Generalization Bounds for Meta-Learning via PAC-Bayes and Uniform Stability

Alec Farid **Anirudha Majumdar**
Department of Mechanical and Aerospace Engineering, Princeton University
{afarid, ani.majumdar}@princeton.edu

Abstract

We are motivated by the problem of providing strong generalization guarantees in the context of meta-learning. Existing generalization bounds are either challenging to evaluate or provide vacuous guarantees in even relatively simple settings. We derive a probably approximately correct (PAC) bound for gradient-based meta-learning using two different generalization frameworks in order to deal with the qualitatively different challenges of generalization at the "base" and "meta" levels. We employ bounds for uniformly stable algorithms at the base level and bounds from the PAC-Bayes framework at the meta level. The result of this approach is a novel PAC bound that is tighter when the base learner adapts quickly, which is precisely the goal of meta-learning. We show that our bound provides a tighter guarantee than other bounds on a toy non-convex problem on the unit sphere and a text-based classification example. We also present a practical regularization scheme motivated by the bound in settings where the bound is loose and demonstrate improved performance over baseline techniques.

1 Introduction

A major challenge with current machine learning systems is the need to acquire large amounts of training data in order to learn a new task. Over the past few decades, meta-learning [62, 70] has emerged as a promising avenue for addressing this challenge. Meta-learning relies on the intuition that a new task often bears significant similarity to previous tasks; hence, a learner can learn to perform a new task very quickly by exploiting data from previously-encountered related tasks. The meta-learning problem formulation thus assumes access to datasets from a variety of tasks during meta-training. The goal of the meta learner is then to learn inductive biases from these tasks in order to train a base learner to achieve few-shot generalization on a new task.

Over the past few years, there has been tremendous progress in practical algorithms for meta-learning (see, e.g., [61, 55, 25, 32]). Techniques such as model-agnostic meta-learning (MAML) [25] have demonstrated the ability to perform few-shot learning in a variety of supervised learning and reinforcement learning domains. However, our theoretical understanding of these techniques lags significantly behind successes on the empirical front. In particular, the problem of deriving *generalization bounds* for meta-learning techniques remains an outstanding challenge. Current methods for obtaining generalization guarantees for meta-learning [5, 34, 77] either (i) produce bounds that are extremely challenging to compute or (ii) produce vacuous or near-vacuous bounds in even highly simplified settings (see Section 5 for numerical examples). Indeed, we note that existing work on generalization theory for meta-learning techniques do not explicitly report numerical values for generalization bounds. This is in contrast to the state of generalization theory in the supervised learning setting, where recent techniques demonstrate the ability to obtain non-vacuous generalization guarantees on benchmark problems (e.g. visual classification problems [24, 78, 54]).

The generalization challenge in meta-learning is similar to, but distinct from, the supervised learning case. In particular, any generalization bound for meta-learning must account for *two levels* of

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

generalization. First, one must account for generalization at the base level, i.e., the ability of the base learner to perform well on new data from a given task. This is particularly important in the few-shot learning setting. Second, one must account for generalization at the meta level, i.e., the ability of the meta learner to generalize to new tasks not encountered during meta-training. Moreover, the generalization performance at the two levels is coupled since the meta learner is responsible for learning inductive biases that the base learner can exploit for future tasks.

The key technical insight of this work is to bound the generalization error at the two levels (base and meta) using two *different* generalization theory frameworks that each are particularly well-suited for addressing the specific challenges of generalization. At the base level, we utilize the fact that a learning algorithm that exhibits uniform stability [14, 15] also generalizes well in expectation (see Section 4.1 for a formal statement). Intuitively, uniform stability quantifies the sensitivity of the output of a learning algorithm to changes in the training dataset. As demonstrated by [29], limiting the number of training epochs of a gradient-based learning algorithm leads to uniform stability. In other words, a gradient-based algorithm that *learns quickly* is stable. Since the goal of meta-learning is *precisely* to train the base learner to learn quickly, we posit that generalization bounds based on stability are particularly well-suited to bounding the generalization error at the base level. At the meta level, we employ a generalization bound based on *Probably Approximately Correct (PAC)-Bayes* theory. Originally developed two decades ago [43, 38], there has been a recent resurgence of interest in PAC-Bayes due to its ability to provide strong generalization guarantees for neural networks [24, 8, 54]. Intuitively, the challenge of generalization at the meta level (i.e., generalizing to new tasks) is similar to the challenge of generalizing to new data in the standard supervised learning setting. In both cases, one must prevent over-fitting to the particular tasks/data that have been seen during meta-training/training. Thus, the strong empirical performance of PAC-Bayes theory in supervised learning problems makes it a promising candidate for bounding the generalization error at the meta level.

Contributions. The primary contributions of this work are the following. First, we leverage the insights above in order to develop a novel generalization bound for gradient-based meta-learning using uniform stability and PAC-Bayes theory (Theorem 3). Second, we develop a regularization scheme for MAML [25] that explicitly minimizes the derived bound (Algorithm 1). We refer to the resulting approach as *PAC-BUS* since it combines PAC-Bayes and Uniform Stability to derive generalization guarantees for meta-learning. Third, we demonstrate our approach on two meta-learning problems: (i) a toy non-convex classification problem on the unit-ball (Section 5.1), and (ii) the *Mini-Wiki* benchmark introduced in [34] (Section 5.2). Even in these relatively small-scale settings, we demonstrate that recently-developed generalization frameworks for meta-learning provide either near-vacuous or loose bounds, while PAC-BUS provides significantly stronger bounds. Fourth, we demonstrate our approach in larger-scale settings where it remains challenging to obtain non-vacuous bounds (for our approach as well as others). Here, we propose a practical regularization scheme which re-weights the terms in the rigorously-derived PAC-BUS upper bound (*PAC-BUS(H)*; Algorithm 3 in the appendix). Recent work [77] introduces a challenging variant of the *Omniglot* benchmark [35] which highlights and tackles challenges with *memorization* in meta-learning. We show that PAC-BUS(H) is able to prevent memorization on this variant (Section 5.3).

2 Problem formulation

Samples, tasks, and datasets. Formally, consider the setting where we have an unknown meta distribution P_t over tasks (roughly, "tasks" correspond to different, but potentially related, learning problems). A sampled task $t \sim P_t$ induces an (unknown) distribution $P_{z|t}$ over sample space $\mathcal{Z}$. We assume that all sampling is independent and identically distributed (i.i.d.). Note that the sample space $\mathcal{Z}$ is shared between tasks, but the distribution $P_{z|t}$ may be different. We then sample within-task samples $z \sim P_{z|t}$ and within-task datasets $S = \{z_1, z_2, \ldots, z_m\} \sim P_{z|t}^m$. We assume that each sample z has a single corresponding label $o(z)$, where the function o is an oracle which outputs the correct label of z. At meta-training time, we assume access to l datasets, which we call $\mathbf{S} = \{S_1, S_2, \ldots, S_l\}$. Each dataset S_i in $\mathbf{S}$ is drawn by first selecting a task t_i from P_t, and then drawing $S_i \sim P_{z|t_i}^m$.

Hypotheses and losses. Let h denote a hypothesis and $L(h, z)$ be the loss incurred by hypothesis h on sample z. The loss is computed by comparing $h(z)$ with the true label $o(z)$. For simplicity, we assume that there is no noise on the labels; we can thus assume that all loss functions have access to

the label oracle function o and thus the loss depends only on hypothesis h and sample z. We note that this assumption is not required for our analysis and is made for the ease of exposition. Overloading the notation, we let $L(h, P_{z|t}) := \mathbb{E}_{z\sim P_{z|t}} L(h, z)$ and $\widehat{L}(h, S) := \frac{1}{|S|}\sum_{i=1}^{|S|} L(h, z_i)$.

Meta-learning. As with model-agnostic meta-learning (MAML) [25], we let meta parameters $\theta \in \mathbb{R}^{n_\theta}$ correspond to an initialization of the base learner's hypothesis. Let h_θ be the θ-initialized hypothesis. Generally, the initialization θ is learned from the multiple datasets we have access to at meta-training time. In this work, we will learn a *distribution* P_θ over initializations so that we can use bounds from the PAC-Bayes framework. At test time, a new task $t \sim P_t$ is sampled and we are provided with a new dataset $S \sim P_{z|t}^m$. The base learner uses an algorithm A (e.g., gradient descent), the dataset S, and the initialization $\theta \sim P_\theta$ in order to fine-tune the hypothesis and perform well on future samples drawn from $P_{z|t}$. We denote the base learner's updated hypothesis by $h_{A(\theta,S)}$. More formally, our goal is to learn a distribution P_θ with the following objective:

$$\min_{P_\theta} \mathcal{L}(P_\theta, P_t) := \min_{P_\theta} \mathop{\mathbb{E}}_{t\sim P_t} \mathop{\mathbb{E}}_{S\sim P_{z|t}^m} \mathop{\mathbb{E}}_{\theta\sim P_\theta} L(h_{A(\theta,S)}, P_{z|t}). \tag{1}$$

We are particularly interested in the the few-shot learning case, where the number of samples which the base learner can use to adapt is small. A common technique to improve test performance in the few-shot learning case is to allow for validation data at meta-training time. Thus, in addition to a generalization guarantee on meta-learning without validation data, we will derive a bound when allowing for the use of validation data $S_{\text{va}} \sim P_{z|t}^n$ during meta-training.

3 Related work

Meta-learning. Meta-learning is a well-studied technique for exploiting similarities between learning tasks [62, 70]. Often used to reduce the need for large amounts of training data, a number of approaches for meta-learning have been explored over decades [11, 13, 16, 31, 72, 61, 55, 32]. Recently, methods based on model-agnostic meta-learning (MAML) [25] have demonstrated strong performance across different application domains and benchmarks such as *Omniglot* [35] and *Mini-ImageNet* [74]. These methods operate by optimizing a set of initial parameters that can be quickly fine-tuned via gradient descent on a new task. The approaches mentioned above typically do not provide any generalization guarantees, and none of them compute explicit numerical bounds on generalization performance. Our approach has the structure of gradient-based meta-learning while providing guarantees on generalization.

Generalization bounds for supervised learning. Multiple frameworks have been developed for providing generalization guarantees in the classical supervised learning setting. Early breakthroughs include Vapnik-Chervonenkis (VC) theory [71, 6], Rademacher complexity [65], and the minimum description length principle [12, 56, 36]. More recent frameworks include algorithmic stability bounds [14, 19, 29, 57, 1] and PAC-Bayes theory [67, 43, 64]. The connection between stability and learnability has been established in [66, 73, 29], and suggests that algorithmic stability bounds are a strong choice of generalization framework. PAC-Bayes theory in particular provides some of the tightest known generalization bounds for classical supervised learning approaches such as support vector machines [64, 38, 26, 57, 4, 48]. Since its development, researchers have continued to tighten [38, 42, 54] and generalize the framework [17, 18, 59]. Exciting recent results [24, 45, 46, 10, 8, 54] have demonstrated the promise of PAC-Bayes to provide strong generalization bounds for neural networks on supervised learning problems (see [33] for a recent review of generalization bounds for neural networks). It is also possible to combine frameworks such as PAC-Bayes and uniform stability to derive bounds for supervised learning [39]. We will use these two frameworks to bound generalization in the two levels of meta-learning. In contrast to the standard supervised learning setting, generalization bounds for meta-learning are less common and remain loose.

Generalization bounds for meta-learning. As described in Section 1, meta-learning bounds must account for two "levels" of generalization (base level and meta level). The approach presented in [41] utilizes algorithmic stability bounds at both levels. However, this requires both meta and base learners to be uniformly stable. This is a strong requirement that is challenging to ensure at the meta level. Another recent method, known as follow-the-meta-regularized-leader (FMRL) [34], provides guarantees for a regularized meta-learning version of the follow-the-leader (FTL) method for online learning, see e.g. [30]. The generalization bounds provided are derived from the application of online-to-batch techniques [3, 22]. A regret bound for meta-learning using an aggregation technique at the meta-level and an algorithm with a uniform generalization bound at the base level is provided

in [3]. The techniques mentioned do not present an algorithm which makes use of validation data (in contrast to our approach). Using validation data (i.e., held-out data) is a common technique for improving performance in meta-learning and is particularly important for the few-shot learning case.

Another method for deriving a generalization bound on meta-learning is to use PAC-Bayes bounds at both the base and meta levels [52, 53]. In [5], generalization bounds based on such a framework are provided along with practical optimization techniques. However, the method requires one to maintain distributions over distributions of initializations, which can result in large computation times during training and makes it extremely challenging to numerically compute the bound. Moreover, the approach also does not allow one to incorporate validation data to improve the bound. Recent work has made progress on some of these challenges. In [60], the computational efficiency of training is improved but the challenges associated with numerically computing the generalization bound or incorporating validation data are not addressed. State-of-the-art work tightens the two-level PAC-Bayes guarantee, addresses computation times for training and evaluation of the bound, and allows for validation data [77]. However, all of the two-level PAC-Bayes bounds require a separate PAC-Bayes bound for each task, and thus a potentially loose union bound.

We present a framework which, to our knowledge, is the first to combine algorithmic stability and PAC-Bayes bounds (at the base- and meta- levels respectively) in order to derive a meta-learning algorithm with associated generalization guarantees. As outlined in Section 1, we believe that the algorithmic stability and PAC-Bayes frameworks are particularly well-suited to tackling the specific challenges of generalization at the different levels. We also highlight that *none* of the approaches mentioned above report numerical values for generalization bounds, even for relatively simple problems. Here, we empirically demonstrate that prior approaches tend to provide either near-vacuous or loose bounds even in relatively small-scale settings while our proposed method provides significantly stronger bounds.

4 Generalization bound on meta-learning

We use two different frameworks for the two levels of generalization required in a meta-learning bound. We utilize the PAC-Bayes framework to bound the expected training loss on future tasks, and uniform stability bounds to argue that if we have a low training loss when using a uniformly stable algorithm, then we achieve a low test loss. The following section will introduce these frameworks independently. We then present the overall meta-learning bound and associated algorithm to find a distribution over initialization parameters (i.e., meta parameters) that minimizes the upper bound.

4.1 Preliminaries: two generalization frameworks

4.1.1 Uniform stability

Let $S = \{z_1, z_2, \ldots, z_m\} \in \mathcal{Z}^m$ be a set of m elements of $\mathcal{Z}$. Let $S^i = \{z_1, \ldots, z_{i-1}, z_i', z_{i+1}, \ldots, z_m\}$ be identical to dataset S except that the i^{th} sample z_i is replaced by some $z_i' \in \mathcal{Z}$. Note that our analysis can be extended to allow for losses bounded by some finite M, but we work with losses bounded within $[0, 1]$ for the sake of simplicity. With these precursors, we define an analogous notion of *uniform stability* to [29, Definition 2.1] for deterministic algorithms A and distributions P_θ over initializations.[1]

Definition 1 (Uniform Stability) *A deterministic algorithm A has $\beta > 0$ uniform stability with respect to loss L if $\forall\, z \in \mathcal{Z}$, $\forall\, S \in \mathcal{Z}^m$, $\forall\, i \in \{1, \ldots, m\}$, and all distributions P_θ over initializations, the following holds:*

$$\mathop{\mathbb{E}}_{\theta \sim P_\theta} |L(h_{A(\theta,S)}, z) - L(h_{A(\theta,S^i)}, z)| \leq \beta. \tag{2}$$

We define β_{US} as the minimal such β.

In this work, we will bound β_{US} as a function of the algorithm, form of the loss, and number of samples that the algorithm uses (See Appendix A.4 for further details on the bounds on β_{US} for our setup). We then establish a relationship between uniform stability and generalization in expectation. The following is adapted from [29, Theorem 2.2] for the notion of uniform stability presented in Definition 1:

[1] We use deterministic algorithms to avoid excess computation when calculating the provided meta-learning upper bounds. See Appendix A.4 for further details.

Theorem 1 (Algorithmic Stability Generalization in Expectation) *Fix a task $t \sim P_t$. The following inequality holds for hypothesis $h_{A(\theta,S)}$ learned using β_{US} uniformly stable algorithm A with respect to loss L:*

$$\mathop{\mathbb{E}}_{S\sim P_{z|t}^m} \mathop{\mathbb{E}}_{\theta\sim P_\theta} L(h_{A(\theta,S)}, P_{z|t}) \leq \mathop{\mathbb{E}}_{S\sim P_{z|t}^m} \mathop{\mathbb{E}}_{\theta\sim P_\theta} \widehat{L}(h_{A(\theta,S)}, S) + \beta_{\mathrm{US}}. \tag{3}$$

Proof. The proof is similar to the one presented for [29, Theorem 2.2] and is presented in Appendix A.1. □

4.1.2 PAC-Bayes theory

For the meta-level bound, we make use of the PAC-Bayes generalization bound introduced in [43]. Note that other PAC-Bayes bounds such as the quadratic variant [58] and PAC-Bayes-λ variant [69] may be used and substituted in the following analysis. We first present a general version of the PAC-Bayes bound and then specialize it to our meta-learning setting in Section 4.2. Let $f(\theta, s)$ be an arbitrary loss function which only depends on parameters θ and the sample s which has been drawn from an arbitrary distribution P_s. The following bound is a tightened version of the bound presented in [43] for when $l \geq 8$.

Theorem 2 (PAC-Bayes Generalization Bound [40]) *For any data-independent prior distribution $P_{\theta,0}$ over θ, some loss function f where $0 \leq f(\theta, s) \leq 1, \forall\, s, \forall\, \theta$, $l \geq 8$, and $\delta \in (0, 1)$, with probability at least $1-\delta$ over a sampling of $\{s_1, s_2, \ldots, s_l\} \sim P_s^l$, the following holds simultaneously for all distributions P_θ over θ:*

$$\mathop{\mathbb{E}}_{s\sim P_s} \mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta, s) \leq \frac{1}{l}\sum_{i=1}^{l} \mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta, s_i) + R_{\mathrm{PAC-B}}(P_\theta, P_{\theta,0}, \delta, l), \tag{4}$$

where the PAC-Bayes "regularizer" term is defined as follows

$$R_{\mathrm{PAC-B}}(P_\theta, P_{\theta,0}, \delta, l) := \sqrt{\frac{D_{\mathrm{KL}}(P_\theta \| P_{\theta,0}) + \ln \frac{2\sqrt{l}}{\delta}}{2l}}, \tag{5}$$

and D_{KL} is the Kullback-Leibler (KL) divergence.

4.2 Meta-learning bound

In order to obtain a generalization guarantee for meta-learning, we utilize the two frameworks above. We first specialize the PAC-Bayes bound in Theorem 2 to bound the expected training loss on future tasks. We then utilize Theorem 1 to demonstrate that if we have a low expected training loss when using a uniformly stable algorithm, then we achieve a low expected test loss. These two steps allow us to combine the generalization frameworks above to derive an upper bound on (1) which can be computed with known quantities. With the following assumption, the resulting generalization bound is presented in Theorem 3.

Assumption 1 (Bounded loss.) *The loss function L is bounded: $0 \leq L(h, z) \leq 1$ for any h in the hypothesis space for the given problem, and any z in the sample space.*

Theorem 3 (Meta-Learning Generalization Guarantee) *For hypotheses $h_{A(\theta,S)}$ learned with β_{US} uniformly stable algorithm A, data-independent prior $P_{\theta,0}$ over initializations θ, loss L which satisfies Assumption 1, $l \geq 8$, and $\delta \in (0, 1)$, with probability at least $1 - \delta$ over a sampling of the meta-training dataset $\boldsymbol{S} \sim P_S^l$, the following holds simultaneously for all distributions P_θ over θ:*

$$\mathcal{L}(P_\theta, P_t) \leq \frac{1}{l}\sum_{i=1}^{l} \mathop{\mathbb{E}}_{\theta\sim P_\theta} \widehat{L}(h_{A(\theta,S_i)}, S_i) + R_{\mathrm{PAC-B}}(P_\theta, P_{\theta,0}, \delta, l) + \beta_{\mathrm{US}}. \tag{6}$$

Proof. The proof can be split into three steps:
Step 1.
Let P_s in Theorem 2 be the marginal distribution P_S over datasets of size m (see Appendix A.2 for details) and note that sampling $S \sim P_S$ is equivalent to first sampling $t \sim P_t$ and then sampling $S \sim P_{z|t}^m$. Additionally let $f(\theta, S) := \widehat{L}(h_{A(\theta,S)}, S)$ where $A(\theta, S)$ is any deterministic algorithm.

Plugging in these definitions into Inequality (4) results in the following inequality which holds under the same assumptions as Theorem 2, and with probability at least $1-\delta$ over the sampling of $\mathbf{S} \sim P_S^l$:

$$\begin{aligned}\mathop{\mathbb{E}}_{S\sim P_S}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)},S) &= \mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)},S)\\ &\leq \frac{1}{l}\sum_{i=1}^{l}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S_i)},S_i) + R_{\mathrm{PAC-B}}(P_\theta,P_{\theta,0},\delta,l). \qquad (7)\end{aligned}$$

Step 2.
Now assume that algorithm A is β_{US} uniformly stable. For a fixed task $t \sim P_t$ we have the following by Theorem 1:

$$\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}L(h_{A(\theta,S)},P_{z|t}) \leq \mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)},S) + \beta_{\mathrm{US}}.$$

Take the expectation over $t \sim P_t$. We then have:

$$\mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}L(h_{A(\theta,S)},P_{z|t}) \leq \mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)},S) + \beta_{\mathrm{US}}, \qquad (8)$$

since $\mathbb{E}_{t\sim P_t}\ \beta_{\mathrm{US}} = \beta_{\mathrm{US}}$. This establishes a bound on the true expected loss for a new task after running algorithm A on a training dataset corresponding to the new task.
Step 3.
Note that (7) provides an upper bound on the first term of the RHS of (8) when algorithm A is β_{US} uniformly stable. Thus we have the following by plugging (7) in the RHS of (8):
Under the same assumptions as both Theorems 1 and 2, and with probability at least $1-\delta$ over the sampling of $\mathbf{S} \sim P_S^l$:

$$\mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}L(h_{A(\theta,S)},P_{z|t}) \leq \frac{1}{l}\sum_{i=1}^{l}\mathop{\mathbb{E}}_{\theta\sim P_\theta}L(h_{A(\theta,S)},S_i) + R_{\mathrm{PAC-B}}(P_\theta,P_{\theta,0},\delta,l) + \beta_{\mathrm{US}},$$

completing the proof. □

Theorem 3 is presented for any distributions P_θ and $P_{\theta,0}$ over initializations. However, in practice we will use multivariate Gaussian distributions for both. The specialization of Theorem 3 to Gaussian distributions is provided in Appendix A.3.1. Next, we allow for validation data $S_{\mathrm{va}} \sim P_{z|t}^n$ at meta-training time so that the bound is more suited to the few-shot learning case. We compute the upper bound using the evaluation data $S_{\mathrm{ev}} = \{S, S_{\mathrm{va}}\}$ sampled from the marginal distribution $P_{S_{\mathrm{ev}}}$ over datasets of size $m+n$. However, we still only require m samples at meta-test time; see Appendix A.3.2 for the derivation. Note that the training data S is often excluded from the data used to update the meta-learner. However, this is necessary for our approach to obtain a guarantee on few-shot learning performance. The result is a guarantee with high probability over a sampling of $\mathbf{S}_{\mathrm{ev}} \sim P_{S_{\mathrm{ev}}}^l$:

$$\mathcal{L}(P_\theta,P_t) \leq \frac{1}{l}\sum_{i=1}^{l}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S_i)},S_{\mathrm{ev},i}) + R_{\mathrm{PAC-B}}(P_\theta,P_{\theta,0},\delta,l) + \frac{m\beta_{\mathrm{US}}}{m+n}. \qquad (9)$$

4.3 PAC-BUS algorithm

Recall that we aim to find a distribution P_θ over initializations that minimizes $\mathcal{L}(P_\theta, P_t)$ as stated in Equation (1). We cannot minimize $\mathcal{L}(P_\theta, P_t)$ directly due to the expectations taken over unknown distributions P_t and $P_{z|t}$ for sampled task t, but we may indirectly minimize it by minimizing the upper bounds in Inequalities (6) or (9).

Computing the upper bound requires evaluating an expectation taken over $\theta \sim P_\theta$. In general, this is intractable. However, we aim to minimize this upper bound to provide the tightest guarantee possible. Similar to the method in [24], we use an unbiased estimator of $\mathbb{E}_{\theta\sim P_\theta}L(h_{A(\theta,S)},\cdot)$. Let P_θ be a multivariate Gaussian distribution over initializations θ with mean μ and covariance $\mathrm{diag}(s)$; thus $P_\theta = \mathcal{N}(\mu, \mathrm{diag}(s))$ and $P_{\theta,0} = \mathcal{N}(\mu_0, \mathrm{diag}(s_0))$. Further, let $\psi := (\mu, \log(s))$, and use the shorthand $\mathcal{N}_{\psi_0}$ for the prior and $\mathcal{N}_\psi$ for the posterior distribution over initializations. We use the following estimator of $\mathbb{E}_{\theta\sim P_\theta}L(h_{A(\theta,S)},\cdot)$:

$$L(h_{A(\theta,S)},\cdot), \quad \theta \sim \mathcal{N}_\psi. \qquad (10)$$

Algorithm 1 PAC-BUS: meta-learning via PAC-Bayes and Uniform Stability

Input: Fixed prior distribution $\mathcal{N}_{\psi_0}$ over initializations
Input: β_{US} uniformly stable Algorithm A
Input: Meta-training dataset $\mathbf{S}$, learning rate γ
Initialize: $\psi \leftarrow \psi_0$
Output: Optimized ψ^*
$B(\psi, \theta'_1, \theta'_2 \ldots, \theta'_l) := \frac{1}{l}\sum_{i=1}^{l} \widehat{L}(h_{\theta'_i}, S_i) + R_{\text{PAC-B}}(\mathcal{N}_\psi, \mathcal{N}_{\psi_0}, \delta, l) + \beta_{\mathrm{US}}$
while not converged **do**
  Sample $\theta \sim \mathcal{N}_\psi$
  for $i = 1$ **to** l **do**
    $\theta'_i \leftarrow A(\theta, S_i)$
  end for
  $\psi \leftarrow \psi - \gamma \nabla_\psi B(\psi, \theta'_1, \theta'_2 \ldots, \theta'_l)$
end while

We present the resulting training technique in Algorithm 1. This algorithm can be used to learn a distribution over initializations that minimizes the upper bound presented in Theorem 3 and its specializations. This is presented for the case when A is β_{US} uniformly stable for some β_{US}. For gradient-based algorithms, the learning rate α often appears directly in the bound for β_{US} [29]. Thus it is potentially beneficial to update α as well. We present Algorithm 1 without learning the learning rate. To meta-learn the learning rate, we can augment ψ_0 to include a parameterization of a prior distribution over learning rates and update it using the same gradient step presented in 1 for ψ.

Determining the gradient of $B(\psi, \theta'_1, \theta'_2 \ldots, \theta'_l)$ with respect to ψ requires computing the Hessian of the loss function if algorithm $A(\theta, S)$ uses a gradient update to compute θ'_i. First order approximations often perform similarly to the second-order meta-learning techniques [27, 25, 47], and can be used to speed up the training. Additionally, Algorithm 1 can be modified to use mini-batches of tasks instead of all tasks in the meta update to improve training times; we present an algorithm which uses mini-batches of tasks in Appendix A.5.1.

In practice, we are interested in algorithms such as stochastic gradient descent (SGD) and gradient descent (GD) for the base learner. We can obtain bounds on the uniform stability constant β_{US} when using gradient methods with the results from [29]. See Appendix A.4 for details on the β_{US} bounds we use in this work. With a bound on β_{US}, we can calculate all the terms in $B(\psi, \theta'_1, \theta'_2 \ldots, \theta'_l)$ and use Algorithm 1 to minimize the meta-learning upper bound. When evaluating the upper bound, we use the sample convergence bound [37, 24] to upper bound the expectation taken over $\theta \sim P_\theta$. See Appendix A.6 for details.

5 Examples

We demonstrate our approach on three examples below. All examples we provide are few-shot meta-learning problems. To adapt at the base level, m examples from each class are given for an "m-shot" learning problem. If applicable, n samples can be given as validation data for each task during the meta-training step. In the first two examples, our primary goal is to demonstrate the tightness of our generalization bounds compared to other meta-learning bounds. We also present empirical test performance on held-out data; however, we emphasize that the focus of our work is to obtain improved generalization guarantees (and not necessarily to improve empirical test performance). In the third example, we present an algorithm that is motivated by our theoretical framework and demonstrate its ability to improve empirical performance on a challenging task. All the code required to run the following examples is available at `https://github.com/irom-lab/PAC-BUS`.

5.1 Example: classification on the unit ball

We evaluate the tightness of the generalization bound in Equation (9) on a toy two-class classification problem where the sample space $\mathcal{Z}$ is the unit ball $B^2(0, 1)$ in two dimensions with radius 1 and centered at the origin. Data points for each task are sampled from $P_{z|t}$, where a task corresponds to a particular concept which labels the data as $(+)$ if within $B^2(c_t, r_t)$ and $(-)$ otherwise. Center c_t is sampled uniformly from the $y \geq 0$ semi-ball $B^2_{y\geq 0}(0, 0.4)$ of radius 0.4. The radius r_t is then sampled uniformly from $[0.1, 1 - \|c_t\|]$. Notably, the decision boundary between classes is nonlinear. Thus, generalization bounds which rely on convex losses (such as [34]) will have difficulty with

Table 1: We present the generalization bounds (for $\delta = 0.01$) provided by each method if applicable, and use the sample convergence bound [37] for MR-MAML, and PAC-BUS, but not MLAP-M.[2] Note that for these methods, we specifically minimize their respective meta-learning bounds. We also report the meta-test loss (the softmax activated cross-entropy loss – CEL_s) for all methods. We present the mean and standard deviation after 5 trials. We highlight that our approach provides the strongest generalization guarantee.

Classification on Ball	MAML [25]	MLAP-M [5]	MR-MAML [77]	PAC-BUS (ours)
Bound ↓	None	1.0538 ± 0.0012[2]	0.3422 ± 0.0006	$\mathbf{0.2213 \pm 0.0012}$
Test Loss ↓	0.1701 ± 0.0070	0.1645 ± 0.0045	$\mathbf{0.1584 \pm 0.0012}$	0.1657 ± 0.0014

providing guarantees for networks that perform well. We choose the softmax-activated cross-entropy loss, CEL_s, as the loss function. Before running Algorithm 1, we address a few technical challenges that arise from Assumption 1 as well as computing c_L and c_S. We address these in Appendix A.5.

We then apply Algorithm 1 using the few-shot learning bound in Inequality (9). We present the guarantee on the meta-test loss associated with each training method in Table 1. In addition, we present the average meta-test loss after training with 10 samples. We compare our bounds and empirical performance with the meta-learning by adjusting priors (MLAP) technique [5] and the meta-regularized MAML (MR-MAML) technique [77]. All methods are given held-out data to learn a prior before minimizing their respective upper bounds (see Appendix A.11.1 for further details on the prior training step). Additionally, since all bounds require the loss to be within $[0, 1]$, networks N are constrained such that the Frobenius norm of the output is bounded by r, i.e., $\|N(z)\|_F \leq r$. We compare the aforementioned methods' meta-test loss to MAML with weights constrained in the same manner (note that MAML does not provide a guarantee). Upper bounds which use the PAC-Bayes framework are computed with many evaluations from the posterior distribution. This allows us to apply the sample convergence bound [37] (as in Equation (35) for our bound) unless otherwise noted.

We find that PAC-BUS provides a significantly stronger guarantee compared with the other methods. Note that the guarantee provided by MLAP-M [5] is vacuous because the meta-test loss is bounded between 0 and 1, while the guarantee is above 1.

5.2 Example: Mini-Wiki

Next, we present results on the *Mini-Wiki* benchmark introduced in [34]. This is derived from the Wiki3029 dataset presented in [9]. The dataset is comprised of 4-class, m-shot learning tasks with sample space $\mathcal{Z} = \{z \in \mathbb{R}^d \mid \|z\|_2 = 1\}$. Sentences from various Wikipedia articles are passed through the continuous-bag-of-words GloVe embedding [51] into dimension d = 50 to generate samples. For this learning task, we use a k-class version of CEL_s and logistic regression. Since this example is convex, we can use GD and bound β_{US} with Theorem 4 in the appendix [29]. We keep the loss bounded by constraining the network $\|N(z)\|_F \leq r$ and scale the loss as in the previous example. The tightness of the bounds on c_L and c_S affected the upper bound in Inequality (9) more than in the previous example, so we bound them as tightly as possible. See Appendix A.9 for the calculations.

We apply Algorithm 1 using the bound which allows for validation data, Inequality (9), to learn on 4-way *Mini-Wiki* $m = \{1, 3, 5\}$-shot. The results are presented in Table 2. We compare our results with the FMRL variant which provides a guarantee [34], follow-the-last-iterate (FLI)-Batch, and with MR-MAML [77]. FLI-Batch does not require bounded losses explicitly, but requires that the parameters of the network lie within a ball of radius r. For the logistic regression used in the example, this is equivalent to $\|N(z)\|_F \leq r$. Thus, we scale the loss and use the same r for each method to provide a fair comparison. We also show the results of training with MAML constrained in the same way for reference. Each method is given the same amount of held-out data for training a prior (see Appendix A.11.2 for further details on training the prior).

[2]Due to high computation times associated with estimating the MLAP upper bound, this value is not computed with the sample convergence bound as the other upper bounds are. Thus, the value presented does not carry a guarantee, but would be similar if computed with the sample convergence bound. The value is shown to give a qualitative sense of the guarantee.

Table 2: We compare the generalization bounds (for $\delta = 0.01$) provided by each method where applicable and use the sample convergence bound for MR-MAML and PAC-BUS. Since we specifically minimize these methods' upper bounds, we can fairly compare the relative tightness of each bound. We also report the meta-test loss (CEL_s) for each method for exposition. We report the mean and standard deviation after 5 trials. We highlight that our approach provides the strongest guarantee.

4-Way *Mini-Wiki*	1-shot ↓	3-shot ↓	5-shot ↓
FLI-Batch Bound [34]	0.6638 ± 0.0011	0.6366 ± 0.0006	0.6343 ± 0.0014
MR-MAML Bound [77]	0.7400 ± 0.0003	0.7312 ± 0.0003	0.7283 ± 0.0005
PAC-BUS Bound (ours)	$\mathbf{0.4999 \pm 0.0003}$	$\mathbf{0.5058 \pm 0.0002}$	$\mathbf{0.5101 \pm 0.0002}$
MAML [25]	$\mathbf{0.3916 \pm 0.0009}$	$\mathbf{0.3868 \pm 0.0005}$	$\mathbf{0.3883 \pm 0.0005}$
FLI-Batch [34]	0.4091 ± 0.0008	0.4078 ± 0.0005	0.4097 ± 0.0012
MR-MAML [77]	0.3922 ± 0.0009	0.3869 ± 0.0003	0.3884 ± 0.0005
PAC-BUS (ours)	0.3922 ± 0.0009	0.3878 ± 0.0003	0.3895 ± 0.0005

As in the previous example, PAC-BUS provides a significantly tighter guarantee than the other methods (Table 2). We see similar empirical meta-test loss for MAML [25], MR-MAML [77], and PAC-BUS with slightly higher loss for FLI-Batch [34]. In addition, we computed the meta-test accuracy as the percentage of correctly classified sentences. See Table 4 in Section A.11.2 for these results along with other experimental details.

5.3 Example: memorizable Omniglot

We have demonstrated the ability of our approach to provide strong generalization guarantees for meta-learning in the settings above. We now consider a more complex setting where we are unable to obtain strong guarantees. In this example, we employ a learning heuristic based on the PAC-BUS upper bound, *PAC-BUS(H)*; see Appendix A.5.2 for the details and the Algorithm. We relax Assumption 1 and no longer constrain the network as in previous sections. Instead, we maintain and update estimates of the Lipschitz and smoothness constants of the network, using [68], and incorporate them into the uniform stability regularizer term, β_{US}. We then scale each regularizer term (i.e., $R_{\text{PAC}-\text{B}}(P_\theta, P_{\theta,0}, \delta, l)$ and β_{US}) by hyper-parameters λ_1 and λ_2 respectively. Analogous to the technique described in [77], we aim to incorporate the form of the theoretically-derived regularizer into the loss, without requiring it to be as restrictive during learning. The result is a regularizer that punishes large deviation from the prior $P_{\theta,0}$ and too much adaptation at the base-learning level.

We test our method on *Omniglot* [35] for 20-way, $m = \{1, 5\}$-shot classification in the non-mutually exclusive (NME) case [77]. In [77], the problem of memorization in meta-learning is explored and demonstrated with non-mutually exclusive learning problems. *NME Omniglot* corresponds to randomization of class labels for a task at test time only. This worsens the performance of any network that memorized class labels; see [77] for more details.[3] We compare our method to an analogous heuristic presented in [77], which also has a $D_{\text{KL}}(P_\theta \| P_{\theta,0})$ term in the loss. Thus, this heuristic (referred to as MR-MAML(W) [77]) regularizes the change in weights of the network. Additionally, we compare to the heuristic described in [34] (FLI-Online) which performs better in practice than the FLI-Batch method. We do not provide data for training a prior in this case since we do not aim to compute a bound in this example. We use standard MAML as a reference. See Table 3 for the results.

We see that MAML [25] and FLI-Online [34] do not prevent memorization on *NME Omniglot* [77]. This is especially apparent in the 1-shot learning case, where their performance suffers significantly due to this memorization. Both MR-MAML(W) [77] and PAC-BUS(H) prevent memorization, with PAC-BUS(H) outperforming MR-MAML(W). Note that PAC-BUS(H) outperforms MR-MAML(W) by a wider margin in the 1-shot case as compared with the 5-shot case. We believe this is due to the effectiveness of the uniform stability regularizer at the base level. MR-MAML(W) suffers more in the 1-shot case because over-adaptation is more likely with fewer within-task examples.

6 Conclusion and discussion

We presented a novel generalization bound for gradient-based meta-learning: PAC-BUS. We use different generalization frameworks for tackling the distinct challenges of generalization at the two

[3] We use a slightly different task setup as the one in [77]; see Appendix A.11.3 for the details of our setup.

Table 3: We present the meta-test accuracy as a percentage on non-mutually-exclusive *Omniglot* [77]. In contrast to the previous examples, here we aim to achieve the best empirical performance for each method. In particular, this task compares each methods' ability to prevent memorization. We report the mean and standard deviation after 5 trials.

20-WAY *Omniglot*	NME 1-SHOT ↑	NME 5-SHOT ↑
MAML [25]	23.4 ± 2.2	75.1 ± 4.8
FLI-ONLINE [34]	22.4 ± 0.5	39.1 ± 0.5
MR-MAML(W) [77]	84.2 ± 2.2	94.3 ± 0.3
PAC-BUS(H) (OURS)	$\mathbf{87.9 \pm 0.5}$	$\mathbf{95.0 \pm 0.9}$

levels of meta-learning. In particular, we employ uniform stability bounds and PAC-Bayes bounds at the base- and meta-learning levels respectively. On a toy non-convex problem and the *Mini-Wiki* meta-learning task [34], we provide significantly tighter generalization guarantees as compared to state-of-the-art meta-learning bounds while maintaining comparable empirical performance. To our knowledge, this work presents the first numerically-evaluated generalization guarantees associated with a proposed meta-learning bound. On memorizable *Omniglot* [35, 77], we show that a heuristic based on the PAC-BUS bound prevents memorization of class labels in contrast to MAML [25], and better performance than meta-regularized MAML [77]. We believe our framework is well suited to the few-shot learning problems for which we present empirical results, but our framework is potentially applicable to a broad range of different settings (e.g., reinforcement learning).

We note a few challenges with our method as motivation for future work. Our bound is vacuous on larger scale learning problems such as *Omniglot*. This is partially caused by a larger KL-divergence term in the PAC-Bayes bound when using deep convolutional networks (due to the increased dimensionality of the weight vector). In addition, we do not have a theoretical analysis on the convergence properties of the algorithms presented, so we must experimentally determine the number of samples required for tight bounds. In the results of Section 5.1 and 5.2, despite an improved bound over other methods, our method does not necessarily improve empirical test performance. We emphasize that our focus in this work was on deriving stronger generalization guarantees rather than improving empirical performance. However, obtaining approaches that provide both stronger guarantees and empirical performance is an important direction for future work.

Future work can also explore ways in which to incorporate tighter PAC-Bayes bounds or those with less restrictive assumptions. One interesting avenue is to extend PAC-BUS by using a PAC-Bayes bound for unbounded loss functions for the meta-generalization step (e.g. as presented in [28]). Another promising direction is to incorporate regularization on the weights of the network directly (e.g., L_2 regularization or gradient clipping) to create networks with smaller Lipschitz and smoothness constants. Additionally, it would be interesting to explore learning of the base-learner's algorithm while maintaining uniform stability. For example, one could parameterize a set of uniformly stable algorithms and learn a posterior distribution over the parameters.

Broader impact. The approach we present in this work aims to strengthen performance guarantees for gradient-based meta-learning. We believe that strong generalization guarantees in meta-learning, especially in the few-shot learning case, could lead to broader application of machine learning in real-world applications. One such example is for medical diagnosis, where abundant training data for certain diseases may be difficult to obtain. Another example on which poor performance is not an option is any safety critical robotic system, such as ones which involve human interaction.

Meta-learning methods typically require a lot of data and training time, and ours is not an exception. In our case, it took multiple weeks of computation time on Amazon Web Services (AWS) instances to train and compute all networks and results we present in this paper. This creates challenges with accessibility and energy usage.

Acknowledgments

The authors are grateful to the anonymous reviewers for their valuable feedback and suggestions, and to Thomas Griffiths for helpful feedback on this work. The authors were supported by the Office of Naval Research [N00014-21-1-2803, N00014-18-1-2873], the NSF CAREER award [2044149], the Google Faculty Research Award, and the Amazon Research Award.

References

[1] Karim Abou-Moustafa and Csaba Szepesvári. An Exponential Efron-Stein Inequality for L_q Stable Learning Rules. *Proceedings of the 30th International Conference on Algorithmic Learning Theory*, 2019.

[2] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[3] Pierre Alquier, The Tien Mai, and Massimiliano Pontil. Regret Bounds for Lifelong Learning. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[4] Amiran Ambroladze, Emilio Parrado-hernández, and John Shawe-taylor. Tighter PAC-Bayes Bounds. *Advances in Neural Information Processing Systems 19*, 2007.

[5] Ron Amit and Ron Meir. Meta-Learning by Adjusting Priors Based on Extended PAC-Bayes Theory. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[6] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[7] Sebastien Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. learn2learn: A library for meta-learning research. arxiv preprint. *preprint arXiv:2008.12284*, 2020.

[8] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger Generalization Bounds for Deep Nets via a Compression Approach. *arXiv preprint arXiv:1802.05296*, 2018.

[9] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A Theoretical Analysis of Contrastive Unsupervised Representation Learning. *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[10] Peter L. Bartlett, Dylan J. Foster, and Matus J Telgarsky. Spectrally-Normalized Margin Bounds for Neural Networks. In *Advances in Neural Information Processing Systems 30*, pages 6240–6249, 2017.

[11] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the Optimization of a Synaptic Learning Rule. *Proceedings of the Conference on Optimality in Artificial and Biological Neural Networks*, pages 6–8, 1992.

[12] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's Razor. *Information Processing Letters*, 24(6):377–380, 1987.

[13] Léon Bottou and Vladimir Vapnik. Local Learning Algorithms. *Neural Computation*, 4: 888–900, 1992.

[14] Olivier Bousquet and André Elisseeff. Stability and Generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[15] Olivier Bousquet, Yegor Klochkov, and Nikita Zhivotovskiy. Sharper Bounds for Uniformly Stable Algorithms. *Proceedings of the 33rd Conference on Learning Theory*, 2020.

[16] Rich Caruana. Multitask Learning. *Machine Learning*, 28:41–75, 1997.

[17] Olivier Catoni. *Statistical Learning Theory and Stochastic Optimization*. École d'Été de Probabilités de Saint-Flour 2001. Springer, 2004.

[18] Olivier Catoni. *PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning*, volume 56 of *Lecture notes - Monograph Series*. Institute of Mathematical Statistics, 2007.

[19] Alain Celisse and Benjamin Guedj. Stability Revisited: New Generalisation Bounds for the Leave-one-Out. *arXiv preprint arXiv:1608.06412*, 2016.

[20] Andrew Collette. *Python and HDF5*. O'Reilly, 2013.

[21] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[22] Giulia Denevi, Dimitris Stamos, Carlo Ciliberto, and Massimiliano Pontil. Online-Within-Online Meta-Learning. *Advances in Neural Information Processing Systems 32*, 2019.

[23] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[24] Gintare Karolina Dziugaite and Daniel M. Roy. Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

[25] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[26] Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian Learning of Linear Classifiers. In *Proceedings of the 26th International Conference on Machine Learning*, pages 353–360. ACM, 2009.

[27] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[28] Maxime Haddouche, Benjamin Guedj, Omar Rivasplata, and John Shawe-Taylor. PAC-Bayes Unleashed: Generalisation Bounds With Unbounded Losses. *arXiv preprint arXiv:2006.07279*, 2020.

[29] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train Faster, Generalize Better: Stability of Stochastic Gradient Descent. *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[30] Elad Hazan. Introduction to Online Convex Optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

[31] Tom Heskes. Solving a Huge Number of Similar Tasks: a Combination of Multi-Task Learning and a Hierarchical Bayesian Approach. *Proceedings of the 15th International Conference on Machine Learning*, 1998.

[32] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-Learning in Neural Networks: A Survey. *arXiv preprint arXiv:2004.05439*, 2020.

[33] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic Generalization Measures and Where to Find Them. *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[34] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Provable Guarantees for Gradient-Based Meta-Learning. *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[35] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One Shot Learning of Simple Visual Concepts. *Cognitive Science*, 33, 2011.

[36] John Langford. Tutorial on Practical Prediction Theory for Classification. *Journal of Machine Learning Research*, 6(Mar):273–306, 2005.

[37] John Langford and Rich Caruana. (Not) Bounding the True Error. *Advances in Neural Information Processing Systems 14*, 2002.

[38] John Langford and John Shawe-Taylor. PAC-Bayes & margins. *Advances in Neural Information Processing Systems 15*, 2003.

[39] Ben London. A PAC-Bayesian Analysis of Randomized Learning with Application to Stochastic Gradient Descent. *Advances in Neural Information Processing Systems 30*, 2017.

[40] Andreas Maurer. A Note on the PAC Bayesian Theorem. *arXiv preprint arXiv:0411099*, 2004.

[41] Andreas Maurer. Algorithmic Stability and Meta-Learning. *Journal of Machine Learning Research*, 6:967–994, 2005.

[42] David McAllester. A PAC-Bayesian Tutorial with A Dropout Bound. *arXiv preprint arXiv:1307.2118*, 2013.

[43] David A McAllester. PAC-Bayesian Model Averaging. *Proceedings of the 12th Conference on Learning Theory*, 1999.

[44] MOSEK ApS. Mosek fusion api for python 9.0.105, 2019. URL `https://docs.mosek.com/9.0/pythonfusion/index.html`.

[45] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks. *preprint arXiv:1707.09564*, 2017.

[46] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring Generalization in Deep Learning. In *Advances in Neural Information Processing Systems 30*, pages 5949–5958, 2017.

[47] Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[48] Emilio Parrado-Hernández, Amiran Ambroladze, John Shawe-Taylor, and Shiliang Sun. PAC-Bayes Bounds with Data Dependent Priors. *Journal of Machine Learning Research*, 13: 3507–3531, 2012.

[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, pages 8024–8035, 2019.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[51] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[52] Anastasia Pentina and Christoph H. Lampert. A PAC-Bayesian Bound for Lifelong Learning. *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[53] Anastasia Pentina and Christoph H. Lampert. Lifelong Learning with Non-i.i.d. Tasks. *Advances in Neural Information Processing Systems 28*, 2015.

[54] María Pérez-Ortiz, Omar Rivasplata, John Shawe-Taylor, and Csaba Szepesvári. Tighter Risk Certificates for Neural Networks. *arXiv preprint arXiv:2007.12911*, 2020.

[55] Sachin Ravi and Hugo Larochelle. Optimization as a Model for Few-Shot Learning. *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[56] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, 1989.

[57] Omar Rivasplata, Emilio Parrado-Hernandez, John Shawe-Taylor, Shiliang Sun, and Csaba Szepesvari. PAC-Bayes Bounds for Stable Algorithms with Instance-Dependent Priors. *Advances in Neural Information Processing Systems 31*, 2018.

[58] Omar Rivasplata, Vikram M. Tankasali, and Csaba Szepesvari. PAC-Bayes with Backprop. *arXiv preprint arXiv:1908.07380*, 2019.

[59] Omar Rivasplata, Ilja Kuzborskij, Csaba Szepesvari, and John Shawe-Taylor. PAC-Bayes Analysis Beyond the Usual Bounds. *Advances in Neural Information Processing Systems 33*, 2020.

[60] Jonas Rothfuss, Vincent Fortuin, and Andreas Krause. PACOH: Bayes-Optimal Meta-Learning with PAC-Guarantees. *arXiv preprint arXiv:2002.05551*, 2020.

[61] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-Learning with Memory-Augmented Neural Networks. *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[62] Jurgen Schmidhuber. Evolutionary Principles in Self-Referential Learning. On Learning how to Learn: The Meta-Meta-Meta...-Hook. Diploma thesis, Technische Universitat Munchen, Germany, 1987.

[63] Rolf Schneider. *Convex Bodies: The Brunn–Minkowski Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 2013.

[64] Matthias Seeger. PAC-Bayesian Generalisation Error Bounds for Gaussian Process Classification. *Journal of Machine Learning Research*, 3(Oct):233–269, 2002.

[65] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[66] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, Stability and Uniform Convergence. *Journal of Machine Learning Research*, 11:2635–2670, 2010.

[67] John Shawe-Taylor and Robert C. Williamson. A PAC Analysis of a Bayesian Estimator. *Proceedings of the 10th Conference on Computational Learning Theory*, 1997.

[68] Aman Sinha, Hongseok Namkoong, Riccardo Volpi, and John Duchi. Certifying Some Distributional Robustness with Principled Adversarial Training. *arXiv preprint arXiv:1710.10571*, 2020.

[69] Niklas Thiemann, Christian Igel, Olivier Wintenberger, and Yevgeny Seldin. A Strongly Quasiconvex PAC-Bayesian Bound. *Machine Learning Research*, 76:1–26, 2017.

[70] Sebastian Thrun and Lorien Pratt. *Learning to Learn*. Springer Science & Business Media, 1998.

[71] Vladimir N. Vapnik and A. Ya Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Dokl. Akad. Nauk*, 181(4), 1968.

[72] Ricardo Vilalta and Youssef Drissi. A Perspective View and Survey of Meta-Learning. *Artificial Intelligence Review*, 18:77–95, 2002.

[73] Silvia Villa, Lorenzo Rosasco, and Tomaso Poggio. On Learnability, Complexity and Stability. In *Empirical Inference*, pages 59–69. Springer, 2013.

[74] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. *Advances in Neural Information Processing Systems 29*, 2016.

[75] Jeremy Watt, Reza Borhani, and Aggelos K. Katsaggelos. *Machine Learning Refined: Foundations, Algorithms, and Applications*. Cambridge University Press, 2016.

[76] Wolfram Research, Inc. Mathematica, Version 12.0, 2019. URL `https://www.wolfram.com/mathematica/`. Champaign, IL.

[77] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-Learning without Memorization. *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[78] Wenda Zhou, Victor Veitch, Morgane Austern, Ryan Adams, and Peter Orbanz. Nonvacuous Generalization Bounds at the ImageNet Scale: A PAC-Bayesian Compression Approach. *Proceedings of the 7th International Conference on Learning Representations*, 2019.

A Appendix

A.1 Proof of Theorem 1

Theorem 1 (Algorithmic Stability Generalization in Expectation) *Fix a task $t \in P_t$. The following inequality holds for hypothesis $h_{A(\theta,S)}$ learned using β_{US} uniformly stable algorithm A with respect to loss L:*

$$\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta} L(h_{A(\theta,S)}, P_{z|t}) \leq \mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta} \widehat{L}(h_{A(\theta,S)}, S) + \beta_{\mathrm{US}}. \tag{11}$$

Proof. Let $S = \{z_1, z_2 \ldots, z_m\} \sim P_{z|t}^m$ and $S' = \{z_1', z_2' \ldots, z_m'\} \sim P_{z|t}^m$ be two independent random samples and let $S^i = \{z_1, \ldots, z_{i-1}, z_i', z_{i+1}, \ldots, z_m\}$ be identical to S except with the i^{th} sample replaced with z_i'. Fix a distribution P_θ over initializations. Consider the following

$$\begin{aligned}
\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta} \widehat{L}(h_{A(\theta,S)}, S) &= \mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\left[\frac{1}{m}\sum_{i=1}^m L(h_{A(\theta,S)}, z_i)\right] && (12)\\
&= \mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{S'\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\left[\frac{1}{m}\sum_{i=1}^m L(h_{A(\theta,S_i)}, z_i)\right] && (13)\\
&= \mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{S'\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\left[\frac{1}{m}\sum_{i=1}^m L(h_{A(\theta,S)}, z_i')\right] + \delta && (14)\\
&= \mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta} L(h_{A(\theta,S)}, P_{z|t}) + \delta && (15)
\end{aligned}$$

where

$$\delta = \mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{S'\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\left[\frac{1}{m}\sum_{i=1}^m L(h_{A(\theta,S_i)}, z_i') - \sum_{i=1}^m L(h_{A(\theta,S)}, z_i')\right]. \tag{16}$$

We bound δ with the supremum over datasets S and S' differing by a single sample

$$\delta \leq \sup_{S,S',z}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\left[L(h_{A(\theta,S)}, z) - L(h_{A(\theta,S)}, z)\right] \leq \beta_{\mathrm{US}} \tag{17}$$

by Definition 1. □

A.2 Definition of Marginal Distribution P_S

In this section we formally define the marginal distribution P_S which we make use of in the proof of Theorem 3. This is the distribution over datasets one obtains by first sampling a task t from P_t, and then sampling a dataset S from $P_{z|t}^m$. Consider the following equations (for simplicity, we use summations instead of integrals to compute expectations; $p(t)$ represents the probability of sampling t and $p(S|t)$ is the probability of sampling S given t):

$$\begin{aligned}
\mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta, S) &= \sum_t p(t)\sum_S p(S|t)\mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta, S) && (18)\\
&= \sum_{t,S} p(t)p(S|t)\mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta, S) && (19)\\
&= \sum_{t,S} p(S,t)\mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta, S) && (20)\\
&= \sum_S \left(\mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta, S)\underbrace{\sum_t p(S,t)}_{=p(S)}\right). && (21)
\end{aligned}$$

Here $p(S)$ corresponds to the marginal distribution over datasets S. Note that the last line above holds because $\mathbb{E}_{\theta\sim P_\theta} f(\theta, S)$ does not depend on t.

Definition 2 (Marginal Distribution Over Datasets S) *Let $P_S := p(S)$ from above.*

Thus we have

$$\mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta,S)=\sum_S p(S)\mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta,S)=\mathop{\mathbb{E}}_{S\sim P_S}\mathop{\mathbb{E}}_{\theta\sim P_\theta} f(\theta,S). \tag{22}$$

A.3 Specializing the Bound

A.3.1 Meta-Learning Bound for Gaussian Distributions

In practice, the distribution P_θ over initializations will be a multivariate Gaussian distribution. Thus, in this section, we present a specialization of the bound for Gaussian distributions. Let P_θ have mean μ and covariance Σ; thus $P_\theta=\mathcal{N}(\mu,\Sigma)$ and analogously $P_{\theta,0}=\mathcal{N}(\mu_0,\Sigma_0)$. We can then apply the analytical form for the KL-divergence between two multivariate Gaussian distributions to the bound presented in Theorem 3. The result is the following bound holding under the same assumptions as Theorem 3:

$$\begin{aligned}\mathcal{L}(P_\theta,P_t)\leq\frac{1}{l}\sum_{i=1}^{l}\mathop{\mathbb{E}}_{\theta\sim P_\theta}L(h_{A(\theta,S_i)},S_i)+\beta_{\mathrm{US}}\\ +\sqrt{\frac{(\mu-\mu_0)\Sigma_0^{-1}(\mu-\mu_0)+\ln\frac{|\Sigma_0|}{|\Sigma|}+\mathrm{tr}(\Sigma_0^{-1}\Sigma)-n_{\mathrm{dim}}+2\ln\frac{2\sqrt{l}}{\delta}}{4l}},\end{aligned} \tag{23}$$

where n_{dim} is the number of dimensions of the Gaussian distribution. We implement the above bound in code instead of the non-specialized form of the KL divergence to speed up computations and simplify gradient computations.

A.3.2 Few-Shot Learning Bound with Validation Data

In this section, we will assume that, in addition to the training data $S\sim P_{z|t}^m$, we have access to validation data $S_{\mathrm{va}}\sim P_{z|t}^n$ at meta-training time. We will show that a meta-learning generalization bound can still be obtained in this case. Notably, this will not require validation data at meta-testing time.

We begin by bounding the expected loss on evaluation data $S_{\mathrm{ev}}=\{S,S_{\mathrm{va}}\}$ after training on S. Note that for other meta-learning techniques, the training data S is often excluded from the data used to update the meta-learner. Including it here helps relate the loss on $P_{z|t}$ to the loss on S_{ev} after adaptation with S (see derivation below), and is necessary to achieve a guarantee on performance for the few-shot learning case. From Inequality (4), we set the arbitrary distribution P_s to the marginal distribution $P_{S_{\mathrm{ev}}}$ over datasets of size $m+n$ and $f(\theta,s):=\widehat{L}(h_{A(\theta,S)},S_{\mathrm{ev}})$. Note that with this marginal distribution, we have an equivalence of sampling given by

$$\mathop{\mathbb{E}}_{S_{\mathrm{ev}}\sim P_{S_{\mathrm{ev}}}}[\cdot]=\mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S_{\mathrm{ev}}\sim P_{z|t}^{m+n}}[\cdot]. \tag{24}$$

The following inequality holds with high probability over a sampling of $\mathbf{S}_{\mathrm{ev}}=\{S_{\mathrm{ev},1},S_{\mathrm{ev},2},\ldots,S_{\mathrm{ev},l}\}\sim P_{S_{\mathrm{ev}}}^l$:

$$\begin{aligned}&\mathop{\mathbb{E}}_{S_{\mathrm{ev}}\sim P_{S_{\mathrm{ev}}}}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)},S_{\mathrm{ev}})=\\ &\mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S_{\mathrm{ev}}\sim P_{z|t}^{m+n}}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)},S_{\mathrm{ev}})\leq\frac{1}{l}\sum_{i=1}^{l}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S_i)},S_{\mathrm{ev},i})+R_{\mathrm{PAC-B}}(P_\theta,P_{\theta,0},\delta,l).\end{aligned} \tag{25}$$

In the next steps, we aim to isolate for a $\widehat{L}(h_{A(\theta,S)},S)$ term so that we may still combine with Inequality (8) as we did in Section 4.2. We decompose the LHS of Inequality (25),

$$\frac{1}{m+n}\mathop{\mathbb{E}}_{t\sim P_t}\Big[m\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)},S)+n\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{S_{\mathrm{va}}\sim P_{z|t}^n}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)},S_{\mathrm{va}})\Big]. \tag{26}$$

Since the validation data S_{va} is sampled independently from S, the expected training loss on the validation data is the true expected loss over sample space $P_{z|t}$,

$$\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{S_{\mathrm{va}}\sim P_{z|t}^n}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)},S_{\mathrm{va}})=\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}L(h_{A(\theta,S)},P_{z|t}). \tag{27}$$

We plug Equality (27) into Equation (26), and then the decomposition in Equation (26) into Inequality (25). We can then isolate for the $\widehat{L}(h_{A(\theta,S)}, S)$ term,

$$\mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S)}, S) \leq \frac{m+n}{m}\left[\frac{1}{l}\sum_{i=1}^{l}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S_i)}, S_{\text{ev},i}) + R_{\text{PAC}-\text{B}}(P_\theta, P_{\theta,0}, \delta, l)\right] - \frac{n}{m}\mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta} L(h_{A(\theta,S)}, P_{z|t}), \tag{28}$$

and plug into the LHS of Equation (8). By simplifying, we find that

$$\mathop{\mathbb{E}}_{t\sim P_t}\mathop{\mathbb{E}}_{S\sim P_{z|t}^m}\mathop{\mathbb{E}}_{\theta\sim P_\theta} L(h_{A(\theta,S)}, P_{z|t}) \leq \frac{1}{l}\sum_{i=1}^{l}\mathop{\mathbb{E}}_{\theta\sim P_\theta}\widehat{L}(h_{A(\theta,S_i)}, S_{\text{ev},i}) + R_{\text{PAC}-\text{B}}(P_\theta, P_{\theta,0}, \delta, l) + \frac{m\beta_{\text{US}}}{m+n}. \tag{29}$$

This resulting bound is very similar to the one in Inequality (6). We compute the loss term in the upper bound with evaluation data and as a result, the size of the uniform stability regularization term is reduced.

A.4 Bounds on the Uniform Stability Constant

In this section, we present bounds from [29] on the uniform stability constant β_{US} which are applicable to our settings. We first formalize the definitions of Lipschitz continuous ("Lipschitz" with constant c_L) and Lipschitz smoothness ("smooth" with constant c_S).

Definition 3 (c_L-Lipschitz) *Function f is c_L-Lipschitz if $\forall\ \theta, \theta' \in \mathbb{R}^{n_\theta}, \forall\ z \in \mathcal{Z}$ the following holds:*

$$|f(\theta, z) - f(\theta', z)| \leq c_L\|\theta - \theta\|. \tag{30}$$

Definition 4 (c_S-smooth) *Function f is c_S-smooth if $\forall\ \theta, \theta' \in \mathbb{R}^{n_\theta}, \forall\ z \in \mathcal{Z}$ the following holds:*

$$\|\nabla f(\theta, z) - \nabla f(\theta', z)\| \leq c_S\|\theta - \theta'\|. \tag{31}$$

Using a convex loss and stochastic gradient descent (SGD) allows us to directly bound the uniform stability constant β_{US} [29]:

Theorem 4 (Convex Loss SGD is Uniformly Stable [29]) *Assume that convex loss function L is c_S-smooth and c_L-Lipschitz $\forall\ z \in \mathcal{Z}$. Suppose we run SGD on S with step size $\alpha \leq \frac{2}{c_S}$ for T steps. Then SGD satisfies β_{US} uniform stability with*

$$\beta_{\text{US}} \leq \frac{2c_L^2}{m}T\alpha. \tag{32}$$

Note that the bounds on β_{US} presented in [29] guarantee β_{US} uniform stability in expectation for a randomized algorithm A. However, for deterministic algorithms, this reduces to β_{US} uniform stability. Using the uniform stability in expectation definition introduces another expectation (over a draw of algorithm A) into the upper bound of the meta-learning generalization guarantee in Inequality (6). So as to not increase the computation required to estimate the upper bound, we let A be deterministic. This is achieved either by fixing the order of the samples on which we perform gradient updates for SGD, or by using gradient descent (GD). Additionally, in the convex case, T steps of GD satisfies the same bound on β_{US} as T steps of SGD; see Appendix A.8.1 for the proof. For non-convex losses, a bound on β_{US} is still achieved when algorithm A is SGD [29]:

Theorem 5 (Non-Convex Loss SGD is Uniformly Stable [29]) *Let non-convex loss L be c_S-smooth and c_L-Lipschitz $\forall\ z \in P_{z|t}$ and satisfy Assumption 1. Suppose we run T steps of SGD with monotonically non-increasing step size $\alpha_t \leq \frac{c}{t}$. Then SGD satisfies β_{US} uniform stability with*

$$\beta_{\text{US}} \leq \frac{1 + \frac{1}{c_S c}}{n-1}(2c_L^2 c)^{\frac{1}{c_S c+1}}T^{\frac{c_S c}{c_S c+1}} \tag{33}$$

Note that this bound does not hold when GD is used.

Algorithm 2 PAC-BUS using Mini-Batches of Tasks

Input: Fixed prior distribution $\mathcal{N}_{\psi_0}$ over initializations
Input: β_{US} uniformly stable Algorithm A
Input: Meta-training dataset $\mathbf{S} = \{S_1, S_2, \ldots, S_l\}$, learning rate γ
Initialize: $\psi \leftarrow \psi_0$
Output: Optimized ψ^*
$B(\psi, \theta'_1, \theta'_2 \ldots, \theta'_k) := \frac{1}{l} \sum_{i=1}^{k} \widehat{L}(h_{\theta'_i}, S_i) + R_{\text{PAC-B}}(\mathcal{N}_\psi, \mathcal{N}_{\psi_0}, \delta, l) + \beta_{\text{US}}$
while not converged **do**
  Sample $\theta \sim \mathcal{N}_\psi$
  for $i = 1$ **to** k **do**
    $j \sim \text{Uniform}\{1, 2, \ldots, l\}$
    $\theta'_i \leftarrow A(\theta, S_j)$
  end for
  $\psi \leftarrow \psi - \gamma \nabla_\psi B(\psi, \theta'_1, \theta'_2 \ldots, \theta'_k)$
end while

A.5 Algorithms

Before running the algorithms presented in this paper, we must deal with a few technical challenges that arise from our method's assumptions and terms which need to be computed. In this paragraph, we discuss the approach we take to deal with these challenges. For arbitrary networks, the softmax-activated cross entropy loss (CEL_s) is not bounded and would not satisfy Assumption 1. We thus constrain the network parameters to lie within a ball and scale the loss function such that all samples $z \in \mathcal{Z}$ achieve a loss within $[0, 1]$; see Appendix A.7 for details. However, the PAC-BUS framework works with distributions P_θ over initializations. One option is to let P_θ be a projected multivariate Gaussian distribution. This prevents the network's output from becoming arbitrarily large. However, the upper bound in Inequality (9) requires the KL-divergence between the prior and posterior distribution over initializations. This is difficult to calculate for projected multivariate Gaussian distributions and would require much more computation during gradient steps. Since the KL-divergence between projected Gaussians is less than that between Gaussians (due to the data processing inequality [21]), we can loosen the upper bound in (6) and (9) by computing the upper bound using the non-projected distributions (but using the projected Gaussians for the algorithm). After sampling a base learner's initialization, we re-scale the network such that its parameters lie within a ball of radius r. We also re-scale the base learner's parameters after each gradient step to guarantee that the loss stays within $[0, 1]$. Projection after gradient steps is not standard SGD, but we show that it maintains the same bound on β_{US}; see Section A.8.2 for details of the proof. Thus, we let algorithm A be SGD with projections after each update and use Theorem 5 to bound β_{US} for non-convex losses [29]. Additionally, we can upper bound the Lipschitz c_L and smoothness c_S constants for the network using the methods presented in [68]. After working through these technicalities, we can compute all terms in the upper bound.

A.5.1 PAC-BUS using Mini-Batches of Tasks

We present the PAC-BUS algorithm modified for mini-batches of tasks to improve training times. For batches of size k, the algorithm is presented in 2.

A.5.2 PAC-BUS(H)

In addition to providing algorithms which minimize the upper bound in Inequalities (6) and (9), we are also interested in a regularization scheme which re-weights the regularizer terms in these bounds. For larger scale and complex settings, it is challenging to provide a non-vacuous guarantee on performance, but weighting regularizer terms has been shown to be an effective training technique [77]. We calculate β_{US} with a one-gradient-step version of Theorem 5. This Theorem requires the algorithm A to be SGD, but we let A be a single step of GD to improve training times. We also relax Assumption 1 Since the β_{US} depends on both c_L and c_S, we update estimates of them after each iteration by sampling multiple $\theta \sim P_\theta$, bound the c_L and c_S for those sets of parameters using Section 4 of [68], and then choose the maximum to compute β_{US}. This is in contrast to limiting the network parameters directly by bounding the output of the loss. Instead, the β_{US} term in the upper

Algorithm 3 PAC-BUS(H): Meta-learning heuristic based on PAC-BUS upper bound

Input: Fixed prior distribution $\mathcal{N}_{\psi_0}$ over initializations
Input: Meta-training dataset $\mathbf{S}$, learning rates α and γ
Input: Scale factors λ_1, λ_2 for regularization terms
Initialize: $\psi \leftarrow \psi_0$
Output: Optimized ψ^*
$B(\psi, c_L, c_S, \theta'_1, \theta'_2, \ldots, \theta'_l) := \frac{1}{l}\sum_{i=1}^{l} \widehat{L}(h_{\theta'_i}, S_i) + \lambda_1 R_{\text{PAC-B}}(\mathcal{N}_\psi, \mathcal{N}_{\psi_0}, \delta, l) + \lambda_2 \beta_{\text{US}}(c_L, c_S)$
Estimate c_L and c_S using $\mathcal{N}_{\psi_0}$
while not converged **do**
  Sample $\theta \sim \mathcal{N}_\psi$
  for $i = 1$ **to** l **do**
    $\theta'_i \leftarrow \theta - \alpha \nabla_\theta \widehat{L}(h_\theta, S_i)$
  end for
  $\psi \leftarrow \psi - \gamma \nabla_\psi B(\psi, c_L, c_S, \theta'_1, \theta'_2, \ldots, \theta'_l)$
  Estimate c_L and c_S using $\mathcal{N}_\psi$
end while

bound and the scale factor will determine how much to restrict the network parameters. The resulting method is presented in Algorithm 3. In order to provide strong performance in practice, we tune λ_1 and λ_2.

A.6 Sample Convergence Bound

After training is complete, we aim to compute the upper bound. However, this requires evaluating an expectation $\theta \sim P_\theta$, which may be intractable. Providing a valid PAC guarantee without needing to evaluate the expectation taken over $\theta \sim P_\theta$ requires the use of the sample convergence bound [37]. We have the following guarantee with probability $1 - \delta'$ over a random draw of $\{\theta_1, \theta_2 \ldots, \theta_N\} \sim P_\theta^N$ for any dataset S [37],

$$D_{\text{KL}}\left(\sum_{j=1}^{N} L(h_{A(\theta_j,S)}, \cdot) \middle\| \mathop{\mathbb{E}}_{\theta \sim P_\theta} L(h_{A(\theta,S)}, \cdot)\right) \leq \frac{\log(\frac{2}{\delta'})}{N}. \tag{34}$$

We can invert this KL-style bound (i.e. a bound of the form $D_{\text{KL}}(p\|q^*) \leq c$) by solving the optimization problem, $q^* \leq D_{\text{KL}}^{-1}(q\|c) := \sup\{q \in [0,1] : D_{\text{KL}}(p\|q) \leq c\}$, as described in [24]. After the inversion is performed on Inequality (34), we use a union bound to combine the result with Inequality (6) and retain a guarantee with probability $1 - \delta - \delta'$ as in [24],

$$\mathcal{L}(P_\theta, P_t) \leq \frac{1}{l}\sum_{i=1}^{l} D_{\text{KL}}^{-1}\left(\sum_{j=1}^{N} L(h_{A(\theta_j,S_i)}, S_i) \middle\| \frac{\log(\frac{2}{\delta'})}{N}\right) + R_{\text{PAC-B}}(P_\theta, P_{\theta,0}, \delta, l) + \beta_{\text{US}}. \tag{35}$$

An analogous bound is achieved when combined with Inequality (9). Thus, after training, we evaluate Inequality (35) to provide the guarantee. Note that use of the sample convergence bound is a loosening step. However, in our experiments, the upper bound in Inequality (35) is less than 5% looser than unbiased estimates of Inequality (6). This can be reduced further at the expense of computation time (if we utilize a larger number of samples in the concentration inequality).

A.7 Constraining Parameters and Scaling Losses

In order to maintain a guarantee, the PAC-Bayes upper bound in Theorem 2 requires a loss function bounded between 0 and 1. However, the losses we use are not bounded in general. Let N_θ be an arbitrary network parameterized by θ and $N_\theta(z)$ be the output of the network given sample $z \in \mathcal{Z}$. Consider arbitrary loss f, which maps the network's output to a real number. If $\|N_\theta(z)\| \leq r, \forall\, \theta \in \mathbb{R}^{n_\theta}, \forall\, z \in \mathcal{Z}$, then we can perform a linear scaling of f to map it onto the interval $[0,1]$. We define the minimum and maximum value achievable by loss function f as follows

$$M_f := \max_{z \in \mathcal{Z},\ \theta \in \mathbb{R}^{n_\theta},\ \|N_\theta(z)\| \leq r} f(\theta, z) \tag{36}$$

$$m_f := \min_{z \in \mathcal{Z},\ \theta \in \mathbb{R}^{n_\theta},\ \|N_\theta(z)\| \leq r} f(\theta, z). \tag{37}$$

Now we can define a scaled function

$$f_S(\theta, z) := \frac{f(\theta, z) - m_f}{M_f - m_f} \tag{38}$$

such that $f_S(\theta, z) \in [0, 1]$. Note that the Lipschitz and smoothness constants of f_S are also scaled by $\frac{1}{M_f - m_f}$. When we choose loss CEL_s, the k-class cross entropy loss with softmax activation, we have

$$M_{\text{CEL}_s} := \log\Big(\frac{e^{-r} + (k-1)}{e^{-r}}\Big), \quad m_{\text{CEL}_s} := \log\Big(\frac{e^{r} + (k-1)}{e^{r}}\Big). \tag{39}$$

However, we must restrict the parameters in such a way that satisfies $\|N_\theta(z)\| \leq r$. For arbitrary networks structures, this is not straightforward, so we only analyze the case we use in this paper. Consider an L-layer network with ELU activation. Let parameters θ contain weights $\mathbf{W}_1, \ldots, \mathbf{W}_L$, and biases $b_1, \ldots, b_L$, and assume bounded input $\|z\| \leq r_z, \forall\, \mathcal{Z}$.

$$\|N_\theta(z)\| = \|\text{ELU}\big(\mathbf{W}_L \text{ELU}(\mathbf{W}_{L-1}(\cdots) + b_{L-1}) + b_L\big)\| \tag{40}$$

$$\leq \|\mathbf{W}_L\|_F(\|\mathbf{W}_{L-1}\|_F(\cdots) + \|b_{L-1}\|) + \|b_L\| \leq r \tag{41}$$

We can satisfy $\|N_{\mathbf{W}}(z)\| \leq r$ by restricting

$$\|\theta\|^2 = \sum_{i=1}^{L} \|\mathbf{W}_i\|_F^2 + \sum_{i=1}^{L} \|b_i\|^2 \leq \left(\frac{r}{\max(1, r_z)}\right)^2. \tag{42}$$

Equation (42) implies Equation (41) by applying the inequality of arithmetic and geometric means. Thus, we ensure $\|\theta\| \leq r/\max(1, r_z)$ by projecting the network parameters onto the ball of radius $\min(r, \frac{r}{r_z})$ after each gradient update.

A.8 Uniform Stability Considerations

A.8.1 Uniform Stability for Gradient Descent

In this section, we will prove that T steps of GD has the same uniform stability constant as T steps of SGD in the convex case. This will allow us to use GD when attempting to minimize a convex loss, Section 5.2. Let the gradient update rule G be given by $G(\theta, z) = \theta - \alpha \nabla_\theta f(\theta, z)$ for *convex* loss function f, initialization $\theta \in \mathbb{R}^{n_\theta}$, sample $z \in \mathcal{Z}$, and positive learning rate α. We define two key properties for gradient updates: expansiveness and boundedness [29].

Definition 5 (c_E-expansive, Definition 2.3 in [29]) *Update rule G is c_E-expansive if $\forall\ \theta, \theta' \in \mathbb{R}^{n_\theta}, \forall\ z \in \mathcal{Z}$ the following holds:*

$$\|G(\theta, z) - G(\theta', z)\| \leq c_E \|\theta - \theta'\|. \tag{43}$$

Definition 6 (c_B-bounded, Definition 2.4 in [29]) *Update rule G is c_B-bounded if $\forall\ \theta \in \mathbb{R}^{n_\theta}, \forall\ z \in \mathcal{Z}$ the following holds:*

$$\|\theta - G(\theta, z)\| \leq c_B. \tag{44}$$

Now, consider dataset $S \in \mathcal{Z}^m$ and define $\bar{f}(\theta, S) := \frac{1}{m}\sum_{i=1}^m f(\theta, z_i)$. We also define $\bar{G}(\theta, S) := \theta - \alpha \nabla_\theta \bar{f}(\theta, S) = \sum_{i=1}^m G(\theta, z_i)$. Assume that $G(\theta, z)$, is c_E-expansive and c_B-bounded $\forall\ z \in \mathcal{Z}$. We then bound the expansiveness of $\bar{G}(\theta, S)$,

$$\|\bar{G}(\theta, S) - \bar{G}(\theta', S)\| \leq \frac{1}{m}\sum_{i=1}^{m} \|G(\theta, z_i) - G(\theta', z_i)\| \leq \frac{1}{m}\sum_{i=1}^{m} c_E \|\theta - \theta'\| = c_E \|\theta - \theta'\|. \tag{45}$$

To compute the boundedness, consider

$$\|\theta - \bar{G}(\theta, S)\| \leq \frac{1}{m}\sum_{i=1}^{m} \|\theta - G(\theta, z_i)\| \leq \frac{1}{m}\sum_{i=1}^{m} c_B = c_B. \tag{46}$$

For a single gradient step on sample z, we see the same bounds on c_E and c_B when performing a single GD step on dataset S. Thus, if Lemmas 2.5, 3.3, and 3.7 in [29] are true for gradient updates G, they are also true for gradient updates $\bar{G}$. We can then run through the proof of Theorem 3.8 in [29] to show that it holds for T steps of GD if it holds for T steps of SGD.

Let $S \sim P_S$ be a dataset of size m and S' be an identical dataset with one element changed. We run T steps of GD updates, $\bar{G}$, on each of S and S'. This results in parameters $\theta_1, \ldots, \theta_T$ and $\theta'_1, \ldots, \theta'_T$ respectively. Fix learning rate $\alpha \leq \frac{2}{c_S}$ and consider

$$\mathop{\mathbb{E}}_{S,S'}\|\theta_{t+1} - \theta'_{t+1}\| = \mathop{\mathbb{E}}_{S,S'}\|\bar{G}(\theta_t, S) - \bar{G}(\theta'_t, S')\| \tag{47}$$

$$\leq \frac{1}{m}\sum_{j=1, i\neq j}^{m} \mathop{\mathbb{E}}_{S,S'}\|G(\theta_t, z_j) - G(\theta'_t, z_j)\| + \frac{1}{m}\mathop{\mathbb{E}}_{S,S'}\|G(\theta_t, z_i) - G(\theta'_t, z_i))\| \tag{48}$$

$$\leq \frac{m-1}{m}\mathop{\mathbb{E}}_{S,S'}\|\theta_t - \theta'_t\| + \frac{1}{m}\mathop{\mathbb{E}}_{S,S'}\|\theta_t - \theta'_t\| + \frac{2\alpha c_L}{m} = \mathop{\mathbb{E}}_{S,S'}\|\theta_t - \theta'_t\| + \frac{2\alpha c_L}{m} \tag{49}$$

The steps above follow from Lemmas 2.5, 3.3, and 3.7 in [29] and the linearity of expectation. The rest of the proof follows naturally and results in a uniformly stable constant $\beta_{\text{US}} \leq \frac{2c_L^2}{m} T\alpha$ for T steps of GD. Thus, we have the following result.

Corollary 1 *Assume that loss convex function f is c_S-smooth and c_L Lipschitz $\forall\, z \in \mathcal{Z}$. Suppose T steps of SGD on S satisfies β_{US} uniform stability. This implies that T steps of GD on S satisfies β_{US} uniform stability.*

A.8.2 Uniform Stability Under Projections

Projecting parameters onto a ball after gradient updates does not constitute standard SGD nor GD, so we analyze the stability constant after T steps of $G_P(\theta, z) = \text{Proj}[\theta - \alpha\nabla_\theta f(\theta, z)]$. Assume $\|z\| \leq r_z, \forall\, z \in \mathcal{Z}$. The function Proj scales parameters to satisfy $\|\theta\| \leq \max(r, \frac{r}{r_z})$ if it is not already satisfied. See Appendix A.7 for an explanation of this restriction.

As in Appendix A.8.1, we compute bounds on the expansiveness and boundedness of G_P. Suppose θ is a vector containing all weights of an L-layer network. Network hyper-parameters such as learning rate and activation parameters do not need to be projected, so they will not be included. Assume that θ, θ' already satisfy $\|N_\theta(z)\| \leq r, \forall\, z \in \mathcal{Z}$. Consider

$$\|G_P(\theta, z) - G_P(\theta', z)\| = \|\text{Proj}(G(\theta, z)) - \text{Proj}(G(\theta', z))\| \leq \|G(\theta, z) - G(\theta', z)\| \leq c_E\|\theta - \theta'\|. \tag{50}$$

Note that any required scaling is equivalent to orthogonal projection of the parameters onto a euclidean norm ball of radius r in R^d, where d is the number of parameters in the network. Thus, the first inequality follows from the fact that orthogonal projections onto closed convex sets satisfy the contractive property [63]. Next, consider

$$\|\theta - G_P(\theta, z)\| = \|\text{Proj}(\theta) - \text{Proj}(G_P(\theta, z))\| \leq \|\theta - G(\theta, z)\| \leq c_B. \tag{51}$$

The equality follows from the assumption that θ already satisfies the norm constraint. As above, the first inequality follows from the fact that the Proj function satisfies the contractive property [63].

With these bounds, gradient update G_P satisfies Lemmas 2.5, 3.3, and 3.7 from [29] if G does. Note that an analogous procedure can be used to show that scaling after a GD update, $\bar{G}_P$, also satisfies these Lemmas. When function f or $\bar{f}$ is convex, the proof of Theorem 3.8 in [29] applies, and shows that using gradient updates G_P or $\bar{G}_P$ achieve the same bound on the uniform stability constant β_{US}. Thus, when f is convex, we may use G_P or $\bar{G}_P$ to compute updates and maintain the guarantee presented in Theorem 1. Suppose now that f is not convex. Using Lemmas 2.5, 3.3, 3.7, and 3.11 from [29], the proof of Theorem 3.12 in [29] follows naturally to achieve a bound on SGD using projected gradient updates G_P when f is not convex.

A.9 Lipschitz and Smoothness Constant Calculation

Recall Definitions 3 and 4 for a function which is c_L-Lipschitz and c_S-smooth from Appendix A.4. We define the softmax activation function.

Definition 7 (Softmax Function) $s : \mathbb{R}^k \to \mathbb{P}^k$

$$s(u)_i = \frac{e^{u_i}}{\sum_{j=1}^{k} e^{u_j}}\ ,\ \forall\, i. \tag{52}$$

Where every element in $\mathbb{P}^k$ is a probability distribution in k dimensions (i.e. if $v \in \mathbb{P}^k$, then $\sum_{i=1}^k v_i = 1$ and $v_i \geq 0 \; \forall \; i$). Since the stability constant β_{US} depends directly on the Lipschitz constant of the loss function, and β_{US} appears in the regularizer of the final bound, we will be as tight as possible when bounding the Lipschitz constant to keep the generalization as tight as possible. Section 6.2 of [75] describes an approach for bounding the Lipschitz constant for the 2-class, sigmoid activated, cross entropy loss. We are interested in the k-class case with softmax activation, and also aim to bound the smoothness constant. We begin with a similar analysis to the one described in [75].

Given unit-length column vector $z \in \mathbb{R}^d$ and row vector $y \in \mathbb{P}^k$, with weight matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$ (representing a single-layer network), the loss function is given by:

$$\text{CEL}_s(\mathbf{W}) = -\sum_{i=1}^{k} y_i \log(s(z^T \mathbf{W})_i). \tag{53}$$

Note that while y is any probability distribution, in practice, y will be an indicator vector, describing the correct label with a 1 in the index of the correct class and 0 elsewhere. However, the analysis that follows does not depend on this assumption.

We will take the Hessian of this loss to determine convexity and the Lipschitz constant. However, since the weights are given by a matrix, the Hessian would be a 4-tensor. To simplify the analysis, we will define

$$\mathbf{w} = \begin{bmatrix} \mathbf{W}_{:,1} \\ \mathbf{W}_{:,2} \\ \vdots \\ \mathbf{W}_{:,k} \end{bmatrix}. \tag{54}$$

Where $\mathbf{W}_{:,i}$ is the i^{th} column of $\mathbf{W}$ such that $\mathbf{w} \in \mathbb{R}^{dk}$. We also let

$$\mathbf{z}(i)^T = \begin{bmatrix} \bar{0} & \dots & \bar{0} & z^T & \bar{0} & \dots & \bar{0} \end{bmatrix} \tag{55}$$

such that z is placed in the i^{th} group of d elements and $\bar{0}$ is a row vector of d zeros. Vector $\mathbf{z}(i) \in \mathbb{R}^{dk}$ since there are k groups. With these definitions, we write the softmax activated network defined by $\mathbf{W}$ with input z:

$$s(z^T \mathbf{W})_i = \frac{e^{\mathbf{z}(i)^T \mathbf{w}}}{\sum_{j=1}^k e^{\mathbf{z}(j)^T \mathbf{w}}}. \tag{56}$$

We can simplify this by plugging in for the definition of s:

$$\text{CEL}_s(\mathbf{w}) := \text{CEL}_s(\mathbf{W}) = -\sum_{i=1}^{k} y_i \left[\mathbf{z}(i)^T \mathbf{w} - \log\left(\sum_{j=1}^{k} e^{\mathbf{z}(j)^T \mathbf{w}} \right) \right] \tag{57}$$

$$= -\sum_{i=1}^{k} y_i \mathbf{z}(i)^T \mathbf{w} + \log\left(\sum_{i=1}^{k} e^{\mathbf{z}(i)^T \mathbf{w}} \right). \tag{58}$$

These are equivalent because $\sum_{i=1}^k y_i = 1$. For readability, we let $p_i := s(z^T \mathbf{W})_i$. With these preliminaries the Hessian will be a 2-tensor and the $\nabla^3_{\mathbf{w}}$ term will be a 3-tensor. We compute the gradient and Hessian and $\nabla^3_{\mathbf{w}}$ term:

$$\nabla_{\mathbf{w}} \text{CEL}_s(\mathbf{w}) = -\sum_{i=1}^{k} y_i \mathbf{z}(i) + \sum_{i=1}^{k} \mathbf{z}(i) p_i \tag{59}$$

$$\nabla^2_{\mathbf{w}} \text{CEL}_s(\mathbf{w}) = \sum_{i=1}^{k} \mathbf{z}(i)\mathbf{z}(i)^T p_i - \left(\sum_{i=1}^{k} \mathbf{z}(i) p_i \right) \left(\sum_{j=1}^{k} \mathbf{z}(j)^T p_j \right). \tag{60}$$

We write $\nabla^3_{\mathbf{w}} \text{CEL}_s(\mathbf{w})$ termwise to simplify notation:

$$\nabla^3_{\mathbf{w}} \text{CEL}_s(\mathbf{w}) = \begin{cases} (p_i - 3p_i^2 + 2p_i^3) z \otimes z^T \otimes z^\perp & i = j = l \\ (-p_i p_l + 2p_i^2 p_l) z \otimes z^T \otimes z^\perp & i = j \neq l \\ (-p_j p_i + 2p_j^2 p_i) z \otimes z^T \otimes z^\perp & j = l \neq i \\ (-p_l p_j + 2p_l^2 p_j) z \otimes z^T \otimes z^\perp & l = i \neq j \\ (2 p_i p_j p_l) z \otimes z^T \otimes z^\perp & i \neq j \neq l \end{cases} \tag{61}$$

Where $\otimes$ is the tensor product and $z \otimes z^T \otimes z^\perp \in \mathbb{R}^{d\times d\times d}$ is a 3-tensor with the abuse of notation: $z \in \mathbb{R}^{d\times 1\times 1}$, $z^T \in \mathbb{R}^{1\times d\times 1}$, and $z^\perp \in \mathbb{R}^{1\times 1\times d}$. Thus $\nabla^3_{\mathbf{w}}\text{CEL}_s(\mathbf{w}) \in \mathbb{R}^{dk\times dk\times dk}$.

For twice-differentiable functions, the Lipschitz constant is given by the greatest eigenvalue of the Hessian. Correspondingly, the smoothness constant is given by the greatest eigenvalue of the $\nabla^3_{\mathbf{w}}$ term for thrice-differentiable functions. Thus, we aim to bound the largest value that the Rayleigh quotient can take for any unit-length vector x. For the Hessian:

$$x^T\nabla^2_{\mathbf{w}}\text{CEL}_s(\mathbf{w})x \leq |x^T\nabla^2_{\mathbf{w}}\text{CEL}_s(\mathbf{w})x| = \|x^T\nabla^2_{\mathbf{w}}\text{CEL}_s(\mathbf{w})x\|_F \tag{62}$$

$$\leq \|x\|^2\|\nabla^2_{\mathbf{w}}\text{CEL}_s(\mathbf{w})\|_F = \|\nabla^2_{\mathbf{w}}\text{CEL}_s(\mathbf{w})\|_F \tag{63}$$

$$= \sqrt{\sum_{i=1}^k \|zz^T\|_F(p_i - p_i^2)^2 + \sum_{i=1}^k\sum_{j=1,j\neq i}^k \|zz^T\|_F(p_ip_j)^2} \tag{64}$$

$$= \sqrt{\sum_{i=1}^k (p_i - p_i^2)^2 + \sum_{i=1}^k\sum_{j=1,j\neq i}^k (p_ip_j)^2}. \tag{65}$$

The Frobenius norm is maximized when $p_i = \frac{1}{k}$ for $k > 1$:

$$\|\nabla^2_{\mathbf{w}}\text{CEL}_s(\mathbf{w})\|_F \leq \sqrt{k\Big(\frac{1}{k} - \frac{1}{k^2}\Big)^2 + k(k-1)\Big(\frac{1}{k^2}\Big)^2} \tag{66}$$

$$= \frac{\sqrt{k-1}}{k}. \tag{67}$$

Thus, for $\text{CEL}_s(\mathbf{w})$, the Lipschitz constant, $c_L \leq \frac{\sqrt{k-1}}{k}$ when $k > 1$. We can also show that the Rayleigh quotient is lower bounded by 0 by following analogous steps in [75] (these steps are omitted from this appendix), and thus $\text{CEL}_s(\mathbf{w})$ is convex. Next, we examine the Rayleigh quotient of the $\nabla^3_{\mathbf{w}}\text{CEL}_s(\mathbf{w})$. Analogous to the procedure for the Hessian, we make use of a 3-tensor analog of the Frobenius norm: $\|M\|_{3,F} := \sqrt{\sum_{i=1}^k\sum_{j=1}^k\sum_{l=1}^k M(i,j,l)^2}$. Thus we have the following inequality

$$x^T \otimes [x^\perp \otimes \nabla^3_{\mathbf{w}}\text{CEL}_s(\mathbf{w})] \otimes x \leq \|\nabla^3_{\mathbf{w}}\text{CEL}_s(\mathbf{w})\|_{3,F}. \tag{68}$$

Since $\|z \otimes z^T \otimes z^\perp\|_{3,F} = 1$, we can write this as

$$\|\nabla^3_{\mathbf{w}}\text{CEL}_s(\mathbf{w})\|_{3,F} \leq \sqrt{\begin{aligned}&\sum_{i=1}^k (p_i - 3p_i^2 + 2p_i^3)^2 + \sum_{i=1}^k\sum_{j=1,j\neq i}^k (-p_ip_l + 2p_i^2p_l)^2 \\ &+ \sum_{j=1}^k\sum_{l=1,l\neq j}^k (-p_jp_i + 2p_j^2p_i)^2 + \sum_{l=1}^k\sum_{i=1,i\neq l}^k (-p_lp_j + 2p_l^2p_j)^2 \\ &+ \sum_{i=1}^k\sum_{j=1,j\neq i}^k\sum_{l=1,l\neq j}^k (2p_ip_jp_l)^2.\end{aligned}} \tag{69}$$

This is maximized when $p_i = \frac{1}{k}$ for $k > 2$, which was verified with the symbolic integrator Mathematica [76]. Simplifying results in:

$$\|\nabla^3_{\mathbf{w}}\text{CEL}_s(\mathbf{w})\|_{3,F} \leq \sqrt{\frac{(k-1)(k-2)}{k^3}}. \tag{70}$$

Thus for $\text{CEL}_s(\mathbf{w})$, the smoothness constant, $c_S \leq \sqrt{\frac{(k-1)(k-2)}{k^3}}$ when $k > 2$. When $k = 2$, $p_1, p_2 = \frac{1}{2} \pm \frac{\sqrt{3}}{6}$ and $c_S \leq \sqrt{\frac{2}{27}}$.

A.10 Study on Base-learning Learning Rate and Number of Update Steps

In this section we present additional results on the performance of the algorithms with different iterations and learning rates using the same example setup as in Section 5.1. Note that we have

not used the sample convergence bound (see Appendix A.6) and present results for a single sample $\theta \sim P_\theta$. The true values of the upper bounds for MLAP-M [5], MR-MAML [77], and PAC-BUS (our method) are unlikely to change by more than 5% as the sample complexity bound does not loosen the guarantee very much. We present these results to provide a qualitative sense of the guarantees and their trends for varying base-learning learning rates and number of update steps.

Below we present test losses for MAML [25] (as a baseline), MLAP-M [5], MR-MAML [77], and PAC-BUS for base-learning rates (lr_b) of 0.01 to 10 using $\{1, 3, 10\}$ adaptation steps.

MAML Test Loss, lr_b =	0.01	0.03	0.1	0.3	1	3	10
Adaptation steps = 1	0.184±0.007	0.184±0.008	0.168±0.006	0.152±0.004	0.120±0.001	0.114±0.001	0.133±0.007
Adaptation steps = 3	0.177±0.006	0.179±0.002	0.149±0.002	0.126±0.001	0.115±0.001	0.106±0.001	0.123±0.004
Adaptation steps = 10	0.179±0.004	0.155±0.002	0.128±0.001	0.124±0.001	0.113±0.001	0.104±0.002	0.129±0.008

MLAP-M Test Loss, lr_b =	0.01	0.03	0.1	0.3	1	3	10
Adaptation steps = 1	0.181±0.010	0.175±0.014	0.150±0.006	0.129±0.009	0.083±0.001	0.065±0.003	0.220±0.044
Adaptation steps = 3	0.178±0.006	0.159±0.007	0.102±0.005	0.081±0.003	0.064±0.001	0.050±0.004	0.379±0.021
Adaptation steps = 10	0.161±0.005	0.115±0.002	0.078±0.004	0.063±0.002	0.050±0.001	0.045±0.002	0.919±0.036

MR-MAML Test Loss, lr_b =	0.01	0.03	0.1	0.3	1	3	10
Adaptation steps = 1	0.171±0.003	0.169±0.003	0.163±0.003	0.146±0.002	0.127±0.001	0.128±0.000	0.178±0.008
Adaptation steps = 3	0.170±0.002	0.166±0.001	0.146±0.002	0.128±0.001	0.123±0.001	0.118±0.001	0.163±0.022
Adaptation steps = 10	0.165±0.002	0.152±0.002	0.129±0.001	0.126±0.001	0.118±0.001	0.115±0.001	0.139±0.009

PAC-BUS Test Loss, lr_b =	0.01	0.03	0.1	0.3	1	3	10
Adaptation steps = 1	0.176±0.002	0.171±0.003	0.160±0.001	0.145±0.002	0.127±0.001	0.129±0.002	0.164±0.019
Adaptation steps = 3	0.170±0.002	0.165±0.002	0.145±0.001	0.129±0.002	0.123±0.001	0.120±0.001	0.144±0.014
Adaptation steps = 10	0.163±0.001	0.150±0.001	0.130±0.001	0.126±0.002	0.119±0.002	0.115±0.002	0.130±0.004

Next, we present the computed bounds for MLAP-M [5], MR-MAML [77], and PAC-BUS for the same set of hyper-parameters.

MLAP-M Bound, lr_b =	0.01	0.03	0.1	0.3	1	3	10
Adaptation steps = 1	**1.003±0.000**	1.015±0.001	1.223±0.020	1.946±0.043	3.113±0.154	5.435±0.220	21.874±0.420
Adaptation steps = 3	1.008±0.000	1.087±0.027	1.864±0.062	3.072±0.157	4.147±0.095	6.760±0.233	28.356±1.826
Adaptation steps = 10	1.050±0.006	1.535±0.044	2.574±0.064	4.009±0.107	5.98±0.057	10.119±0.087	47.971±1.346

MR-MAML Bound, lr_b =	0.01	0.03	0.1	0.3	1	3	10
Adaptation steps = 1	0.344±0.002	0.343±0.002	0.335±0.002	0.320±0.001	0.300±0.000	0.303±0.001	0.351±0.006
Adaptation steps = 3	0.344±0.002	0.340±0.002	0.320±0.002	0.302±0.002	0.296±0.001	**0.292±0.001**	0.335±0.018
Adaptation steps = 10	0.339±0.001	0.324±0.002	0.303±0.000	4.752±0.808	5.330±0.187	6.316±0.639	9.134±1.448

PAC-BUS Bound, lr_b =	0.01	0.03	0.1	0.3	1	3	10
Adaptation steps = 1	0.216±0.002	0.216±0.002	0.204±0.002	0.188±0.002	**0.169±0.000**	0.171±0.001	0.207±0.021
Adaptation steps = 3	0.252±0.001	0.247±0.002	0.228±0.002	0.211±0.001	0.204±0.001	0.200±0.002	0.228±0.017
Adaptation steps = 10	0.383±0.002	0.372±0.001	0.350±0.001	1.160±0.093	1.288±0.056	1.650±0.055	2.221±0.256

These results show the dependence that the PAC-BUS upper bound (specifically the uniform stability regularizer term β_{US}) has on the learning rate and number of base-learning update steps whereas the bound for MR-MAML does not suffer with increasing base-learning steps or learning rate. However, once the learning rate and number of adaptation steps are too large, all bounds worsen significantly. The tightest guarantee obtained using PAC-BUS is significantly stronger than those for any tuning of MR-MAML and MLAP-M. We bold the tightest guarantee achieved in the tables above to highlight this.

A.11 Additional Experimental Details

In this section, we report information about the data used, the procedure for prior, train, and test splits, as well as other experimental details. Code capable of reproducing the results in this paper is publicly available at `https://github.com/irom-lab/PAC-BUS`. All results provided in this

paper were computed on an Amazon Web Services (AWS) p2 instances. Tuning and intermediate results were computed on a desktop computer with a 12-core Intel i7-8700k CPU and an NVIDIA Titan Xp GPU. In addition, we made use of several existing software assets: SciKit-learn [50] (BSD license), PyTorch [49] (BSD license), CVXPY [23, 2] (Apache License, Version 2.0), MOSEK [44] (software was used with a personal academic license, see `https://www.mosek.com/products/license-agreement` for more details), learn2learn [7] (MIT License), and h5py [20] (Python license, see `https://docs.h5py.org/en/stable/licenses.html` for more details).

A.11.1 Circle Class

We randomly sample points from the unit ball $B^2(0,1)$ and classify them as $(+)$ or $(-)$ according to whether or not the points are outside the ball $B^2(c_t, r_t)$. For the tasks which are used to train a prior, we sample c_t from $[0.1, 0.5]$ and r_t from $[0.1, 1 - \|c_t\|]$. For the meta-training and meta-testing tasks, we sample c_t from $[0.1, 0.4]$ and r_t from $[0.1, 1 - \|c_t\|]$.

For all methods, we train the prior on 500 tasks, train the network on 10000 tasks, and test on 1000 tasks. We report the meta-test loss and a guarantee on the loss if applicable. A single task is a 2-class 10-sample (i.e. there are 10 samples given in total for training, not 10 samples from each class) learning problem. The evaluation dataset S_{ev} consists of a dataset S of 10 base-learner training samples and a dataset S_{va} of 250 validation samples. For PAC-BUS, we searched for the meta-learning rate in $[1\text{e}{-4}, 1]$, the base-learning rate in $[0.01, 10]$, and the number of base-learning update steps in $[1, 10]$. The resulting parameters for the 10-shot learning problems are: meta-learning rate $1\text{e}{-3}$, base-learning rate 0.05, and 1 base-learning update step. Note that in this example and the *Mini-wiki* example, we select the number of base-learning steps such that the upper bound is minimized. A lower loss may have been achievable with more base-learning update steps, but we aim to produce the tightest bound possible. Training for each method took less than 1 hour on the AWS p2 instance and computing the sample convergence upper bound took approximately 3 days when applicable.

A.11.2 Mini-wiki

In Table 4, we present additional results – the percentage of correctly classified sentences on test tasks (after the base learner's adaptation step). Note that we present these results with the same posterior as was used to generate the results in Table 2.

Table 4: Meta-test accuracy as a percentage for MAML, FLI-Batch, MR-MAML, and PAC-BUS. We report the mean and standard deviation after 5 trials.

4-WAY *Mini-Wiki*	1-SHOT ↑	3-SHOT ↑	5-SHOT ↑
MAML [25]	$\mathbf{60.2 \pm 0.9}$	68.3 ± 0.7	$\mathbf{71.9 \pm 0.6}$
FLI-BATCH [34]	46.0 ± 5.9	48.7 ± 4.9	54.5 ± 2.4
MR-MAML [77]	59.9 ± 0.8	$\mathbf{68.4 \pm 0.7}$	71.8 ± 0.7
PAC-BUS (OURS)	59.9 ± 0.8	68.1 ± 0.7	71.2 ± 0.7

We use the *Mini-wiki* dataset from [34], which consists of 813 classes each with at least 1000 example sentences from that class's corresponding Wikipedia article. The dataset was derived from the Wiki3029 dataset presented in [9], which was created from a public domain (CC0 license) Wikipedia dump. Although the Wikipedia dump is open source, it is possible that content which is copyrighted was used since the datasets are large and it is difficult to moderate all content on the website. In addition, it is possible that the dataset has some offensive content such as derogatory terms or curse words. However, since these are in the context Wikipedia articles, the authors trust that the original article was not written maliciously, but for the purposes of education. We use the first 62 classes of *Mini-wiki* for training the prior, the next 625 for the meta-training, and the last 126 for meta-testing. Before creating learning tasks, we remove all sentences with fewer than 120 characters.

For all methods, we train the prior on 100 tasks, train the network on 1000 tasks, and test on 200 tasks. We report the meta-test score, the meta-test loss, and a guarantee on the loss if applicable. A single task is a 4-class $\{1, 3, 5\}$-shot learning problem. The evaluation dataset S_{ev} consists of a dataset S of $\{1, 3, 5\}$ base-learner training samples and a dataset S_{va} of $\{250, 250, 250\}$ validation samples respectively. For PAC-BUS, we search for the meta-learning rate in $[0.01, 1]$ the base-learning rate in $[1\text{e}{-3}, 100]$, and the number of base-learning update steps in $[1, 50]$. The resulting parameters for the

$\{1, 3, 5\}$-shot learning problems are: meta-learning rate $\{0.1, 0.1, 0.1\}$, base-learning rate $\{2.5, 5, 5\}$, and $\{2, 4, 5\}$ base-learning update steps respectively. Training for each method took less than 1 hour on the AWS p2 instance and computing the sample convergence upper bound took approximately 2 days when applicable.

A.11.3 Omniglot

We use the *Omniglot* dataset from [35], which consists of 1623 characters each with 20 examples. The dataset was collected using Amazon's Mechanical Turk (AMT) and is available on GitHub with an MIT license. This dataset was collected voluntarily by AMT workers. Since the dataset is small enough, it can be checked visually for personally-identifiable information. We use the first 1200 characters for meta-training and the remaining 423 for meta-testing. The image resolution is reduced to 28×28. In the non-mutually exclusive setting, the 1200 training characters are randomly partitioned into 20 equal-sized groups which are assigned a fixed class label from 1 to 20. Note that this is distinct from the method described in [77] where the data is partitioned into 60 disjoint sets. Both experimental setups cause memorization, but the setup used in [77] causes more severe memorization than ours. This is why our implementation of MAML performs better than the results for MAML reported in [77]. However, our implementation of MR-MAML(W) method performs similarly to what is reported in [77].

For all methods, we trained on 100000 batches of 16 tasks and report the meta-test score on 8000 test tasks. We also used 5 base-learning update steps for all methods. A single task is a 20-way $\{1, 5\}$-shot learning problem. The evaluation dataset S_{ev} consists of a dataset S of $\{1, 5\}$ base-learner training samples and a dataset S_{va} of $\{4, 5\}$ validation samples respectively. For PAC-BUS(H), we searched for the regularization scales λ_1 and λ_2 in $[1\mathrm{e}{-7}, 1]$ and $[1\mathrm{e}{-4}, 1\mathrm{e}4]$ respectively. Additionally, the meta-learning rate was selected from $[5\mathrm{e}{-4}, 0.1]$, and the base-learning rate was selected from $[0.01, 10]$. The resulting parameters for the $\{1, 5\}$-shot learning problems are: $\lambda_1 = \{1\mathrm{e}{-3}, 1\mathrm{e}{-4}\}$, $\lambda_2 = \{10, 10\}$, meta-learning rate $\{1\mathrm{e}{-3}, 1\mathrm{e}{-3}\}$, and base-learning rate $\{0.5, 0.5\}$ respectively. Training for each method took approximately 3 days on the AWS p2 instance.