LEARNING FLEXIBLE CLASSIFIERS WITH SHOT-CONDITIONAL EPISODIC (SCONE) TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Early few-shot classification work advocates for episodic training, i.e. training over learning episodes each posing a few-shot classification task. However, the role of this training regime remains poorly understood, and its usefulness is still debated. Standard classification training methods ("pre-training") followed by episodic fine-tuning have recently achieved strong results. This work aims to understand the role of this episodic fine-tuning phase through an exploration of the effect of the "shot" setting (number of examples per class) that is used during fine-tuning. We discover that fine-tuning on episodes of a particular shot can specialize the pre-trained model to solving episodes of that shot at the expense of performance on other shots, in agreement with a trade-off recently observed in the context of end-to-end episodic training. To amend this, we propose a shot-conditional form of episodic fine-tuning, inspired from recent work that trains a single model on a distribution of losses. Our investigation shows that this improves overall performance, without suffering disproportionately on any shot. We also examine the usefulness of this approach on the large-scale Meta-Dataset benchmark where test episodes exhibit varying shots and imbalanced classes. We find that our flexible model improves performance in that challenging environment.

1 INTRODUCTION

Few-shot classification is the problem of learning a classifier using only a few examples. Specifically, the aim is to utilize a training dataset towards obtaining a flexible model that has the ability to 'quickly' learn about new classes from few examples. Success is evaluated on a number of *test episodes*, each posing a classification task between previously-unseen *test* classes. In each such episode, we are given a few examples, or "shots", of each new class that can be used to adapt this model to the task at hand, and the objective is to correctly classify a held-out set of examples of the new classes.

A simple approach to this problem is to learn a classifier over the training classes, parameterized as a neural network feature extractor followed by a classification layer. While the classification layer is not useful at test time due to the class shift, the embedding weights that are learned during this "pre-training" phase evidently constitute a strong representation that can be used to tackle test tasks when paired with a simple "inference algorithm" (e.g. nearest-neighbour, logistic regression) to make predictions for each example in the test episode given the episode's small training set. Alternatively, early influential works on few-shot classification (Vinyals et al., 2016) advocate for *episodic training*, a regime where the training objective is expressed in terms of performance on a number of *training episodes* of the same structure as the test episodes, but with the classes sampled from the training set. It was hypothesized that this episodic approach captures a more appropriate inductive bias for the problem of few-shot classification and would thus lead to better generalization.

However, there is an ongoing debate about whether episodic training is in fact required for obtaining the best few-shot classification performance. Notably, recent work (Chen et al., 2019; Dhillon et al., 2020) proposed strong "pre-training" baselines that leverage common best practices for supervised training (e.g. normalization schemes, data augmentation) to obtain a powerful representation that works well for this task. Interestingly, other recent work combines the pre-training of a single classifier with episodic fine-tuning by removing the classification head and continuing to train the embedding network using the episodic inference algorithm that will be applied at test time (Triantafillou et al., 2020; Chen et al., 2020). The success of this hybrid approach suggests that perhaps the two regimes

have complementary strengths, but the role of this episodic fine-tuning is poorly understood: what is the nature of the modification it induces into the pre-trained solution? Under which conditions is it required in order to achieve the best performance?

As a step towards answering those questions, we investigate the effect of the shot used during episodic fine-tuning on the resulting model's performance on test tasks of a range of shots. We are particularly interested in understanding whether the shot of the training episodes constitutes a source of information that the model can leverage to improve its few-shot classification performance on episodes of that shot at test time. Our analysis reveals that indeed a particular functionality that this fine-tuning phase may serve is to specialize a pre-trained model to solving tasks of a particular shot; accomplished by performing the fine-tuning on episodes of that shot. However, perhaps unsurprisingly, we find that specializing to a given shot comes at the expense of hurting performance for other shots, in agreement with (Cao et al., 2020)'s theoretical finding in the context of Prototypical Networks (Snell et al., 2017) where inferior performance was reported when the shot at training time did not match the shot at test time.

Given those trade-offs, how can our newfound understanding of episodic fine-tuning as shot-specialization help us in practice? It is unrealistic to assume that we will always have the same number of labeled examples for every new class we hope to learn at test time, so we are interested in approaches that operate well on tasks of a range of shots. However, it is impractical to fine-tune a separate episodic model for every shot, and intuitively that seems wasteful as we expect that tasks of similar shots should require similar models. Motivated by this, we propose to train a single shot-conditional model for specializing the pre-trained solution to a wide spectrum of shots without suffering trade-offs. This leads to a compact but flexible model that can be conditioned to be made appropriate for the shot appearing in each test episode.

In what follows we provide some background on few-shot classification and episodic models and then introduce our proposed shot-conditioning approach and related work. We then present our experimental analysis on the effect of the shot chosen for episodic fine-tuning, and we observe that our shot-conditional training approach is beneficial for obtaining a general flexible model that does not suffer the trade-offs inherent in naively specializing to any particular shot. Finally, we experiment with our proposed shot-conditional approach in the large-scale Meta-Dataset benchmark for few-shot classification, and demonstrate its effectiveness in that challenging environment.

2 BACKGROUND

Problem definition Few-shot classification aims to classify test examples of unseen classes from a small labeled training set. The standard evaluation procedure involves sampling *classification episodes* by picking N classes at random from a test set of classes $\mathcal{C}^{test}$ and sampling two disjoint sets of examples from the N chosen classes: a *support* set (or training set) of k labeled examples per class, and a *query* set (or test set) of unlabeled examples, forming N-way, k-shot episodes. The model is allowed to use the support set, in addition to knowledge acquired while training on a disjoint set of classes $\mathcal{C}^{train}$, to make a prediction for examples in the query set, and is evaluated on its query set accuracy averaged over multiple test episodes.

Episodic training Early few-shot classification approaches (Vinyals et al., 2016) operate under the assumption that obtaining a model capable of few-shot classification requires training it on (mini-batches of) learning episodes, instead of (mini-batches of) individual examples as in standard supervised learning. These learning episodes are sampled in the same way as described above for test episodes, but with classes sampled from $\mathcal{C}^{train}$ this time. In other words, the model is trained to minimize a loss of the form:

$$\mathbb{E}_{\mathcal{S},\mathcal{Q}\sim P^{N,k}_{train}}\left[\frac{1}{|\mathcal{Q}|}\sum_{(x^*,y^*)\in\mathcal{Q}} -\log p_\theta(y^* \mid x^*, \mathcal{S})\right] \tag{1}$$

where $\mathcal{S}$ and $\mathcal{Q}$ are support and query sets sampled from the distribution $P^{N,k}_{train}$ of N-way, k-shot training episodes induced by $\mathcal{C}^{train}$, and θ represents the model's parameters. This training regime is often characterized as *meta-learning* or *learning to learn*, i.e. learning over many episodes how to learn within an episode (from few labeled examples). Episodic models differ by their "inference

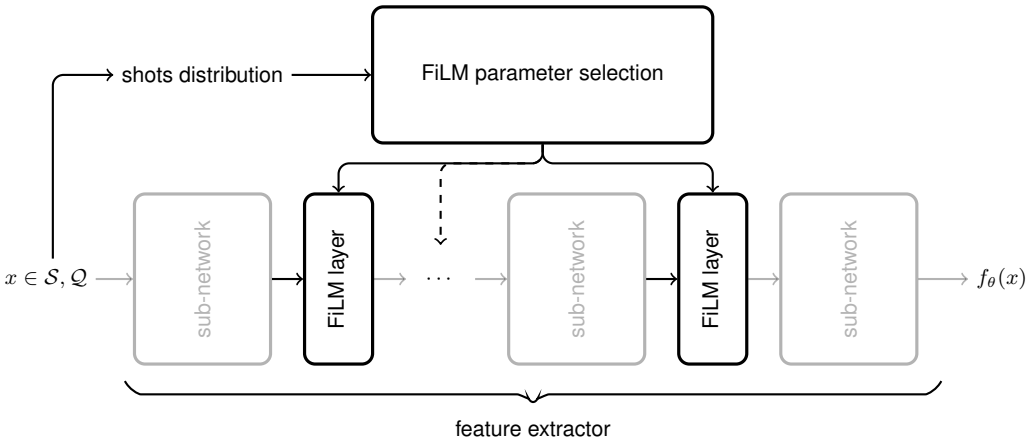


Figure 1: SCONE conditions the feature extractor f_θ on an episode's shot distribution.

algorithm", i.e. the manner in which $p_\theta(y^* \mid x^*, \mathcal{S})$ is computed to classify query examples based on the support set.

Prototypical Networks Prototypical Networks (Snell et al., 2017) is a simple but effective episodic model which constructs a prototype ϕ_c for each class c in an episode as

$$\phi_c = \frac{1}{|\mathcal{S}_c|} \sum_{x \in \mathcal{S}_c} f_\theta(x), \tag{2}$$

where f is an embedding function parametrized by θ and $\mathcal{S}_c$ represents the set of support examples belonging to class c, and classifies a given query example as

$$p(y^* = c \mid x^*, \mathcal{S}) = \frac{\exp(-||x^* - \phi_c||_2^2)}{\sum_{c'} \exp(-||x^* - \phi_{c'}||_2^2)}. \tag{3}$$

3 SHOT CONDITIONAL EPISODIC (SCONE) TRAINING

In this section we introduce Shot CONditional Episodic (SCONE) training for the purpose of specializing a strong pre-trained model to solving few-shot classification tasks of a range of different shots, without suffering disproportionately for any shot.

Training objective Training episodically involves minimizing the objective shown in Equation 1. We first sample an episode from $P_{train}^{k,N}$ and compute a prediction $p_\theta(y^* \mid x^*, \mathcal{S})$ for each query example x^*. We then compute the cross-entropy loss on the query set using those predictions and perform a parameter update by backpropagating its gradient with respect to θ *into the inference algorithm*. In this work we concern ourselves with models that use an embedding function f_θ to obtain a representation for the support and query examples of each episode on top of which the inference algorithm is applied. In Prototypical Networks, for instance, f_θ contains *all* of the model's learnable parameters.

SCONE trains on episodes of varying shots and conditions the model on each episode's shot distribution. (Figure 1) by minimizing

$$\mathbb{E}_{k \sim P_k} \left[\mathbb{E}_{\mathcal{S}, \mathcal{Q} \sim P_{train}^{N,k}} \left[\frac{1}{|\mathcal{Q}|} \sum_{(x^*, y^*) \in \mathcal{Q}} -\log p_{\theta_k}(y^* \mid x^*, \mathcal{S}) \right] \right], \tag{4}$$

where P_k is the distribution over shots at training time and θ_k depends on an episode's sampled shots. In the Appendix, we include an algorithm box outlining SCONE fine-tuning.

Conditioning mechanism Rather than learning a separate set of model parameters for each shot setting, we modulate a subset of its parameters using FiLM (Perez et al., 2018), a simple conditioning mechanism which performs an affine feature-wise transformation of its input x based on conditioning information k (in our case, the episode's number of shots):

$$\text{FiLM}(x) = \gamma(k) \odot x + \beta(k). \tag{5}$$

The dependency of γ and β on k is handled by maintaining distinct values for each shot setting and selecting the appropriate γ and β based on an episode's shot. Equivalently, we can think of our approach as a compact representation of many shot-specific feature extractors which share all but their FiLM layer parameters.

More concretely, we maintain a set of FiLM parameters for each shot in the [1, MAX-SHOT] range (where MAX-SHOT is a hyperparameter) and let all shots settings greater than or equal to MAX-SHOT share the same FiLM parametrization. As is often the case in practice, instead of inserting FiLM layers in the network's architecture, we modulate the scaling and shifting parameter values of existing batch normalization layers (Dumoulin et al., 2017; De Vries et al., 2017). When performing episodic fine-tuning, we initialize all sets of FiLM parameters to those learned during pre-training (i.e. the learned batch normalization scaling and shifting coefficients). These different sets of FiLM parameters are then free to deviate from each other as a result of fine-tuning. We found it beneficial to penalize the L2-norm of β (regularizing the offset towards 0) and the L2 norm of $\gamma - 1$ (regularizing the scaling towards 1). For this purpose, we introduce a hyperparameter that controls the strength of this FiLM weight decay.

Handling class-imbalanced episodes SCONE can also be used on imbalanced episodes, where different classes have different shots. In that case, instead of selecting a single set of FiLM parameters, we compute the FiLM parameters for an episode as the convex combination of the FiLM parameters associated with all shots found in the episode, where the weights of that combination are determined based on the frequency with which each shot appears in the episode.

Concretely, the episode's "shot distribution" s (a vector of length MAX-SHOT) is obtained by averaging the one-hot representations of the shots of the classes appearing in an episode. In the special case of a class-balanced episode, the resulting average will be exactly a one-hot vector. This shot distribution is then used for the purpose of selecting the episode's FiLM parameters. This can be thought of as an embedding lookup $s^T \mathcal{F}$ in a matrix $\mathcal{F}$ of FiLM parameters using a shot distribution s.

Smoothing the shot distribution We expect similar shots to require similar feature extractors, which we incorporate as an inductive bias by smoothing the shots distribution using an exponential moving average (with an exponential decay factor m) before normalizing it again. We treat m as a hyperparameter that can be used both at training and evaluation time. We include the details of our smoothing procedure in the Appendix.

4 Related Work

Few-shot classification A plethora of models have been recently proposed for few-shot classification, and we refer the reader to (Hospedales et al., 2020) for a broad survey. Before episodic training was introduced, few-shot classifiers often relied on metric learning (Koch et al., 2015; Triantafillou et al., 2017). This theme persisted in early episodic models like Matching Networks (Vinyals et al., 2016) and Prototypical Networks (Snell et al., 2017) where classification is made via nearest-neighbour comparisons in the embedding space. Matching Networks apply a soft k-NN algorithm where the label of a query example is predicted to be the weighted average of the (one-hot) support labels with the weights determined by the similarity of that query to each support example.

Gradient-based episodic models are another popular family of approaches following the influential MAML paper (Finn et al., 2017). To create a classifier for each given episode, this approach fine-tunes the embedding weights along with a linear classifier head using gradient descent on the support set. Intuitively, this results in learning an embedding space that serves as a useful starting point from which a few steps of gradient descent suffice to adapt the model to each episode's classification task. Proto-MAML (Triantafillou et al., 2020) is a simple extension that initializes the linear classifier for each episode from the prototypes of the classes appearing in that episode.

Recently, the field has shifted towards studying few-shot classification in more realistic environments like *tiered*-ImageNet (Ren et al., 2018) and Meta-Dataset (Triantafillou et al., 2020), which has encouraged research into newly-introduced challenges, such as accounting for multiple diverse datasets. Along these lines, Requeima et al. (2019); Bateni et al. (2019) proposed novel task conditioning approaches, Saikia et al. (2020) introduced an improved hyperparameter tuning approach, and Dvornik et al. (2020) proposed a method for selecting an appropriate set of features for each test episode out of a universal feature representation.

Understanding episodic learning Our work inscribes itself in a recent line of work attempting to understand the differences between episodic and non-episodic learning. Goldblum et al. (2020) attempts to understand episodic learning from the perspective of how classes cluster in feature-space (for models that learn a final classification layer on top of a feature extractor) as well as from the perspective of local minima clusters (for gradient-based meta-learners). Huang et al. (2020); Chao et al. (2020) draw parallels between learning episodes and supervised learning examples, Bronskill et al. (2020) discusses batch normalization in episodic learning, drawing parallels from its use in non-episodic learning and Chen et al. (2020) contrasts episodic and non-episodic learning in their ability to generalize to new examples of previously seen classes or new examples of *unseen* classes. Finally, Cao et al. (2020) theoretically investigates the role of the shot in Prototypical Networks to explain the observed performance drop when there is a mismatch between the shots at training and test time. Instead, we empirically study the effect of the shot chosen during episodic fine-tuning of a pre-trained solution, in a larger-scale and more diverse environment.

Feature-wise conditioning Feature-wise transformations such as FiLM (Perez et al., 2018) are used as a conditioning mechanism in a variety of problem settings; see Dumoulin et al. (2018) for a survey on the topic. In the few-shot classification setting, FiLM is used as a way to condition metric learners' backbones on the support set (Oreshkin et al., 2018; Requeima et al., 2019; Bateni et al., 2019), as well as a way to represent many pre-trained classifiers using a shared parametrization (Dvornik et al., 2020). Notably, TADAM (Oreshkin et al., 2018), CNAPs (Requeima et al., 2019) and Simple-CNAPs (Bateni et al., 2019) also use task conditioning, but they use the mean of the support set for this and thus the 'shot' information is discarded. The purpose of our conditioning mechanism is instead to make the backbone shot-aware. The idea of shot-conditional learners is inspired by recent work that investigates loss-conditional training using feature-wise transformations (Dosovitskiy & Djolonga, 2020; Babaeizadeh & Ghiasi, 2020).

5 EXPERIMENTS

5.1 EXPLORING THE ROLE OF 'SHOTS' DURING EPISODIC FINE-TUNING

In this subsection, we examine the effect of the 'shot' that is used during the episodic fine-tuning phase, and in particular how it impacts the resulting model's ability to solve test episodes of different shots. We consider either using a fixed shot k throughout the fine-tuning phase, or fine-tuning on episodes of a distribution of shots. In the latter case, we explore both standard fine-tuning as well as SCONE fine-tuning that equips the model with the shot-conditioning mechanism described in the previous section.

Experimental setup We ran this round of experiments on ImageNet using the class splits proposed in Meta-Dataset. First, we pre-trained a standard classifier on the set of training classes of ImageNet. We then removed the topmost classification layer, leaving us with a pre-trained backbone that we used as the initialization for the subsequent episodic fine-tuning round. We ran four variants of episodic fine-tuning: exclusively on 1-shot episodes ('Fine-tune on 1-shot'), exclusively on 5-shot episodes ('Fine-tune on 5-shot'), on episodes whose shot is drawn uniformly from the range $[1, 40]$ ('Fine-tune on all shots'), and on episodes with that same shot distribution but using SCONE ('SCONE Fine-tune on all shots'), which additionally equips the backbone with the shot conditioning mechanism described in the previous section. In all cases, we fix the 'way' to 5. We use Prototypical Networks as the episodic model and we perform early stopping and model selection on the validation set of classes, where the validation performance of a variant is computed on episodes of the same (distribution of) shot(s) that it is trained on. All models are finally tested on a held-out test set of classes that is not seen during pre-training nor episodic fine-tuning, on 5-way episodes of different shot settings.

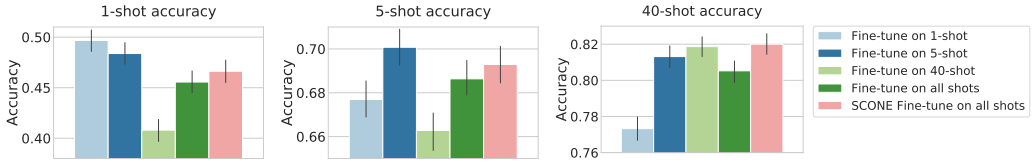


Figure 2: Test accuracy on three different evaluation shots. Fine-tuning exclusively on a particular shot leads to the best test accuracy on that shot but poor accuracy on different shots. Fine-tuning on a range of shots is a reasonable general solution, but its performance can be improved when using SCONE , thanks to its conditioning mechanism that offers a compact form of shot specialization.

As mentioned in the previous section, when applying SCONE training, we penalize the L2 norm of FiLM parameters. For a fair comparison with the other models, we applied the same regularization to the batch normalization parameters of all models during the episodic fine-tuning phase, and we found this to be generally helpful. We tuned the strength of this regularization separately for each model and picked the variant that worked best on the validation set, which we report in the Appendix. We set the SCONE 's MAX-SHOT hyperparameter to be 40 for this experiment.

Findings We observe from Figure 2 that fine-tuning on a fixed shot yields the best results on test episodes of that shot. For example, 1-shot accuracies show that 'Fine-tune on 1-shot' surpasses the performance of all other variants on 1-shot test episodes, with the analogous findings in 1-shot and 5-shot accuracies for 5-shot and 40-shot, respectively. Therefore, a particular functionality that the episodic fine-tuning phase may serve is to specialize the pre-trained model for performing well on tasks of a particular shot. However, as illustrated in all sub-plots of Figure 2, this shot specialization comes at the cost of severely reduced performance on tasks of very different shots. For instance, the model that is specialized for 40-shot tasks ('Fine-tune on 40-shot') performs very poorly on 1-shot test tasks and vice-versa.

In practice, it may be desirable to perform well on more than a single shot setting at test time, without having to fine-tune multiple separate shot-specialized models. A reasonable approach to that is episodically fine-tuning on a range of shots, to obtain a general model. Indeed, Figure 2 shows that 'Fine-tune on all shots' does not perform too poorly on any shot but, perhaps unsurprisingly, in any given setting, it falls short of the performance of the corresponding shot-specialized model.

Finally, we observe that SCONE fine-tuning outperforms its shot-unaware counterpart in all settings ('SCONE Fine-tune on all shots' vs 'Fine-tune on all shots'). This constitutes evidence that SCONE fine-tuning indeed leads to a more flexible model that can adapt to the shot of each episode via its conditioning mechanism, without suffering the trade-offs inherent in naively specializing a model exclusively to any particular shot. We can view a SCONE model as a very compact way of representing multiple shot-specialized models, where the information required for that specialization resides in the light-weight FiLM parameters.

5.2 Large-scale Experiments on Meta-Dataset

In what follows, we apply SCONE to the diverse and challenging Meta-Dataset benchmark for few-shot classification (Triantafillou et al., 2020). Meta-Dataset is comprised of ten distinct image datasets, including natural images, handwritten characters and sketches. It also defines a generative process for episodes that varies the way and shot across episodes, and within a particular episode varies the shot for different classes, introducing imbalance. The range of shots induced by this episode generator is also larger than what we considered in the previous section. It is a long-tailed distribution under which small and medium shots are more likely but it is possible to also encounter very large shots (e.g. >400), though this would happen very infrequently. We include histograms of the shot distributions of Meta-Dataset's training, validation and test episodes in the Appendix. These experiments aim to investigate whether SCONE is effective on this broader shot distribution and imbalanced episodes.

Prototypical Network on ImageNet For our first set of experiments on Meta-Dataset, we explore different strategies of episodic fine-tuning of the pre-trained classifier's embedding weights using

	Prototypical Networks (ImageNet only)			Meta-Baseline (All datasets)		
Dataset	Standard	L2 BN	SCONE	Classifier-Baseline	Control	SCONE
ILSVRC-2012	50.90 ± 1.12%	**51.81 ± 1.06%**	**52.51 ± 1.11%**	53.44 ± 0.82%	49.83 ± 0.80%	**53.69 ± 0.83%**
Omniglot	63.12 ± 1.37%	63.14 ± 1.32%	**65.60 ± 1.34%**	81.66 ± 0.73%	**89.28 ± 0.51%**	**90.01 ± 0.49%**
Aircraft	54.30 ± 0.97%	53.26 ± 0.97%	**55.38 ± 0.96%**	70.65 ± 0.62%	**81.60 ± 0.49%**	78.27 ± 0.54%
Birds	68.22 ± 0.97%	**69.21 ± 1.01%**	**69.70 ± 1.01%**	76.99 ± 0.64%	**78.75 ± 0.59%**	**79.62 ± 0.58%**
DTD	66.62 ± 0.90%	68.33 ± 0.81%	**69.58 ± 0.77%**	71.28 ± 0.56%	70.47 ± 0.58%	**71.89 ± 0.59%**
Quickdraw	59.79 ± 0.98%	59.17 ± 0.96%	**60.81 ± 0.95%**	64.09 ± 0.67%	**72.79 ± 0.59%**	**71.95 ± 0.56%**
Fungi	36.77 ± 1.07%	**38.96 ± 1.10%**	**39.66 ± 1.12%**	50.23 ± 0.81%	55.28 ± 0.73%	**57.04 ± 0.74%**
VGG Flower	86.61 ± 0.87%	**87.70 ± 0.77%**	**88.03 ± 0.73%**	89.14 ± 0.44%	90.13 ± 0.43%	**91.09 ± 0.39%**
Traffic Signs	48.64 ± 1.06%	46.54 ± 1.03%	**48.24 ± 1.09%**	68.87 ± 0.61%	**70.37 ± 0.56%**	**70.33 ± 0.56%**
MSCOCO	43.02 ± 1.09%	**43.11 ± 1.05%**	**44.25 ± 1.11%**	53.92 ± 0.78%	47.85 ± 0.81%	**52.94 ± 0.82%**
Average	57.80 ± %	58.12 ± %	59.38%	68.03%	70.63%	71.68%

Table 1: Left: Prototypical Networks fine-tuned on ImageNet ('Standard') with the addition of L2 regularization on the batch normalization weights ('L2 BN') and with SCONE ('SCONE '). Right: Our reproduction of the Classifier-Baseline (Chen et al., 2020) trained on all datasets, and two variants that freeze those weights and episodically fine-tune using Meta-Baseline (Chen et al., 2020) to optimize either only the batch norm parameters ('Control'), or only SCONE 's parameters ('SCONE '). In all cases, the reported numbers are query set accuracies averaged over test episodes (600 for Prototypical Networks and 1K for Meta-Baseline; the discrepancy due to the different defaults in the respective code-bases). We computed 95% confidence intervals and use a statistical test to determine via a statistical test which entries to bold. We report details on that in the Appendix.

Prototypical Networks. For this, we use Meta-Dataset's sampling algorithm to draw training episodes of varying shots and ways from ImageNet. We compare standard episodic fine-tuning ('Standard') to SCONE episodic fine-tuning ('SCONE '). Since SCONE uses L2-regularization on the sets of FiLM parameters, for a fair comparison we include a variant of standard episodic fine-tuning with L2-regularization on the batch normalization parameters ('L2 BN'). We evaluate the resulting models on the held-out set of ImageNet classes as well as the remaining 9 datasets. We set SCONE 's MAX-SHOT to 200. We tune the learning rate and decay schedule separately for each variant and we perform model selection of SCONE 's hyperparameters using the validation set. All additional details are reported in the Appendix, and we plan to open source our code upon publication.

Meta-Baseline on all datasets Next, we experiment with the recent Meta-Baseline model (Chen et al., 2020). Meta-Baseline also consists of a pre-training phase ('Classifier-Baseline') followed by an episodic fine-tuning phase ('Meta-Baseline'). Classifier-Baseline refers to simply training a classifier on the set of training classes. This variant is evaluated on few-shot episodes by discarding the ultimate classification layer and utilizing a cosine similarity-based nearest-centroid inference algorithm on the learned embeddings. Meta-Baseline then fine-tunes Classifier-Baseline's pre-trained embeddings on the episodic objective of the aforementioned nearest-centroid algorithm.

When training on all datasets of Meta-Dataset, they obtained strong results using their Classifier-Baseline which is in this case trained in a multi-task setup with separate output heads for the different datasets. They found that episodically fine-tuning that solution on all datasets did not help in general (it improved performance on some datasets but hurt performance on a larger number of datasets).

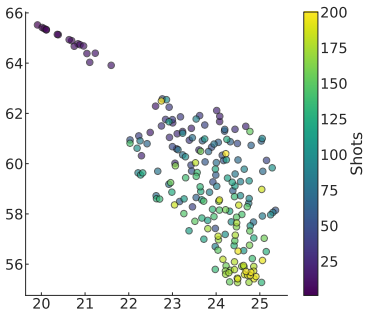


Figure 3: UMAP projection of the learned FiLM parameters for each "shot" setting, color-coded by shots.

Inspired by that finding, we experimented with a SCONE training phase on top of Classifier-Baseline's strong pre-trained solution where we froze the embedding weights to that powerful representation and we optimized only the set of SCONE 's FiLM parameters for shot conditioning. We performed this fine-tuning on training episodes from all datasets, using Meta-Baseline's nearest centroid method as the episodic model. As a control experiment, we performed the same episodic fine-tuning but without shot-conditioning, where we optimized only the batch normalization parameters, keeping the remainder of the

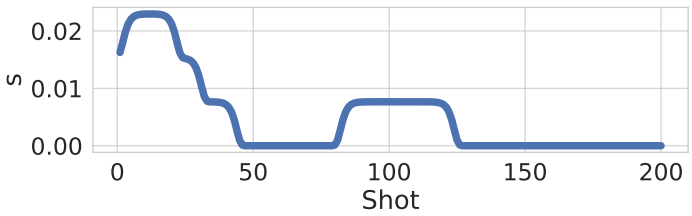


Figure 4: The shot distribution s produced according to our smoothing procedure for a hypothetical 4-way episode where the shots for the four classes are: 1, 10, 23, and 103.

embedding weights frozen ('Control'). This control can be thought of as a special case of SCONE where MAX-SHOT is set to 1.

Findings The results of this investigation are shown in Table 1. Following (Triantafillou et al., 2020), we run a test of statistical significance described in the Appendix to determine when to bold an entry. From the first three columns we can see that SCONE fine-tuning using outperforms standard episodic fine-tuning in the context of Prototypical Networks. Interestingly, penalizing the L2-norm of batch normalization parameters during episodic fine-tuning is beneficial even when not using SCONE , but it does not reach the performance obtained by shot-conditioning. Similarly, in the context of Meta-Baseline (last three columns), episodically fine-tuning the batch normalization parameters of the otherwise-frozen embedding is helpful, but learning a separate set of FiLM parameters for each shot yields additional gains in this setting too. Overall, despite the simplicity of SCONE , these results demonstrate its effectiveness on different shot distributions, and in different backbones.

FiLM parameter visualization Finally, as a sanity check, we perform a UMAP projection (McInnes et al., 2018) of the learned FiLM parameters for each shot setting (Figure 3). As expected, similar shot settings tend to learn similar sets of FiLM parameters, which is reflective of the fact that they rely on similar features for classification.

Example smoothed shot distribution To gain an intuition on the effect of our smoothing procedure, we illustrate in Figure 4 the result of smoothing an example shot distribution using $m = 1 - 1e - 06$, which is the value of the smoothing hyperparameter that we used for our Prototypical Network experiments on Meta-Dataset. For this, we consider a hypothetical 4-way episode where the shots for the four classes are: 1, 10, 23, and 103. We observe that the largest peak is in the range of small values, due to the first three shots of the episode, with the fourth shot causing a second peak around the value 103. As a reminder, this shot distribution defines the weights of the convex combination of FiLM parameters that will be used for the episode. In practice therefore, we are activating 'blocks' of FiLM parameters that are relevant for each episode, instead of strictly activating only the FiLM parameters of the observed shots.

6 CONCLUSION

In summary, we present an analysis aiming to understand the role of episodic fine-tuning on top of a pre-trained model for few-shot classification. We discover that this fine-tuning phase can be used to specialize the pre-trained model to episodes of a given shot, leading to strong performance on test episodes of that shot at the expense of inferior performance on other shots. To eliminate that trade-off, we propose a shot-conditional episodic training approach that trains a model on episodes of a range of shots and can be conditioned at test time to modify its behavior appropriately depending on the shot of the given test episode. Our experimental analysis suggests that our proposed shot-conditioning mechanism is beneficial both in smaller-scale experiments, as well as in the large-scale and diverse Meta-Dataset benchmark, in the context of two different episodic models.

REFERENCES

Mohammad Babaeizadeh and Golnaz Ghiasi. Adjustable real-time style transfer. In *Proceedings of the International Conference on Learning Representations*, 2020.

Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. *arXiv preprint arXiv:1912.03432*, 2019.

John Bronskill, Jonathan Gordon, James Requeima, Sebastian Nowozin, and Richard E Turner. Tasknorm: Rethinking batch normalization for meta-learning. *arXiv preprint arXiv:2003.03284*, 2020.

Tianshi Cao, Marc Law, and Sanja Fidler. A theoretical analysis of the number of shots in few-shot learning. In *Proceedings of the International Conference on Learning Representations*, 2020.

Wei-Lun Chao, Han-Jia Ye, De-Chuan Zhan, Mark Campbell, and Kilian Q Weinberger. Revisiting meta-learning as supervised learning. *arXiv preprint arXiv:2002.00573*, 2020.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *Proceedings of the International Conference on Learning Representations*, 2019.

Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020.

Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pp. 6594–6604, 2017.

Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *Proceedings of the International Conference on Learning Representations*, 2020.

Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *Proceedings of the International Conference on Learning Representations*, 2020.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *Proceedings of the International Conference on Learning Representations*, 2017.

Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 2018.

Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Selecting relevant features from a universal representation for few-shot classification. *arXiv preprint arXiv:2003.09338*, 2020.

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

Micah Goldblum, Steven Reich, Liam Fowl, Renkun Ni, Valeriia Cherepanova, and Tom Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *Proceedings of the International Conference on Machine Learning*, 2020.

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

Meiyu Huang, Xueshuang Xiang, and Yao Xu. Training few-shot classification via the perspective of minibatch and pretraining. *arXiv preprint arXiv:2004.05910*, 2020.

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

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 721–731, 2018.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems*, pp. 7957–7968, 2019.

Tonmoy Saikia, Thomas Brox, and Cordelia Schmid. Optimized generic feature learning for few-shot classification across domains. *arXiv preprint arXiv:2001.07926*, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017.

Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, pp. 2255–2265, 2017.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-Dataset: A dataset of datasets for learning to learn from few examples. In *Proceedings of the International Conference on Learning Representations*, 2020.

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

A Appendix

SCONE 's training algorithm in more detail

For clarity, we provide pseudocode for SCONE 's training algorithm including our procedure for shot smoothing in Algorithm 1. We will also release our code upon publication for reproducibility.

Hypothesis testing We follow the same procedure as in (Triantafillou et al., 2020) to determine which entries to bold in our tables. Specifically, we perform a 95% confidence interval statistical test on the difference between the mean accuracies of the two entries of that row. If we are not able to reject the null hypothesis that the difference between their means is 0, we bold both entries. If we are able to reject that hypothesis, we bold whichever entry has the largest mean accuracy.

Experimental details

We plan to open source our code upon publication, including all experimental details. In the meantime, we outline these details below for completeness.

Architecture We use ResNet-18 as the feature extractor for all of our experiments, following the implementation in (Triantafillou et al., 2020). For the SCONE variants, we add FiLM to all of the batch normalization layers throughout the network.

Algorithm 1 SCONE training

Input: Distributions of training episodes P_{train}, pre-trained embedding weights θ, pre-trained batch norm weights γ and β, embedding function f, learning rate ϵ (a float), smoothing co-efficient m (a float in the range $[0, 1]$) and maximum supported shot MAX-SHOT (an int).

Output: Finetuned embedding weights θ' and FiLM parameters $\mathcal{F} = \{\gamma', \beta'\}$.

procedure SMOOTH-SHOT($s, m,$ MAX-SHOT)
  if $s >$ MAX-SHOT **then**
    $s \leftarrow$ MAX-SHOT ▷ Cap s to the max supported shot
  end if
  $s \leftarrow s - 1$ ▷ So that s is in the range $[0,$ MAX-SHOT $- 1]$
  $\tilde{s} \leftarrow$ ONE-HOT(s, DEPTH=MAX-SHOT) ▷ Init the smoothed shot
  for $0 \leq j \leq$ MAX-SHOT **do**
    $l \leftarrow s - j - 1$ ▷ The index j slots to the left of s
    $l \leftarrow$ ONE-HOT(l, DEPTH=MAX-SHOT) $* m$ ▷ Outputs the zero vector if $l < 0$
    $r \leftarrow s + j + 1$ ▷ The index j slots to the right of s
    $r \leftarrow$ ONE-HOT(r, DEPTH=MAX-SHOT) $* m$ ▷ Outputs the zero vector if $r < 0$
    $\tilde{s} \leftarrow \tilde{s} + l + r$
    $m \leftarrow m^2$ ▷ Adjust the next iteration's smoothing
  end for
end procedure

$\theta' \leftarrow \theta$ ▷ Init the embedding weights from the pre-trained embeddings
for $1 \leq k \leq$ MAX-SHOT **do** ▷ Init the FiLM params from the pre-trained batch norm params
  $\gamma'(k) \leftarrow \gamma$
  $\beta'(k) \leftarrow \beta$
end for
while validation accuracy is improving **do**
  Sample a training episode with support set $\mathcal{S}$ and query set $\mathcal{Q}$
  Let $k_1, \ldots k_N$ be the shots of the episode's classes.
  $s \leftarrow$ ZEROS(MAX-SHOT) ▷ Init the (unnormalized) shot distribution
  for each class i **do**
    $s_i \leftarrow$ ONE-HOT(k_i, DEPTH $=$ MAX-SHOT)
    $s_i \leftarrow$ SMOOTH-SHOT($s_i, m,$ MAX-SHOT) ▷ Smooth the one-hot shot of class i
    $s \leftarrow s + s_i$
  end for
  $s \leftarrow s \div$ SUM(s) ▷ Normalize to get the episode's shot distribution
  $\gamma'_s \leftarrow s^T \gamma'$ ▷ Select the FiLM params for the episode
  $\beta'_s \leftarrow s^T \beta'$
  Let $\mathcal{S}^H = \{f(x; \theta', \gamma'_s, \beta'_s), y\}_{(x,y) \in \mathcal{S}}$ ▷ The embedded support set
  Let $\mathcal{Q}^H = \{f(x; \theta', \gamma'_s, \beta'_s), y\}_{(x,y) \in \mathcal{Q}}$ ▷ The embedded query set
  $\mathcal{L} \leftarrow \frac{1}{|\mathcal{Q}^H|} \sum_{(h^*, y^*) \in \mathcal{Q}^H} -\log p(y^* \mid h^*, \mathcal{S}^H)$ ▷ Compute the episode's loss
  $\theta' \leftarrow \theta' - \epsilon \dfrac{\partial \mathcal{L}}{\partial \theta'}$ ▷ Update the model via gradient descent
  $\gamma' \leftarrow \gamma' - \epsilon \dfrac{\partial \mathcal{L}}{\partial \gamma'}$
  $\beta' \leftarrow \beta' - \epsilon \dfrac{\partial \mathcal{L}}{\partial \beta'}$
end while

Image processing For all experiments, we use Meta-Dataset's input pipeline to obtain images, and we follow the image processing performed in (Chen et al., 2020) which yields images of size 128 x 128. We apply standard data augmentation consisting of horizontal flipping and random cropping followed by standardization using a commonly-used mean and standard deviation as in (Chen et al., 2020). For episodic models, data augmentation is applied in both the support and query sets. No data augmentation is used at validation nor test time.

Optimization We use ADAM with exponential learning rate decay and weight decay of $1e-8$ to optimize all models in this work. We tune the initial learning rate, the decay rate, and the number of updates between each learning rate decay separately for each model presented in the paper. The initial learning rate values we considered are 0.0005 and 0.001, with a decay factor of 0.8 or 0.9 applied every 1000, 2000, 3000 steps. We ran a variant for every combination of those values. We also tune the weight decay applied to the FiLM parameters (for SCONE variants) or the batch normalization parameters (for non-SCONE variants). We tried the values: $1e-8$, $1e-6$, $1e-4$.

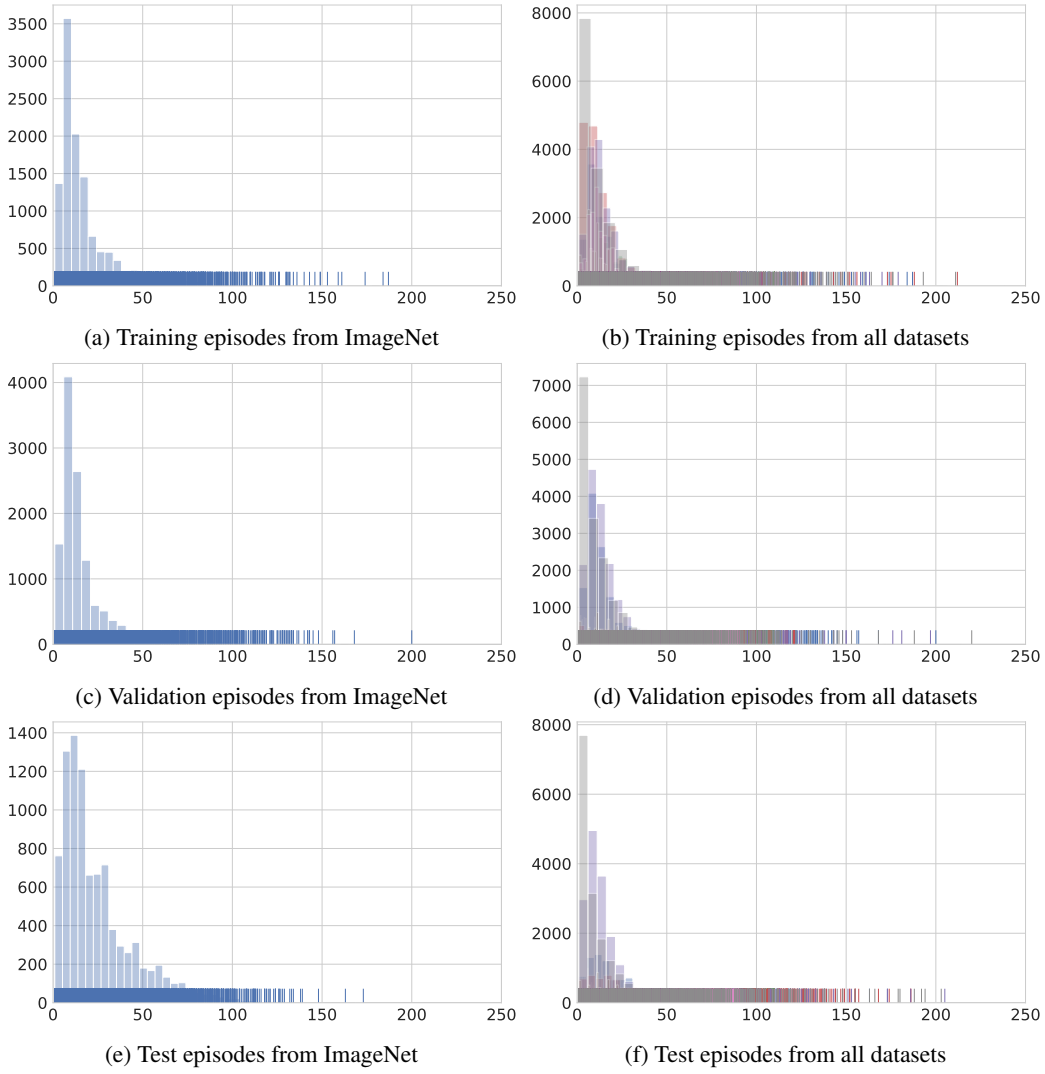
SCONE hyperparameters For the SCONE variants, aside from the above hyperparameters, we additionally tune the smoothing parameter m described in the main paper that is used for training and for evaluation. We did not tune the MAX-SHOT hyperparameter mentioned in the main paper as we found that our initial choices worked reasonably. Specifically, we set it to 40 for the smaller-scale experiments where the maximum shot was 40, and to 200 for the large-scale experiments. The latter choice was performed heuristically since shots much larger than 200 are unlikely under the shot distribution induced by Meta-Dataset's episode generator. For more information on that shot distribution, we refer the reader to the next section.

SCONE smoothing hyperparameter We tuned the value of this hyperparameter that will be used both at training and at evaluation. At training time, we considered values in the range $0, 0.2, 0.4, 0.6, 0.9$ for Prototypical Network experiments, and we picked the variant that worked best according to the validation performance that was computed without smoothing. Once the model was trained and all of the remaining hyperparamters were tuned, we performed a final validation round to tune the evaluation-time smoothing that will be used in the chosen model. We found it beneficial to use larger values here, picking the value of $1-1e-06$ for example for the Prototypical Network on ImageNet. In the Meta-Baseline codebase, we trained with larger values of smoothing (the best we found was $1-1e-10$) and didn't find it beneficial to additionally smooth at evaluation time.

Model selection For each experiment, we perform early stopping according to the performance on the validation set. For the models that train on a single shot k in the smaller-scale experiments, the validation performance that we monitor for early stopping is the average query set accuracy on k-shot 5-way episodes drawn from the validation set. For the models in the small-scale experiments that train on a distribution of shots, we use the average validation performance over 5-way episodes whose shot is sampled according to the same distribution used for training the respective model. For the larger-scale Meta-Dataset experiments, we draw validation episodes only from the validation set of ImageNet for the experiments that train on ImageNet only, or from the validation sets of all datasets for the experiments that train on all datasets. In both cases, the validation episodes are drawn using Meta-Dataset's episode generator that yields episodes of variable ways and variable shots with class imbalance. In all cases, the average validation performance is computed over 600 validation episodes and is monitored every 2K training updates. We apply exponential smoothing to the resulting validation "curve" (using the default value of 0.6 in TensorBoard). Then, we choose the update step at which the highest peak of that curve is found and we use the checkpoint corresponding to that update step for testing.

DISTRIBUTION OF SHOTS IN META-DATASET EPISODES

For reference, Figure 5 displays histograms of the number of shots produced by Meta-Dataset's episode sampling algorithm. These are computed by sampling 600 episodes per dataset for each of the training, validation and test splits of Meta-Dataset.



(a) Training episodes from ImageNet

(b) Training episodes from all datasets

(c) Validation episodes from ImageNet

(d) Validation episodes from all datasets

(e) Test episodes from ImageNet

(f) Test episodes from all datasets

Figure 5: Histogram of shots appearing in episodes generated using Meta-Dataset's sampling algorithm for the different splits.