Appendix: A Baseline for Few-Shot Image Classification

Guneet S. Dhillon[1], **Pratik Chaudhari**[2,*], **Avinash Ravichandran**[1], **Stefano Soatto**[1,3]
[1]Amazon Web Services, [2]University of Pennsylvania, [3]University of California, Los Angeles
{guneetsd, ravinash, soattos}@amazon.com, pratikac@seas.upenn.edu

A Problem definition and related work

We first introduce some notation and formalize the few-shot image classification problem. Let (x, y) denote an image and its ground-truth label respectively. The training and test datasets are $\mathcal{D}_{\mathrm{s}} = \{(x_i, y_i)\}_{i=1}^{N_{\mathrm{s}}}$ and $\mathcal{D}_{\mathrm{q}} = \{(x_i, y_i)\}_{i=1}^{N_{\mathrm{q}}}$ respectively, where $y_i \in C_{\mathrm{t}}$ for some set of classes C_{t}. The training and test datasets are disjoint. In the few-shot learning literature, training and test datasets are commonly referred to as support and query datasets respectively, and are collectively called a few-shot episode. The number of *ways*, or classes, is $|C_{\mathrm{t}}|$. The set $\{x_i \mid y_i = k,\ (x_i, y_i) \in \mathcal{D}_{\mathrm{s}}\}$ of samples is the *support* of class k and its cardinality is (non-zero) s *support shots* (more generally referred to as *shots*). The set $\{x_i \mid y_i = k, (x_i, y_i) \in \mathcal{D}_{\mathrm{q}}\}$ of samples is the *query* of class k and its cardinality is q *query shots*. The goal is to learn a function F to exploit the training set $\mathcal{D}_{\mathrm{s}}$ to predict the label of a test datum x, where $(x, y) \in \mathcal{D}_{\mathrm{q}}$, by

$$\hat{y} = F(x;\ \mathcal{D}_{\mathrm{s}}). \tag{1}$$

Typical approaches for supervised learning replace $\mathcal{D}_{\mathrm{s}}$ above with a statistic, $\theta^* = \theta^*(\mathcal{D}_{\mathrm{s}})$ that is, ideally, sufficient to classify the training data, as measured by, say, the cross-entropy loss

$$\theta^*(\mathcal{D}_{\mathrm{s}}) = \arg\min_{\theta} \frac{1}{N_{\mathrm{s}}} \sum_{(x,y)\in\mathcal{D}_{\mathrm{s}}} -\log p_\theta(y|x), \tag{2}$$

where $p_\theta(\cdot|x)$ is the probability distribution on the set of classes C_{t} as predicted by the model in response to an input x. When presented with a test datum, the classification rule is typically chosen to be of the form

$$F_{\theta^*}(x; \mathcal{D}_{\mathrm{s}}) \triangleq \arg\max_{k} p_{\theta^*}(k|x), \tag{3}$$

where the training set $\mathcal{D}_{\mathrm{s}}$ is *represented* by the parameters (weights) θ^*. This form of the classifier entails a loss of generality *unless* θ^* is a minimal sufficient statistic, $p_{\theta^*}(y|x) = p(y|x)$, which is of course never the case, especially given few labeled data in $\mathcal{D}_{\mathrm{s}}$. However, it conveniently separates training and inference phases, so we never have to revisit the training set. This is desirable in ordinary image classification where the training set consists of millions of samples, but not in few-shot learning. We therefore adopt the more general form of F in (1).

If we call the test datum $x = x_{N_{\mathrm{s}}+1}$, then we can obtain the general form of the classifier by

$$\hat{y} = F(x; \mathcal{D}_{\mathrm{s}}) = \arg\min_{y_{N_{\mathrm{s}}+1}} \min_{\theta} \frac{1}{N_{\mathrm{s}}+1} \sum_{i=1}^{N_{\mathrm{s}}+1} -\log p_\theta(y_i|x_i). \tag{4}$$

In addition to the training (support) set, one typically also has a *meta-training* set,

$$\mathcal{D}_{\mathrm{m}} = \{(x_i, y_i)\}_{i=1}^{N_{\mathrm{m}}}, \quad \text{where } y_i \in C_{\mathrm{m}},$$

*Work done while at Amazon Web Services

3rd Workshop on Meta-Learning at NeurIPS 2019, Vancouver, Canada.

with classes C_{m} disjoint from C_{t}. The goal of meta-training is to use $\mathcal{D}_{\mathrm{m}}$ to infer the parameters of the few-shot learning model: $\hat{\theta}(\mathcal{D}_{\mathrm{s}}; \mathcal{D}_{\mathrm{m}}) = \arg\min_\theta \frac{1}{N_{\mathrm{m}}} \sum_{(x,y)\in\mathcal{D}_{\mathrm{m}}} \ell(y, F_\theta(x; \mathcal{D}_{\mathrm{s}}))$ where ℓ is a meta-training loss that depends on the specific method.

A.1 Related work

A.1.1 Few-shot learning

The meta-training loss is designed to make few-shot training efficient [1, 2, 3, 4]. This approach partitions the problem into a base-level that performs standard supervised learning and a meta-level that accrues information from the base-level. Two main approaches have emerged to do so.

Gradient-based approaches: These approaches treat the updates of the base-level as a learnable mapping [5]. This mapping can be learnt using temporal models [6, 7], or one can back-propagate the gradient across the base-level updates [8, 9]. It is however challenging to perform this dual or bi-level optimization, respectively. These approaches have not been shown to be competitive on large datasets. A recent line of work learns the base-level in closed-form using simpler models such as SVMs [10, 11] which restricts the capacity of the base-level although it alleviates the optimization problem.

Metric-based approaches: A large majority of state-of-the-art algorithms are metric-based meta-learners. These techniques learn an embedding over the meta-training tasks that can be used to compare [12, 13] or cluster [14, 15] query samples. A number of recent results build upon this idea with increasing levels of sophistication in how they learn the embedding [14, 16, 17], creating exemplars from the support set and picking a metric for the embedding [18, 17, 19]. There are numerous hyper-parameters and design choices involved in implementing these approaches which makes it hard to evaluate them systematically [20].

A.1.2 Transductive learning

This approach is more efficient at using few labeled data than supervised learning [21, 22, 23]. The idea is to use information from the test datum x to restrict the hypothesis space while searching for the classifier $F(x, \mathcal{D}_{\mathrm{s}})$ *at test time*. This search can be bootstrapped using a model trained on $\mathcal{D}_{\mathrm{m}}$ and $\mathcal{D}_{\mathrm{s}}$. Our approach is closest to this line of work. We use the backbone trained on the meta-training set $\mathcal{D}_{\mathrm{m}}$ and initialize a classifier using the support set $\mathcal{D}_{\mathrm{s}}$. Both the classifier and the backbone are then fine-tuned to adapt to the new test datum x.

In the few-shot learning context, there are recent papers such as [24, 25] that are motivated from transductive learning and exploit the unlabeled query samples. The former updates batch-normalization parameters using the query samples while the latter uses label propagation to estimate labels of all the query samples at once.

A.1.3 Semi-supervised learning

We penalize the entropy of the predictions on the query samples at test time. This is a simple technique in the semi-supervised learning literature and is closest to [26]. Modern augmentation techniques such as [27, 28, 29] or graph-based approaches [30] can also be used with our approach; we used the entropic penalty for the sake of simplicity. Semi-supervised few-shot learning is typically formulated as having access to extra unlabeled data during meta-training or few-shot training [31, 32]. Note that this is different from our approach which uses the unlabeled query samples for transductive learning.

A.1.4 Initialization for fine-tuning

We use recent ideas from the deep metric learning literature [33, 20, 34, 35] to initialize the fine-tuning for the backbone. These works connect the softmax cross-entropy loss with cosine distance.

B Experimental Setup

Datasets: We use the following datasets for our benchmarking experiments.

- The Mini-ImageNet dataset [14] which is a subset of Imagenet-1k [36] and consists of 84 $\times$ 84 sized images with 600 images per class. There are 64 training, 16 validation and 20 test

classes. There are multiple versions of this dataset in the literature; we obtained the dataset from the authors of [17][2].

- The Tiered-ImageNet dataset [32] is a larger subset of Imagenet-1k with 608 classes split as 351 training, 97 validation and 160 testing classes, each with about 1300 images of size 84 $\times$ 84. This dataset ensures that training, validation and test classes do not have a semantic overlap and is a potentially harder few-shot learning dataset.
- We also consider two smaller CIFAR-100 [37] derivatives, both with 32 $\times$ 32 sized images. The first is the CIFAR-FS dataset [10] which splits classes randomly into 64 training, 16 validation and 20 test with 600 images in each. The second is the FC-100 dataset [16] which splits CIFAR-100 into 60 training, 20 validation and 20 test classes with minimal semantic overlap, containing 600 images per class.

Meta-training: We use various backbone architectures for our experiments - conv $(64)_{\times 4}$ [14, 15], ResNet-12 [38, 16, 11], WRN-28-10 and WRN-16-4 [39, 40, 41]. Experiments on conv $(64)_{\times 4}$ and ResNet-12 are in Appendix D. These networks are trained using standard data augmentation, cross-entropy loss with label smoothing [42] of ϵ=0.1, mixup regularization [43] of α=0.25, SGD with batch-size of 256, Nesterov's momentum of 0.9, weight-decay of 10^{-4} and no dropout. Batch-normalization [44] is used, but its parameters are excluded from weight decay [45]. We use cyclic learning rates [46] and half-precision distributed training on 8 Nvidia V100 GPUs [47, 48] to reduce training time.

Meta-training dataset: Some papers in the literature use only the training classes as the meta-training set and report few-shot results, while others use both training and validation classes for meta-training. For completeness we report results using both methodologies; the former is denoted as (train) while the latter is denoted as (train + val).

Fine-tuning: We perform fine-tuning on one GPU in full-precision for 25 epochs and a fixed learning rate of 5×10^{-5} with Adam [49] without any regularization. We do not use mini-batches but make two weight updates in each epoch: one for the cross-entropy term using support samples and one for the Shannon Entropy term using query samples (cf. Section 2.3).

Data augmentation: Input images are normalized using the mean and standard-deviation computed on Imagenet-1k. Our Data augmentation consists of left-right flips with probability of 0.5, padding the image with 4px and adding brightness and contrast changes of $\pm$ 40%. The augmentation is kept the same for both meta-training and fine-tuning.

Hyper-parameters: We used images from Imagenet-1k belonging to the training classes of Mini-ImageNet as the validation set for pre-training the backbone for Mini-ImageNet. We used the validation set of Mini-ImageNet to choose hyper-parameters for fine-tuning. **All hyper-parameters are kept constant for experiments on benchmark datasets**, namely Mini-ImageNet, Tiered-ImageNet, CIFAR-FS and FC-100.

Evaluation: Few-shot episodes contain classes that are sampled uniformly from classes in the test sets of the respective datasets; support and query samples are further sampled uniformly for each class; the query shot is fixed to 15 for all experiments unless noted otherwise. All networks are evaluated over 1,000 few-shot episodes unless noted otherwise. To enable easy comparison with existing literature, we report an estimate of the mean accuracy and the 95% confidence interval of this estimate.

All experiments in Section 3 and Appendix D use the (train + val) setting, pre-training on both the training and validation data of the corresponding datasets.

C Setup for Imagenet-21k

The blue region in Fig. 4 denotes our training set with 7,491 classes. The green region shows 13,007 classes with at least 10 images each, and is the test set. We do not use the red region consisting of 1,343 classes with less than 10 images each. We train the same backbone WRN-28-10 with the same procedure as that in Appendix B on 84 $\times$ 84 resized images, albeit for only 24 epochs. Since we use the same hyper-parameters as the other benchmark datasets, we did not create validation sets for meta-training or the few-shot fine-tuning phases. We create few-shot episodes from the test set in the same way as Appendix B. We evaluate using fewer few-shot episodes (80) on this dataset because we would like to demonstrate the performance across a large number of different ways.

[2] https://github.com/gidariss/FewShotWithoutForgetting

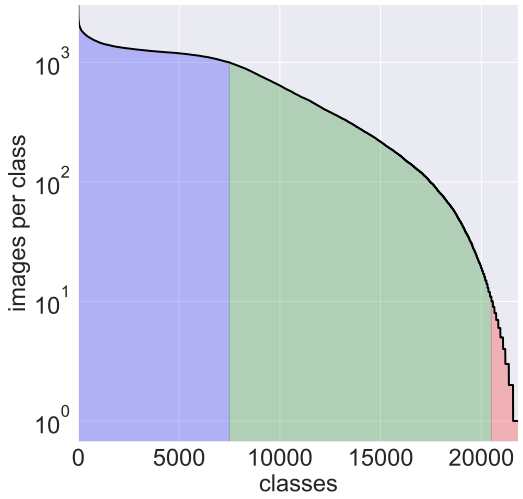

Figure 4: **Imagenet-21k is a highly imbalanced dataset.** The most frequent class has about 3K images while the rarest class has a single image.

The improvements of transductive fine-tuning are minor for Imagenet-21k (cf. Section 3.2) because the accuracies are extremely high even at initialization. We noticed a slight degradation of the accuracy due to transductive fine-tuning at high ways because the entropic term in in the loss function (cf. Section 2.3) is much larger than the the cross-entropy loss. The experiments for Imagenet-21k therefore scale down the entropic term by $\log |C_{\mathrm{t}}|$ and forego the ReLU before the input to the classifier (cf. Section 2.2). We find that the difference in accuracy between support-based initialization and transductive fine-tuning, after this change, is small for high ways.

D Analysis

This section contains additional experiments and analysis, complementing Section 3.

D.1 Transductive fine-tuning changes the embedding dramatically

Fig. 5 demonstrates this effect. The logits for query samples are far from those of their respective support samples and metric-based loss functions, e.g., those for prototypical networks [15] would have a poor loss on this episode; indeed the accuracy after the support-based initialization is 64%. Logits for the query samples change dramatically during transductive fine-tuning and majority of the query samples cluster around their respective supports. The post transductive fine-tuning accuracy of this episode is 73.3%. This suggests that modifying the embedding using the query samples is crucial to obtaining good performance on new classes. This example also demonstrates that the support-based initialization is efficient, logits of the support samples are relatively unchanged during the transductive fine-tuning phase.

D.2 Using features of the backbone as input to the classifier

Instead of re-initializing the final fully-connected layer of the backbone to classify new classes, we simply append the classifier on top of it. We implemented the former, more common, approach and found that it achieves an accuracy of $64.20 \pm 0.65\%$ and $81.26 \pm 0.45\%$ for 1-shot 5-way and 5-shot 5-way respectively on Mini-ImageNet, while the accuracy on Tiered-ImageNet is $67.14 \pm 0.74\%$ and $86.67 \pm 0.46\%$ for 1-shot 5-way and 5-shot 5-way respectively. These numbers are significantly lower for the 1-shot 5-way protocol on both datasets compared to their counterparts in Section 3.1. However, the 5-shot 5-way accuracy is marginally higher in this experiment than that in Section 3.1. Logits of the backbone are well-clustered and that is why they work better for low-shot scenarios [24].

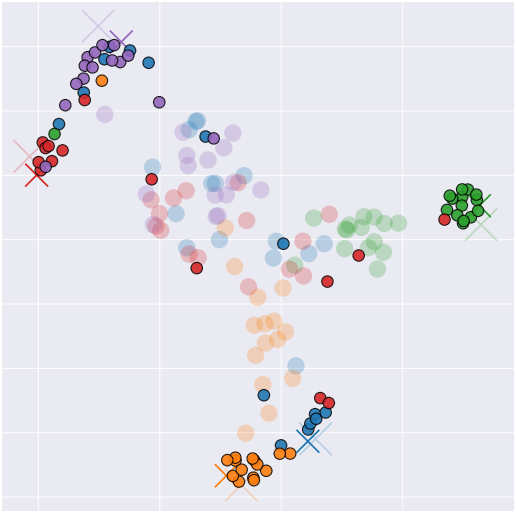

Figure 5: **t-SNE [50] embedding of the logits for 1-shot 5-way few-shot episode of Mini-ImageNet.** Colors denote the ground-truth labels; crosses denote the support samples; circles denote the query samples; translucent markers and opaque markers denote the embeddings before and after transductive fine-tuning respectively. Even though query samples are far away from their respective supports in the beginning, they move towards the supports by the end of transductive fine-tuning. Logits of support samples are relatively unchanged which suggests that the support-based initialization is effective.

D.3 Freezing the backbone restricts performance

The previous observation suggests that the network changes a lot in the fine-tuning phase. Freezing the backbone severely restricts the changes in the network to only changes to the classifier. As a consequence, the accuracy of freezing the backbone is 58.38 ± 0.66 % and 75.46 ± 0.52% on Mini-ImageNet and 67.06 ± 0.69% and 83.20 ± 0.51% on Tiered-ImageNet for 1-shot 5-way and 5-shot 5-way respectively. While the 1-shot 5-way accuracies are much lower than their counterparts in Section 1, the gap in the 5-shot 5-way scenario is smaller.

D.4 Large backbone vs. small backbone

The expressive power of the backbone plays an important role in the efficacy of fine-tuning. We observed that a WRN-16-4 architecture (2.7M parameters) performs worse than WRN-28-10 (36M parameters). The former obtains 63.28 ± 0.68% and 77.39 ± 0.5% accuracy on Mini-ImageNet and 69.04 ± 0.69% and 83.55 ± 0.51% accuracy on Tiered-ImageNet on 1-shot 5-way and 5-shot 5-way protocols respectively. While these numbers are comparable to those of state-of-the-art algorithms, they are lower than their counterparts for WRN-28-10 in Section 3.1. This suggests that a larger network is effective in learning richer features from the meta-training classes, and fine-tuning is effective in taking advantage of this to further improve performance on samples belonging to few-shot classes.

The above experiment also helps understand why fine-tuning as a strong baseline for few-shot learning has not been noticed in the literature before, the baseline architectures were smaller. This observation was partially made by [20]. They however only concluded that gaps between meta-learning and fine-tuning-based approaches are reduced with a larger backbone. We take this point further and show in Section 3.1 that fine-tuning a large backbone can be better than meta-learning approaches even for the same backbone. The support-based initialization is another reason why our approach is so effective. With this initialization, we need to fine-tune the classifier, transductively or not, only for a few epochs (25 in our experiments).

D.5 Latency with a smaller backbone

The WRN-16-4 architecture (2.7M parameters) is much smaller than WRN-28-10 (36M parameters) and transductive fine-tuning on the former is much faster. As compared to our implementation of [15] with the same backbone, WRN-16-4 is 20-70$\times$ slower (0.87 vs. 0.04 seconds for a query shot of 1,

and 2.85 vs. 0.04 seconds for a query shot of 15) for the 1-shot 5-way scenario. The latency with respect to metric-based approaches is thus smaller for WRN-16-4. Compare this to the computational complexity experiment in Section 3.3.

As discussed in Appendix D.4, the accuracy of WRN-16-4 is $63.28 \pm 0.68\%$ and $77.39 \pm 0.5\%$ for 1-shot 5-way and 5-shot 5-way on Mini-ImageNet respectively. As compared to this, our implementation of [15] using a WRN-16-4 backbone obtains $57.29 \pm 0.40\%$ and $75.34 \pm 0.32\%$ accuracies for the same settings; the former number in particular is significantly worse than its counterpart for WRN-16-4.

D.6 Using mixup during pre-training

Mixup improves the few-shot accuracy by about 1%: The accuracy for WRN-28-10 trained without mixup is $67.06 \pm 0.71\%$ and $79.29 \pm 0.51\%$ on Mini-ImageNet for 1-shot 5-way and 5-shot 5-way respectively.

D.7 Comparisons against backbone architectures in the current literature

We include experiments using conv $(64)_{\times 4}$ [14, 15], ResNet-12 [38, 16, 11] and WRN-16-4 [41] in Table 1, in addition to WRN-28-10 in Section 3.1, in order to facilitate comparisons of the proposed baseline for different architectures. Our results are comparable or better than existing results for a given backbone architectures, except for those in [51], which use a graph-based transduction algorithm, for conv $(64)_{\times 4}$ on Mini-ImageNet and Tiered-ImageNet. In line with our goal of simplicity, we kept the hyper-parameters for pre-training and fine-tuning the same as the ones used for WRN-28-10 (cf. Section 2, Appendix B).

D.8 A proposal for reporting few-shot classification performance

As discussed in Section 1, we need better metrics to report the performance of few-shot algorithms. There are two main issues: (i) standard deviation of the few-shot accuracy across different sampled episodes for a given algorithm, dataset and few-shot protocol is very high (cf. Section 1), and (ii) different models and hyper-parameters for different few-shot protocols makes evaluating algorithmic contributions difficult (cf. Section 3.1). This section takes a step towards resolving these issues.

Hardness of an episode: Classification performance on the few-shot episode is determined by the relative location of the features corresponding to query samples. If query samples belonging to different classes have similar features one expects the accuracy of the classifer to be low. On the other hand, if the features are far apart the classifier can distinguish easily between the classes to obtain a high accuracy. The following definition characterizes this intuition.

For training data $\mathcal{D}_{\mathrm{s}}$ and test data $\mathcal{D}_{\mathrm{q}}$, we will define the hardness Ω_{φ} as the average log-odds of a test datum being classified incorrectly by the prototypical loss [15]. More precisely,

$$\Omega_{\varphi}(\mathcal{D}_{\mathrm{q}};\, \mathcal{D}_{\mathrm{s}}) = \frac{1}{N_{\mathrm{q}}} \sum_{(x,y)\in\mathcal{D}_{\mathrm{q}}} \log \frac{1 - p(y \mid x)}{p(y \mid x)}, \tag{5}$$

where $p(\cdot \mid x_q)$ is a unit temperature softmax distribution with logits $z_{y_q} = w\varphi(x_q)$ for a query sample x_q where w is the weight matrix constructed using support-based initialization (cf. Section 2.1) and $\mathcal{D}_{\mathrm{s}}$. We imagine $\varphi(x)$ to be the ℓ_2 normalized embedding computed using any rich-enough feature generator, say a deep network trained for standard image classification.

Note that Ω_{φ} does not depend on the few-shot learner and gives a measure of how difficult the classification problem is for any few-shot episode, using a generic feature extractor.

Fig. 6 demonstrates how to use the hardness metric. Few-shot **accuracy degrades linearly with hardness**. Performance for all hardness can thus be estimated simply by testing for two different ways. We advocate **selecting few-shot learning hyper-parameters using the area under the fitted curves as a metric** instead of tuning them specifically for each few-shot protocol. The advantage of such a test methodology is that it predicts the performance of the model across multiple few-shot protocols systematically.

Different algorithms can be compared directly, e.g., transductive fine-tuning (solid lines) and support-based initialization (dotted lines). For instance, the former leads to large improvements on easy episodes, the performance is similar for hard episodes, especially for Tiered-ImageNet and Imagenet-21k.

Table 1: **Few-shot accuracies on benchmark datasets for 5-way few-shot episodes.** The notation conv $(64^k)_{\times 4}$ denotes a CNN with 4 layers and 64^k channels in the k^{th} layer. The rows are sorted by the backbone architecture. Best results in each column and for a given backbone architecture are shown in bold. Results where the support-based initialization is better than existing algorithms are denoted by †. The notation (train + val) indicates that the backbone was trained on both training and validation data of the datasets; the backbone is trained only on the training set when not indicated. The authors in [11] use a $1.25\times$ wider ResNet-12 which we denote as ResNet-12 *.

		Mini-ImageNet		**Tiered-ImageNet**		**CIFAR-FS**		**FC-100**	
Algorithm	Architecture	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)
MAML [9]	conv $(32)_{\times 4}$	48.70 ± 1.84	63.11 ± 0.92						
Matching networks [14]	conv $(64)_{\times 4}$	46.6	60						
LSTM meta-learner [7]	conv $(64)_{\times 4}$	43.44 ± 0.77	60.60 ± 0.71						
Prototypical Networks [15]	conv $(64)_{\times 4}$	49.42 ± 0.78	68.20 ± 0.66						
Transductive Propagation [51]	conv $(64)_{\times 4}$	**55.51 ± 0.86**	**69.86 ± 0.65**	**59.91 ± 0.94**	73.30 ± 0.75				
Support-based initialization (train)	conv $(64)_{\times 4}$	50.69 ± 0.63	66.07 ± 0.53	58.42 ± 0.69	**73.98 ± 0.58**†	**61.77 ± 0.73**	**76.40 ± 0.54**	**36.07 ± 0.54**	48.72 ± 0.57
Fine-tuning (train)	conv $(64)_{\times 4}$	49.43 ± 0.62	66.42 ± 0.53	57.45 ± 0.68	**73.96 ± 0.56**	59.74 ± 0.72	76.37 ± 0.53	35.46 ± 0.53	49.43 ± 0.57
Transductive fine-tuning (train)	conv $(64)_{\times 4}$	50.46 ± 0.62	66.68 ± 0.52	58.05 ± 0.68	**74.24 ± 0.56**	**61.73 ± 0.72**	**76.92 ± 0.52**	**36.62 ± 0.55**	**50.24 ± 0.58**
R2D2 [10]	conv $(96^k)_{\times 4}$	51.8 ± 0.2	68.4 ± 0.2			65.4 ± 0.2	79.4 ± 0.2		
TADAM [16]	ResNet-12	58.5 ± 0.3	**76.7 ± 0.3**					40.1 ± 0.4	**56.1 ± 0.4**
Transductive Propagation [51]	ResNet-12	59.46	75.64						
Support-based initialization (train)	ResNet-12	54.21 ± 0.64	70.58 ± 0.54	66.39 ± 0.73	81.93 ± 0.54	65.69 ± 0.72	79.95 ± 0.51	35.51 ± 0.53	48.26 ± 0.54
Fine-tuning (train)	ResNet-12	56.67 ± 0.62	74.80 ± 0.51	64.45 ± 0.70	**83.59 ± 0.51**	64.66 ± 0.73	**82.13 ± 0.50**	37.52 ± 0.53	55.39 ± 0.57
Transductive fine-tuning (train)	ResNet-12	**62.35 ± 0.66**	74.53 ± 0.54	**68.41 ± 0.73**	**83.41 ± 0.52**	**70.76 ± 0.74**	81.56 ± 0.53	**41.89 ± 0.59**	54.96 ± 0.55
MetaOpt SVM [11]	ResNet-12 *	62.64 ± 0.61	78.63 ± 0.46	65.99 ± 0.72	81.56 ± 0.53	72.0 ± 0.7	84.2 ± 0.5	41.1 ± 0.6	55.5 ± 0.6
Support-based initialization (train)	WRN-28-10	56.17 ± 0.64	73.31 ± 0.53	67.45 ± 0.70	82.88 ± 0.53	70.26 ± 0.70	83.82 ± 0.49	36.82 ± 0.51	49.72 ± 0.55
Fine-tuning (train)	WRN-28-10	57.73 ± 0.62	**78.17 ± 0.49**	66.58 ± 0.70	**85.55 ± 0.48**	68.72 ± 0.67	**86.11 ± 0.47**	38.25 ± 0.52	**57.19 ± 0.57**
Transductive fine-tuning (train)	WRN-28-10	**65.73 ± 0.68**	**78.40 ± 0.52**	**73.34 ± 0.71**	**85.50 ± 0.50**	**76.58 ± 0.68**	**85.79 ± 0.50**	**43.16 ± 0.59**	**57.57 ± 0.55**
Support-based initialization (train + val)	conv $(64)_{\times 4}$	**52.77 ± 0.64**	**68.29 ± 0.54**	**59.08 ± 0.70**	**74.62 ± 0.57**	**64.01 ± 0.71**	**78.46 ± 0.53**	**40.25 ± 0.56**	54.53 ± 0.57
Fine-tuning (train + val)	conv $(64)_{\times 4}$	51.40 ± 0.61	**68.58 ± 0.52**	58.04 ± 0.68	**74.48 ± 0.56**	62.12 ± 0.71	**77.98 ± 0.52**	39.09 ± 0.55	54.83 ± 0.55
Transductive fine-tuning (train + val)	conv $(64)_{\times 4}$	**52.30 ± 0.61**	**68.78 ± 0.53**	**58.81 ± 0.69**	**74.71 ± 0.56**	**63.89 ± 0.71**	**78.48 ± 0.52**	**40.33 ± 0.56**	**55.60 ± 0.56**
Support-based initialization (train + val)	ResNet-12	56.79 ± 0.65	72.94 ± 0.55	67.60 ± 0.71	83.09 ± 0.53	69.39 ± 0.71	83.27 ± 0.50	43.11 ± 0.58	58.16 ± 0.57
Fine-tuning (train + val)	ResNet-12	58.64 ± 0.64	**76.83 ± 0.50**	65.55 ± 0.70	**84.51 ± 0.50**	68.11 ± 0.70	**85.19 ± 0.48**	42.84 ± 0.57	**63.10 ± 0.57**
Transductive fine-tuning (train + val)	ResNet-12	**64.50 ± 0.68**	**76.92 ± 0.55**	**69.48 ± 0.73**	**84.37 ± 0.51**	**74.35 ± 0.71**	84.57 ± 0.53	**48.29 ± 0.63**	**63.38 ± 0.58**
MetaOpt SVM (train + val) [11]	ResNet-12 *	64.09 ± 0.62	80.00 ± 0.45	65.81 ± 0.74	81.75 ± 0.53	72.8 ± 0.7	85.0 ± 0.5	47.2 ± 0.6	62.5 ± 0.6
Activation to Parameter (train + val) [39]	WRN-28-10	59.60 ± 0.41	73.74 ± 0.19						
LEO (train + val) [40]	WRN-28-10	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09				
Support-based initialization (train + val)	WRN-28-10	58.47 ± 0.66	75.56 ± 0.52	67.34 ± 0.69†	83.32 ± 0.51†	72.14 ± 0.69	85.21 ± 0.49	45.08 ± 0.61	60.05 ± 0.60
Fine-tuning (train + val)	WRN-28-10	59.62 ± 0.66	**79.93 ± 0.47**	66.23 ± 0.68	**86.08 ± 0.47**	70.07 ± 0.67	**87.26 ± 0.45**	43.80 ± 0.58	64.40 ± 0.58
Transductive fine-tuning (train + val)	WRN-28-10	**68.11 ± 0.69**	**80.36 ± 0.50**	**72.87 ± 0.71**	**86.15 ± 0.50**	**78.36 ± 0.70**	**87.54 ± 0.49**	**50.44 ± 0.68**	**65.74 ± 0.60**

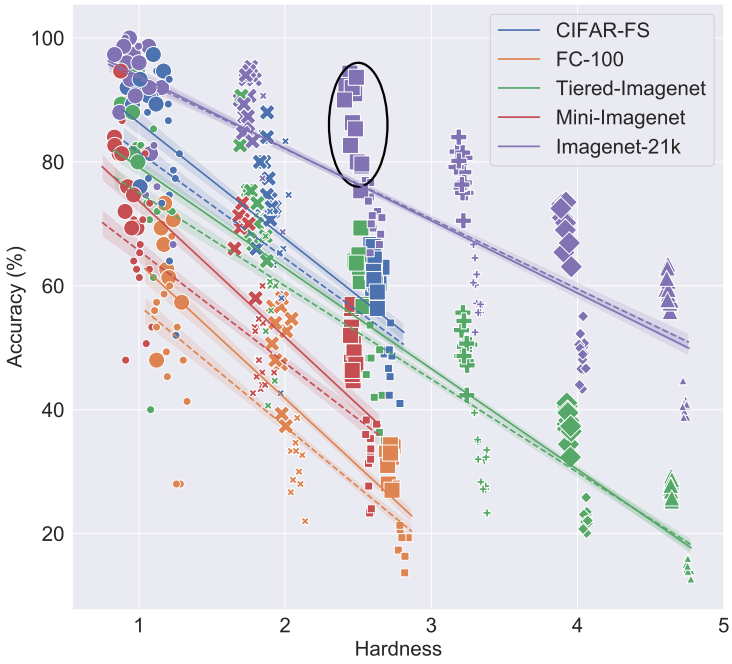

Figure 6: **Comparing accuracy of transductive fine-tuning (solid lines) vs. support-based initialization (dotted lines) for different datasets, ways (5,10,20,40,80 and 160) and support shots (1 and 5)**. E.g., the ellipse contains accuracies of different 5-shot 10-way episodes for Imagenet-21k. Abscissae are computed using (5) and a Resnet-152 [52] network trained for standard image classification on the Imagenet-1k dataset. Markers indicate the accuracy of transductive fine-tuning of a few-shot episodes; markers for support-based initialization are hidden to avoid clutter. Shape of the markers denotes different ways; ways increase from left to right (5,10,20,40,80 and 160). Size of the markers denotes the support shot (1 and 5); it increases from the bottom to the top. Regression lines are drawn for each algorithm and dataset by combining the episodes of all few-shot protocols. This plot is akin to a precision-recall curve and allows comparing two algorithms, or models, for different test scenarios. The area in the first quadrant under the fitted regression lines is 295 vs. 284 (CIFAR-FS), 167 vs. 149 (FC-100), 208 vs. 194 (Mini-ImageNet), 280 vs. 270 (Tiered-ImageNet) and 475 vs. 484 (Imagenet-21k) for transductive fine-tuning and support-based initialization.

The **high standard deviation of accuracy** of few-shot learning algorithms (cf. Section 1) can be seen as the spread of the cluster corresponding to each few-shot protocol, e.g., the ellipse denotes 5-shot 10-way protocol for Imagenet-21k. It is the nature of few-shot learning that episodes have very different hardness even if the way and shot are fixed. Episodes within the ellipse lie on a (different) line which indicates that hardness is a good indicator of accuracy.

Fig. 6 also shows that due to fewer test classes, CIFAR-FS, FC-100 and Mini-ImageNet have less diversity in the hardness of episodes while Tiered-ImageNet and Imagenet-21k allow sampling of both very hard and very easy diverse episodes. For a given few-shot protocol, the hardness of episodes in the former three is almost the same as that of the latter two datasets. This indicates that **CIFAR-FS, FC-100 and Mini-ImageNet may be good benchmarks for applications with few classes.**

The hardness metric in (5) naturally builds upon existing ideas in metric-based few-shot learning, namely [15]. We propose it as a means to evaluate few-shot learning algorithms uniformly across different few-shot protocols for different datasets; ascertaining its efficacy and comparisons to other metrics will be part of future work.

References

[1] Paul E Utgoff. Shift of bias for inductive concept learning. *Machine learning: An artificial intelligence approach*, 2:107–148, 1986.

[2] Jurgen Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987.

[3] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[4] Jonathan Baxter. *Learning internal representations*. Flinders University of S. Aust., 1995.

[5] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pages 6–8. Univ. of Texas, 1992.

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

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

[8] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, 2015.

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

[10] Luca Bertinetto, João F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv:1805.08136*, 2018.

[11] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. *arXiv:1904.03758*, 2019.

[12] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[13] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

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

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

[16] 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*, pages 719–729, 2018.

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

[18] Kelsey R Allen, Hanul Shin, Evan Shelhamer, and Josh B Tenenbaum. Variadic learning by bayesian nonparametric deep embedding. 2018.

[19] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training, 2019.

[20] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. 2018.

[21] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[22] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Icml*, volume 99, pages 200–209, 1999.

[23] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

[24] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv:1803.02999*, 2018.

[25] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. 2018.

[26] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[27] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. *arXiv:1507.00677*, 2015.

[28] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171, 2016.

[29] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in neural information processing systems*, pages 6510–6520, 2017.

[30] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907*, 2016.

[31] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv:1711.04043*, 2017.

[32] 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:1803.00676*, 2018.

[33] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Deep transfer metric learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 325–333, 2015.

[34] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5822–5830, 2018.

[35] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017.

[36] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[37] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[39] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018.

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

[41] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv:1605.07146*, 2016.

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[43] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv:1710.09412*, 2017.

[44] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

[45] Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv:1807.11205*, 2018.

[46] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[47] Jeremy Howard et al. fastai. `https://github.com/fastai/fastai`, 2018.

[48] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv:1710.03740*, 2017.

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

[50] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[51] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, and Yi Yang. Transductive propagation network for few-shot learning. *arXiv:1805.10002*, 2018.

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *arXiv:1603.05027*, 2016.