PROGRESSIVE KNOWLEDGE DISTILLATION FOR GENERATIVE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

While modern generative models are able to synthesize high-fidelity, visually appealing images, successfully generating examples that are useful for recognition tasks remains an elusive goal. To this end, our key insight is that the examples should be synthesized to recover classifier decision boundaries that would be learned from a large amount of real examples. More concretely, we treat a classifier trained on synthetic examples as “student” and a classifier trained on real examples as “teacher”. By introducing knowledge distillation into a meta-learning framework, we encourage the generative model to produce examples in a way that enables the student classifier to mimic the behavior of the teacher. To mitigate the potential gap between student and teacher classifiers, we further propose to distill the knowledge in a progressive manner, either by gradually strengthening the teacher or weakening the student. We demonstrate the use of our model-agnostic distillation approach to deal with data scarcity, significantly improving few-shot learning performance on miniImageNet and ImageNet1K benchmarks.

1 INTRODUCTION

Over the past decade, generative image modeling has progressed remarkably with the emergence of deep learning techniques. Modern generative models, such as the variants of generative adversarial networks (GANs) (Goodfellow et al., 2014; Karras et al., 2018; Brock et al., 2019) and variational auto-encoders (VAEs) (Kingma & Welling, 2014; Razavi et al., 2019), are able to synthesize high-fidelity, visually appealing images, with successful applications ranging from super-resolution (Ledig et al., 2017) to artistic manipulation (Zhu et al., 2017). However, when it comes to their use in discriminative visual recognition tasks, these images are still far from satisfactory. The performance of the classifiers trained on synthetic images is substantially inferior to that of the classifiers trained on real images (Dai et al., 2017; Shmelkov et al., 2018).

In this paper, we make a step towards building generative models that are recognition task oriented, thus enabling synthesizing examples in a way that helps the classification algorithm learn better classifiers. This is of great promise to deal with data scarcity in real-world scenarios, such as addressing few-shot learning which aims to recognize novel categories from one, or only a few, annotated examples (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017). Instead of matching training data distribution or aiming for realism, our key insight is that the examples should be synthesized to recover or stabilize classifier decision boundaries that would be learned from real samples.

More precisely, let us consider a 2-way 2-shot classification problem in Figure 1. We aim to learn a good classifier (the purple boundary in Figure 1a) that distinguishes the two classes based on 2 training examples per class. Ideally, the hope is that the boundary in Figure 1a should be as close as possible to the classifier (the red boundary in Figure 1b) that would be learned from a large set of real samples (on the order of hundreds or thousands of). To this end, we synthesize additional examples for each class based on its available 2 examples, so that the resulting classifier (the red solid boundary in Figure 1c) produced by the synthesized examples together with the few real examples remains unchanged from the desired classifier (the red dashed boundary in Figure 1b or Figure 1c).

To minimize the discrepancy between the classifier trained on synthetic examples and the classifier trained on real examples, we leverage the idea of knowledge distillation proposed by Hinton et al. (2015). While knowledge distillation was developed for model compression, in which a lightweight “student” model is trained to mimic the behavior of a larger, high-capacity “teacher” model, here...
Figure 1: Knowledge distillation of generative models for few-shot learning. We aim to recognize two novel classes from 2 examples per class (Figure 1a). The desired classifier is the one that would be learned from abundant real examples (Figure 1b). To this end, we distill the knowledge of the desired large-sample (dashed) classifier into a generative model, and thus enable it to produce additional examples from the few real examples in a way that minimizes the discrepancy between the (solid) classifier trained on synthesized examples together with the few real examples and the large-sample (dashed) classifier (Figure 1c). Real examples are shown as squares, synthetic examples as triangles, and classifier decision boundaries as solid or dashed lines.

we focus on models of the same capacity but trained on different types of data. Specifically, we treat the classifier trained on synthetic examples along with few real examples as the student, and treat the classifier trained on a large amount of real examples as the teacher. Using the distillation loss function (Hinton et al., 2015), our generative model is encouraged to produce such kind of examples that enable the student classifier to output the distribution of class probabilities predicted by the teacher. To make the generative model applicable to a broad range of categories, we further incorporate the distillation process into a meta-learning framework as in (Wang et al., 2018). Through meta-learning, we construct a variety of few-shot learning tasks from base categories with abundant labeled examples, thus being able to learn a generic, category-shared generative model. For a novel few-shot recognition task on unseen categories, we use the learned generative model to synthesize additional examples and produce an augmented training set for learning classifiers.

While we show that the basic framework of meta-learning with distillation already performs well, directly distilling the knowledge into the generative model might be still challenging. This is because the decision boundaries of the student and teacher classifiers could be far away from each other at the beginning of the training, if the teacher is produced by a large amount of real examples while the student has access to only few real examples. To mitigate this issue, we propose to distill the knowledge in a progressive manner and explore two different avenues of dual directions — (1) we start with a teacher and a student trained on a small number of real examples, and we gradually strengthen the teacher by re-training it with increasing number of real examples; (2) we start with a teacher and a student trained on a large number of real examples, and we gradually weaken the student by removing its real examples. During both of the processes, the generative model is trained progressively as well by producing more synthetic examples. Finally, we introduce ensemble of distillation and train independently several distillation processes on different student-teacher pairs, thus leading to a diverse collection of generative models and effectively reducing the variance of few-shot classifiers.

We demonstrate that our progressive distillation facilitates learning generative models to be directly useful for discriminative recognition tasks, significantly improving few-shot learning performance on both the widely benchmarked miniImageNet and much larger-scale ImageNet1K datasets. In particular, our approach is general and model-agnostic, which can synthesize in different feature spaces and can be combined with different meta-learning models to improve their performance.

2 RELATED WORK

Generative models. Largely initiated by generative adversarial networks (GANs) (Goodfellow et al., 2014) and variational auto-encoders (VAEs) (Kingma & Welling, 2014), deep generative models are now able to synthesize images which are difficult to distinguish from natural images without close inspection (Mao et al., 2017; Arjovsky et al., 2017; Karras et al., 2018; Brock et al., 2019; Razavi et al., 2019). Recently, generative models have also shown great potential as a way of data augmentation for few-shot learning (Antoniou et al., 2017; Zhang et al., 2018; Gao et al., 2018; Wang et al., 2018) and semi-supervised learning (Dai et al., 2017), but the improvement of recognition performance is still limited (Shmelkov et al., 2018). The generation can be performed either in image space (Chen et al., 2019b) or in a pre-trained feature space (Harisharan & Girshick, 2017) by using an auto-encoder architecture (Schwartz et al., 2018), GAN-like generator (Wang et al., 2018), or the combination of GANs and auto-encoders (Xian et al., 2018; 2019). Our work
is independent of these different types of generators, and we focus primarily on how to train the generator to improve its use for recognition tasks by leveraging large amounts of auxiliary data.

**Few-shot learning and meta-learning.** Meta-learning, or the ability to learn to learn (Thrun, 1998), is a powerful framework for tackling the problem of learning with limited data. Most of modern approaches fall into one of the categories between optimization and metric learning based methods. Optimization based methods learn how to do fast adaptation to novel tasks, by using memory based architectures (Ravi & Larochelle, 2017) or with very few gradient descent steps (Finn et al., 2017). Adaptation could be done in the original feature space (Finn et al., 2017; Antioniu et al., 2019; Antioniu & Storkey, 2019) or in an embedded space (Kusu et al., 2019). Metric learning methods focus on learning a similarity metric between examples belonging to the same class. Several distance functions have been explored, from the Euclidean distance (Snell et al., 2017) and the cosine distance (Chen et al., 2019a; Gidaris & Komodakis, 2018; Dvornik et al., 2019) to more complex parametric functions and metrics (Koch et al., 2015; Sung et al., 2018; Vinyals et al., 2016; Li et al., 2019; Koch et al., 2015), or using an additional task-specific metric (Oreshkin et al., 2018). Most methods often treat each category separately without considering the relations between them. Graph neural networks are thus introduced to leverage those relations (Satorras & Estrach, 2018; Kim et al., 2019; Gidaris & Komodakis 2019). To conduct meta-learning more effectively, recent approaches often first compute a set of features of the images using a trained feature extractor network. Given that high-dimensional features have better modeling capacity but are computationally expensive to work with, each meta-learning task is then formulated as a convex optimization problem and solved in its low-dimensional dual space (Bertinetto et al., 2018) [Lee et al., 2019]. Our generative component is model-agnostic and can be integrated into different meta-learning methods.

**Knowledge distillation.** Compressing one cumbersome or several models into a smaller model is a classic idea (Domingos, 1997; Buciluă et al., 2006) and has been popularized by the distillation formulation of Hinton et al. (2015). Recent work focuses on advanced techniques to guide the distillation process (Mirzadeh et al., 2019; Xu et al., 2018) and its applications to practical problems, such as object detection (Xu et al., 2019; Wei et al., 2018) and distributed machine learning (Anil et al., 2018). To the best of our knowledge, our work is the first to introduce knowledge distillation for learning generative models. Importantly, different from existing work that addresses models of different capacity, we consider models of the same capacity but trained on real or synthetic data.

## 3 GENERATIVE MODELING THROUGH KNOWLEDGE DISTILLATION FOR FEW-SHOT LEARNING

**Few-shot learning setting.** We are given a set of base categories \( C_{\text{base}} \) and a set of novel categories \( C_{\text{novel}} \), where \( C_{\text{base}} \cap C_{\text{novel}} = \emptyset \). We have a base dataset \( D_{\text{base}} \) with a large amount of annotated training examples per class and a novel dataset \( D_{\text{novel}} \) with only few annotated training examples per class. The goal of few-shot learning is to learn a good classification model for \( C_{\text{novel}} \) based on the small dataset \( D_{\text{novel}} \). Recent work achieves this by leveraging a meta-learning procedure (Vinyals et al., 2016), which learns from a collection of sampled few-shot classification tasks. Given a set of \( C \) and a set of data \( D \), a \( M \)-way \( k \)-shot task is composed of a subset \( L \) of \( M \) categories from \( C \), a support set \( S_{\text{train}} \) of \( k \) examples from \( D \) for each class in \( L \), and a test set \( S_{\text{test}} \) of one example from \( D \) for each class in \( L \). Meta-learning is performed in two phases as follows.

During *meta-training*, a classifier learns from a collection of \( M \)-way \( k \)-shot tasks sampled from \( C_{\text{base}} \) and \( D_{\text{base}} \). While our work is agnostic to different classification models, here we consider a variant of prototypical networks (Snell et al., 2017), using the cosine distance function instead of the standard Euclidean distance (Chen et al., 2019a). In each iteration, we compute a prototype representation for each class in \( L \). Each example is fed to an embedding function \( f_\theta \) with learnable parameters \( \theta \). The prototype of a class \( c \) is the mean of the output through \( f_\theta \) of examples from \( c \) in \( S_{\text{train}} \). We can then feed the examples in \( S_{\text{test}} \) to the classifier and update the parameters \( \theta \). During *meta-testing*, we use the same approach and build our previously meta-learned classifier with one unique \( M \)-way \( k \)-shot task, using \( C_{\text{novel}} \) instead of \( C_{\text{base}} \) and \( D_{\text{novel}} \) instead of \( D_{\text{base}} \). We evaluate the final classifier on unseen examples with labels from \( C_{\text{novel}} \).

**Meta-learning with generative models.** Incorporating a generative model which produces additional examples for data augmentation has been shown to facilitate meta-learning (Wang et al., 2018; Gao et al., 2018; Schwartz et al., 2018). While our distillation approach does not rely on
sampled from a Gaussian distribution, and \( w \) is the random noise vector of dimension \( d_{\text{noise}} \) sampled from a Gaussian distribution, and \( w \) is the parameters of \( G \). We improve this generator architecture by introducing the mean of the category of interest as another input. Our generator thus becomes a function \( G(x, z, q; w; \mu) : \mathbb{R}^{d + d_{\text{noise}}} \rightarrow \mathbb{R}^d \), where \( q \) is the mean of the available examples from the category of \( x \). The synthesized example \( G(x, z, q; w) \) is of the same category as \( x \).

Now the procedure of meta-learning integrated with the generator \( G \) is illustrated in Figure 2. During each iteration of meta-training, the support set \( S_{\text{train}} \) is first augmented by a generated set \( S_{\text{train}}^G \). Specifically, for each class \( y \), we sample \( k_{\text{train}} \) examples \((x, y)\) in \( S_{\text{train}} \), sample associated random noise vectors \( z \), compute \( q \) using examples in \( S_{\text{train}} \), and then add \((x', y)\) to \( S_{\text{train}}^G \), where \( x' = G(x, z, q; w) \). Our final training set is \( S_{\text{aug}} = S_{\text{train}} \cup S_{\text{train}}^G \). As long as \( G \) is differentiable with respect to the generated set \( S_{\text{train}}^G \), the gradients of the final classification loss function can be back-propagated into \( G \) to produce useful synthetic examples. Through meta-training over a large amount of iterations, the generator learns to capture shared modes of variation across different categories and can thus generalize to unseen categories. During meta-testing, we use the learned \( G \) to synthesize additional examples for recognizing categories in \( C_{\text{novel}} \).

**Distilling knowledge into generative models.** The end-to-end optimization of the classification objective enables the generator to synthesize discriminative examples that contribute to formulating classifier decision boundaries. However, since the synthetic examples are generated based on a small support set, the resulting classifier could be still far away from the desired classifier that would be learned from a large set of real examples. This makes it critical to close the gap between these two classifiers. In fact, during meta-training, a large amount of annotated examples are already available for the base categories \( C_{\text{base}} \), which allows us to explicitly obtain the classifier trained on the full set and use it to guide the learning of the generator.

Formally, we treat the classifier trained on the augmented set of the synthetic examples and the few support examples as a student model, and we treat the classifier trained on the original full set of real examples as a teacher model. Our goal then is to minimize the discrepancy between the student and its teacher. While a naïve approach would be to directly characterize the difference between their model parameters, it turns out to be challenging due to the high dimensionality of the parameter space. Inspired by knowledge distillation (Hinton et al., 2015), we instead enforce the student to mimic the distribution of class probabilities predicted by the teacher.

As shown in Figure 2, meta-training a student model, or essentially the generator \( G \), is conducted in the following way. We first sample a large set of examples \( S_{\text{teacher}} \) with \( k_{\text{teacher}} \) examples per class in \( C_{\text{base}} \) and train a teacher classifier using all the examples in \( S_{\text{teacher}} \). During each iteration of meta-training, we augment \( S_{\text{train}} \) by generating new examples using the generator \( G \). We train the student classifier on \( S_{\text{aug}} \) through the knowledge distillation loss function in (Hinton et al., 2015):

\[
L_{kd}(s, t, y) = L_{CE}(s, y) + \gamma T^2 L_{CE}(\sigma(s/T), \sigma(t/T)),
\]

which consists of a standard cross-entropy loss (the first term) and an additional component that measures the difference between student and teacher outputs (the second term). \( s \) and \( t \) are the logits

![Figure 2: Framework overview of meta-learning a generative model through knowledge distillation. During each iteration of meta-training, a small support set \( S_{\text{train}} \) is augmented with a set \( S^G_{\text{train}} \) of examples synthesized by a generator \( G \). Examples from the resulting set \( S_{\text{aug}} \) are used to build a student classifier model \( S \). A teacher classifier model \( T \) is built from a set \( S_{\text{teacher}} \) containing a large amount of real examples. Examples from the test set \( S_{\text{test}} \) are fed into the student and the teacher models. The corresponding knowledge distillation loss \( L_{kd} \) is computed over \( S_{\text{test}} \), and its gradients are back-propagated first into \( S \) and then into \( G \).]
produced by the student and the teacher, respectively, for a test example of label \( y \) in \( S_{\text{test}} \) which is associated with \( S_{\text{train}} \). \( \sigma \) denotes the softmax function, \( L_{CE} \) denotes the cross-entropy loss, \( e_y \) is the one-hot encoding of \( y \), and \( \gamma \) is a trade-off hyper-parameter that balances the two terms. Note that \( T \) is a critical learnable parameter called temperature, which smooths the probability distribution produced by the teacher and makes the corresponding decision boundary easier to learn for the student than the original one. Minimizing Eqn. [1] over \( S_{\text{test}} \) thus guides the generator \( G \) towards synthesizing examples that help the student recover the decision boundary from the teacher.

4 Progressive and Ensemble Distillation

Under the framework of meta-learning with distillation, a straightforward way is to build the teacher classifier by using \( k_{\text{teacher}} \) as large as possible (potentially the full set of \( D_{\text{base}} \)) and keep it fixed, and to train a student classifier using only few real examples. By doing so, however, we face the problem that the decision boundaries obtained by those two classifiers could be very far from each other at the beginning of the training, making the learning of the generator difficult. To address this issue, we perform the distillation process in a progressive manner with varied number of real examples. We start with a teacher and a student which have access to a not too different number of real examples, and then progressively change the number of examples, so that the decision boundaries transform in a smooth manner. Concretely, this can be achieved in the following two dual directions.

**Progressive distillation by strengthening the teacher.** In this setting, both the student and the teacher start with a small number of real examples. However, the number of real examples for the teacher gradually increases over the training. The objective for the generator then is to learn to generate additional examples so that its corresponding student can always match the performance of the teacher, whenever the teacher is re-trained with more samples and becomes stronger. More specifically, during meta-training, the support set \( S_{\text{train}} \) of each few-shot task is composed of very few examples per class, \( k_{\text{train}} \), as in regular meta-training. At the beginning, we sample \( S_{\text{teacher}} \) with \( k_{\text{teacher}} \) being set to the value of \( k_{\text{train}} \). We then progressively sample new real examples in the same amount for each class and add them into \( S_{\text{teacher}} \). \( k_{\text{teacher}} \) grows from \( k_{\text{train}} \) to \( k_{\text{max}} \) in a linear or logarithmic scale, where \( k_{\text{max}} \) is the maximum available number of examples per class in \( D_{\text{base}} \). We retrain the teacher classifier every time we add new examples.

**Progressive distillation by weakening the student.** In this setting, both the student and the teacher start with a large number of real examples. However, we gradually remove the real examples for the student over the training. The objective for the generator then is to learn to generate the missing examples based on the remaining real examples. This allows the student to preserve or stabilize the original decision boundary formulated by the large set of examples (i.e., the teacher boundary), when the student has access to less real examples and becomes weaker. More specifically, during meta-training, the support set \( S_{\text{train}} \) of each “few-shot” task is composed of a large number of examples per class, unlike regular meta-training. This number of examples per classes in \( S_{\text{train}} \), \( k_{\text{train}} \), decreases in a linear or logarithmic scale, until it reaches a small value.

**Ensemble learning.** To further benefit from diverse teachers, we introduce ensemble of distillation and train independently several generative models. Each of them is guided by a different teacher. Take the distillation by weakening the student as an example. Empirically, we found that, for a single teacher, starting with \( k_{\text{train}} = k_{\text{max}} \) works as well as starting with lower values. Hence, we sample \( T \) sets \( S_{\text{teacher}} \) with \( k_{\text{teacher}} < k_{\text{max}} \) examples and build \( T \) teachers. Accordingly, we meta-train \( T \) students and the associated generators, each with a different teacher. For a given test example, the final label prediction is the average of the predictions of each student. In addition to obtaining a diverse collection of generators, our ensemble learning effectively helps reduce the variance of few-shot student classifiers, consistent with the recent work of [Dvornik et al., 2019].

5 Evaluation

We now present experiments to evaluate our framework of meta-learning with knowledge distillation on few-shot classification tasks, and study the effect of knowledge distillation for generative modeling. While our work is agnostic to the choice of classification models, here we focus on a simple cosine classifier, which has been recently shown to achieve very competitive few-shot learning performance [Chen et al., 2019]. And we explore learning the variants of this classifier based on
different types of features. We evaluate on two standard benchmarks: miniImageNet (Vinyals et al., 2016; Ravi & Larochelle, 2017) and ImageNet1K (Hariharan & Girshick, 2017; Wang et al., 2018).

5.1 ImageNet1K

Dataset. The ImageNet1k dataset was proposed by Hariharan & Girshick (2017) and improved by Wang et al. (2018). The dataset is a subset of the ILSVRC-12 dataset (Russakovsky et al., 2015) and contains 389 base categories and 611 novel categories, with 193 of the base categories and 300 of the novel categories used for cross validation and the remaining 196 base categories and 311 novel categories used for the final evaluation. The evaluation is done in 311-way (the number of novel classes), \( k \in \{1, 2, 5, 10, 20\} \)-shot settings.

Pre-trained features. We follow the approach of Hariharan & Girshick (2017), where they first train a feature extractor using regular training and then perform meta-training with the features vectors extracted from the last layer before softmax of the feature extractor. We used pre-trained features from a ResNet-10 architecture (He et al., 2015) coupled with a standard linear classifier (Hariharan & Girshick, 2017) or a cosine distance based classifier (Gidaris & Komodakis, 2018).

Evaluation protocol. For fair comparisons, we report the mean top-5 accuracies, calculated under different protocols depending on the set of pre-trained features. When using the pre-trained features from Hariharan & Girshick (2017), we use their protocol and average over 5 pre-determined \( k \)-shot tasks. When using the pre-trained feature obtained from Gidaris & Komodakis (2018), we follow their protocol and average over 100 randomly sampled \( k \)-shot tasks. When we learn an ensemble of models with mixed features, we use the protocol of Hariharan & Girshick (2017).

Comparison to baselines and concurrent work. We compare against several baselines and competitors as follows. (1) For each of the pre-trained features, we focus on comparing the variants of our distillation approach with the cosine classifier and the cosine classifier integrated with a plain generator without distillation. (2) We compare with concurrent, state-of-the-art meta-learning based few-shot learning approaches, including prototypical nets (Snell et al., 2017), matching nets (Vinyals et al., 2016), prototype matching nets (Wang et al., 2018), cosine classifier & attention weight generator (Cosine Att. Weight) (Gidaris & Komodakis, 2018). (3) We also include results of other approaches that incorporate a generator into standard learning (e.g., logistic regression Gen) (Hariharan & Girshick, 2017) or meta-learning (e.g., prototype matching nets Gen) (Wang et al., 2018).

Table 1 summarizes the results. Note that the 95% confidence intervals for the recognition accuracy on the ImageNet1K benchmark are of the order of 0.2% (Gidaris & Komodakis, 2018). We thus observe that in all cases our approach substantially outperforms the baselines, irrespective of the choice of the pre-trained features. This indicates that the sample generation is effective in different feature spaces. In addition, guided by our distillation process, a simple cosine classifier achieves superior performance than state-of-the-art approaches that are based on more complex classification models, such as the attention based classifier (Gidaris & Komodakis, 2018). Our approach could be combined with these classification models as well to further improve their performance, which is an interesting direction for future research.

Ablation and diagnostic analysis. To unpack the performance gain and understand the impact of different components and design choices, we perform a series of ablations summarized in Table 1.

Impact of pre-trained features. While our approach consistently outperforms the baselines irrespective of the types of the pre-trained features, the improvement is more pronounced when the features are pre-trained with a standard linear classifier. This shows that our generator is able to reconcile the conflict between the pre-trained feature and the final recognition classifier. For the cosine classifier which we used as the final classifier, the features pre-trained with a linear classifier are not consistent with it, resulting to poor performance of the plain cosine classifier without any generation. However, benefited from the end-to-end distillation, our generator spends its capacity on suppressing such inconsistency which throws the classifier off, thus significantly boosting the performance.

Strengthening the teacher vs. weakening the student. Comparing the two directions for progressive distillation, we observe that both of them outperform the normal distillation without progression, and that weakening the student achieves better results. It comes from the fact that, if both teacher and student start being weak, the learning problem could actually be hard due to the high variance...
Table 1: Top-5 accuracy comparison with state of the art and ablation study on ImageNet1K 311-way, k-shot classification. Gen: with a sample generator. Dist: with normal distillation to learn the generator. Dist↓: progressive distillation in the way of weakening the student. Dist Ensemble: ensemble of distillation. ‘Standard’: features pre-trained with a standard ResNet10 feature extractor network using a linear classifier. ‘Cosine’: features pre-trained with a ResNet10 network using a cosine distance based classifier. ‘Mixed’: ensemble of distillation with half models trained with ‘Standard’ features and half models trained with ‘Cosine’ features. The 95% confidence intervals for all number are of the order of 0.2%. We report in red and blue the best and second best accuracy for each $k$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Features</th>
<th>k=1</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosine Classifier (baseline)</td>
<td>Standard</td>
<td>37.8</td>
<td>51.0</td>
<td>65.5</td>
<td>72.5</td>
<td>76.6</td>
</tr>
<tr>
<td>Cosine Classifier Gen (baseline)</td>
<td>Standard</td>
<td>42.6</td>
<td>53.9</td>
<td>66.4</td>
<td>72.6</td>
<td>76.3</td>
</tr>
<tr>
<td>Cosine Classifier Dist</td>
<td>Standard</td>
<td>44.5</td>
<td>56.2</td>
<td>68.6</td>
<td>74.2</td>
<td>77.3</td>
</tr>
<tr>
<td>Cosine Classifier Dist↓</td>
<td>Standard</td>
<td>44.7</td>
<td>56.6</td>
<td>68.8</td>
<td>74.2</td>
<td>77.3</td>
</tr>
<tr>
<td>Cosine Classifier Dist↓ Ensemble</td>
<td>Standard</td>
<td>45.1</td>
<td>56.2</td>
<td>68.8</td>
<td>74.8</td>
<td>78.3</td>
</tr>
<tr>
<td>Cosine Classifier Dist↓ Ensemble Mixed</td>
<td>Standard</td>
<td>46.2</td>
<td>58.3</td>
<td>70.0</td>
<td>75.6</td>
<td>78.7</td>
</tr>
<tr>
<td>Cosine Classifier (baseline)</td>
<td>Cosine</td>
<td>45.8</td>
<td>57.0</td>
<td>68.9</td>
<td>74.3</td>
<td>77.4</td>
</tr>
<tr>
<td>Cosine Classifier Gen (baseline)</td>
<td>Cosine</td>
<td>47.0</td>
<td>57.8</td>
<td>69.1</td>
<td>74.3</td>
<td>77.6</td>
</tr>
<tr>
<td>Cosine Classifier Dist↓</td>
<td>Cosine</td>
<td>47.2</td>
<td>58.2</td>
<td>69.2</td>
<td>74.4</td>
<td>77.5</td>
</tr>
<tr>
<td>Cosine Classifier Dist↓ Ensemble</td>
<td>Cosine</td>
<td>47.8</td>
<td>58.7</td>
<td>69.5</td>
<td>74.5</td>
<td>77.6</td>
</tr>
<tr>
<td>Cosine Classifier Dist↓ Ensemble Mixed</td>
<td>Mixed</td>
<td>46.9</td>
<td>59.0</td>
<td>70.4</td>
<td>75.8</td>
<td>78.8</td>
</tr>
</tbody>
</table>

**Concurrent work**

- Prototypical Nets (Snell et al., 2017)
- Matching Nets (Vinyals et al., 2016)
- Logistic regression (Hariharan & Girshick, 2017)
- Prototype Matching Nets Gen (Wang et al., 2018)
- Cosine Att. Weight (Gidaris & Komodakis, 2018)

of both teacher and student. By contrast, this is not the case when both start with a relatively large number of real examples, which makes the distillation process more stable.

Schedule of progressive distillation. Empirically, we found that using the logarithmic scale when changing the number of examples on which the student or teacher model is trained performs better than using a linear scale. The reported number in Table 1 is thus on logarithmic scale. This is consistent with the general observation that recognition performance changes on a logarithmic scale as the number of training samples varies (Sun et al., 2017; Wang et al., 2017).

A single super teacher vs. ensemble of teachers. Comparing with a single super teacher that is trained on all available real examples, ensemble learning based on multiple teachers, each of which is trained on a randomly sampled large set of real examples, achieves superior performance. Note that for each pair of student and teacher, we used the same set of pre-trained features, in order to guarantee that the performance boost comes from the diversity of the teacher classification networks and not from the diversity of the representations learned by the feature extractor networks. This is different from the ensemble learning in (Dvornik et al., 2019). By training and benefiting diversity feature extractors as in (Dvornik et al., 2019), our performance could be further improved. We demonstrate this by training an ensemble with two sets of pre-trained features, denoted as ‘mixed’.

5.2 miniImageNet

**Dataset and pre-trained features.** The miniImageNet dataset (Vinyals et al., 2016) contains 100 classes with 600 images from each class. We use the split proposed by Ravi & Larochelle (2017), where the classes are divided into 64 training classes, 16 validation classes, and 20 test classes. The evaluation is conducted in 5-way, 1-shot and 5-way, 5-shot setting. We average over 1000 randomly sampled tasks and report accuracies and the 95% confidence intervals. We use a ResNet18 feature extractor trained with the ‘baseline’ method from Chen et al. (2019a).

**Results.** From Table 2, we observe a same trend of performance improvement as on ImageNet1K, when adding the different components of our approach. The relative improvement of progressive distillation over the normal distillation is more notable on ImageNet1K than miniImageNet, showing the importance of performing distillation progressively when the gap between student and teacher is large in more challenging classification problems. In addition, consistent with recent work, our generator further improves the performance when trained using additional validation data.
Table 2: Comparison with state of the art and ablation study on miniImageNet. Please refer to the caption of Table 1 for notation. * indicates using validation data during meta-training phase.

<table>
<thead>
<tr>
<th>Method</th>
<th>K=1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosine Classifier (baseline)</td>
<td>51.99 ± 0.59</td>
<td>74.32 ± 0.49</td>
</tr>
<tr>
<td>Cosine Classifier Gen (baseline)</td>
<td>57.88 ± 0.61</td>
<td>75.95 ± 0.48</td>
</tr>
<tr>
<td>Cosine Classifier Dist</td>
<td>59.20 ± 0.63</td>
<td>76.36 ± 0.50</td>
</tr>
<tr>
<td>Cosine Classifier Dist↓ Ensemble</td>
<td>59.56 ± 0.62</td>
<td>76.57 ± 0.50</td>
</tr>
<tr>
<td>Cosine Classifier Dist↓ Ensemble*</td>
<td>61.18 ± 0.59</td>
<td>78.34 ± 0.49</td>
</tr>
</tbody>
</table>

Concurrent work

<table>
<thead>
<tr>
<th>Method</th>
<th>K=1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta-Learning LSTM (Ravi &amp; Larochelle, 2017)</td>
<td>43.44 ± 0.77</td>
<td>60.60 ± 0.71</td>
</tr>
<tr>
<td>Matching Networks (Vinyals et al., 2016)</td>
<td>43.56 ± 0.84</td>
<td>55.31 ± 0.73</td>
</tr>
<tr>
<td>MAML (Finn et al. 2017)</td>
<td>48.70 ± 1.84</td>
<td>63.11 ± 0.92</td>
</tr>
<tr>
<td>Prototypical Networks (Snell et al. 2017)</td>
<td>49.42 ± 0.78</td>
<td>68.20 ± 0.66</td>
</tr>
<tr>
<td>Relation Networks (Sung et al., 2018)</td>
<td>50.44 ± 0.82</td>
<td>65.32 ± 0.70</td>
</tr>
<tr>
<td>R2D2 (Bertinetto et al., 2018)</td>
<td>51.2 ± 0.6</td>
<td>68.8 ± 0.1</td>
</tr>
<tr>
<td>Transductive Prop Nets (Liu et al., 2019)</td>
<td>55.51 ± 0.86</td>
<td>69.86 ± 0.65</td>
</tr>
<tr>
<td>SNAIL (Mishra et al., 2018)</td>
<td>55.71 ± 0.99</td>
<td>68.88 ± 0.92</td>
</tr>
<tr>
<td>Dynamic Few-shot (Gidaris &amp; Komodakis, 2018)</td>
<td>56.20 ± 0.86</td>
<td>73.00 ± 0.64</td>
</tr>
<tr>
<td>AdaResNet (Munkhdalai et al. 2018)</td>
<td>56.88 ± 0.62</td>
<td>71.94 ± 0.57</td>
</tr>
<tr>
<td>TADAM (Oreshkin et al., 2018)</td>
<td>58.50 ± 0.30</td>
<td>76.70 ± 0.30</td>
</tr>
<tr>
<td>Activation to Parameter* (Qiao et al., 2018)</td>
<td>59.60 ± 0.41</td>
<td>73.74 ± 0.19</td>
</tr>
<tr>
<td>LEO* (Rusu et al., 2019)</td>
<td>61.76 ± 0.08</td>
<td>77.59 ± 0.12</td>
</tr>
</tbody>
</table>

Figure 3: Visualization with t-SNE of the evolution of the decision boundary for two novel classes, when meta-training the generator through progressive distillation by weakening the student. Real examples (small dots) are progressively removed, and synthesized examples (triangles) are generated in a way that helps maintain the student decision boundary (red dashed line) as close as possible to the desired decision boundary that would be formulated by a large set of real examples (black solid line).

Figure 4: Visualization of synthesized examples for four novel classes. The single black framed image come from the original dataset and is used as a seed for synthesising new examples. Color framed images correspond to the nearest neighbor real images of the synthesized examples in the feature space. Best viewed in color.

5.3 Visualizing and Understanding Progressive Distillation

To further understand how the progressive distillation procedure helps learning a classifier and refining the generator, we visualize it with real data using t-SNE (van der Maaten & Hinton, 2008). We first visualize in Figure 3 the evolution of the decision boundary for two novel classes during progressive distillation by weakening the student. We then visualize in Figure 4 the synthesized examples in the pixel space, using their nearest neighbor real image in the feature space.

6 Conclusion

In this paper, we introduced a general framework of meta-learning with knowledge distillation to guide the learning of generative models to be directly useful for discriminative recognition tasks. We apply our approach to few-shot learning, where the amount of available data is very limited and therefore this kind of generation is in particular helpful. By progressively distilling the knowledge and benefiting from diverse teachers, our approach achieves state-of-the-art results on heavily benchmarked miniImageNet and ImageNet1K few-shot classification datasets.
REFERENCES


Zheng Xu, Yen-Chang Hsu, and Jiawei Huang. Training shallow and thin networks for acceleration via knowledge distillation with conditional adversarial networks. In ICLR, 2018.


A IMPLEMENTATION DETAILS

In this section, we provide implementation details about the distillation procedure and the evaluation on the ImageNet1k and miniImageNet benchmarks.

**ImageNet1k.** The embedding function $f_{\theta}$ of our cosine classifier has a different architecture depending on the pre-trained features used. When using features trained with a linear classifier [Hariharan & Girshick, 2017], the embedding function of the student model is a 2 layers fully-connected network. The teacher model, however, is trained using a simple linear layer embedding. We found that the distillation is very hard when the difference between teacher and student is large. Reducing the capacity of the embedding is a way to reduce the gap between the two, without reducing the number of examples on which the teacher is trained. We found, however, that the addition of an embedding function was not helpful when using pre-trained features from a classifier based on the cosine distance [Gidaris & Komodakis, 2018]. This is because the embedding function helps shape the synthesised features for the classifier used during distillation. Therefore, when using a cosine classifier, the features are already more consistent with the classifier.

During progressive distillation by weakening the student, we start training the teacher with $k_{\text{teacher}} = 256$, and we decrease that number to 1 in a logarithmic scale over 50000 iterations. We initialize the temperature to 7 and the scale factor of the cosine distance to 75, and learn those parameters. $\gamma$ is set to 150. We save the student model once for each time $k_{\text{train}}$ takes its value in $[16, 8, 4, 2, 1]$. We use each of those models respectively when testing in a $k = [20, 10, 5, 2, 1]$ setting. For distillation with ensemble, we learn 12 pairs of student and teacher. The performance
does not improve when increasing this number. The ImageNet1k dataset contains about 1000 examples per class. Using 12 teachers built on 256 randomly sampled examples per class from the 1000 examples per class is good to cover almost all the examples and to make sure the different students learn from all the examples in the dataset both by distillation and by meta-learning.

miniImageNet. When evaluating on miniImageNet, we don not use any embedding function. Few-shot learning on the miniImageNet benchmark is relatively easy compared to the ImageNet1k benchmark, and using an embedding function leads to strong overfitting.

During progressive distillation by weakening the student, we start training the teacher with $k_{\text{teacher}} = 128$, and we decrease that number to 1 in a logarithmic scale over 6000 iterations. We initialize the temperature to 7, the scale factor of the cosine distance to 150, and $\gamma$ to 5. For distillation with ensemble we learn 20 pairs of student and teacher.