Rethinking Few-shot Image Classification: A Good Embedding is All You Need?

Yonglong Tian[1]*, Yue Wang[1]*, Dilip Krishnan[2], Joshua B. Tenenbaum[1], and Phillip Isola[1]

[1] MIT
[2] Google Research

Abstract. The focus of recent meta-learning research has been on the development of learning algorithms that can quickly adapt to test time tasks with limited data and low computational cost. Few-shot learning is widely used as one of the standard benchmarks in meta-learning. In this work, we show that a simple baseline: learning a supervised or self-supervised representation on the meta-training set, followed by training a linear classifier on top of this representation, outperforms state-of-the-art few-shot learning methods. An additional boost can be achieved through the use of self-distillation. This demonstrates that using a good learned embedding model can be more effective than sophisticated meta-learning algorithms. We believe that our findings motivate a rethinking of few-shot image classification benchmarks and the associated role of meta-learning algorithms. Code: `http://github.com/WangYueFt/rfs/`.

1 Introduction

Few-shot learning measures a model's ability to quickly adapt to new environments and tasks. This is a challenging problem because only limited data is available to adapt the model. Recently, significant advances [54, 51, 49, 11, 43, 45, 52, 33, 41, 58, 26, 28] have been made to tackle this problem using the ideas of meta-learning or "learning to learn".

Meta-learning defines a family of tasks, divided into disjoint meta-training and meta-testing sets. Each task consists of limited training data, which requires fast adaptability [42] of the learner (e.g., the deep network that is fine-tuned). During meta-training/testing, the learner is trained and evaluated on a task sampled from the task distribution. The performance of the learner is evaluated by the average test accuracy across many meta-testing tasks. Methods to tackle this problem can be cast into two main categories: optimization-based methods and metric-based methods. Optimization-based methods focus on designing algorithms that can quickly adapt to each task; while metric-based methods aim to find good metrics (usually kernel functions) to side-step the need for inner-loop optimization for each task.

*: Equal contribution. Correspondence to: {yonglong,yuewangx}@mit.edu

Meta-learning is evaluated on a number of domains such as few-shot classification and meta-reinforcement learning. Focusing on few-shot classification tasks, a question that has been raised in recent work is whether it is the meta-learning algorithm or the learned representation that is responsible for the fast adaption to test time tasks. [37] suggested that feature reuse is main factor for fast adaptation. Recently, [9] proposed transductive fine-tuning as a strong baseline for few-shot classification; and even in a regular, inductive, few-shot setup, they showed that fine-tuning is only slightly worse than state-of-the-art algorithms. In this setting, they fine-tuned the network on the meta-testing set and *used* information from the testing data. Besides, [5] shows an improved fine-tuning model performs slightly worse than meta-learning algorithms.

In this paper, we propose an extremely simple baseline that suggests that good learned representations are more powerful for few-shot classification tasks than the current crop of complicated meta-learning algorithms. Our baseline consists of a *linear* model learned on top of a pre-trained embedding. Surprisingly, we find this outperforms *all other meta-learning algorithms* on few-shot classification tasks, often by large margins. The differences between our approach and that of [9] are: we *do not* utilize information from testing data (since we believe that inductive learning is more generally applicable to few-shot learning); and we use a fixed neural network for feature extraction, rather than fine-tuning it on the meta-testing set. The concurrent works [6, 21] are inline with ours.

Our model learns representations by training a neural network on the entire meta-training set: we merge all meta-training data into a single task and a neural network is asked to perform either ordinary classification or self-supervised learning, on this combined dataset. The classification task is equivalent to the pre-training phase of TADAM [33] and LEO [41]. After training, we keep the pre-trained network up to the penultimate layer and use it as a feature extractor. During meta-testing, for each task, we fit a linear classifier on the features extracted by the pre-trained network. In contrast to [9] and [37], we *do not* fine-tune the neural network. Furthermore, we show that self-distillation on this baseline provides an additional boost.

Contributions. Our key contributions are:

- A surprisingly simple baseline for few-shot learning, which achieves the state-of-the-art. This baseline suggests that many recent meta-learning algorithms are *no better* than simply learning a good representation through a proxy task, e.g., image classification.
- Building upon the simple baseline, we use self-distillation to further improve performance. Our combined method achieves an average of 3% improvement over the previous state-of-the-art on widely used benchmarks. On the new benchmark Meta-Dataset [50], our method outperforms previous best results by more than 7% on average.
- Beyond supervised training, we show that representations learned with state-of-the-art self-supervised methods achieve similar performance as fully supervised methods.

2 Related works

Metric-based meta-learning. The core idea in metric-based meta-learning is related to nearest neighbor algorithms and kernel density estimation. Metric-based methods embed input data into fixed dimensional vectors and use them to design proper kernel functions. The predicted label of a query is the weighted sum of labels over support samples. Metric-based meta-learning aims to learn a task-dependent metric. [23] used Siamese network to encode image pairs and predict confidence scores for each pair. Matching Networks [51] employed two networks for query samples and support samples respectively and used an LSTM with read-attention to encode a full context embedding of support samples. Prototypical Networks [43] learned to encode query samples and support samples into a shared embedding space; the metric used to classify query samples is the distance to prototype representations of each class. Instead of using distances of embeddings, Relation Networks [45] leveraged relational module to represent an appropriate metric. TADAM [33] proposed metric scaling and metric task conditioning to boost the performance of Prototypical Networks.

Optimization-based meta-learning. Deep learning models are neither designed to train with very few examples nor to converge very fast. To fix that, optimization-based methods intend to learn with a few examples. Meta-learner [38] exploited an LSTM to satisfy two main desiderata of few-shot learning: quick acquisition of task-dependent knowledge and slow extraction of transferable knowledge. MAML [11] proposed a general optimization algorithm; it aims to find a set of model parameters, such that a small number of gradient steps with a small amount of training data from a new task will produce large improvements on that task. In that paper, first-order MAML was also proposed, which ignored the second-order derivatives of MAML. It achieved comparable results to complete MAML with orders of magnitude speedup. To further simplify MAML, Reptile [32] removed re-initialization for each task, making it a more natural choice in certain settings. LEO [41] proposed that it is beneficial to decouple the optimization-based meta-learning algorithms from high-dimensional model parameters. In particular, it learned a stochastic latent space from which the high-dimensional parameters can be generated. MetaOptNet [26] replaced the linear predictor with an SVM in the MAML framework; it incorporated a differentiable quadratic programming (QP) solver to allow end-to-end learning. For a complete list of recent works on meta-learning, we refer readers to [55].

Towards understanding MAML. To understand why MAML works in the first place, many efforts have been made either through an optimization perspective or a generalization perspective. Reptile [32] showed a variant of MAML works even without re-initialization for each task, because it tends to converge towards a solution that is close to each task's manifold of optimal solutions. In [37], the authors analyzed whether the effectiveness of MAML is due to rapid learning of each task or reusing the high quality features. It concluded that feature reuse is the dominant component in MAML's efficacy, which is reaffirmed by experiments conducted in this paper.

Meta-learning datasets. Over the past several years, many datasets have been proposed to test meta-learning or few-shot learning algorithms. Omniglot [24] was one of the earliest few-shot learning datasets; it contains thousands of handwritten characters from the world's alphabets, intended for one-shot "visual Turing test". In [25], the authors reported the 3-year progress for the Omniglot challenge, concluding that human-level one-shot learnability is still hard for current meta-learning algorithms. [51] introduced mini-ImageNet, which is a subset of ImageNet [8]. In [40], a large portion of ImageNet was used for few-shot learning tests. Meta-dataset [50] summarized recent datasets and tested several representative methods in a uniform fashion.

Knowledge distillation. The idea of knowledge distillation (KD) dates back to [4]. The original idea was to compress the knowledge contained in an ensemble of models into a single smaller model. In [19], the authors generalized this idea and brought it into the deep learning framework. In KD, knowledge is transferred from the teacher model to the student model by minimizing a loss in which the target is the distribution of class probabilities induced by the teacher model. It was shown that KD has several benefits for optimization and knowledge transfer between tasks [59, 13, 14]. BAN [12] introduced sequential distillation, which also improved the performance of teacher models. In natural language processing (NLP), BAM [7] used BAN to distill from single-task models to a multi-task model, helping the multi-task model surpass its single-task teachers. Another two related works are [30] which provides theoretical analysis of self-distillation and CRD [47] which shows distillation improves the transferability across datasets.

3 Method

We establish preliminaries about the meta-learning problem and related algorithms in §3.1; then we present our baseline in §3.2; finally, we introduce how knowledge distillation helps few-shot learning in §3.3. For ease of comparison to previous work, we use the same notation as [26].

3.1 Problem formulation

The collection of meta-training tasks is defined as $\mathcal{T} = \{(\mathcal{D}_i^{train}, \mathcal{D}_i^{test})\}_{i=1}^{I}$, termed as meta-training set. The tuple $(\mathcal{D}_i^{train}, \mathcal{D}_i^{test})$ describes a training and a testing dataset of a task, where each dataset contains a small number of examples. Training examples $\mathcal{D}^{train} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$ and testing examples $\mathcal{D}^{test} = \{(\mathbf{x}_q, y_q)\}_{q=1}^{Q}$ are sampled from the same distribution.

A base learner $\mathcal{A}$, which is given by $y_* = f_\theta(\mathbf{x}_*)$ ($*$ denotes t or q), is trained on $\mathcal{D}^{train}$ and used as a predictor on $\mathcal{D}^{test}$. Due to the high dimensionality of $\mathbf{x}_*$, the base learner $\mathcal{A}$ suffers high variance. So training examples and testing examples are mapped into a feature space by an embedding model $\boldsymbol{\Phi}_* = f_\phi(\mathbf{x}_*)$. Assume the embedding model is fixed during training the base learner on each

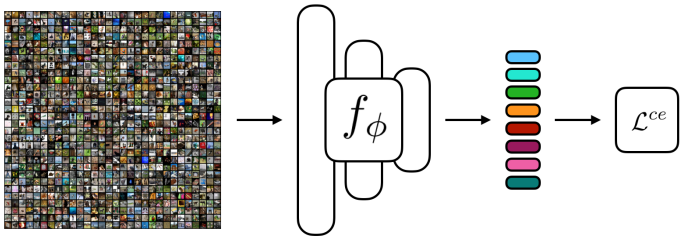

Fig. 1. In meta-training, we train on an image classification task on the merged meta-training data to learn an embedding model. This model is then re-used at meta-testing time to extract embedding for a simple linear classifier.

task, then the objective of the base learner is

$$\begin{aligned}\theta &= \mathcal{A}(\mathcal{D}^{train}; \phi) \\ &= \arg\min_{\theta} \mathcal{L}^{base}(\mathcal{D}^{train}; \theta, \phi) + \mathcal{R}(\theta),\end{aligned} \tag{1}$$

where $\mathcal{L}$ is the loss function and $\mathcal{R}$ is the regularization term.

The objective of the meta-learning algorithms is to learn a good embedding model, so that the average test error of the base learner on a distribution of tasks is minimized. Formally,

$$\phi = \arg\min_{\phi} \mathbb{E}_{\mathcal{T}}[\mathcal{L}^{meta}(\mathcal{D}^{test}; \theta, \phi)], \tag{2}$$

where $\theta = \mathcal{A}(\mathcal{D}^{train}; \phi)$.

Once meta-training is finished, the performance of the model is evaluated on a set of held-out tasks $\mathcal{S} = \{(\mathcal{D}_j^{train}, \mathcal{D}_j^{test})\}_{j=1}^{J}$, called meta-testing set. The evaluation is done over the distribution of the test tasks:

$$\mathbb{E}_{\mathcal{S}}[\mathcal{L}^{meta}(\mathcal{D}^{test}; \theta, \phi), \text{where } \theta = \mathcal{A}(\mathcal{D}^{train}; \phi)]. \tag{3}$$

3.2 Learning embedding model through classification

As we show in §3.1, the goal of meta-training is to learn a transferrable embedding model f_ϕ, which generalizes to any new task. Rather than designing new meta-learning algorithms to learn the embedding model, we propose that a model pre-trained on a classification task can generate powerful embeddings for the downstream base learner. To that end, we merge tasks from meta-training set into a single task, which is given by

$$\begin{aligned}\mathcal{D}^{new} &= \{(\mathbf{x}_i, y_i)\}_{k=1}^{K} \\ &= \cup\{\mathcal{D}_1^{train}, \ldots, \mathcal{D}_i^{train}, \ldots, \mathcal{D}_I^{train}\},\end{aligned} \tag{4}$$

where $\mathcal{D}_i^{train}$ is the task from $\mathcal{T}$. The embedding model is then

$$\phi = \arg\min_{\phi} \mathcal{L}^{ce}(\mathcal{D}^{new}; \phi), \tag{5}$$

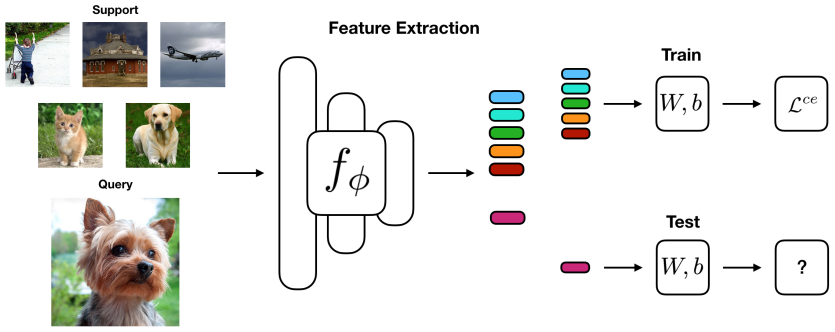

Fig. 2. We show a meta-testing case for 5-way 1-shot task: 5 support images and 1 query image are transformed into embeddings using the fixed neural network; a linear model (logistic regression (LR) in this case) is trained on 5 support embeddings; the query image is tested using the linear model.

and $\mathcal{L}^{ce}$ denotes the cross-entropy loss between predictions and ground-truth labels. We visualize the task in Figure 1.

As shown in Figure 2, for a task $(\mathcal{D}_j^{train}, \mathcal{D}_j^{test})$ sampled from meta-testing distribution, we train a base learner on $\mathcal{D}_j^{train}$. The base learner is instantiated as multivariate logistic regression. Its parameters $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$ include a weight term $\boldsymbol{W}$ and a bias term $\boldsymbol{b}$, given by

$$\theta = \arg\min_{\{\boldsymbol{W},\boldsymbol{b}\}} \sum_{t=1}^{T} \mathcal{L}_t^{ce}(\boldsymbol{W} f_\phi(\mathbf{x}_t) + \boldsymbol{b}, y_t) + \mathcal{R}(\boldsymbol{W}, \boldsymbol{b}). \tag{6}$$

We also evaluate other base learners such as nearest neighbor classifier with $\mathcal{L}$-2 distance and/or cosine distance in §4.8.

In our method, the crucial difference between meta-training and meta-testing is the embedding model parameterized by ϕ is carried over from meta-training to meta-testing and kept unchanged when evaluated on tasks sampled from meta-testing set. The base learner is re-initialized for every task and trained on $\mathcal{D}^{train}$ of meta-testing task. Our method is the same with the pre-training phase of methods used in [41, 33]. Unlike other methods [9, 37], we *do not* fine-tune the embedding model f_ϕ during the meta-testing stage.

3.3 Sequential self-distillation

Knowledge distillation [19] is an approach to transfer knowledge embedded in an ensemble of models to a single model, or from a larger teacher model to a smaller student model. Instead of using the embedding model directly for meta-testing, we distill the knowledge from the embedding model into a new model with an identical architecture, training on the same merged meta-training set. The new embedding model parameterized by ϕ' is trained to minimize a weighted sum of

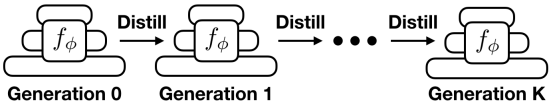

Fig. 3. Sequential self-distillation: a vanilla model, termed as *Generation 0*, is trained with standard cross-entropy loss; then, the k-th generation is learned with knowledge distilled from the (k-1)-th generation.

the cross-entropy loss between the predictions and ground-truth labels and the Kullback–Leibler divergence (KL) between predictions and soft targets:

$$\begin{aligned}\phi' = \arg\min_{\phi'}(&\alpha\mathcal{L}^{ce}(\mathcal{D}^{new};\phi')+\\ &\beta KL(f(\mathcal{D}^{new};\phi'),f(\mathcal{D}^{new};\phi))),\end{aligned} \tag{7}$$

where usually $\beta = 1-\alpha$.

We exploit the Born-again [12] strategy to apply KD sequentially to generate multiple generations, which is shown in Figure 3. At each step, the embedding model of k-th generation is trained with knowledge transferred from the embedding model of (k-1)-th generation:

$$\begin{aligned}\phi_k = \arg\min_{\phi}(&\alpha\mathcal{L}^{ce}(\mathcal{D}^{new};\phi)+\\ &\beta KL(f(\mathcal{D}^{new};\phi),f(\mathcal{D}^{new};\phi_{k-1}))).\end{aligned} \tag{8}$$

We repeat the operation K times, we use ϕ_K as the embedding model to extract features for meta-testing. We analyze the sequential self-distillation in §4.7.

4 Experiments

We conduct experiments on five widely used few-shot image recognition benchmarks: miniImageNet [51], tieredImageNet [40], CIFAR-FS [3], and FC100 [33], and Meta-Dataset [50].

4.1 Setup

Architecture. Following previous works [29, 33, 26, 39, 9], we use a ResNet12 as our backbone: the network consists of 4 residual blocks, where each has 3 convolutional layers with 3×3 kernel; a 2×2 max-pooling layer is applied after each of the first 3 blocks; and a global average-pooling layer is on top of the fourth block to generate the feature embedding. Similar to [26], we use Dropblock as a regularizer and change the number of filters from (64,128,256,512) to (64,160,320,640). As a result, our ResNet12 is identical to that used in [39, 26].

Optimization setup. We use SGD optimizer with a momentum of 0.9 and a weight decay of $5e^{-4}$. Each batch consists of 64 samples. The learning rate is

		miniImageNet 5-way		tieredImageNet 5-way	
model	**backbone**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
MAML [11]	32-32-32-32	48.70 ± 1.84	63.11 ± 0.92	51.67 ± 1.81	70.30 ± 1.75
Matching Networks [51]	64-64-64-64	43.56 ± 0.84	55.31 ± 0.73	-	-
IMP [2]	64-64-64-64	49.2 ± 0.7	64.7 ± 0.7	-	-
Prototypical Networks† [43]	64-64-64-64	49.42 ± 0.78	68.20 ± 0.66	53.31 ± 0.89	72.69 ± 0.74
TAML [22]	64-64-64-64	51.77 ± 1.86	66.05 ± 0.85	-	-
SAML [16]	64-64-64-64	52.22 ± n/a	66.49 ± n/a	-	-
GCR [27]	64-64-64-64	53.21 ± 0.80	72.34 ± 0.64	-	-
KTN(Visual) [34]	64-64-64-64	54.61 ± 0.80	71.21 ± 0.66	-	-
PARN[57]	64-64-64-64	55.22 ± 0.84	71.55 ± 0.66	-	-
Dynamic Few-shot [15]	64-64-128-128	56.20 ± 0.86	73.00 ± 0.64	-	-
Relation Networks [45]	64-96-128-256	50.44 ± 0.82	65.32 ± 0.70	54.48 ± 0.93	71.32 ± 0.78
R2D2 [3]	96-192-384-512	51.2 ± 0.6	68.8 ± 0.1	-	-
SNAIL [29]	ResNet-12	55.71 ± 0.99	68.88 ± 0.92	-	-
AdaResNet [31]	ResNet-12	56.88 ± 0.62	71.94 ± 0.57	-	-
TADAM [33]	ResNet-12	58.50 ± 0.30	76.70 ± 0.30	-	-
Shot-Free [39]	ResNet-12	59.04 ± n/a	77.64 ± n/a	63.52 ± n/a	82.59 ± n/a
TEWAM [35]	ResNet-12	60.07 ± n/a	75.90 ± n/a	-	-
MTL [44]	ResNet-12	61.20 ± 1.80	75.50 ± 0.80	-	-
Variational FSL [60]	ResNet-12	61.23 ± 0.26	77.69 ± 0.17	-	-
MetaOptNet [26]	ResNet-12	62.64 ± 0.61	78.63 ± 0.46	65.99 ± 0.72	81.56 ± 0.53
Diversity w/ Cooperation [10]	ResNet-18	59.48 ± 0.65	75.62 ± 0.48	-	-
Fine-tuning [9]	WRN-28-10	57.73 ± 0.62	78.17 ± 0.49	66.58 ± 0.70	85.55 ± 0.48
LEO-trainval† [41]	WRN-28-10	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09
Ours-simple	ResNet-12	62.02 ± 0.63	79.64 ± 0.44	69.74 ± 0.72	84.41 ± 0.55
Ours-distill	ResNet-12	**64.82 ± 0.60**	**82.14 ± 0.43**	**71.52 ± 0.69**	**86.03 ± 0.49**

Table 1. Comparison to prior work on miniImageNet and tieredImageNet. Average few-shot classification accuracies (%) with 95% confidence intervals on miniImageNet and tieredImageNet meta-test splits. Results reported with input image size of 84x84. a-b-c-d denotes a 4-layer convolutional network with a, b, c, and d filters in each layer. † results obtained by training on the union of training and validation sets.

initialized as 0.05 and decayed with a factor of 0.1 by three times for all datasets, except for miniImageNet where we only decay twice as the third decay has no effect. We train 100 epochs for miniImageNet, 60 epochs for tieredImageNet, and 90 epochs for both CIFAR-FS and FC100. During distillation, we use the same learning schedule and set $\alpha = \beta = 0.5$.

Data augmentation. When training the embedding network on transformed meta-training set, we adopt random crop, color jittering, and random horizontal flip as in [26]. For meta-testing stage, we train an N-way logistic regression base classifier. We use the implementations in scikit-learn [1].

4.2 Results on ImageNet derivatives

The miniImageNet dataset [51] is a standard benchmark for few-shot learning algorithms for recent works. It consists of 100 classes randomly sampled from the ImageNet; each class contains 600 downsampled images of size 84x84. We follow the widely-used splitting protocol proposed in [38], which uses 64 classes for meta-training, 16 classes for meta-validation, and 20 classes for meta-testing.

		CIFAR-FS 5-way		FC100 5-way	
model	**backbone**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
MAML [11]	32-32-32-32	58.9 ± 1.9	71.5 ± 1.0	-	-
Prototypical Networks [43]	64-64-64-64	55.5 ± 0.7	72.0 ± 0.6	35.3 ± 0.6	48.6 ± 0.6
Relation Networks [45]	64-96-128-256	55.0 ± 1.0	69.3 ± 0.8	-	-
R2D2 [3]	96-192-384-512	65.3 ± 0.2	79.4 ± 0.1	-	-
TADAM [33]	ResNet-12	-	-	40.1 ± 0.4	56.1 ± 0.4
Shot-Free [39]	ResNet-12	69.2 ± n/a	84.7 ± n/a	-	-
TEWAM [35]	ResNet-12	70.4 ± n/a	81.3 ± n/a	-	-
Prototypical Networks [43]	ResNet-12	72.2 ± 0.7	83.5 ± 0.5	37.5 ± 0.6	52.5 ± 0.6
MetaOptNet [26]	ResNet-12	72.6 ± 0.7	84.3 ± 0.5	41.1 ± 0.6	55.5 ± 0.6
Ours-simple	ResNet-12	71.5 ± 0.8	86.0 ± 0.5	42.6 ± 0.7	59.1 ± 0.6
Ours-distill	ResNet-12	**73.9 ± 0.8**	**86.9 ± 0.5**	**44.6 ± 0.7**	**60.9 ± 0.6**

Table 2. Comparison to prior work on CIFAR-FS and FC100. Average few-shot classification accuracies (%) with 95% confidence intervals on CIFAR-FS and FC100. a-b-c-d denotes a 4-layer convolutional network with a, b, c, and d filters in each layer.

The tieredImageNet dataset [40] is another subset of ImageNet but has more classes (608 classes). These classes are first grouped into 34 higher-level categories, which are further divided into 20 training categories (351 classes), 6 validation categories (97 classes), and 8 testing categories (160 classes). Such construction ensures the training set is distinctive enough from the testing set and makes the problem more challenging.

Results. During meta-testing, we evaluate our method with 3 runs, where in each run the accuracy is the mean accuracy of 1000 randomly sampled tasks. We report the median of 3 runs in Table 1. Our simple baseline with ResNet-12 is already comparable with the state-of-the-art MetaOptNet [26] on miniImageNet, and outperforms all previous works by at least 3% on tieredImageNet. The network trained with distillation further improves by 2-3%.

We notice that previous works [36, 41, 33, 44] have also leveraged the standard cross-entropy pre-training on the meta-training set. In [33, 41], a wide ResNet (WRN-28-10) is trained to classify all classes in the meta-training set (or combined meta-training and meta-validation set), and then frozen during the meta-training stage. [9] also conducts pre-training but the model is fine-tuned using the support images in meta-testing set, achieving 57.73 ± 0.62. We adopt the same architecture and gets 61.1 ± 0.86. So fine-tuning on small set of samples makes the performance worse. Another work [33] adopts a multi-task setting by jointly training on the standard classification task and few-shot classification (5-way) task. In another work [44], the ResNet-12 is pre-trained before mining hard tasks for the meta-training stage.

4.3 Results on CIFAR derivatives

The CIFAR-FS dataset [3] is a derivative of the original CIFAR-100 dataset by randomly splitting 100 classes into 64, 16 and 20 classes for training, validation, and testing, respectively. The FC100 dataset [33] is also derived from CIFAR-

Trained on ILSVRC train split					
Dataset	Best from [50]	LR (ours)	SVM (ours)	LR-distill (ours)	SVM-distill (ours)
ILSVRC	50.50	60.14	56.48	**61.48**	58.33
Omniglot	63.37	64.92	65.90	64.31	**66.77**
Aircraft	**68.69**	63.12	61.43	62.32	64.23
Birds	68.66	77.69	74.61	**79.47**	76.63
Textures	69.05	78.59	74.25	**79.28**	76.66
Quick Draw	51.52	**62.48**	59.34	60.83	59.02
Fungi	39.96	47.12	41.76	**48.53**	44.51
VGG Flower	87.15	**91.60**	90.32	91.00	89.66
Traffic Signs	66.79	77.51	**78.94**	76.33	78.64
MSCOCO	43.74	57.00	50.81	**59.28**	54.10
Mean Accuracy	60.94	68.02	65.38	**68.28**	66.86

Table 3. Results on Meta-Dataset. Average accuracy (%) is reported with variable number of ways and shots, following the setup in [50]. We compare four variants of out method (LR, SVM, LR-distill, and SVM-distill) to the best accuracy over 7 methods in [50]. In each episode, 1000 tasks are sampled for evaluation.

100 dataset in a similar way to tieredImagNnet. This results in 60 classes for training, 20 classes for validation, and 20 classes for testing.

Results. Similar to previous experiments, we evaluate our method with 3 runs, where in each run the accuracy is the mean accuracy of 3000 randomly sampled tasks. Table 2 summarizes the results, which shows that our simple baseline is comparable to Prototypical Networks [43] and MetaOptNet [26] on CIFAR-FS dataset, and outperforms both of them on FC100 dataset. Our distillation version achieves the new state-of-the-art on both datasets. This verifies our hypothesis that a good embedding plays an important role in few-shot recognition.

4.4 Results on Meta-Dataset

Meta-Dataset [50] is a new benchmark for evaluating few-shot methods in large-scale settings. Compared to miniImageNet and tieredImageNet, Meta-Dataset provides more diverse and realistic samples.

Setup. The ILSVRC (ImageNet) subset consists of 712 classes for training, 158 classes for validation, and 130 classes for testing. We follow the setting in Meta-Dateset [50] where the embedding model is trained solely on the ILSVRC training split. We use ResNet-18 [18] as the backbone network. The input size is 128×128. In the pre-training stage, we use SGD optimizer with a momentum of 0.9. The learning rate is initially 0.1 and decayed by a factor of 10 for every 30 epochs. We train the model for 90 epochs in total. The batch size is 256. We use standard data augmentation, including randomly resized crop and horizontal flip. In the distillation stage, we set $\alpha = 0.5$ and $\beta = 1.0$. We perform distillation twice and use the model from the second generation for meta-testing. We do not use test-time augmentation in meta-testing. In addition to logistic regression (LR), we also provide results of linear SVM for completeness.

		miniImageNet 5-way	
model	**backbone**	**1-shot**	**5-shot**
Supervised	ResNet50	**57.56 ± 0.79**	73.81 ± 0.63
MoCo [17]	ResNet50	54.19 ± 0.93	73.04 ± 0.61
CMC [46]	ResNet50*	56.10 ± 0.89	**73.87 ± 0.65**

Table 4. Comparsions of embeddings from supervised pre-training and self-supervised pre-training (Moco and CMC). * the encoder of each view is 0.5× width of a normal ResNet-50.

					miniImageNet		tieredImageNet		CIFAR-FS		FC100	
NN	**LR**	**$\mathcal{L}$-2**	**Aug**	**Distill**	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
✓					56.29	69.96	64.80	78.75	64.36	78.00	38.40	49.12
	✓				58.74	78.31	67.62	84.77	66.92	84.78	40.36	57.23
	✓	✓			61.56	79.27	69.53	85.08	71.24	85.63	42.77	58.86
	✓	✓	✓		62.02	79.64	69.74	85.23	71.45	85.95	42.59	59.13
	✓	✓	✓	✓	64.82	82.14	71.52	86.03	73.89	86.93	44.57	60.91

Table 5. Ablation study on four benchmarks with ResNet-12 as backbone network. "NN" and "LR" stand for nearest neighbour classifier and logistic regression. "$\mathcal{L}$-2" means feature normalization after which feature embeddings are on the unit sphere. "Aug" indicates that each support image is augmented into 5 samples to train the classifier. "Distill" represents the use of knowledge distillation.

We select the best results from [50] for comparison – for each testing subset, we pick the best accuracy over 7 methods and 3 different architectures including 4-layer ConvNet, Wide ResNet, and ResNet-18. As shown in Table 3, our simple baselines clearly outperform the best results from [50] on 9 out of 10 testing datasets, often by a large margin. Our baseline method using LR outperforms previous best results by more than 7% on average. Also, self-distillation improves `max(LR, SVM)` in 7 out of the 10 testing subsets. Moreover, we notice empirically that logistic regression (LR) performs better than linear SVM.

4.5 Embeddings from self-supervised representation learning

Using unsupervised learning [56, 46, 17, 48] to improve the generalization of the meta-learning algorithms [53] removes the needs of data annotation. In addition to using embeddings from supervised pre-training, we also train a linear classifier on embeddings from self-supervised representation learning. Following MoCo [17] and CMC [46] (both are inspired by InstDis [56]), we train a ResNet50 [18] (without using labels) on the merged meta-training set to learn an embedding model. We compare unsupervised ResNet50 to a supervised ResNet50. From Table 4, we observe that using embeddings from self-supervised ResNet50 is only slightly worse than using embeddings from supervised ResNet50 (in 5-shot setting, the results are comparable). This observation shows the potential of self-supervised learning in the scenario of few-shot learning.

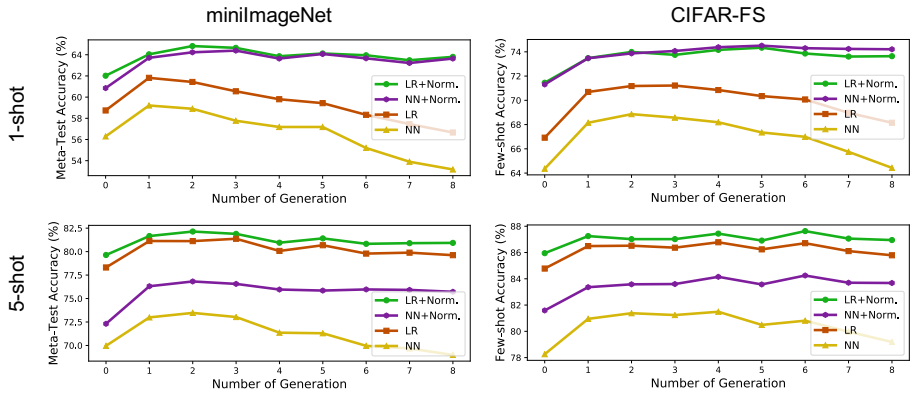

Fig. 4. Evaluation on different generations of distilled networks. The 0-th generation (or root generation) indicates the vanilla network trained with only standard cross-entropy loss. The k-th generation is trained by combining the standard classification loss and the knowledge distillation (KD) loss using the (k-1)-th generation as the teacher model. Logistic regression (LR) and nearest neighbours (NN) are evaluated.

4.6 Ablation experiments

In this section, we conduct ablation studies to analyze how each component affects the few-shot recognition performance. We study the following five components of our method: (a) we chose logistic regression as our base learner, and compare it to a nearest neighbour classifier with euclidean distance; (b) we find that normalizing the feature vectors onto the unit sphere, e.g., $\mathcal{L}$-2 normalization, could improve the classification of the downstream base classifier; (c) during meta-testing, we create 5 augmented samples from each support image to alleviate the data insufficiency problem, and using these augmented samples to train the linear classifier; (d) we distill the embedding network on the training set by following the sequential distillation [12] strategy.

Table 5 shows the results of our ablation studies on miniImageNet, tieredImageNet, CIFAR-FS, and FC100. In general, logistic regression significantly outperforms the nearest neighbour classifier, especially for the 5-shot case; $\mathcal{L}$-2 normalization consistently improves the 1-shot accuracy by 2% on all datasets; augmenting the support images leads to marginal improvement; even with all these techniques, distillation can still provide 2% extra gain.

4.7 Effects of distillation

We use sequential self-distillation to get an embedding model, similar to the one in Born-again networks [12]. We investigate the effect of this strategy on the performance of downstream few-shot classification.

In addition to logistic regression and nearest-neighbour classifiers, we also look into a cosine similarity classifier, which is equivalent to the nearest-neighbour

		miniImageNet 5-way		tieredImageNet 5-way	
model	**backbone**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
Ours	64-64-64-64	55.25 ± 0.58	71.56 ± 0.52	56.18 ± 0.70	72.99 ± 0.55
Ours-distill	64-64-64-64	55.88 ± 0.59	71.65 ± 0.51	56.76 ± 0.68	73.21 ± 0.54
Ours-trainval	64-64-64-64	56.32 ± 0.58	72.46 ± 0.52	56.53 ± 0.68	73.15 ± 0.58
Ours-distill-trainval	64-64-64-64	**56.64 ± 0.58**	**72.85 ± 0.50**	**57.35 ± 0.70**	**73.98 ± 0.56**
Ours	ResNet-12	62.02 ± 0.63	79.64 ± 0.44	69.74 ± 0.72	84.41 ± 0.55
Ours-distill	ResNet-12	64.82 ± 0.60	82.14 ± 0.43	71.52 ± 0.69	86.03 ± 0.49
Ours-trainval	ResNet-12	63.59 ± 0.61	80.86 ± 0.47	71.12 ± 0.68	85.94 ± 0.46
Ours-distill-trainval	ResNet-12	**66.58 ± 0.65**	**83.22 ± 0.39**	**72.98 ± 0.71**	**87.46 ± 0.44**
Ours	SEResNet-12	62.29 ± 0.60	79.94 ± 0.46	70.31 ± 0.70	85.22 ± 0.50
Ours-distill	SEResNet-12	65.96 ± 0.63	82.05 ± 0.46	71.72 ± 0.69	86.54 ± 0.49
Ours-trainval	SEResNet-12	64.07 ± 0.61	80.92 ± 0.43	71.76 ± 0.66	86.27 ± 0.45
Ours-distill-trainval	SEResNet-12	**67.73 ± 0.63**	**83.35 ± 0.41**	**72.55 ± 0.69**	**86.72 ± 0.49**

Table 6. Comparisons of different backbones on miniImageNet and tieredImageNet.

		CIFAR-FS 5-way		FC100 5-way	
model	**backbone**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
Ours	64-64-64-64	62.7 ± 0.8	78.7 ± 0.5	39.6 ± 0.6	53.5 ± 0.5
Ours-distill	64-64-64-64	63.8 ± 0.8	79.5 ± 0.5	40.3 ± 0.6	54.1 ± 0.5
Ours-trainval	64-64-64-64	63.5 ± 0.8	79.8 ± 0.5	43.2 ± 0.6	58.5 ± 0.5
Ours-distill-trainval	64-64-64-64	**64.9 ± 0.8**	**80.3 ± 0.5**	**44.6 ± 0.6**	**59.2 ± 0.5**
Ours	ResNet-12	71.5 ± 0.8	86.0 ± 0.5	42.6 ± 0.7	59.1 ± 0.6
Ours-distill	ResNet-12	73.9 ± 0.8	86.9 ± 0.5	44.6 ± 0.7	60.9 ± 0.6
Ours-trainval	ResNet-12	73.1 ± 0.8	86.7 ± 0.5	49.5 ± 0.7	66.4 ± 0.6
Ours-distill-trainval	ResNet-12	**75.4 ± 0.8**	**88.2 ± 0.5**	**51.6 ± 0.7**	**68.4 ± 0.6**
Ours	SEResNet-12	72.0 ± 0.8	86.0 ± 0.6	43.4 ± 0.6	59.1 ± 0.6
Ours-distill	SEResNet-12	74.2 ± 0.8	87.2 ± 0.5	44.9 ± 0.6	61.4 ± 0.6
Ours-trainval	SEResNet-12	73.3 ± 0.8	86.8 ± 0.5	49.9 ± 0.7	66.8 ± 0.6
Ours-distill-trainval	SEResNet-12	**75.6 ± 0.8**	**88.2 ± 0.5**	**52.0 ± 0.7**	**68.8 ± 0.6**

Table 7. Comparisons of different backbones on CIFAR-FS and FC100.

classifier but with normalized features (noted as "NN+Norm."). The plots of 1-shot and 5-shot results on miniImageNet and CIFAR-FS are shown in Figure 4. The 0-th generation (or root generation) refers to the vanilla model trained with only standard cross-entropy loss, and the (k-1)-th generation is distilled into k-th generation. In general, few-shot recognition performance keeps getting better in the first two or three generations. After certain number of generations, the accuracy starts decreasing for logistic regression and nearest neighbour. Normalizing the features can significantly alleviate this problem. In Table 1, Table 2, and Table 5, we evalute the model of the second generation on miniImageNet, CIFAR-FS and FC100 datasets; we use the first generation on tieredImageNet. Model selection is done on the validation set.

4.8 Choice of base classifier

One might argue in the 1-shot case, that a linear classifier should behavior similarly to a nearest-neighbour classifier. However in Table 5 and Figure 4, we find

that logistic regression is clearly better than nearest-neighbour. We argue that this is casued by the scale of the features. After we normalize the features by the $\mathcal{L}$-2 norm, logistic regression ("LR+Norm") performs similarly to the nearest neighbour classifier ("NN+Norm."), as shown in the first row of Figure 4. However, when increasing the size of the support set to 5, logistic regression is significantly better than nearest-neighbour even after feature normalization

4.9 Comparsions of different network backbones.

To further verify our assumption that the key success of few-shot learning algorithms is due to the quality of embeddings, we compare three alternatives in Table 6 and Table 7: a ConvNet with four four convolutional layers (64, 64, 64, 64); a ResNet12 as in Table 1; a ResNet12 with sequeeze-and-excitation [20] modules. For each model, we have four settings: training on meta-training set; training and distilling on meta-training set; training on meta-training set and meta-validation set; training and distilling on meta-training set and meta-validation set. The results consistently improve with more data and better networks. This is inline with our hypothesis: embeddings are the most critical factor to the performance of few-shot learning/meta learning algorithms; better embeddings will lead to better few-shot testing performance (even with a simple linear classier). In addition, our ConvNet model also outperforms other few-shot learning and/or meta learning models using the same network. This verifies that in both small model regime (ConvNet) and large model regime (ResNet), few-shot learning and meta learning algorithms are *no better* than learning a good embedding model.

4.10 Multi-task vs multi-way classification?

We are interested in understanding whether the efficacy of our simple baseline is due to multi-task or multi-way classification. We compare to training an embedding model through *multi-task* learning: a model with shared embedding network and different classification heads is constructed, where each head is only classifying the corresponding category; then we use the embedding model to extract features as we do with our baseline model. This achieves 58.53 ± 0.8 on mini-ImageNet 5-way 1-shot case, compared to our baseline model which is 62.02 ± 0.63. So we argue that the speciality of our setting, where the few-shot classification tasks are mutually exclusive and can be merged together into a single *multi-way* classification task, makes the simple model effective.

Acknowledgement

The authors thank Hugo Larochelle and Justin Solomon for helpful discussions and feedback on this manuscript. This research was supported in part by iFlytek. This material was also based in part upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-19-C-1001.

References

1. Machine learning in python. `https://scikit-learn.org/stable/`
2. Allen, K., Shelhamer, E., Shin, H., Tenenbaum, J.: Infinite mixture prototypes for few-shot learning. In: ICML (2019)
3. Bertinetto, L., Henriques, J.F., Torr, P.H., Vedaldi, A.: Meta-learning with differentiable closed-form solvers. arXiv preprint arXiv:1805.08136 (2018)
4. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: SIGKDD (2006)
5. Chen, W.Y., Liu, Y.C., Kira, Z., Wang, Y.C., Huang, J.B.: A closer look at few-shot classification. In: ICLR (2019)
6. Chen, Y., Wang, X., Liu, Z., Xu, H., Darrell, T.: A new meta-baseline for few-shot learning. ArXiv **abs/2003.04390** (2020)
7. Clark, K., Luong, M.T., Manning, C.D., Le, Q.V.: Bam! born-again multi-task networks for natural language understanding. In: ACL (2019)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
9. Dhillon, G.S., Chaudhari, P., Ravichandran, A., Soatto, S.: A baseline for few-shot image classification. In: ICLR (2020)
10. Dvornik, N., Schmid, C., Mairal, J.: Diversity with cooperation: Ensemble methods for few-shot classification. In: ICCV (2019)
11. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML (2017)
12. Furlanello, T., Lipton, Z.C., Tschannen, M., Itti, L., Anandkumar, A.: Born-again neural networks. In: ICML (2018)
13. Gan, C., Gong, B., Liu, K., Su, H., Guibas, L.J.: Geometry guided convolutional neural networks for self-supervised video representation learning. In: CVPR (2018)
14. Gan, C., Zhao, H., Chen, P., Cox, D., Torralba, A.: Self-supervised moving vehicle tracking with stereo sound. In: ICCV (2019)
15. Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: CVPR (2018)
16. Hao, F., He, F., Cheng, J., Wang, L., Cao, J., Tao, D.: Collect and select: Semantic alignment metric learning for few-shot learning. In: ICCV (2019)
17. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.B.: Momentum contrast for unsupervised visual representation learning. ArXiv **abs/1911.05722** (2019)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CVPR (2016)
19. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015)
20. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR (2018)
21. Huang, S., Tao, D.: All you need is a good representation: A multi-level and classifier-centric representation for few-shot learning. ArXiv **abs/1911.12476** (2019)
22. Jamal, M.A., Qi, G.J.: Task agnostic meta-learning for few-shot learning. In: CVPR (2019)
23. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop (2015)
24. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. Science (2015)

25. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: The omniglot challenge: a 3-year progress report. Current Opinion in Behavioral Sciences (2019)
26. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. In: CVPR (2019)
27. Li, A., Luo, T., Xiang, T., Huang, W., Wang, L.: Few-shot learning with global class representations. In: ICCV (2019)
28. Li, H., Eigen, D., Dodge, S., Zeiler, M., Wang, X.: Finding task-relevant features for few-shot learning by category traversal. In: CVPR (2019)
29. Mishra, N., Rohaninejad, M., Chen, X., Abbeel, P.: A simple neural attentive meta-learner. arXiv preprint arXiv:1707.03141 (2017)
30. Mobahi, H., Farajtabar, M., Bartlett, P.L.: Self-distillation amplifies regularization in hilbert space. arXiv preprint arXiv:2002.05715 (2020)
31. Munkhdalai, T., Yuan, X., Mehri, S., Trischler, A.: Rapid adaptation with conditionally shifted neurons. arXiv preprint arXiv:1712.09926 (2017)
32. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. ArXiv **abs/1803.02999** (2018)
33. Oreshkin, B., López, P.R., Lacoste, A.: Tadam: Task dependent adaptive metric for improved few-shot learning. In: NIPS (2018)
34. Peng, Z., Li, Z., Zhang, J., Li, Y., Qi, G.J., Tang, J.: Few-shot image recognition with knowledge transfer. In: ICCV (2019)
35. Qiao, L., Shi, Y., Li, J., Wang, Y., Huang, T., Tian, Y.: Transductive episodic-wise adaptive metric for few-shot learning. In: ICCV (2019)
36. Qiao, S., Liu, C., Shen, W., Yuille, A.L.: Few-shot image recognition by predicting parameters from activations. In: CVPR (2018)
37. Raghu, A., Raghu, M., Bengio, S., Vinyals, O.: Rapid learning or feature reuse? towards understanding the effectiveness of maml. arXiv preprint arXiv:1909.09157 (2019)
38. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: ICLR (2017)
39. Ravichandran, A., Bhotika, R., Soatto, S.: Few-shot learning with embedded class models and shot-free meta training. In: ICCV (2019)
40. Ren, M., Ravi, S., Triantafillou, E., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S.: Meta-learning for semi-supervised few-shot classification. In: ICLR (2018)
41. Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R.: Meta-learning with latent embedding optimization. In: ICLR (2019)
42. Scott, T., Ridgeway, K., Mozer, M.C.: Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning. In: NIPS (2018)
43. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: NIPS (2017)
44. Sun, Q., Liu, Y., Chua, T.S., Schiele, B.: Meta-transfer learning for few-shot learning. In: CVPR (2019)
45. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. In: CVPR (2018)
46. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. arXiv preprint arXiv:1906.05849 (2019)
47. Tian, Y., Krishnan, D., Isola, P.: Contrastive representation distillation. arXiv preprint arXiv:1910.10699 (2019)
48. Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., Isola, P.: What makes for good views for contrastive learning? arXiv preprint arXiv:2005.10243 (2020)

49. Triantafillou, E., Zemel, R.S., Urtasun, R.: Few-shot learning through an information retrieval lens. In: NIPS (2017)
50. Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P.A., et al.: Meta-dataset: A dataset of datasets for learning to learn from few examples. arXiv preprint arXiv:1903.03096 (2019)
51. Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., Wierstra, D.: Matching networks for one shot learning. In: NIPS (2016)
52. Wang, Y.X., Girshick, R.B., Hebert, M., Hariharan, B.: Low-shot learning from imaginary data. CVPR (2018)
53. Wang, Y.X., Hebert, M.: Learning from small sample sets by combining unsupervised meta-training with cnns. In: Advances in Neural Information Processing Systems 29 (2016)
54. Wang, Y., Hebert, M.: Learning to learn: Model regression networks for easy small sample learning. In: ECCV (2016)
55. Weng, L.: Meta-learning: Learning to learn fast. lilianweng.github.io/lil-log (2018), `http://lilianweng.github.io/lil-log/2018/11/29/meta-learning.html`
56. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: CVPR (2018)
57. Wu, Z., Li, Y., Guo, L., Jia, K.: Parn: Position-aware relation networks for few-shot learning. In: ICCV (2019)
58. Ye, H.J., Hu, H., Zhan, D.C., Sha, F.: Learning embedding adaptation for few-shot learning. CoRR **abs/1812.03664** (2018)
59. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: CVPR (2017)
60. Zhang, J., Zhao, C., Ni, B., Xu, M., Yang, X.: Variational few-shot learning. In: ICCV (2019)