Uniform Sampling over Episode Difficulty

Sébastien M. R. Arnold[1*†] **Guneet S. Dhillon**[2*] **Avinash Ravichandran**[2] **Stefano Soatto**[2,3]
[1]University of Southern California, [2]Amazon Web Services,
[3]University of California, Los Angeles
seb.arnold@usc.edu, {guneetsd, ravinash, soattos}@amazon.com

Abstract

Episodic training is a core ingredient of few-shot learning to train models on tasks with limited labelled data. Despite its success, episodic training remains largely understudied, prompting us to ask the question: what is the best way to sample episodes? In this paper, we first propose a method to approximate episode sampling distributions based on their difficulty. Building on this method, we perform an extensive analysis and find that sampling uniformly over episode difficulty outperforms other sampling schemes, including curriculum and easy-/hard-mining. As the proposed sampling method is algorithm agnostic, we can leverage these insights to improve few-shot learning accuracies across many episodic training algorithms. We demonstrate the efficacy of our method across popular few-shot learning datasets, algorithms, network architectures, and protocols.

1 Introduction

Large amounts of high-quality data have been the key for the success of deep learning algorithms. Furthermore, factors such as data augmentation and sampling affect model performance significantly. Continuously collecting and curating data is a resource (cost, time, storage, etc.) intensive process. Hence, recently, the machine learning community has been exploring methods for performing transfer-learning from large datasets to unseen tasks with limited data.

A popular genre of these approaches is called meta-learning few-shot approaches, where, in addition to the limited data from the task of interest, a large dataset of a disjoint tasks is available for (pre-)training. These approaches are prevalent in the area of computer vision [31] and reinforcement learning [6]. A key component of these methods is the notion of episodic training, which refers to sampling tasks from the larger dataset for training. By learning to solve these tasks correctly, the model can generalize to new tasks.

However, sampling for episodic training remains surprisingly understudied despite numerous methods and applications that build on it. To the best of our knowledge, only a handful of works [69, 55, 33] explicitly considered the consequences of sampling episodes. In comparison, stochastic [46] and mini-batch [4] sampling alternatives have been thoroughly analyzed from the perspectives of optimization [17, 5], information theory [27, 9], and stochastic processes [68, 66], among many others. Building a similar understanding of sampling for episodic training will help theoreticians and practitioners develop improved sampling schemes, and is thus of crucial importance to both.

In this paper, we explore many sampling schemes to understand their impact on few-shot methods. Our work revolves around the following fundamental question: *what is the best way to sample episodes?* Our focus will be restricted to image classification in the few-shot learning setting – where "best" is taken to mean "higher transfer accuracy of unseen episodes" – and leave analyses and applications to other areas for future work.

[*]Equal contributions
[†]Work done while at Amazon Web Services

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

Contrary to prior work, our experiments indicate that sampling uniformly with respect to *episode difficulty* yields higher classification accuracy — a scheme originally proposed to regularize metric learning [62]. To better understand these results, we take a closer look at the properties of episodes and what makes them difficult. Building on this understanding, we propose a method to approximate different sampling schemes, and demonstrate its efficacy on several standard few-shot learning algorithms and datasets.

Concretely, we make the following contributions:

- We provide a detailed empirical analysis of episodes and their difficulty. When sampled randomly, we show that episode difficulty (approximately) follows a normal distribution and that the difficulty of an episode is largely independent of several modeling choices including the training algorithm, the network architecture, and the training iteration.
- Leveraging our analysis, we propose *simple and universally applicable* modifications to the episodic sampling pipeline to approximate *any* sampling scheme. We then use this scheme to thoroughly compare episode sampling schemes – including easy/hard-mining, curriculum learning, and uniform sampling – and report that sampling uniformly over episode difficulty yields the best results.
- Finally, we show that *sampling matters for few-shot classification* as it improves transfer accuracy for a diverse set of popular [53, 20, 15, 41] and state-of-the-art [64] algorithms on standard and cross-domain benchmarks.

2 Preliminaries

2.1 Episodic sampling and training

We define episodic sampling as subsampling few-shot tasks (or episodes) from a larger *base* dataset [7]. Assuming the base dataset admits a generative distribution, we sample an episode in two steps[3]. First, we sample the episode classes $\mathcal{C}_\tau$ from class distribution $p(\mathcal{C}_\tau)$; second, we sample the episode's data from data distribution $p(x, y \mid \mathcal{C}_\tau)$ conditioned on $\mathcal{C}_\tau$. This gives rise to the following log-likelihood for a model l_θ parameterized by θ:

$$\mathcal{L}(\theta) = \mathop{\mathrm{E}}_{\tau \sim q(\cdot)} \left[\log l_\theta(\tau) \right], \tag{1}$$

where $q(\tau)$ is the episode distribution induced by first sampling classes, then data. In practice, this expectation is approximated by sampling a batch of episodes $\mathcal{B}$ each with their set τ_Q of *query* samples. To enable transfer to unseen classes, it is also common to include a small set τ_S of *support* samples to provide statistics about τ. This results in the following Monte-Carlo estimator:

$$\mathcal{L}(\theta) \approx \frac{1}{|\mathcal{B}|} \sum_{\tau \in \mathcal{B}} \frac{1}{|\tau_Q|} \sum_{(x,y) \in \tau_Q} \log l_\theta(y \mid x, \tau_S), \tag{2}$$

where the data in τ_Q and τ_S are both distributed according to $p(x, y \mid \mathcal{C}_\tau)$. In few-shot image classification, the n-way k-shot setting corresponds to sampling $n = |\mathcal{C}_\tau|$ classes, each with $k = |\tau_S|$ support data points. The implicit assumption in the vast majority of few-shot methods is that both classes and data are sampled with *uniform* probability – *but are there better alternatives?* We carefully examines those underlying assumptions in the forthcoming sections.

2.2 Few-shot algorithms

We briefly present a few representative episodic learning algorithms. A more comprehensive treatment of few-shot algorithms is presented in Wang et al. [59] and Hospedales et al. [24]. A core question in few-shot learning lies in evaluating (and maximizing) the model likelihood l_θ. These algorithms can be divided in two major families: gradient-based methods, which adapt the model's parameters to the episode; and metric-based methods, which compute similarities between support and query samples in a learned embedding space.

Gradient-based few-shot methods are best illustrated through Model-Agnostic Meta-Learning [15] (MAML). The intuition behind MAML is to learn a set of initial parameters which can quickly

[3] We use the notation for a probability distribution and its probability density function interchangeably.

specialize to the task at hand. To that end, MAML computes l_θ by adapting the model parameters θ via one (or more) steps of gradient ascent and then computes the likelihood using the adapted parameters θ'. Concretely, we first compute the likelihood $p_\theta(y \mid x)$ using the support set τ_S, adapt the model parameters, and then evaluate the likelihood:

$$l_\theta(y \mid x, \tau_S) = p_{\theta'}(y \mid x) \quad \text{s.t.} \quad \theta' = \theta + \alpha \nabla_\theta \sum_{(x,y)\in\tau_S} \log p_\theta(y \mid x),$$

where $\alpha > 0$ is known as the adaptation learning rate. A major drawback of training with MAML lies in back-propagating through the adaptation phase, which requires higher-order gradients. To alleviate this computational burden, Almost No Inner Loop [41] (ANIL) proposes to only adapt the last classification layer of the model architecture while tying the rest of the layers across episodes. They empirically demonstrate little classification accuracy drop while accelerating training times four-fold.

Akin to ANIL, metric-based methods also share most of their parameters across tasks; however, their aim is to learn a metric space where classes naturally cluster. To that end, metric-based algorithms learn a feature extractor ϕ_θ parameterized by θ and classify according to a non-parametric rule. A representative of this family is Prototypical Network [53] (ProtoNet), which classifies query points according to their distance to *class prototypes* — the average embedding of a class in the support set:

$$l_\theta(y \mid x, \tau_S) = \frac{\exp\left(-d(\phi_\theta(x), \phi_\theta^y)\right)}{\sum_{y'\in\mathcal{C}_\tau} \exp\left(-d(\phi_\theta(x), \phi_\theta^{y'})\right)} \quad \text{s.t.} \quad \phi_\theta^c = \frac{1}{k} \sum_{\substack{(x,y)\in\tau_S \\ y=c}} \phi_\theta(x),$$

where $d(\cdot,\cdot)$ is a distance function such as the Euclidean distance or the negative cosine similarity, and ϕ_θ^c is the class prototype for class c. Other classification rules include support vector clustering [32], neighborhood component analysis [30], and the earth-mover distance [67].

2.3 Episode difficulty

Given an episode τ and likelihood function l_θ, we define *episode difficulty* to be the negative log-likelihood incurred on that episode:

$$\Omega_{l_\theta}(\tau) = -\log l_\theta(\tau),$$

which is a surrogate for how hard it is to classify the samples in τ_Q correctly, given l_θ and τ_S. By definition, this choice of episode difficulty is tied to the choice of the likelihood function l_θ.

Dhillon et al. [11] use a similar surrogate as a means to systematically report few-shot performances. We use this definition because it is equivalent to the loss associated with the likelihood function l_θ on episode τ, which is readily available at training time.

3 Methodology

In this section, we describe the core assumptions and methodology used in our study of sampling methods for episodic training. Our proposed method builds on importance sampling [21] (IS) which we found compelling for three reasons: (i) IS is *well understood* and solidly grounded from a theoretical standpoint, (ii) IS is *universally applicable* thus compatible with all episodic training algorithms, and (iii) IS is *simple to implement* with little requirement for hyper-parameter tuning.

Why should we care about episodic sampling? A back-of-the-envelope calculation[4] suggests that there are on the order of 10^{162} different training episodes for the smallest-scale experiments in Section 5. Since iterating through each of them is infeasible, we ought to express some preference over which episodes to sample. In the following, we describe a method that allows us to specify this preference.

3.1 Importance sampling for episodic training

Let us assume that the sampling scheme described in Section 2.1 induces a distribution $q(\tau)$ over episodes. We call it the *proposal distribution*, and assume knowledge of its density function. We wish

[4]For a base dataset with K classes and N input-output pairs per class, there are a total of $\binom{K}{k}\binom{N}{n}^k$ possible episodes that can be created when sampling n pairs each from k classes.

Algorithm 1: Episodic training with Importance Sampling

Input: target (p) and proposal (q) distributions, likelihood function l_θ, optimizer OPT.
Randomly initialize model parameters θ.
repeat
  Sample a mini-batch $\mathcal{B}$ of episodes from $q(\tau)$.
  for each episode τ in mini-batch $\mathcal{B}$ **do**
    Compute episode likelihood: $l_\theta(\tau)$.
    Compute importance weight: $w(\tau) = \frac{p(\tau)}{q(\tau)}$.
  end for
  Aggregate: $\mathcal{L}(\theta) \leftarrow \sum_{\tau \in \mathcal{B}} w(\tau) \log l_\theta(\tau)$.
  Compute effective sample size $\text{ESS}(\mathcal{B})$.
  Update model parameters: $\theta \leftarrow \text{OPT}(\frac{\mathcal{L}(\theta)}{\text{ESS}(\mathcal{B})})$.
until parameters θ have converged.

to estimate the expectation in Eq. (1) when sampling episodes according to a *target distribution* $p(\tau)$ of our choice, rather than $q(\tau)$. To that end, we can use an importance sampling estimator which simply re-weights the observed values for a given episode τ by $w(\tau) = \frac{p(\tau)}{q(\tau)}$, the ratio of the target and proposal distributions:

$$\mathop{\mathrm{E}}_{\tau \sim p(\cdot)} \left[\log l_\theta(\tau)\right] = \mathop{\mathrm{E}}_{\tau \sim q(\cdot)} \left[w(\tau) \log l_\theta(\tau)\right].$$

The importance sampling identity holds whenever $q(\tau)$ has non-zero density over the support of $p(\tau)$, and effectively allows us to sample from *any* target distribution $p(\tau)$.

A practical issue of the IS estimator arises when some values of $w(\tau)$ become much larger than others; in that case, the likelihoods $l_\theta(\tau)$ associated with mini-batches containing heavier weights dominate the others, leading to disparities. To account for this effect, we can replace the mini-batch average in the Monte-Carlo estimate of Eq. (2) by the *effective sample size* $\text{ESS}(\mathcal{B})$ [29, 34]:

$$\mathop{\mathrm{E}}_{\tau \sim p(\cdot)} \left[\log l_\theta(\tau)\right] \approx \frac{1}{\text{ESS}(\mathcal{B})} \sum_{\tau \in \mathcal{B}} w(\tau) \log l_\theta(\tau) \quad \text{s.t.} \quad \text{ESS}(\mathcal{B}) = \frac{\left(\sum_{\tau \in \mathcal{B}} w(\tau)\right)^2}{\sum_{\tau \in \mathcal{B}} w(\tau)^2}, \tag{3}$$

where $\mathcal{B}$ denotes a mini-batch of episodes sampled according to $q(\tau)$. Note that when $w(\tau)$ is constant, we recover the standard mini-batch average setting as $\text{ESS}(\mathcal{B}) = |\mathcal{B}|$. Empirically, we observed that normalizing with the effective sample size avoided instabilities. This method is summarized in Algorithm 1.

3.2 Modeling the proposal distribution

A priori, we do not have access to the proposal distribution $q(\tau)$ (nor its density) and thus need to estimate it empirically. Our main assumption is that sampling episodes from $q(\tau)$ induces a normal distribution over episode difficulty. With this assumption, we model the proposal distribution by this induced distribution, therefore replacing $q(\tau)$ with $\mathcal{N}(\Omega_{l_\theta}(\tau) \mid \mu, \sigma^2)$ where μ, σ^2 are the mean and variance parameters. As we will see in Section 5.2, this normality assumption is experimentally supported on various datasets, algorithms, and architectures.

We consider two settings for the estimation of μ and σ^2: offline and online. The *offline* setting consists of sampling $1,000$ training episodes before training, and computing μ, σ^2 using a model pre-trained on the same base dataset. Though this setting seems unrealistic, *i.e.* having access to a pre-trained model, several meta-learning few-shot methods start with a pre-trained model which they further build upon. Hence, for such methods there is no overhead. For the *online* setting, we estimate the parameters on-the-fly using the model currently being trained. This is justified by the analysis in Section 5.2 which shows that episode difficulty transfers across model parameters during training. We update our estimates of μ, σ^2 with an exponential moving average:

$$\mu \leftarrow \lambda\mu + (1-\lambda)\Omega_{l_\theta}(\tau) \quad \text{and} \quad \sigma^2 \leftarrow \lambda\sigma^2 + (1-\lambda)(\Omega_{l_\theta}(\tau) - \mu)^2,$$

where $\lambda \in [0, 1]$ controls the adjustment rate of the estimates, and the initial values of μ, σ^2 are computed in a warm-up phase lasting 100 iterations. Keeping $\lambda = 0.9$ worked well for all our

experiments (Section 5). We opted for this simple implementation as more sophisticated approaches like West [61] yielded little to no benefit.

3.3 Modeling the target distribution

Similar to the proposal distribution, we model the target distribution by its induced distribution over episode difficulty. Our experiments compare four different approaches, all of which share parameters μ, σ^2 with the normal model of the proposal distribution. For numerical stability, we truncate the support of all distributions to $[\mu - 2.58\sigma, \mu + 2.58\sigma]$, which gives approximately 99% coverage for the normal distribution centered around μ.

The first approach (HARD) takes inspiration from hard negative mining [51], where we wish to sample only more challenging episodes. The second approach (EASY) takes a similar view but instead only samples easier episodes. We can model both distributions as follows:

$$\mathcal{U}(\Omega_{l_\theta}(\tau) \mid \mu, \mu + 2.58\sigma) \tag{HARD}$$

and

$$\mathcal{U}(\Omega_{l_\theta}(\tau) \mid \mu - 2.58\sigma, \mu) \tag{EASY}$$

where $\mathcal{U}$ denotes the uniform distribution. The third (CURRICULUM) is motivated by curriculum learning [2], which slowly increases the likelihood of sampling more difficult episodes:

$$\mathcal{N}(\Omega_{l_\theta}(\tau) \mid \mu_t, \sigma^2) \tag{CURRICULUM}$$

where μ_t is linearly interpolated from $\mu - 2.58\sigma$ to $\mu + 2.58\sigma$ as training progresses. Finally, our fourth approach, UNIFORM, resembles distance weighted sampling [62] and consists of sampling uniformly over episode difficulty:

$$\mathcal{U}(\Omega_{l_\theta}(\tau) \mid \mu - 2.58\sigma, \mu + 2.58\sigma). \tag{UNIFORM}$$

Intuitively, UNIFORM can be understood as a uniform prior over unseen test episodes, with the intention of performing well across the entire difficulty spectrum. This acts as a regularizer, forcing the model to be equally discriminative for both easy and hard episodes.

4 Related Works

This paper studies task sampling in the context of few-shot [36, 14] and meta-learning [49, 56].

Few-shot learning. This settings has received a lot of attention over recent years [58, 43, 47, 18]. Broadly speaking, state-of-the-art methods can be categorized in two major families: metric-based and gradient-based.

Metric-based methods learn a shared feature extractor which is used to compute the distance between samples in embedding space [53, 3, 44, 30]. The choice of metric mostly differentiates one method from another; for example, popular choices include Euclidean distance [53], cosine similarity [20], support vector machines [32], set-to-set functions [64], or the earth-mover distance [67].

Gradient-based algorithms such as MAML [15], propose an objective to learn a network initialization that can quickly adapt to new tasks. Due to its minimal assumptions, MAML has been extended to probabilistic formulations [22, 65] to incorporate learned optimizers — implicit [16] or explicit [40] — and simplified to avoid expensive second-order computations [37, 42]. In that line of work, ANIL [41] claims to match MAML's performance when adapting only the last classification layer – thus greatly reducing the computational burden and bringing gradient and metric-based methods closer together.

Sampling strategies. Sampling strategies have been studied for different training regimes. Wu et al. [62] demonstrated that "sampling matters" in the context of metric learning. They propose to sample a triplet with probability proportional to the distance of its positive and negative samples, and observe stabilized training and improved accuracy. This observation was echoed by Katharopoulos and Fleuret [27] when sampling mini-batches: carefully choosing the constituting samples in the mini-batch improves convergence rate and asymptotic performance. Like ours, their method builds on importance sampling [52, 12, 26] but whereas they compute importance weights using the magnitude of the model's gradients, we use the episode's difficulty. Similar insights were also observed in

reinforcement learning, where Schaul et al. [48] suggests a scheme to sample transitions according to the temporal difference error.

Closer to our work, Sun et al. [55] present a hard-mining scheme where the most challenging classes across episodes are pooled together and used to create new episodes. Observing that the difficulty of a class is intrinsically linked to the other classes in the episode, Liu et al. [33] propose a mechanism to track the difficulty across every class pair. They use this mechanism to build a curriculum [2, 63] of increasingly difficult episodes. In contrast to these two approaches, our proposed method makes use of importance sampling to mimic the target distribution rather than sampling from it directly. This helps achieve fast and efficient sampling without any pre-processing requirements.

5 Experiments

We first validate the assumptions underlying our proposed IS estimator and shed light on the properties of episode difficulty. Then, we answer the question we pose in the introduction, namely: *what is the best way to sample episodes?* Finally, we ask if better sampling improves few-shot classification.

5.1 Experimental setup

We review the standardized few-shot benchmarks and provide a detailed description in the Appendix.

Datasets. We use two standardized image classification datasets, Mini-ImageNet [58] and Tiered-ImageNet [45], both subsets of ImageNet [10]. Mini-ImageNet consists of 64 classes for training, 16 for validation, and 20 for testing; we use the class splits introduced by Ravi and Larochelle [43]. Tiered-ImageNet contains 608 classes split into 351, 97, and 160 for training, validation, and testing, respectively.

Network architectures. We train two different models, a 4-layer convolution network conv$(64)_4$ Vinyals et al. [58] with 64 channels per layer. We also use ResNet-12, a 12-layer deep residual network [23], originally introduced in Oreshkin et al. [39]. Both architectures use batch normalization [25] after every convolutional layer and ReLU as the non-linearity.

Training algorithms. For the metric-based family, we use ProtoNet with Euclidean [53] and scaled negative cosine similarity measures [20]. Additionally, we use MAML [15] and ANIL [41] as representative gradient-based algorithms.

Hyper-parameters. We tune hyper-parameters for each algorithm and dataset to work well across different few-shot settings and network architectures. Additionally, we keep the hyper-parameters the same across all different sampling methods for a fair comparison. We train for 20k iterations with a mini-batch of size 16 and 32 for Mini-ImageNet and Tiered-ImageNet respectively, and validate every 1k iterations on 1k episodes. The best performing model is finally evaluated on 1k test episodes.

5.2 Understanding episode difficulty

All the models in this section are trained using baseline sampling, *i.e.*, episodic training without importance sampling as described in Section 2.

5.2.1 Episode difficulty is approximately normally distributed

We begin our analysis by verifying that the distribution over episode difficulty induced by $q(\tau)$ is approximately normal. In Fig. 1, we show a density plot for the difficulty of 10k test episodes sampled with $q(\tau)$. The difficulties are computed using conv$(64)_4$ trained with ProtoNet and MAML on Mini-ImageNet for 1-shot 5-way classification.

Episode difficulty follows a bell curve, which is naturally modeled with a normal distribution. Fig. 1 also includes Q-Q plots, typically used to assess normality: the closer the curve is to the identity line, the closer the distribution is to a normal. Finally, we compute the Shapiro-Wilk test for normality [50], which tests for the null hypothesis that the data is drawn from a normal distribution. Since the p-value

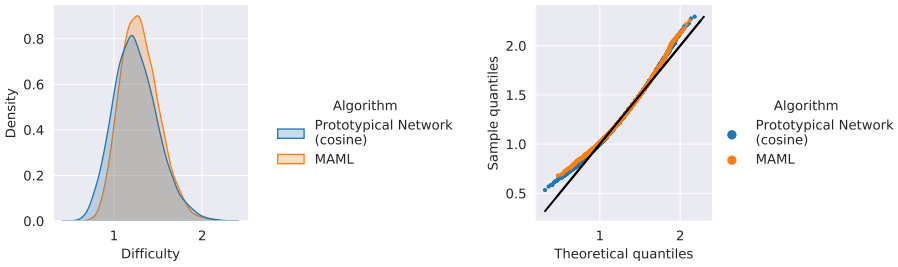

Figure 1: **Episode difficulty is approximately normally distributed.** Density (left) and Q-Q (right) plots of the episode difficulty computed by conv$(64)_4$'s on Mini-ImageNet (1-shot 5-way), trained using ProtoNets (cosine) and MAML (depicted in the legends). The values are computed over 10k test episodes. The density plots follow a bell curve, with the density peak in the middle, which quickly drops-off on either side of the peak. The Q-Q plots are close to the identity line (in black). The closer the curve is to the identity line, the closer the distribution is to a normal. Both suggest that the episode difficulty distribution can be normally approximated.

for this test is sensitive to the sample size[5], we subsample 50 values 100 times and average rejection rates over these subsets. With $\alpha = 0.05$, the null hypothesis is rejected 14% and 17% of the time for Mini-ImageNet and Tiered-ImageNet respectively, thus suggesting that episode difficulty can be reliably approximated with a normal distribution.

5.2.2 Independence from modeling choices

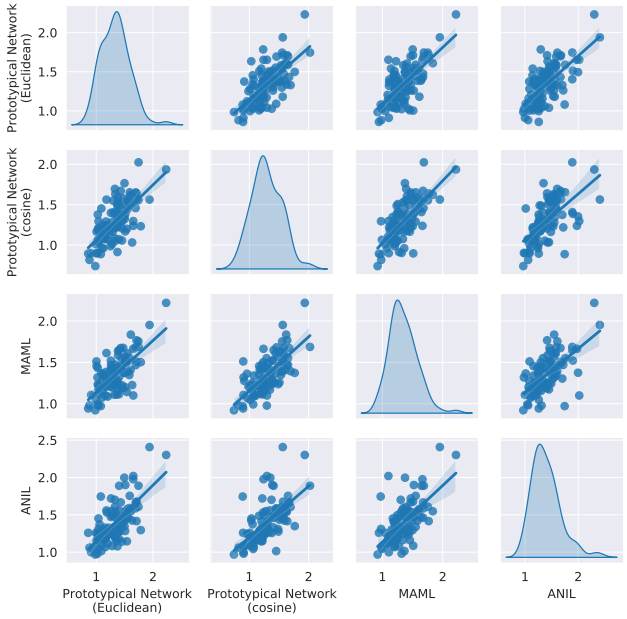

Figure 2: **Episode difficulty transfers across training algorithms.** Scatter plots (with regression lines) of the episode difficulty computed on 1k Mini-ImageNet test episodes (1-shot 5-way) by conv$(64)_4$'s trained using different algorithms. The positive correlation suggests that an episode that is difficult for one training algorithm will be difficult for another.

By definition, the notion of episode difficulty is tightly coupled to the model likelihood l_θ (Section 2), and hence to the modeling variables such as learning algorithm, network architecture, and model parameters. We check if episode difficulty transfers across different choices for these variables. We are concerned with the *relative ranking* of the episode difficulty and not the actual values. To this end, we will use the Spearman rank-order correlation coefficient, a non-parametric measure of the monotonicity of the relationship between two sets of values. This value lies within $[-1; 1]$, with 0 implying no correlation and $+1$ and -1 implying exact positive and negative correlations, respectively.

Training algorithm. Firstly, we check the dependence on the training algorithm. We use all four algorithms to train conv$(64)_4$'s for 1-shot 5-way classification on Mini-ImageNet, then compute episode difficulty over 10k test episodes. The Spearman rank-order correlation coefficients for the difficulty values computed with respect to all possible pairs of training algorithms are > 0.65. This positive correlation is illustrated in Fig. 2 and suggests that an episode that is difficult for one training algorithm is very likely to be difficult for another.

Network architecture. Next we analyze the dependence on the network architecture. We use conv$(64)_4$ and ResNet-12's trained on Mini-ImageNet (1-shot 5-way) with all training algorithms.

[5]For a large sample size, the p-values are not reliable as they may detect trivial departures from normality.

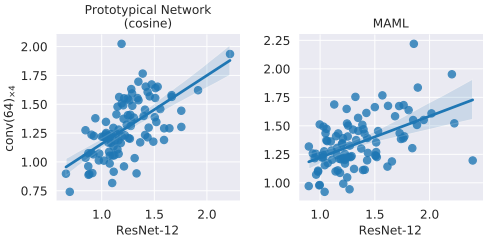

Figure 3: **Episode difficulty transfers across network architectures.** Scatter-plots (with regression lines) of the episode difficulty computed by conv$(64)_4$ and ResNet-12's trained using different algorithms. This is computed for 1k 1-shot 5-way test episodes from Mini-ImageNet. We observe a strong positive correlation between the computed values for both network architectures.

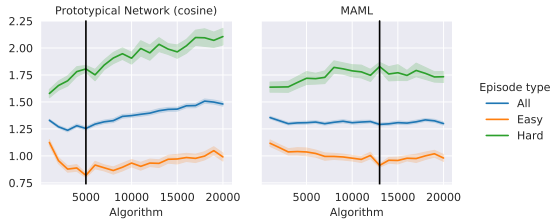

Figure 4: **Episode difficulty is transferred across model parameters during training.** We select the 50 easier and harder episodes and track their difficulty throughout training. The average difficulty of the episodes decreases over time, until convergence (vertical line), after which the model overfits. Additionally, easier episodes remain easy while harder episodes remain hard, indicating that episode difficulty transfers from one set of parameters to the next.

We compute the episode difficulties for 10k test episodes and compute their Spearman rank-order correlation coefficients across the two architectures, for a given algorithm. The correlation coefficients are 0.57 for ProtoNet (Euclidean), 0.72 for ProtoNet (cosine), 0.58 for MAML, and 0.49 for ANIL. Fig. 3 illustrates this positive correlation, suggesting that episode difficulty is transferred across network architectures with high probability.

Model parameters during training. Lastly, we study the dependence on model parameters during training. We select the 50 *easiest* and 50 *hardest* episodes, *i.e.*, episodes with the lowest and highest difficulty respectively, from 1k test episodes. We track the episode difficulty for all episodes over the training phase and visualize the trend in Fig. 4 for conv$(64)_4$'s trained using ProtoNet (Euclidean and cosine), MAML and ANIL on Mini-ImageNet (1-shot 5-way). Throughout training, easy episodes remain easy and hard episodes remain hard, hence suggesting that episode difficulty transfers across different model parameters during training. Since the episode difficulty does not change drastically during the training process, we can estimate it with a running average over the training iterations. This justifies the *online* modeling of the proposal distribution in Section 3.1.

Table 1: **Few-shot accuracies on benchmark datasets for 5-way few-shot episodes in the offline setting.** Mean accuracy and 95% confidence interval computed over 1k test episodes. Best results for a fixed scenario are shown in bold, † indicates matching or improving over baseline sampling.

	Mini-ImageNet				Tiered-ImageNet	
	conv$(64)_4$		ResNet-12		ResNet-12	
	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)
ProtoNet (cosine)	**50.03±0.61**	61.56±0.53	52.85±0.64	62.11±0.52	**60.01±0.73**	72.75±0.59
+ EASY	**49.60±0.61**†	65.17±0.53†	53.35±0.63†	63.55±0.53†	**60.03±0.75**†	74.65±0.57†
+ HARD	49.01±0.60	**66.45±0.50**†	52.65±0.63†	70.15±0.51†	55.44±0.72	75.97±0.55†
+ CURRICULUM	49.38±0.61	64.12±0.53†	53.21±0.65†	65.89±0.52†	**60.37±0.76**†	75.32±0.58†
+ UNIFORM	**50.07±0.59**†	**66.33±0.52**†	**54.27±0.65**†	**70.85±0.51**†	**60.27±0.75**†	**78.36±0.54**†

5.3 Comparing episode sampling methods

We compare different methods for episode sampling. To ensure fair comparisons, we use the offline formulation (Section 3.3) so that all sampling methods share the same pre-trained network (the network trained using baseline sampling) when computing proposal likelihoods. We compute results over 2 datasets, 2 network architectures, 4 algorithms and 2 few-shot protocols, totaling in 24 scenarios. Table 1 presents results on ProtoNet (cosine), while the rest are in Appendix C.

We observe that, although not strictly dominant, UNIFORM tends to outperform other methods as it is within the statistical confidence of the best method in $19/24$ scenarios. For the $5/24$ scenarios

Table 2: **Few-shot accuracies on benchmark datasets for 5-way few-shot episodes in the offline and online settings.** The mean accuracy and the 95% confidence interval are reported for evaluation done over $1,000$ test episodes. Best results for a fixed scenario are shown in bold. Results where a sampling technique is better than or comparable to baseline sampling are denoted by †. The first row in every scenario denotes baseline sampling. UNIFORM (Online) retains most of the performance of the offline formulation while being significantly easier to implement (online is competitive in $15/24$ scenarios vs $16/24$ for offline).

	Mini-ImageNet				Tiered-ImageNet	
	conv$(64)_4$		ResNet-12		ResNet-12	
	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)
ProtoNet (Euclidean)	**49.06±0.60**	65.28±0.52	49.67±0.64	67.45±0.51	**59.10±0.73**	76.95±0.56
+ UNIFORM (Offline)	48.19±0.62	66.73±0.52†	**53.94±0.63**†	**70.79±0.49**†	58.63±0.76†	**78.62±0.55**†
+ UNIFORM (Online)	48.39±0.62	**67.86±0.50**†	52.97±0.64†	**70.63±0.50**†	**59.67±0.70**†	**78.73±0.55**†
ProtoNet (cosine)	**50.03±0.61**	61.56±0.53	52.85±0.64	62.11±0.52	60.01±0.73	72.75±0.59
+ UNIFORM (Offline)	**50.07±0.59**†	**66.33±0.52**†	**54.27±0.65**†	**70.85±0.51**†	60.27±0.75†	**78.36±0.54**†
+ UNIFORM (Online)	**50.06±0.61**†	**65.99±0.52**†	**53.90±0.63**†	68.78±0.51†	**61.37±0.72**†	77.81±0.56†
MAML	**46.88±0.60**	55.16±0.55	49.92±0.65	63.93±0.59	55.37±0.74	**72.93±0.60**
+ UNIFORM (Offline)	**46.67±0.63**†	**62.09±0.55**†	**52.65±0.65**†	**66.76±0.57**†	54.58±0.77	72.00±0.66
+ UNIFORM (Online)	**46.70±0.61**†	**61.62±0.54**†	51.17±0.68†	65.63±0.57†	**57.15±0.74**†	71.67±0.67
ANIL	**46.59±0.60**	**63.47±0.55**	**49.65±0.65**	59.51±0.56	**54.77±0.76**	69.28±0.67
+ UNIFORM (Offline)	**46.93±0.62**†	62.75±0.60	**49.56±0.62**†	**64.72±0.60**†	54.15±0.79†	**70.44±0.69**†
+ UNIFORM (Online)	**46.82±0.63**†	62.63±0.59	**49.82±0.68**†	**64.51±0.62**†	**55.18±0.74**†	69.55±0.71†

where UNIFORM underperforms, it closely trails behind the best methods: the average degradation is -0.58%, and at most -1.44% (ignoring the standard deviations). Conversely, it boosts accuracy over baseline sampling by as much as 8.74% and on average by 3.17% (ignoring the standard deviations). We attribute this overall good performance to the fact that uniform sampling puts a uniform distribution prior over the (unseen) test episodes, with the intention of performing well across the entire difficulty spectrum. This acts as a regularizer, forcing the model to be equally discriminative for easy and hard episodes. If we knew the test episode distribution, upweighting episodes that are most likely under that distribution will improve transfer accuracy [13]. However, this uninformative prior is the safest choice without additional information about the test episodes.

Second best is baseline sampling as it is statistically competitive on $10/24$ scenarios, while EASY, HARD, and CURRICULUM only appear among the better methods in 4, 4, and 9 scenarios, respectively.

5.4 Online approximation of the proposal distribution

Although the offline formulation is better suited for analysis experiments, it is expensive as it requires a pretraining phase for the proposal network and 2 forward passes during episodic training (one for the episode loss, another for the proposal density). In this section, we show that the online formulation faithfully approximates offline sampling and can retain most of the performance improvements from UNIFORM. We take the same 24 scenarios as in the previous subsection, and compare baseline sampling against offline and online UNIFORM. Table 2 reports the full suite of results.

We observe that baseline is statistically competitive on $8/24$ scenarios; on the other hand, offline and online UNIFORM perform similarly in aggregate, as they are within the best results in $16/24$ and $15/24$ scenarios respectively. Similar to its offline counterpart, online UNIFORM does better than or comparable to baseline sampling in 21 out of 24 scenarios. On the $3/24$ scenarios where online UNIFORM underperforms compared to baseline sampling, the average degradation is -0.92%, and at most -1.26% (ignoring the standard deviations). Conversely, it boosts accuracy over baseline sampling by as much as 6.67% and on average by 2.24% (ignoring the standard deviations). Therefore, using online UNIFORM, while computationally comparable to baseline sampling, results in a boost in few-shot performance; when it underperforms, it trails closely. We also compute the mean accuracy difference between the offline and online formulation, which is $0.07\% \pm 0.35$ accuracy points. This confirms that both the offline and online methods produce quantitatively similar outcomes.

Table 3: **Accuracies for cross-domain episodes after training on 5-way Mini-ImageNet episodes.** The mean accuracy and the 95% confidence interval are reported for evaluation done over $1,000$ test episodes. Best results for a fixed scenario are shown in bold. Results where a sampling technique is better than or comparable to baseline sampling are denoted by †.

	CUB-200		Describable Textures	
	ResNet-12		conv$(64)_4$	
	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)
ProtoNet (cosine)	38.67±0.60	49.75±0.57	32.09±0.45	38.44±0.41
+ UNIFORM (Online)	**40.55±0.60**†	**56.30±0.55**†	**33.63±0.47**†	**43.28±0.44**†
MAML	35.80±0.56	45.16±0.62	29.47±0.46	37.85±0.47
+ UNIFORM (Online)	**37.18±0.55**†	**46.58±0.58**†	**31.84±0.49**†	**40.81±0.44**†

5.5 Better sampling improves cross-domain transfer

To further validate the role of episode sampling as a way to improve generalization, we evaluate the models trained in the previous section on episodes from completely different domains. Specifically, we train models on Mini-ImageNet 5-way episodes and evaluate them on the test episodes of CUB-200 [60], Describable Textures [8], FGVC Aircrafts [35], and VGG Flowers [38], following the splits of Triantafillou et al. [57]. Table 3 displays results for ProtoNet (cosine) and MAML on CUB-200 and Describable Textures episodes, with the full suite available in the Appendix. Out of the 64 total cross-domain scenarios, UNIFORM does statistically better in $49/64$ scenarios, comparable in $12/64$ scenarios and worse in only $3/64$ scenarios. These results further go to show that sampling matters in episodic training.

Table 4: **Few-shot accuracies on benchmark datasets for 5-way few-shot episodes using FEAT.** The mean accuracy and the 95% confidence interval are reported for evaluation done over $10,000$ test episodes with a ResNet-12. UNIFORM (Online) improves FEAT's accuracy in $3/4$ scenarios, demonstrating that sampling matters even for state-of-the-art few-shot methods.

	Mini-ImageNet		Tiered-ImageNet	
	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)
FEAT	66.02±0.20	81.17±0.14	**70.50±0.23**	84.26±0.16
+ UNIFORM (Online)	**66.27±0.20**	**81.54±0.14**	**70.61±0.23**	**84.42±0.16**

5.6 Better sampling improves few-shot classification

The results in the previous subsection suggest that online UNIFORM yields a *simple and universally applicable* method to improve episode sampling. To validate that state-of-the-art methods can also benefit from better sampling, we take the recently proposed FEAT [64] algorithm and augment it with our IS-based implementation of online UNIFORM. Concretely, we use their open-source implementation[6] to train both baseline and UNIFORM sampling. We use the prescribed hyper-parameters without any modifications. Results for ResNet-12 on Mini-ImageNet and Tiered-ImageNet are reported in Table 4, where online UNIFORM outperforms baseline sampling on $3/4$ scenerios and is matched on the remaining one. Thus, better episodic sampling can improve few-shot classification even for the very best methods.

6 Conclusion

This manuscript presents a careful study of sampling in the context of few-shot learning, with an eye on episodes and their difficulty. Following an empirical study of difficulty, we propose an importance sampling-based method to compare different episode sampling schemes. Our experiments suggests that sampling uniformly over difficulty performs best across datasets, training algorithms, network architectures and few-shot protocols. Avenues for future work include devising better sampling strategies, analysis beyond few-shot classification (*e.g.*, regression, reinforcement learning), and a theoretical grounding explaining our observations.

[6] Available at: `https://github.com/Sha-Lab/FEAT`

References

[1] S. M. R. Arnold, P. Mahajan, D. Datta, I. Bunner, and K. S. Zarkias. learn2learn: A library for Meta-Learning research. Aug. 2020. URL http://arxiv.org/abs/2008.12284.

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, June 2009. Association for Computing Machinery. URL https://doi.org/10.1145/1553374.1553380.

[3] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. Sept. 2018. URL https://openreview.net/pdf?id=HyxnZh0ct7.

[4] D. P. Bertsekas and J. N. Tsitsiklis. Neuro-Dynamic programming. 27(6), Jan. 1996. URL https://www.researchgate.net/publication/216722122_Neuro-Dynamic_Programming.

[5] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. June 2016. URL http://arxiv.org/abs/1606.04838.

[6] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422, 2019.

[7] W.-L. Chao, H.-J. Ye, D.-C. Zhan, M. Campbell, and K. Q. Weinberger. Revisiting meta-learning as supervised learning. *arXiv preprint arXiv:2002.00573*, 2020.

[8] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014.

[9] D. Csiba and P. Richtárik. Importance sampling for minibatches. *J. Mach. Learn. Res.*, 19(27):1–21, 2018. URL http://jmlr.org/papers/v19/16-241.html.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto. A baseline for few-shot image classification. In *International Conference on Learning Representations*, 2019.

[12] A. Doucet, N. d. Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York, NY, 2001. URL https://link.springer.com/book/10.1007/978-1-4757-3437-9.

[13] A. Fallah, A. Mokhtari, and A. Ozdaglar. Generalization of model-agnostic meta-learning algorithms: Recurring and unseen tasks. *arXiv preprint arXiv:2102.03832*, 2021.

[14] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, Apr. 2006. URL http://dx.doi.org/10.1109/TPAMI.2006.79.

[15] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[16] S. Flennerhag, A. A. Rusu, R. Pascanu, H. Yin, and R. Hadsell. Meta-Learning with warped gradient descent. Aug. 2019. URL http://arxiv.org/abs/1909.00025.

[17] M. P. Friedlander and M. Schmidt. Hybrid Deterministic-Stochastic methods for data fitting. *SIAM J. Sci. Comput.*, 34(3):A1380–A1405, Jan. 2012. URL https://doi.org/10.1137/110830629.

[18] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. M. A. Eslami. Conditional neural processes. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713, Stockholmsmässan, Stockholm Sweden, 2018. PMLR. URL http://proceedings.mlr.press/v80/garnelo18a.html.

[19] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Dropblock: a regularization method for convolutional networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 10750–10760, 2018.

[20] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[21] P. Glasserman. *Monte Carlo methods in financial engineering*. Springer, New York, 2004. ISBN 0387004513 9780387004518 1441918221 9781441918222.

[22] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting Gradient-Based Meta-Learning as hierarchical bayes. Jan. 2018. URL http://arxiv.org/abs/1801.08930.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] T. Hospedales, A. Antoniou, P. Micaelli, and others. Meta-learning in neural networks: A survey. *arXiv preprint arXiv*, 2020. URL https://arxiv.org/abs/2004.05439.

[25] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[26] T. B. Johnson and C. Guestrin. Training deep models faster with robust, approximate importance sampling. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7265–7275. Curran Associates, Inc., 2018.

[27] A. Katharopoulos and F. Fleuret. Not all samples are created equal: Deep learning with importance sampling. Mar. 2018. URL `http://arxiv.org/abs/1803.00942`.

[28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[29] A. Kong. A note on importance sampling using standardized weights. *University of Chicago, Dept. of Statistics, Tech. Rep*, 348, 1992.

[30] S. Laenen and L. Bertinetto. On episodes, prototypical networks, and few-shot learning. Dec. 2020. URL `http://arxiv.org/abs/2012.09831`.

[31] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

[32] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[33] C. Liu, Z. Wang, D. Sahoo, Y. Fang, K. Zhang, and S. C. H. Hoi. Adaptive task sampling for Meta-Learning. July 2020. URL `http://arxiv.org/abs/2007.08735`.

[34] J. S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and computing*, 6(2):113–119, 1996.

[35] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[36] E. G. Miller, N. E. Matsakis, and P. A. Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 1, pages 464–471 vol.1, June 2000. URL `http://dx.doi.org/10.1109/CVPR.2000.855856`.

[37] A. Nichol, J. Achiam, and J. Schulman. On First-Order Meta-Learning algorithms. Mar. 2018. URL `http://arxiv.org/abs/1803.02999`.

[38] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1447–1454. IEEE, 2006.

[39] B. N. Oreshkin, P. Rodriguez, and A. Lacoste. Tadam: task dependent adaptive metric for improved few-shot learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 719–729, 2018.

[40] E. Park and J. B. Oliva. Meta-Curvature. Feb. 2019. URL `http://arxiv.org/abs/1902.03356`.

[41] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2019.

[42] A. Rajeswaran, C. Finn, S. Kakade, and S. Levine. Meta-Learning with implicit gradients. Sept. 2019. URL `http://arxiv.org/abs/1909.04630`.

[43] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. 2016.

[44] A. Ravichandran, R. Bhotika, and S. Soatto. Few-Shot learning with embedded class models and Shot-Free meta training. May 2019. URL `http://arxiv.org/abs/1905.04398`.

[45] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018.

[46] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat.*, 22(3):400–407, Sept. 1951. URL `https://projecteuclid.org/euclid.aoms/1177729586`.

[47] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-Learning with Memory-Augmented neural networks. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA, 2016. PMLR. URL `http://proceedings.mlr.press/v48/santoro16.html`.

[48] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. Nov. 2015. URL `http://arxiv.org/abs/1511.05952`.

[49] J. Schmidhuber. *Evolutionary Principles in Self-Referential Learning*. PhD thesis, 1987. URL http://people.idsia.ch/~juergen/diploma1987ocr.pdf.

[50] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[51] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016.

[52] P. J. Smith, M. Shafi, and Hongsheng Gao. Quick simulation: a review of importance sampling techniques in communications systems. *IEEE J. Sel. Areas Commun.*, 15(4):597–613, May 1997. URL http://dx.doi.org/10.1109/49.585771.

[53] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090, 2017.

[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[55] Q. Sun, Y. Liu, Z. Chen, T.-S. Chua, and B. Schiele. Meta-Transfer learning through hard tasks. Oct. 2019. URL http://arxiv.org/abs/1910.03648.

[56] S. Thrun and L. Pratt, editors. *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, USA, 1998. URL https://dl.acm.org/citation.cfm?id=296635.

[57] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, and H. Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgAGAVKPr.

[58] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3637–3645, 2016.

[59] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3):1–34, June 2020. URL https://doi.org/10.1145/3386252.

[60] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200. 2010.

[61] D. H. D. West. Updating mean and variance estimates: An improved method. *Commun. ACM*, 22(9): 532–535, Sept. 1979. ISSN 0001-0782. doi: 10.1145/359146.359153. URL https://doi.org/10.1145/359146.359153.

[62] C. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling matters in deep embedding learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2859–2867, Oct. 2017. URL http://dx.doi.org/10.1109/ICCV.2017.309.

[63] X. Wu, E. Dyer, and B. Neyshabur. When do curricula work? Dec. 2020. URL http://arxiv.org/abs/2012.03107.

[64] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8808–8817, 2020.

[65] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian Model-Agnostic Meta-Learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 7332–7342. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/e1021d43911ca2c1845910d84f40aeae-Paper.pdf.

[66] C. Zhang, C. Öztireli, S. Mandt, and G. Salvi. Active mini-batch sampling using repulsive point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5741–5748, 2019.

[67] C. Zhang, Y. Cai, G. Lin, and C. Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12203–12213, 2020.

[68] P. Zhao and T. Zhang. Accelerating minibatch stochastic gradient descent using stratified sampling. May 2014. URL http://arxiv.org/abs/1405.3080.

[69] Y. Zhou, Y. Wang, J. Cai, Y. Zhou, Q. Hu, and W. Wang. Expert training: Task hardness aware Meta-Learning for Few-Shot classification. July 2020. URL http://arxiv.org/abs/2007.06240.

A Experimental setup

A.1 Datasets

We use two standardized few-shot image classification datasets.

Mini-ImageNet: This dataset [58] is a subset of ImageNet [10] and consists of 64 classes for training, 16 for validation, and 20 for testing. There are 600 images per class, with images of size 84×84. Multiple versions of this dataset exist in the literature; we use the version by Ravi and Larochelle [43].

Tiered-ImageNet: A larger subset of ImageNet, Tiered-ImageNet [45] consists of 608 classes split into 351, 97, and 160 for training, validation, and testing, respectively. Each class has about 1300 images of size 84×84. This dataset ensures that the train, validation, and test classes do not have any semantic overlap and is proposed as a harder few-shot learning benchmark.

We also use the evaluation splits of the following four datasets, as defined by Triantafillou et al. [57].

CUB-200: CUB-200 was collected by Welinder et al. [60] and contains 6,033 bird images classified into 200 bird species. The original version of the dataset contains 43 images also present in ImageNet. We remove those duplicates to avoid overestimating the transfer capability of our models. The test split contains 30 classes.

Describable Textures: Proposed by Cimpoi et al. [8], the task of this dataset is to classify images into 47 texture classes. Each of the 5640 images (120 samples per class) contains at least 90% of the class' texture, with sizes between 300x300 and 640x640 pixels. The train split has 33 classes, while validation and test splits both consist of 7 classes.

VGG Flowers: Originally introduced by Nilsback and Zisserman [38], VGG Flowers consists of 102 flower categories with each category containing between 40 and 258 RGB images. While we use Triantafillou et al. [57]'s train (71 classes), validation (15 classes), and test (16 classes) splits, our models operate on the raw images, not the cropped versions.

FGVC Aircrafts: Maji et al. [35] introduced this dataset containing $10,200$ images of aircraft partitioned into 102 classes, each with 100 samples. The test split contains 15 classes. As for VGG Flowers, we do not crop those images using bounding box information, thus increasing classification difficulty.

A.2 Network architectures

We train two of the most popular network architectures in few-shot learning literature.

conv$(64)_4$: This architecture [58] consists of 4 convolutional layers with 64 channels per layer.

ResNet-12: From the family of deep residual networks [23], this architecture has 4 blocks, each block constituting 3 convolutional layers with $64 \times 2^{l-1}$ channels per layer in the l'th block. Two versions of this network architecture exist in the literature; we use the one by Oreshkin et al. [39]. The other version by Lee et al. [32] is $1.25\times$ wider and has more parameters.

Both architectures use batch normalization [25] after every convolutional layer with ReLU as the non-linearity. We do not use dropout [54] or any of its variants, like Ghiasi et al. [19]. For MAML and ANIL, a fully-connected layer is appended at the top of the networks.

A.3 Training algorithms

For the metric-based family, we use ProtoNet with Euclidean [53] and scaled negative cosine similarity measures [20]. Based on the implementation of Gidaris and Komodakis [20], we add a learnable parameter that scales the cosine similarity. Additionally, we use MAML [15] and ANIL [41] as representative gradient-based algorithms. For all algorithms, we use the open-source implementation in `lear2learn` [1][7].

[7] Available at: `https://github.com/learnables/learn2learn`

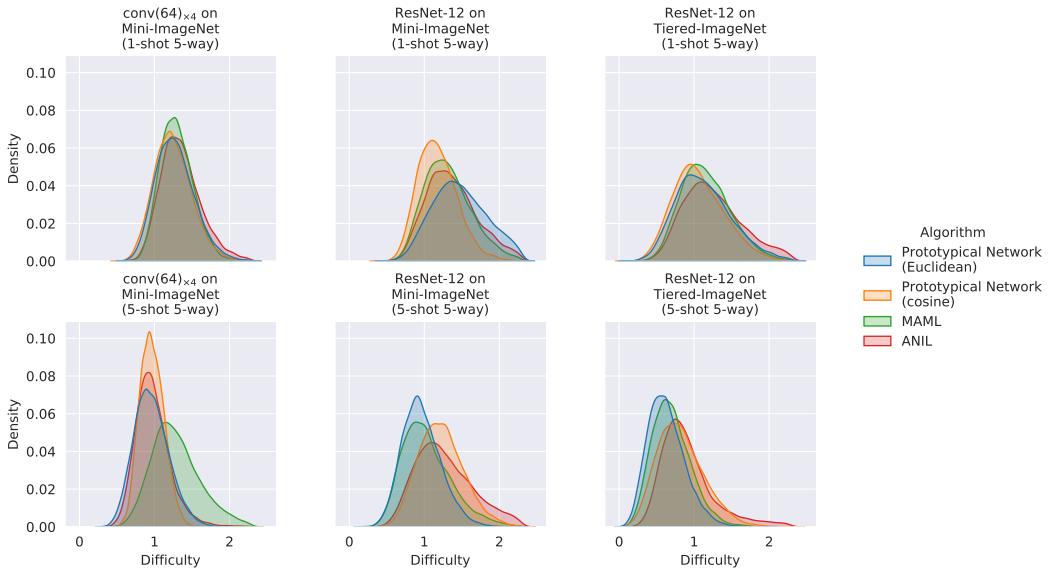

Figure 5: **Episode difficulty is approximately normally distributed - density plots.** Density plots of the episode difficulty computed by conv$(64)_4$'s on Mini-ImageNet (left), ResNet-12's on Mini-ImageNet (center) and ResNet-12's on Tiered-ImageNet (right), trained using ProtoNets (Euclidean and cosine), MAML and ANIL (depicted in the legend). The values are computed over 10k test episodes. The top row is for 1-shot 5-way episodes and the bottom row is for 5-shot 5-way episodes. All the plots follow a bell curve, with the density peak in the middle, which quickly drops-off on either side of the peak.

A.4 Sampling methods

We compare four sampling methods – EASY, HARD, CURRICULUM, and UNIFORM. In each case, we mimic the target distribution using importance sampling (refer to Section 3.3).

We also add baseline sampling in our comparisons. This involves episodic training without the use of any weighting techniques, hence sampling episodes from the distribution $q(\tau)$ without making any changes to it (refer to Section 2 and 3). This is typically used for few-shot episodic training.

A.5 Hyper-parameters

We tune hyper-parameters for each algorithm and dataset to work well across different few-shot settings and network architectures. Additionally, we keep the hyper-parameters the same across all different sampling methods for a fair comparison.

All models are trained using ADAM [28] with a learning rate of 10^{-3} on a single NVIDIATesla V100 GPU. MAML and ANIL use an adaptation learning rate of 0.01 and 0.1 respectively, with 5 adaptation steps taken in both cases. All models are trained for a total of 20k iterations, with a mini-batch of size 16 and 32 for Mini-ImageNet and Tiered-ImageNet respectively. After every 1k iterations, we evaluate on 1k validation episodes. The model with the best performance on this set is finally evaluated on 1k test episodes.

B Episode difficulty is approximately normally distributed

Sampling episodes from $q(\tau)$ (refer to Sections 2) induces a distribution over their difficulty Ω_{l_θ}. Our proposed method estimates this as a normal distribution (refer to Section 3), and here we justify why.

We train conv$(64)_4$'s on Mini-ImageNet and ResNet-12's on both Mini-ImageNet and Tiered-ImageNet using baseline sampling. This is done using all four learning algorithms – ProtoNet (Euclidean and cosine), MAML and ANIL – for 1-shot 5-way and 5-shot 5-way classification. We compute the episode difficulty over 10k test episodes, sampled using the episode distribution $q(\tau)$.

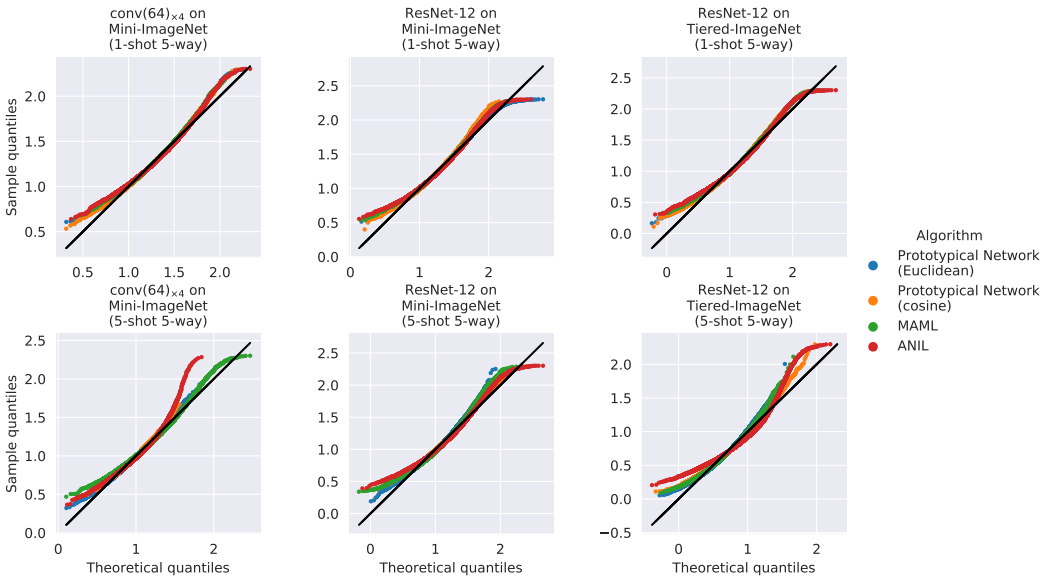

Figure 6: **Episode difficulty is approximately normally distributed - Q-Q plots.** Q-Q plots of the episode difficulty computed by conv$(64)_4$'s on Mini-ImageNet (left), ResNet-12's on Mini-ImageNet (center) and ResNet-12's on Tiered-ImageNet (right), trained using ProtoNets (Euclidean and cosine), MAML and ANIL (depicted in the legend). The values are computed over 10k test episodes and are plotted against normal distributions with the same mean and standard deviation as the episode difficulties. The top row is for 1-shot 5-way episodes and the bottom row is for 5-shot 5-way episodes. We also include the identity line in each plot (in black). The closer the curve is to the identity line, the closer the distribution is to a normal.

Fig. 5 illustrates the density plots of the computed values. We observe that the episode difficulties follow a bell curve in each case, which is naturally modeled with a normal distribution. Fig. 6 includes Q-Q plots for the same, plotted against normal distributions with the same mean and standard deviation as the corresponding episode difficulties. These plots are typically used to assess normality: the closer the curve is to the identity line, the closer the distribution is to a normal, which is also observed here.

Table 5: **Episode difficulty is approximately normally distributed - Shapiro-Wilk normality tests.** We compute the episode difficulty for different datasets, algorithms and network architectures, for both the 1-shot 5-way and 5-shot 5-way settings. This is done for 10k test episodes each. In each case, we subsample 50 values 100 times and run the Shapiro-Wilk test on these subsets (with $\alpha = 0.05$). The rejection rates of the null hypothesis, averaged over everything but the axes mentioned in the column on the left, are mentioned in the column on the right. The average rejection rate does not exceed 20%.

		Rejection rate (%)
Dataset	Mini-ImageNet	14.25
	Tiered-ImageNet	17.38
Shots	1-shot	14.17
	5-shot	16.42
Algorithm	ProtoNet (Euclidean)	19.67
	ProtoNet (cosine)	09.17
	MAML	13.33
	ANIL	19.00
Network Architecture	conv$(64)_4$	09.63
	ResNet-12	18.13

We additionally run the Shapiro-Wilk test for normality [50] on the computed episode difficulties, which tests for the null hypothesis that the data is drawn from a normal distribution. The p-value for this test is sensitive to the sample size: for large sample sizes, trivial departures from the normal

Table 6: **Few-shot accuracies on benchmark datasets for 5-way few-shot episodes in the offline setting.** The mean accuracy and the 95% confidence interval are reported for evaluation done over $1,000$ test episodes. Best results for a fixed scenario are shown in bold. Results where a sampling technique is better than or comparable to baseline sampling are denoted by †. The first row in every scenario denotes baseline sampling. Overall, UNIFORM is among the best sampling methods in $19/24$ scenarios, improving accuracy by 3.17% over the baseline on average.

	Mini-ImageNet				Tiered-ImageNet	
	conv$(64)_4$		ResNet-12		ResNet-12	
	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)
ProtoNet (Euclidean)	**49.06±0.60**	65.28±0.52	49.67±0.64	67.45±0.51	**59.10±0.73**	76.95±0.56
+ EASY	**48.83±0.61**†	65.92±0.55†	51.08±0.63†	67.30±0.52†	57.68±0.75	**78.10±0.53**†
+ HARD	45.69±0.61	**66.47±0.52**†	52.50±0.62†	**71.03±0.51**†	54.85±0.71	76.15±0.56
+ CURRICULUM	48.23±0.63	65.77±0.51†	50.00±0.61†	70.49±0.51†	**59.15±0.76**†	**78.25±0.53**†
+ UNIFORM	48.19±0.62	**66.73±0.52**†	**53.94±0.63**†	**70.79±0.49**†	**58.63±0.76**†	**78.62±0.55**†
ProtoNet (cosine)	**50.03±0.61**	61.56±0.53	52.85±0.64	62.11±0.52	**60.01±0.73**	72.75±0.59
+ EASY	**49.60±0.61**†	65.17±0.53†	53.35±0.63†	63.55±0.53†	**60.03±0.75**†	74.65±0.57†
+ HARD	49.01±0.60	**66.45±0.50**†	52.65±0.63†	70.15±0.51†	55.44±0.72	75.97±0.55†
+ CURRICULUM	49.38±0.61	64.12±0.53†	53.21±0.65†	65.89±0.52†	**60.37±0.76**†	75.32±0.58†
+ UNIFORM	**50.07±0.59**†	**66.33±0.52**†	**54.27±0.65**†	**70.85±0.51**†	**60.27±0.75**†	**78.36±0.54**†
MAML	**46.88±0.60**	55.16±0.55	49.92±0.65	63.93±0.59	**55.37±0.74**	**72.93±0.60**
+ EASY	44.52±0.60	57.36±0.59†	51.62±0.67†	64.33±0.61†	53.39±0.79	69.81±0.68
+ HARD	42.93±0.61	60.42±0.55†	49.57±0.69†	**66.93±0.55**†	50.48±0.73	71.20±0.63
+ CURRICULUM	45.42±0.60	**61.61±0.55**†	**52.21±0.67**†	66.25±0.60†	54.13±0.77	71.47±0.63
+ UNIFORM	**46.67±0.63**†	**62.09±0.55**†	**52.65±0.65**†	**66.76±0.57**†	54.58±0.77	72.00±0.66
ANIL	**46.59±0.60**	**63.47±0.55**	**49.65±0.65**	59.51±0.56	54.77±0.76	69.28±0.67
+ EASY	44.83±0.63	62.23±0.56	49.40±0.64†	56.73±0.60	54.50±0.80†	65.45±0.66
+ HARD	43.30±0.58	59.87±0.55	47.91±0.62	62.05±0.59†	50.22±0.71	62.06±0.65
+ CURRICULUM	45.69±0.60	**63.00±0.54**†	**50.22±0.66**†	61.76±0.57†	**55.59±0.78**†	**69.83±0.73**†
+ UNIFORM	**46.93±0.62**†	62.75±0.60	**49.56±0.62**†	**64.72±0.60**†	54.15±0.79†	**70.44±0.69**†

distribution can be detected, making the p-values unreliable. Instead, we subsample 50 values 100 times and run the test on these subsets (with $\alpha = 0.05$). Table 5 summarizes the rejection rates of the null hypothesis averaged over datasets, shots, algorithms and network architectures. Regardless of which axis the rejection rate is averaged over, it does not exceed 20%. These results suggest that our assumption of estimating the induced distribution over the episode difficulty as a normal distribution is plausible.

C Comparing episode sampling methods

In addition to the discussion in Section 5.3, this section presents the full suite of results for the comparison of different episode sampling methods. We compute results over 2 datasets, 2 network architectures, 4 algorithms and 2 few-shot protocols, resulting in 24 total scenarios. Table 6 contains all performance numbers. As mentioned in the main text, UNIFORM is among the better sampling schemes in $19/24$ scenarios, followed by baseline sampling which is competitive in $10/24$ scenarios. Importantly, when UNIFORM underperforms it is a close second: the average degradation is -0.58%, and at most -1.44%.

D Difference in effectiveness in the 1- and 5-shot settings

The 1-shot setting is inherently noisier than 5-shot. Support samples are randomly drawn from the class-populations, which are then used to construct the few-shot classifier. Sampling only 1 support per-class is more susceptible to outliers in the query set than sampling 5 (the higher the support-shot, the better the estimate of the class-population). This noise propagates to the loss (in the case of baseline sampling) as well as the weighted loss (in the case of UNIFORM). Hence, larger noise

degrades the approximation to a uniform distribution over episode difficulty and ultimately results in UNIFORM not getting as much gain in the 1-shot setting.

We empirically confirm this hypothesis. We use the same 24 scenarios as the ones in Sections 5.3 and 5.4 and compare the training procedures of UNIFORM under 1- vs. 5-shot settings. Using Eq. (3), we compute the per-episode weighted loss during the training process, followed by the per-mini-batch standard deviation. The average deviation is higher under the 1-shot than the 5-shot setting in all scenarios (for both offline and online settings). Additionally, the average deviation is ≈ 1.9 times larger under the 1-shot setting. These experiments confirm the above hypothesis and help explain why UNIFORM (online) outperforms the baseline in (only) 4/12 scenarios, is comparable in 7/12, and underperforms in 1/12.

E Better sampling improves cross-domain few-shot classification

In addition to the results in Section 5, we show that few-shot performance in the cross-domain setting can benefit from better sampling. We train models on Mini-ImageNet (as done in Section 5) and test the few-shot performance on the following datasets: CUB-200 [60], Describable Textures [8], FGVC-Aircraft [35], VGG Flowers [38]. We use conv$(64)_4$ and ResNet-12 network architectures trained using ProtoNet (Euclidean and cosine), MAML and ANIL algorithms for the 5-ways 1- and 5-shot settings. Altogether, these makeup 64 new scenarios. We measure the accuracy on the test splits of [57].

We compare online UNIFORM against baseline sampling and observe that UNIFORM does statistically better in 49/64 scenarios, comparable in 12/64 scenarios, and worse in only 3/64 scenarios. The performance numbers are included in Table 7.

This further goes to show that sampling under the episodic training paradigm matters. Using UNIFORM leads to statistically significant improvements over the ubiquitous baseline sampling in most cases and rarely degrades performance.

F Number of trials

In Tables 2 and 6 we make use of one random seed to give one training job per scenario per sampling method. However we report performances over 1k test episodes, as is typically done in few-shot learning. We additionally ran 3 training jobs for baseline sampling and online UNIFORM, resulting in 3 training jobs per scenario per sampling method. We observe that the difference in accuracy is $.20\%$ and $.02\%$ on average (ignoring the standard deviations) for baseline sampling and online UNIFORM; the effect of multiple random seeds is diminished when testing over many episodes.

Table 7: **Few-shot accuracies on benchmark datasets after training on Mini-ImageNet for 5-way few-shot episodes in the offline and online settings.** The mean accuracy and the 95% confidence interval are reported for evaluation done over 1,000 test episodes. Best results for a fixed scenario are shown in bold. The first row in every scenario denotes baseline sampling. Compared to baseline sampling, online UNIFORM does statistically better in 49/64 scenarios, comparable in 12/64 scenarios and worse in only 3/64 scenarios.

	$\text{conv}(64)_4$		ResNet-12	
	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)
	CUB-200			
ProtoNet (Euclidean)	**37.24±0.53**	52.07±0.53	36.53±0.54	51.49±0.56
+ UNIFORM (Online)	**37.08±0.53**	**53.32±0.53**	**39.48±0.56**	**56.57±0.55**
ProtoNet (cosine)	37.49±0.54	49.31±0.53	38.67±0.60	49.75±0.57
+ UNIFORM (Online)	**41.56±0.58**	**54.17±0.53**	**40.55±0.60**	**56.30±0.55**
MAML	34.52±0.53	**47.11±0.60**	35.80±0.56	45.16±0.62
+ UNIFORM (Online)	**35.84±0.54**	**46.67±0.55**	**37.18±0.55**	**46.58±0.58**
ANIL	35.40±0.54	38.20±0.56	33.20±0.54	39.26±0.58
+ UNIFORM (Online)	**36.89±0.55**	**42.83±0.58**	**34.47±0.56**	**42.08±0.58**
	Describable Textures			
ProtoNet (Euclidean)	32.05±0.45	**45.03±0.44**	31.87±0.45	44.10±0.43
+ UNIFORM (Online)	**32.69±0.49**	**45.23±0.43**	**33.55±0.46**	**47.37±0.43**
ProtoNet (cosine)	32.09±0.45	38.44±0.41	31.48±0.45	39.46±0.41
+ UNIFORM (Online)	**33.63±0.47**	**43.28±0.44**	**32.69±0.48**	**45.56±0.42**
MAML	29.47±0.46	37.85±0.47	**32.19±0.48**	41.14±0.46
+ UNIFORM (Online)	**31.84±0.49**	**40.81±0.44**	31.65±0.46	**43.21±0.44**
ANIL	29.86±0.46	40.69±0.46	28.85±0.41	37.04±0.44
+ UNIFORM (Online)	**31.29±0.48**	**41.42±0.45**	**31.38±0.47**	**39.03±0.47**
	FGVC-Aircraft			
ProtoNet (Euclidean)	**26.03±0.37**	39.41±0.48	25.98±0.39	36.76±0.45
+ UNIFORM (Online)	**26.18±0.38**	**40.23±0.46**	**27.43±0.42**	**38.49±0.46**
ProtoNet (cosine)	**27.11±0.39**	32.14±0.38	25.23±0.39	32.07±0.41
+ UNIFORM (Online)	**27.15±0.38**	**37.78±0.45**	**26.89±0.39**	**37.42±0.44**
MAML	**26.78±0.38**	**34.21±0.41**	25.50±0.39	29.38±0.40
+ UNIFORM (Online)	**26.62±0.39**	**34.41±0.44**	**26.22±0.39**	**30.21±0.43**
ANIL	**25.67±0.37**	27.17±0.36	23.27±0.31	24.52±0.29
+ UNIFORM (Online)	**25.60±0.37**	**27.92±0.39**	**23.78±0.34**	**28.70±0.39**
	VGG Flowers			
ProtoNet (Euclidean)	53.50±0.63	70.96±0.51	**57.74±0.68**	74.87±0.49
+ UNIFORM (Online)	**54.72±0.65**	**73.59±0.49**	55.94±0.67	**76.62±0.50**
ProtoNet (cosine)	52.94±0.62	66.04±0.53	52.98±0.65	66.79±0.51
+ UNIFORM (Online)	**54.23±0.63**	**71.93±0.48**	**57.06±0.65**	**67.31±0.48**
MAML	**49.70±0.60**	**63.69±0.54**	**50.13±0.64**	61.41±0.63
+ UNIFORM (Online)	**49.72±0.60**	**63.52±0.54**	**49.53±0.65**	**63.99±0.58**
ANIL	**47.03±0.65**	46.40±0.66	**42.05±0.67**	40.01±0.65
+ UNIFORM (Online)	**47.48±0.67**	**47.08±0.67**	38.94±0.61	**50.25±0.63**