[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. These conditions 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 $\mathbb{S} = \{S_1, \ldots, S_n\}$, $\mathbb{S}^{-i} := \mathbb{S} \backslash \{S_i\}$, 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 $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, $S^{-j} := S \setminus \{(x_j, y_j)\}$, 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:

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

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 $B(m, T, L, \gamma, c)$:

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

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).