\right), \tag{32}$$

where $C := (1 + 1/(\gamma' c'))(2c'L'^2)^{1/(\gamma' c'+1)}T'^{1-1/(\gamma' c'+1)}$. The simplified expression given in (30) upper-bounds (32). Similarly, if each algorithm A produced by the meta-algorithm $\mathbb{A}$ is an SGM on the $\mathcal{L}_{\text{emp}}$ objective, using Lemma 6 we arrive at the following expression for $B(m, T, L, \gamma, c)$:

$$B(m, T, L, \gamma, c) = 2\beta \le 2\frac{1 + 1/(\gamma c)}{m-1}(2cL^2)^{1/(\gamma c+1)}T^{1-1/(\gamma c+1)} \approx O\left(L^2 T \frac{1}{m}\right) \tag{33}$$

where the approximation ignores terms associated with c and γ. The statement of the theorem now follows from Theorems 4 and 5 and the derived expressions. ■

Besides the implications of our theory discussed in the main text, we can make a few more interesting observations.

What happens if we use empirical estimator of the transfer risk as the objective for MAML? In principle, we can make MAML optimize $\mathcal{L}_{\text{emp}}$ instead of $\mathcal{L}_Q$ in the outer loop. Nichol et al. [Section 6.3, 16] considered an interesting setup in their ablation study, where they analyzed how the overlap between the support and query data affects performance of the the first-order version of MAML. Note that the larger the overlap, the closer MAML's objective becomes to $\mathcal{L}_{\text{emp}}$. Interestingly, they show that larger overlaps lead to the performance degradation on the Omniglot dataset. This result is consistent with our theory—switching MAML's objective to $\mathcal{L}_{\text{emp}}$ necessarily leads to larger meta-generalization error characterized by the additional 2β term in the bound.

Implications for federated learning. In federated learning research, one of the most popular algorithms is federated averaging (FedAvg) [28], which uses model updates that are mathematically equivalent to Reptile. The tasks are defined by the (private) datasets available on different client devices (*e.g.*, mobile phones). Our theory suggests that federated-averaging-style updates might be suboptimal for applications where the available labeled data for each client is very small; at the same time, when each client has sufficient data (as in the EMNIST dataset), we observe empirically superiority of Reptile/FedAvg over MAML (Section 6.3). Designing personalized federated learning algorithms that learn by optimizing a combination of $\mathcal{L}_{\text{emp}}$ (on clients with a lot of data) and $\mathcal{L}_Q$ (on clients with very small datasets) objectives is an interesting research avenue to explore next.

C Details on the Experimental Setup

We provide details on the experimental setup used throughout the paper, including model architectures (often termed *backbone networks* in the few-shot learning literature) and hyperparameters for meta-learning methods. Additionally, our full experimental configurations can be found in the provided supplementary code in the corresponding `conf/` folders, which enables full reproducibility.

C.1 Network Architectures

For all our experiments, we used the standard Conv4 backbone network architectures proposed in the original papers [15, 17, 16]. The embeddings computed by the last hidden layer of the backbone networks were subsequently used for clustering in our active sampling approach. MAML and Reptile used a linear final layer to compute logits from the embeddings, while ProtoNets used the distances between the query and support samples in the embedding space for computing class probabilities.

Omniglot and EMNIST. Input images were resized to 28×28. Models used by all methods consisted of 4 convolutional layers with 64 filters, kernel size of 3, and strides of 2, followed by batch normalization and ReLU activations (with no pooling or dropout in the intermediate layers).

***mini*-ImageNet.** Input images were resized to 84×84. Models used by all methods consisted of 4 convolutional layers with 32 filters, kernel size of 3, and strides of 2, followed by batch normalization and ReLU activations (with no pooling or dropout in the intermediate layers).

Table 3: Meta-test performance with unbounded supervision.

Method	O-5w-1s	O-5w-5s	O-20w-1s	O-20w-5s	MI-5w1d	MI-5w5d
MAML	98.3±0.6	99.9±0.1	95.0±0.5	98.6±0.5	48.7±1.7	63.0±0.9
Reptile	94.9±0.2	98.2±0.5	88.2±0.4	96.4±0.4	47.8±1.3	61.9±1.1
Protonets	97.9±0.4	99.0±0.1	91.9±1.2	98.6±0.5	48.3±0.8	66.2±0.8

C.2 Meta-learning Algorithms

Meta-training (*i.e.*, the outer loop optimization) was performed using Adam optimizer [49] with learning rate of 0.005 and $\beta_1 = 0$ for MAML and ProtoNets. For Reptile, following parameters provided by Nichol et al. [16], the outer loop learning rate was set 1.0 and the optimizer set to stochastic gradient descent (SGD). Details on model adaptation are provided below.

MAML [15]. At training time, we used 5 inner loop gradient descent (GD) steps with a learning rate of 0.01. At evaluation time, the number of inner loop steps was set to 10. To implement first order adaptation updates, we nullified the second order derivatives when computing the meta-training loss.

Reptile [16]. At training time, we used 10 inner loop gradient descent (GD) steps with a learning rate of 0.001 for Omniglot and 0.0005 for *mini*-ImageNet. At evaluation time, the number of inner loop steps was set to 50.

Prototypical Networks [17]. We used a version of the method with the Euclidean distance. The method has no other hyperparameters besides those of the outer loop optimizer.

C.3 Calibration

We selected the hyperparameters described above such that the meta-test performance of all methods nearly matched the reported numbers in the original papers in the limited supervision regime. Results for the calibrated models are reported in Table 3.

A note on Reptile. Nichol et al. [16] used 10-shot tasks at meta-training time and trained for over 100,000 meta-updates (each meta-update was computed on a batch of 20 tasks) in order to attain the performance reported in the original paper. In the limited supervision setting, this would have required a label budget of over 100M (*i.e.*, 1000 times larger than those considered in our study). However, just for calibration purposes, we matched the original setup of Nichol et al. [16].

A note on ProtoNets. To improve performance, Snell et al. [17] proposed to meta-train ProtoNets on tasks with higher number of classes than the tasks used at meta-test time (*e.g.*, meta-training on 60-way tasks while meta-testing on 20-way tasks). Even though training tasks with more classes could be helpful in learning better data representations, increasing the number of classes per task affects the amount of labeled points required per task and may affect performance of non-oracle label selection strategies. Therefore, in our experiments, we decided to stick with a clean setup that matches the number of classes per task at both meta-training and meta-test times, although sacrificing some performance gains. Again, for calibration purposes only, we used an increased number of classes per task at meta-training time.

C.4 Limitations

To avoid a combinatorially large number of combinations of architectures, algorithms, and their hyperparameters, we had to fix many of these variables before experimenting with different labeling budgets and sampling strategies. While this allowed us to conduct a fairly comprehensive study of 3 different meta-learning methods across a variety of regimes, the reported results may be limited to the specific choice of the setup described above; we do not exclude the possibility that the behavior of different methods might vary with the setup (*e.g.*, tuning hyperparameters for each labeling budget separately, while extremely costly, might have rectified poor performance of some of the methods on some of the benchmarks).