Diversity with Cooperation: Ensemble Methods for Few-Shot Classification
Nikita Dvornik, Cordelia Schmid, Julien Mairal

To cite this version:

HAL Id: hal-02080004
https://hal.archives-ouvertes.fr/hal-02080004v2
Submitted on 8 Aug 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Diversity with Cooperation: Ensemble Methods for Few-Shot Classification

Nikita Dvornik, Cordelia Schmid, Julien Mairal
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
firstname.lastname@inria.fr

Abstract

Few-shot classification consists of learning a predictive model that is able to effectively adapt to a new class, given only a few annotated samples. To solve this challenging problem, meta-learning has become a popular paradigm that advocates the ability to “learn to adapt”. Recent works have shown, however, that simple learning strategies without meta-learning could be competitive. In this paper, we go a step further and show that by addressing the fundamental high-variance issue of few-shot learning classifiers, it is possible to significantly outperform current meta-learning techniques. Our approach consists of designing an ensemble of deep networks to leverage the variance of the classifiers, and introducing new strategies to encourage the networks to cooperate, while encouraging prediction diversity. Evaluation is conducted on the mini-ImageNet, tiered-ImageNet and CUB datasets, where we show that even a single network obtained by distillation yields state-of-the-art results.

1. Introduction

Convolutional neural networks [17] have become standard tools in computer vision to model images, leading to outstanding results in many visual recognition tasks such as classification [16], object detection [8, 19, 26], or semantic segmentation [8, 20, 27]. Massively annotated datasets such as ImageNet [28] or COCO [18] seem to have played a key role in this success. However, annotating a large corpus is expensive and not always feasible, depending on the task at hand. Improving the generalization capabilities of deep neural networks and removing the need for huge sets of annotations is thus of utmost importance.

While such a grand challenge may be addressed from different complementary points of views, e.g., large-scale unsupervised learning [4], self-supervised learning [7, 12], or by developing regularization techniques dedicated to deep networks [2, 36], we choose in this paper to focus on variance-reduction principles based on ensemble methods.

Specifically, we are interested in few-shot classification, where a classifier is first trained from scratch on a medium-sized annotated corpus—that is, without leveraging external data or a pre-trained network, and then we evaluate its ability to adapt to new classes, for which only very few annotated samples are provided (typically 1 or 5). Unfortunately, simply fine-tuning a convolutional neural network on a new classification task with very few samples has been shown to provide poor results [9], which has motivated the community to develop dedicated approaches.

The dominant paradigm in few-shot learning builds upon meta-learning [9, 24, 30, 32, 31, 33], which is formulated as a principle to learn how to adapt to new learning problems. These approaches split a large annotated corpus into classification tasks, and the goal is to transfer knowledge across tasks in order to improve generalization. While the meta-learning principle seems appealing for few-shot learning, its empirical benefits have not been clearly established yet. There is indeed strong evidence [5, 11, 23] that train-
Our angle of attack consists of using ensemble methods to outperform the current state of the art in few-shot learning. Meta-learning approaches typically sample few-shot learning classification tasks from the meta-training dataset, and train a model such that it should generalize on a new task that has been left aside. For instance, in [9] a “good network initialization” is learned such that a small number of gradient steps on a new problem is sufficient to obtain a good solution. In [24], the authors learn both the network initialization and an update rule (optimization model) represented by a Long-Term-Short-Memory network (LSTM). Inspired by few-shot learning strategies developed before deep learning approaches became popular [21], distance-based classifiers based on the distance to a centroid were also proposed, e.g., prototypical networks [31], or more sophisticated classifiers with attention [33]. All these methods consider a classical backbone network, and train it from scratch using meta-learning.

Recently, these meta-learning were found to be sub-optimal [11, 22, 23]. Specifically, better results were obtained by training the network on the classical classification task using the meta-training data in a first step, and then only fine-tuning with meta-learning in a second step [22, 29, 35]. Others such as [11, 23] simply freeze the network obtained in the first step, and train a simple prediction layer with meta-learning, which results in similar performance. Finally, the paper [5] demonstrates that simple baselines without meta-learning—based on distance-based classifiers—work equally well. Our paper pushes such principles even further and shows that by appropriate variance-reduction techniques, these approaches can significantly outperform the current state of the art.

### 2. Related Work

In this section, we discuss related work on few-shot learning, meta-learning, and ensemble methods.

**Few-shot classification.** Typical few-shot classification problems consist of two parts called meta-training and meta-testing [5]. During the meta-training stage, one is given a large-enough annotated dataset, which is used to train a predictive model. During meta-testing, novel categories are provided along with few annotated examples, and we evaluate the capacity of the predictive model to retrain or adapt, and then generalize on these new classes.

Meta-learning approaches typically sample few-shot learning classification tasks from the meta-training dataset, and train a model such that it should generalize on a new task that has been left aside. For instance, in [9] a “good network initialization” is learned such that a small number of gradient steps on a new problem is sufficient to obtain a good solution. In [24], the authors learn both the network initialization and an update rule (optimization model) represented by a Long-Term-Short-Memory network (LSTM). Inspired by few-shot learning strategies developed before deep learning approaches became popular [21], distance-based classifiers based on the distance to a centroid were also proposed, e.g., prototypical networks [31], or more sophisticated classifiers with attention [33]. All these methods consider a classical backbone network, and train it from scratch using meta-learning.

Recently, these meta-learning were found to be sub-optimal [11, 22, 23]. Specifically, better results were obtained by training the network on the classical classification task using the meta-training data in a first step, and then only fine-tuning with meta-learning in a second step [22, 29, 35]. Others such as [11, 23] simply freeze the network obtained in the first step, and train a simple prediction layer with meta-learning, which results in similar performance. Finally, the paper [5] demonstrates that simple baselines without meta-learning—based on distance-based classifiers—work equally well. Our paper pushes such principles even further and shows that by appropriate variance-reduction techniques, these approaches can significantly outperform the current state of the art.

**Ensemble methods.** It is well known that ensemble methods reduce the variance of estimators and subsequently may improve the quality of prediction [10]. To gain accuracy from averaging, various randomization or data augmentation techniques are typically used to encourage a high diversity of predictions [3, 6]. While individual classifiers of the ensemble may perform poorly, the quality of the average prediction turns out to be sometimes surprisingly high.

Even though ensemble methods are costly at training time for neural networks, it was shown that a sin-
ingle network trained to mimic the behavior of the ensemble could perform almost equally well [14]—a procedure called distillation—thus removing the overhead at test time. To improve the scalability of distillation in the context of highly-parallelized implementations, an online distillation procedure is proposed in [1]. There, each network is encouraged to agree with the averaged predictions made by other networks of the ensemble, which results in more stable models. The objective of our work is however significantly different. The form of cooperation they encourage between networks is indeed targeted to scalability and stability (due to industrial constraints), but online distilled networks do not necessarily perform better than the basic ensemble strategy. Our goal, on the other hand, is to improve the quality of prediction and do better than basic ensembles.

To this end, we encourage cooperation in a different manner, by encouraging predictions between networks to match in terms of class probabilities conditioned on the prediction not being the ground truth label. While we show that such a strategy alone is useful in general when the number of networks is small, encouraging diversity becomes crucial when this number grows. Finally, we show that distillation can help to reduce the computational overhead at test time.

3. Our Approach

In this section, we present our approach for few-shot classification, starting with preliminary components.

3.1. Mean-centroid classifiers

We now explain how to perform few-shot classification with a fixed feature extractor and a mean centroid classifier.

**Few-shot classification with prototype classifier.** During the meta-training stage, we are given a dataset \( D_b \) with annotations, which we use to train a prediction function \( f_\theta \) represented by a CNN. Formally, after training the CNN on \( D_b \), we remove the final prediction layer and use the resulting vector \( \tilde{f}_\theta(x) \) as a set of visual features for a given image \( x \). The parameters \( \theta \) represent the weights of the network, which are frozen after this training step.

During meta-testing, we are given a new dataset \( D_q = \{x_i, y_i\}_{i=1}^{nk} \), where \( n \) is the number of new categories and \( k \) is the number of available examples for each class. The \( (x_i, y_i) \)'s represent image-label pairs. Then, we build a mean centroid classifier, leading to the class prototypes

\[
c_j = \frac{1}{k} \sum_{i=1}^{k} \tilde{f}_\theta(x_i), \quad j = 1, \ldots, n.
\]

Finally, a test sample \( x \) is assigned to the nearest centroid's class. Simple mean-centroid classifiers have proven to be effective in the context of few-shot classification [5][21][31], which is confirmed in the following experiment.

**Motivation for mean-centroid classifier.** We report here an experiment showing that a more complex model than (1) does not necessarily lead to better results for few-shot learning. Consider indeed a parametrized version of (1):

\[
c_j = \sum_{i=1}^{nk} \alpha_i^j \tilde{f}_\theta(x_i), \quad j = 1, \ldots, n,
\]

where the weights \( \alpha_i^j \) can be learned with gradient descent by maximizing the likelihood of the probabilistic model

\[
p_j(y = l|x) = \frac{\exp(-d(\tilde{f}_\theta(x), c_l))}{\sum_{j=1}^{n} \exp(-d(\tilde{f}_\theta(x), c_j))}
\]

where \( d(\cdot, \cdot) \) is a distance function, such as Euclidian distance or negative cosine similarity. Since the coefficients are learned from data and not set arbitrarily to \( 1/k \) as in (1), one would potentially expect this method to produce better classifiers if appropriately regularized. When we run the evaluation of the aforementioned classifiers on 1000 5-shot learning tasks sampled from miniimagenet-test (see experimental section for details about this dataset), we get similar results on average: 77.28 ± 0.46% for (1) vs. 77.01 ± 0.50% for (2), confirming that learning meaningful parameters in this very-low-sample regime is difficult.

3.2. Learning ensembles of deep networks

During meta-training, one needs to minimize the following loss function over a training set \( \{x_i, y_i\}_{i=1}^{m} \):

\[
L(\theta) = \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, \sigma(f_\theta(x_i))) + \lambda\|\theta\|^2,
\]

where \( f_\theta \) is a CNN as before. The cost function \( \ell(\cdot, \cdot) \) is the cross-entropy between ground-truth labels and predicted class probabilities \( p = \sigma(f_\theta(x)) \), where \( \sigma \) is the normalized exponential function, and \( \lambda \) is a weight decay parameter.

When training an ensemble of \( K \) networks \( f_{\theta_{q}} \) independently, one would solve (4) for each network separately. While these terms may look identical, solutions provided by deep neural networks will typically differ when trained with different initializations and random seeds, making ensemble methods appealing in this context.

In this paper, we are interested in ensemble of networks, but we also want to model relationships between its members; this may be achieved by considering a pairwise penalty function \( \psi \), leading to the joint formulation:

\[
L(\bar{\theta}) = \sum_{j=1}^{K} \left( \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \sigma(f_{\theta_j}(x_i))) + \lambda\|\theta_j\|^2 \right) + \frac{\gamma}{n(K-1)} \sum_{i=1}^{n} \sum_{j \neq l}^{K} \psi(y_i, f_{\theta_j}(x_i), f_{\theta_l}(x_i)),
\]

where \( \gamma \) represents the relative contribution of the pairwise penalty term.
where \( \tilde{\theta} \) is the vector obtained by concatenating all the parameters \( \theta_j \). By carefully designing the function \( \psi \) and setting up appropriately the parameter \( \gamma \), it is possible to achieve desirable properties of the ensemble, such as diversity of predictions or collaboration during training.

### 3.3. Encouraging diversity and cooperation

To reduce the high variance of few-shot learning classifiers, we use ensemble methods trained with a particular interaction function \( \psi \), as in (5). Then, once the parameters \( \theta_j \) have been learned during meta-training, classification in meta-testing is performed by considering a collection of \( K \) mean-centroid classifiers associated to the basic probabilistic model presented in Eq. (3). Given a test image, \( \psi \) is cross-entropy, the \( K \) class probabilities are averaged. Such a strategy was found to perform empirically better than a voting scheme.

As we show in the experimental section, the choice of pairwise relationship function \( \psi \) significantly influences the quality of the ensemble. Here, we describe three different strategies, which all provide benefits in different regimes, starting by a criterion encouraging diversity of predictions.

**Diversity.** One way to encourage diversity consists of introducing randomization in the learning procedure, e.g., by using data augmentation \([3, 10]\) or various initializations. Here, we also evaluate the effect of an interaction function \( \psi \) that acts directly on the network predictions. Given an image \( x \), two models parametrized by \( \theta_i \) and \( \theta_j \) respectively lead to class probabilities \( p_i = \sigma(f_{\theta_i}(x)) \) and \( p_j = \sigma(f_{\theta_j}(x)) \). During training, \( p_i \) and \( p_j \) are naturally encouraged to be close to the assignment vector \( e_y \) in \( \{0, 1\}^d \) with a single non-zero entry at position \( y \), where \( y \) is the class label associated to \( x \) and \( d \) is the number of classes.

From [14], we know that even though only the largest entry of \( p_i \) or \( p_j \) is used to make predictions, other entries—typically not corresponding to the ground truth label \( y \)—carry important information about the network. It becomes then natural to consider the probabilities \( \hat{p}_i \) and \( \hat{p}_j \) conditioned on not being the ground truth label. Formally, these are obtained by setting to zero the entry \( y \) in \( p_i \) and \( p_j \) re-normalizing the corresponding vectors such that they sum to one. Then, we consider the following diversity penalty

\[
\phi(\hat{p}_i, \hat{p}_j) = \cos(\hat{p}_i, \hat{p}_j).
\]

When combined with the loss function, the resulting formulation encourages the networks to make the right prediction according to the ground-truth label, but then they are also encouraged to make different second-best, third-best, and so on, choice predictions (see Figure 1). This penalty turns out to be particularly effective when the number of networks is large, as shown in the experimental section. It typically worsens the performance of individual classifiers on average, but make the ensemble prediction more accurate.

**Cooperation.** Apparently opposite to the previous principle, encouraging the conditional probabilities \( \hat{p}_i \) to be similar—though with a different metric—may also improve the quality of prediction by allowing the networks to cooperate for better learning. Our experiments show that such a principle alone may be effective, but it appears to be mostly useful when the number of training networks is small, which suggests that there is a trade-off between cooperation and diversity that needs to be found.

Specifically, our experiments show that using the negative cosine—in other words, the opposite of (6)—is ineffective. However, a penalty such as the symmetrized KL-divergence turned out to provide the desired effect:

\[
\phi(\hat{p}_i, \hat{p}_j) = \frac{1}{2} (\text{KL}(\hat{p}_i || \hat{p}_j) + \text{KL}(\hat{p}_j || \hat{p}_i)).
\]

By using this penalty, we managed to obtain more stable and faster training, resulting in better performing individual networks, but also—perhaps surprisingly—a better ensemble. Unfortunately, we also observed that the gain of ensembling diminishes with the number of networks in the ensemble since the individual members become too similar.

**Robustness and cooperation.** Given experiments conducted with the two previous penalties, a trade-off between cooperation and diversity seems to correspond to two regimes (low vs. high number of networks). This motivated us to develop an approach designed to achieve the best trade-off. When considering the cooperation penalty (7), we try to increase diversity of prediction by several additional means. i) We randomly drop some networks from the ensemble at each training iteration, which causes the networks to learn on different data streams and reduces the speed of knowledge propagation. ii) We introduce Dropout within each network to increase randomization. iii) We feed each network with a different (crop, color) transformation of the same image, which makes the ensemble more robust to input image transformations. Overall, this strategy was found to perform best in most scenarios (see Figure 2).

### 3.4. Ensemble distillation

As most ensemble methods, our ensemble strategy introduces a significant computational overhead at training time. To remove the overhead at test time, we use a variant of knowledge distillation [13] to compress the ensemble into a single network \( f_w \). Given the meta-training dataset \( D_b \), we consider the following cost function on example \( (x, y) \):

\[
\ell(x, y) = (1 - \alpha) \cdot \hat{\ell}(e_y, \sigma(f_w(x)))
- \alpha \cdot T^2 \cdot \hat{\ell} \left( \frac{1}{K} \sum_{k=1}^{K} \sigma \left( \frac{f_k(x)}{T} \right), \sigma \left( \frac{f_w(x)}{T} \right) \right),
\]

where, \( \hat{\ell} \) is cross-entropy, \( e_y \) is a one-hot embedding of the true label \( y \). The second term performs distillation with parameter \( T \) (see [14]). It encourages the single model \( f_w \) to
be similar to the average output of the ensemble. In our experiments, we are able to obtain a model with performance relatively close to that of the ensemble (see Section 4).

Modeling out-of-distribution behavior. When distillation is performed on the dataset $D_b$, the network $f_w$ mimics the behavior of the ensemble on a specific data distribution. However, new categories are introduced at test time. Therefore, we also tried distillation by using additional unannotated data, which yields slightly better performance.

4. Experiments

We now present experiments to study the effect of cooperation and diversity for ensemble methods, and start with experimental and implementation details.

4.1. Experimental Setup

Datasets. We use mini-ImageNet [24] and tiered-ImageNet [25] which are derived from the original ImageNet [28] dataset and Caltech-UCSD Birds (CUB) 200-2011 [34]. Mini-ImageNet consists of 100 categories—64 for training, 16 for validation and 20 for testing—with 600 images each. Tiered-ImageNet is also a subset of ImageNet that includes 351 class for training, 97 for validation and 160 for testing which is 779,165 images in total. The splits are chosen such that the training classes are sufficiently different from the test ones, unlike in mini-ImageNet. The CUB dataset consists of 11,788 images of birds of more than 200 species. We adopt train, val, and test splits from [35], which were originally created by randomly splitting all 200 species in 100 for training, 50 for validation, and 50 for testing.

Evaluation. In few-shot classification, the test set is used to sample $N$ 5-way classification problems, where only $k$ examples of each category are provided for training and 15 for evaluation. We follow [9, 11, 22, 23, 24] and test our algorithms for $k = 1$ and 5 and $N$ is set to 1,000. Each time, classes and corresponding train/test examples are sampled at random. For all our experiments we report the mean accuracy (in %) over 1,000 tasks and 95% confidence interval.

Implementation details. For all experiments, we use the Adam optimizer [15] with an initial learning rate $10^{-4}$, which is decreased by a factor 10 once during training when no improvement in validation accuracy is observed for $p$ consecutive epochs. For mini-ImageNet, we use $p = 10$, and 20 for the CUB dataset. When distilling an ensemble into one network, $p$ is doubled. We use random crops and color augmentation during training as well as weight decay with parameter $\lambda = 5 \cdot 10^{-4}$. All experiments are conducted with the ResNet18 architecture [13], which allows us to train our ensembles of 20 networks on a single GPU. Input images are then re-scaled to the size 224 $\times$ 224, and organized in mini-batches of size 16. Validation accuracy is computed by running 5-shot evaluation on the validation set. During the meta-testing stage, we take central crops of size 224 $\times$ 224 from images and feed them to the feature extractor. No other preprocessing is used at test time. When building a mean centroid classifier, the distance $d$ in (3) is computed as the negative cosine similarity [31], which is rescaled by a factor 10.

For a fair comparison, we have also evaluated ensembles composed of ResNet18 [13] with input image size 84 $\times$ 84 and WideResNet28 [37] with input size 80 $\times$ 80. All details are reported in Appendix. For reproducibility purposes, our implementation will be made available at [http://lear.inrialpes.fr/research/fewshot_ensemble/](http://lear.inrialpes.fr/research/fewshot_ensemble/)
## 4.2. Ensembles for Few-Shot Classification

In this section, we study the effect of ensemble training with pairwise interaction terms that encourage cooperation or diversity. For that purpose, we analyze the link between the size of ensembles and their 1- and 5-shot classification performance on the mini-ImageNet and CUB datasets.

### Details about the three strategies.

When models are trained jointly, the data stream is shared across all networks and weight updates happen simultaneously. This is achieved by placing all models on the same GPU and optimizing the loss (7). When training a diverse ensemble, we use the cosine function (6) and selected the parameter $\gamma = 1$ that performed best on the validation set among the tested values ($10^i$, for $i = −2, \ldots, 2$) for $n = 5$ and $n = 10$ networks. Then, this value was kept for other values of $n$. To enforce cooperation between networks, we use the symmetrized KL function (7) and selected the parameter $\gamma = 10$ in the same manner. Finally, the robust ensemble strategy is trained with the cooperation relation penalty and the same parameter $\gamma$, but we use Dropout with probability 0.1 before the last layer; each of the network is dropped from the ensemble with probability 0.2 at every iteration; different networks receive different transformation of the same image, i.e. different random crops and color augmentation.

### Results.

Tables 1 and Table A1 of Appendix summarize the few-shot classification accuracies for ensembles trained with our strategies and compare with basic ensembles. On the mini-ImageNet dataset, the results for 1- and 5-shot classification are consistent with each other. Training with cooperation allows smaller ensembles ($n \leq 5$) to perform better, which leads to higher individual accuracy of the ensemble members, as seen in Figure 2. However, when $n \geq 10$, cooperation is less effective, as opposed to the diversity strategy, which benefits from larger $n$. As we can see from Figure 2, individual members of the ensemble become worse, but the ensemble accuracy improves substantially. Finally, the robust strategy seems to perform best for all values of $n$ in almost all settings. The situation for the CUB dataset is similar, although we notice that robust ensembles perform similarly as the diversity strategy for $n = 20$.

## 4.3. Distilling an ensemble

We distill robust ensembles of all sizes to study knowledge transferability with growing ensemble size. To do so, we use the meta-training dataset and optimize the loss (8) with parameters $T = 10$ and $\alpha = 0.8$. For the strategy using external data, we randomly add at each iteration 8 images (without annotations) from the COCO [18] dataset to the 16 annotated samples from the meta-training data. Those images contribute only to the distillation part of the loss (6).

Table 1 and Table A1 of Appendix display model accuracies for mini-ImageNet and CUB datasets respectively. For 5-shot classification on mini-ImageNet, the difference between ensemble and its distilled version is rather low (around 1%), while adding extra non-annotated data helps reducing this gap. Surprisingly, 1-shot classification accuracy is slightly higher for distilled models than for their corresponding full ensembles. On the CUB dataset, distilled models stop improving after $n = 5$, even though the performance of full ensembles keeps growing. This seems to indicate that the capacity of the single network may have been reached, which suggests using a more complex architecture here. Consistently with such hypothesis, adding extra data is not as helpful as for mini-ImageNet, most likely because data distributions of COCO and CUB are more different.

In Tables 2, 3, 5 we also compare the performance of our distilled networks with other baselines from the literature, including current state-of-the-art meta-learning approaches, showing that our approach does significantly better on the mini-ImageNet [24] and tiered-ImageNet [25] datasets.

## 4.4. Study of relationship penalties

There are many possible ways to model relationship between the members of an ensemble. In this subsection, we study and discuss such particular choices.

### Input to relationship function.

As noted by [14], class probabilities obtained by the softmax layer of a network seem to carry a lot of information and are useful for distillation. However, after meta-training, such probabilities are often close to binary vectors with a dominant value associated to the ground-truth label. To make small values more noticeable, distillation uses a parameter $T$, as in (8). Given such a class probability computed by a network, we experimented such a strategy consisting of introducing new probabilities $\tilde{p} = \sigma(p/T)$, where the contributions of non ground-truth values are emphasized. When used within our diversity (6) or cooperation (7) penalties, we however did not see any improvement over the basic ensemble method. Instead, we found that computing the class probabilities conditioned on not being the ground truth label, as explained in Section 3.3, would perform much better.

This is illustrated on the following experiment with two network ensembles of size $n = 5$. We enforce similarity on the full probability vectors in the first one, computed with softmax at $T = 10$ following (11), and with conditionally non-ground-truth probabilities for the second one as defined in Section 3.3. When using the cooperation training formulation, the second strategy turns out to perform about 1% better than the first one (79.79% vs 80.60%), when tested on MiniImageNet. Similar observations have been made using the diversity criterion. In comparison, the basic ensemble method without interactions achieves about 80%.
In principle, any similarity measure could be used to design a penalty encouraging cooperation. Here, we show that in fact, selecting the right criterion for comparing probability vectors (cosine similarity, L2 distance, symmetrized KL divergence), is crucial depending on the desired effect (cooperation or diversity). In Table 2, we perform such a comparison for an ensemble with \( n = 5 \) networks on the MiniImageNet dataset for a 5-shot classification task, when plugging the above function in the formulation [5] with a specific sign. The parameter \( \gamma \) for each experiment is chosen such that the performance on the validation set is maximized.

When looking for diversity, the cosine similarity performs slightly better than negative L2 distance, although the
Table 4: Evaluating different relationship criteria on mini-Imagenet 5-shot. The first row indicates which function was used as a relationship criteria, the first column indicates for which purpose the function is used and the corresponding sign. To evaluate our methods, we performed 1,000 independent experiments on CUB-test and report the average accuracy with 95% confidence intervals. All ensembles are trained on mini-ImageNet-train.

<table>
<thead>
<tr>
<th>Purpose (Sign)</th>
<th>L2</th>
<th>-cos</th>
<th>KL_sim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperation (+)</td>
<td>80.14 ± 0.43</td>
<td>80.29 ± 0.44</td>
<td>80.72 ± 0.42</td>
</tr>
<tr>
<td>Diversity (-)</td>
<td>80.54 ± 0.44</td>
<td>80.82 ± 0.42</td>
<td>79.81 ± 0.43</td>
</tr>
</tbody>
</table>

The overall performance of a single network obtained by distillation (with no computational overhead at test time) leads to state-of-the-art performance for few shot learning, without relying on the meta-learning paradigm. While such a result may sound negative for meta-learning approaches, it may simply mean that a lot of work remains to be done in this area to truly learn how to learn or to adapt.

Acknowledgment

This work was supported by the ERC grant number 714381 (SOLARIS project), the ERC advanced grant AL-LEGRO and grants from Amazon and Intel.

References


Appendix

A. Implementation Details

In this section, we elaborate on training, testing and distillation details of the proposed ensemble methods for different datasets, network architectures and input image resolution.

Training ResNet18 on 84x84 images on mini-ImageNet. For all experiments, we use ResNet18 with input image size 84x84, train with the Adam optimizer with an initial learning rate $3 \cdot 10^{-4}$, which is decreased by a factor 10 once during training when no improvement in validation accuracy is observed for $p$ consecutive epochs. We use $p = 20$ for training individual models, $p = 30$ for training ensembles and when distilling the model. When distilling an ensemble into one network, $p$ is doubled. We use random crops and color augmentation during training as well as weight decay with parameter $\lambda = 5 \cdot 10^{-4}$. At training time we use random crop, color transformation and adding random noise as data augmentation. During the meta-testing stage, we take central crops of size $224 \times 224$ from images and feed them to the feature extractor. No other preprocessing is used at test time. The parameters used in distillation are the same as in Section 4.3 of the paper.

Training WideResNet28 on 80x80 images on mini-ImageNet. For all experiments, we use WideResNet28 with input image size 80x80, train with the Adam optimizer with an initial learning rate $1 \cdot 10^{-4}$, which is decreased by a factor 10 once during training when no improvement in validation accuracy is observed for $p$ consecutive epochs. We use $p = 20$ for training individual models, $p = 30$ for training ensembles and when distilling the model. When distilling an ensemble into one network, $p$ is doubled. We use random crops and color augmentation during training as well as weight decay with parameter $\lambda = 5 \cdot 10^{-4}$. At training time we use random crop, color transformation and adding random noise as data augmentation. During the meta-testing stage, we take central crops of size $80 \times 80$ from images and feed them to the feature extractor. No other preprocessing is used at test time. The parameters used in distillation are the same as in Section 4.3 of the paper.

Training ResNet18 on 224x224 images on tiered-ImageNet. For all experiments, we use ResNet18 with input image size 224x224, train with the Adam optimizer with an initial learning rate $3 \cdot 10^{-4}$, which is decreased by a factor 10 once during training when no improvement in validation accuracy is observed for $p$ consecutive epochs. We use $p = 20$ for training individual models, ensembles and for distillation. We use random crops and color augmentation during training as well as weight decay with parameter $\lambda = 1 \cdot 10^{-4}$. At training time we use random crop and color transformation. During the meta-testing stage, we take central crops of size $224 \times 224$ from images and feed them to the feature extractor. No other preprocessing is used at test time. The parameters used in distillation are the same as in Section 4.3 of the paper.

B. Additional Results

In this section we report and analyze the performance of different ensemble types depending on their size for different network architectures and input image resolutions.

Few-shot Classification with ResNet18 on 224x224 images on CUB. The results for 1- and 5-shot classification on CUB are presented in Table A1. Training details and Figure summary of the results are discussed in Experimental section of the paper.

Few-shot Classification with ResNet18 on 84x84 images on mini-ImageNet. The results for 1- and 5-shot classification on MiniImageNet are presented in Table A3 and summarized in Figure A1. We can see that Cooperation training is the most successful here for all ensemble sizes $< 20$ and other training strategies that introduce diversity tend to perform worse. This happens because single networks are far from overfitting the training set (as opposed to the case with 224x224 input size) and forcing diversity acts as harmful regularization. In contrary, cooperation training enforces useful learning signal and helps ensemble members achieve higher accuracy. Only for $n = 20$ where diversity matters more, robust ensembles perform the best.

Few-shot Classification with WideResNet28 on 80x80 images on mini-ImageNet. Results for 1- and 5-shot classification on MiniImageNet are presented in Table A2 and summarized in Figure A1. In this case we can see again that Diverse training does not help since the networks do not memorize the training set. Robust ensembles outperform other training regimes emphasizing the importance of the proposed solution that generalizes across architectures.
### Few-shot classification accuracy on CUB

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5-shot</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full Ensemble</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Independent</td>
<td>79.47±0.49</td>
<td>81.34±0.46</td>
<td>82.57±0.46</td>
<td>83.16±0.45</td>
<td>83.80±0.45</td>
<td>83.95±0.46</td>
</tr>
<tr>
<td>Diversity</td>
<td>79.47±0.49</td>
<td>81.09±0.45</td>
<td>82.23±0.46</td>
<td>82.91±0.46</td>
<td>84.30±0.44</td>
<td>85.20±0.43</td>
</tr>
<tr>
<td>Cooperation</td>
<td>79.47±0.49</td>
<td>81.69±0.46</td>
<td>82.95±0.47</td>
<td>83.43±0.47</td>
<td>84.01±0.44</td>
<td>84.26±0.44</td>
</tr>
<tr>
<td>Robust</td>
<td>79.47±0.49</td>
<td>82.90±0.46</td>
<td>83.36±0.46</td>
<td>83.62±0.45</td>
<td>84.47±0.46</td>
<td>84.62±0.44</td>
</tr>
<tr>
<td><strong>Distilled Ensembles</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robust-dist</td>
<td>−</td>
<td>82.72±0.47</td>
<td>82.95±0.46</td>
<td>83.27±0.46</td>
<td>83.61±0.46</td>
<td>83.57±0.45</td>
</tr>
<tr>
<td>Robust-dist++</td>
<td>−</td>
<td>82.53±0.48</td>
<td>83.04±0.45</td>
<td>83.37±0.46</td>
<td>83.22±0.46</td>
<td>83.21±0.44</td>
</tr>
<tr>
<td><strong>1-shot</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ensemble type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Independent</td>
<td>64.25±0.73</td>
<td>66.60±0.72</td>
<td>67.64±0.71</td>
<td>68.07±0.70</td>
<td>68.93±0.70</td>
<td>69.64±0.69</td>
</tr>
<tr>
<td>Diversity</td>
<td>64.25±0.73</td>
<td>65.99±0.71</td>
<td>66.71±0.72</td>
<td>68.19±0.71</td>
<td>69.35±0.70</td>
<td>70.07±0.70</td>
</tr>
<tr>
<td>Cooperation</td>
<td>64.25±0.73</td>
<td>67.21±0.71</td>
<td>67.93±0.70</td>
<td>68.22±0.70</td>
<td>68.69±0.70</td>
<td>68.80±0.68</td>
</tr>
<tr>
<td>Robust</td>
<td>64.25±0.73</td>
<td>67.33±0.71</td>
<td>68.01±0.72</td>
<td>68.53±0.70</td>
<td>68.59±0.70</td>
<td>69.47±0.69</td>
</tr>
<tr>
<td><strong>Distilled Ensembles</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robust-dist</td>
<td>−</td>
<td>67.47±0.71</td>
<td>67.29±0.72</td>
<td>68.09±0.70</td>
<td>68.71±0.71</td>
<td>68.77±0.71</td>
</tr>
<tr>
<td>Robust-dist++</td>
<td>−</td>
<td>67.01±0.74</td>
<td>67.62±0.72</td>
<td>68.68±0.71</td>
<td>68.38±0.70</td>
<td>68.68±0.69</td>
</tr>
</tbody>
</table>

Table A1: **Few-shot classification accuracy on CUB.** The first column gives the type of ensemble and the top row indicates the number of networks in an ensemble. Here, dist means that an ensemble was distilled into a single network, and ‘++’ indicates that extra unannotated images were used for distillation. We performed 1000 independent experiments on CUB-test and report the average with 95% confidence interval. All networks are trained on CUB-train set.

### Few-shot classification accuracy on MiniImageNet, using ResNet18 and 84x84 image size

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5-shot</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ensemble type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Independent</td>
<td>70.59±0.51</td>
<td>73.24±0.49</td>
<td>74.29±0.48</td>
<td>74.89±0.47</td>
<td>75.69±0.47</td>
<td>75.93±0.47</td>
</tr>
<tr>
<td>Diversity</td>
<td>70.59±0.51</td>
<td>72.35±0.47</td>
<td>73.44±0.49</td>
<td>74.81±0.48</td>
<td>75.47±0.48</td>
<td>76.36±0.47</td>
</tr>
<tr>
<td>Cooperation</td>
<td>70.59±0.51</td>
<td>74.04±0.47</td>
<td>74.81±0.47</td>
<td>76.37±0.48</td>
<td>76.73±0.48</td>
<td>76.50±0.47</td>
</tr>
<tr>
<td>Robust</td>
<td>70.59±0.51</td>
<td>72.92±0.50</td>
<td>73.09±0.43</td>
<td>75.69±0.42</td>
<td>76.71±0.47</td>
<td>76.90±0.48</td>
</tr>
<tr>
<td><strong>Distilled Ensembles</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robust-dist</td>
<td>−</td>
<td>73.04±0.50</td>
<td>73.58±0.49</td>
<td>74.35±0.48</td>
<td>74.69±0.49</td>
<td>75.24±0.49</td>
</tr>
<tr>
<td>Robust-dist++</td>
<td>−</td>
<td>73.50±0.49</td>
<td>74.17±0.49</td>
<td>74.84±0.49</td>
<td>75.12±0.44</td>
<td>75.62±0.48</td>
</tr>
<tr>
<td><strong>1-shot</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ensemble type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Independent</td>
<td>53.31±0.64</td>
<td>55.72±0.60</td>
<td>56.85±0.64</td>
<td>57.90±0.63</td>
<td>58.21±0.63</td>
<td>58.56±0.61</td>
</tr>
<tr>
<td>Diversity</td>
<td>53.31±0.64</td>
<td>54.61±0.62</td>
<td>55.90±0.62</td>
<td>57.06±0.63</td>
<td>57.49±0.62</td>
<td>58.93±0.64</td>
</tr>
<tr>
<td>Cooperation</td>
<td>53.31±0.64</td>
<td>55.80±0.64</td>
<td>57.13±0.63</td>
<td>58.18±0.64</td>
<td>58.63±0.63</td>
<td>58.73±0.62</td>
</tr>
<tr>
<td>Robust</td>
<td>53.31±0.64</td>
<td>55.95±0.62</td>
<td>56.27±0.64</td>
<td>58.51±0.65</td>
<td>59.38±0.65</td>
<td>59.48±0.65</td>
</tr>
<tr>
<td><strong>Distilled Ensembles</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robust-dist</td>
<td>−</td>
<td>56.84±0.64</td>
<td>56.58±0.65</td>
<td>57.13±0.63</td>
<td>57.41±0.65</td>
<td>58.11±0.64</td>
</tr>
<tr>
<td>Robust-dist++</td>
<td>−</td>
<td>56.53±0.62</td>
<td>57.03±0.64</td>
<td>57.48±0.65</td>
<td>58.05±0.63</td>
<td>58.67±0.65</td>
</tr>
</tbody>
</table>

Table A2: **Few-shot classification accuracy on MiniImageNet, using ResNet18 and 84x84 image size.** The first column gives the strategy, the top row indicates the number $N$ of networks in an ensemble. Here, dist means that an ensemble was distilled into a single network, and ‘++’ indicates that extra unannotated images were used for distillation. We performed 1000 independent experiments on MiniImageNet-test and report the average with 95% confidence interval. All networks are trained on MiniImageNet-train set.
<table>
<thead>
<tr>
<th>Ensemble type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>77.54 ± 0.45</td>
<td>78.78 ± 0.45</td>
<td>79.26 ± 0.43</td>
<td>79.91 ± 0.44</td>
<td>80.12 ± 0.43</td>
</tr>
<tr>
<td>Diversity</td>
<td>77.54 ± 0.45</td>
<td>77.88 ± 0.45</td>
<td>79.15 ± 0.44</td>
<td>79.79 ± 0.44</td>
<td>80.18 ± 0.44</td>
</tr>
<tr>
<td>Cooperation</td>
<td>77.54 ± 0.45</td>
<td>78.96 ± 0.46</td>
<td>80.06 ± 0.44</td>
<td>80.58 ± 0.45</td>
<td>80.87 ± 0.43</td>
</tr>
<tr>
<td>Robust</td>
<td>77.54 ± 0.45</td>
<td>78.99 ± 0.45</td>
<td>80.12 ± 0.43</td>
<td>80.91 ± 0.43</td>
<td>81.72 ± 0.44</td>
</tr>
</tbody>
</table>

**Distilled Ensembles**

<table>
<thead>
<tr>
<th>Ensemble type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robust-dist</td>
<td>-</td>
<td>79.44 ± 0.44</td>
<td>79.84 ± 0.44</td>
<td>80.01 ± 0.42</td>
<td>80.85 ± 0.43</td>
</tr>
<tr>
<td>Robust-dist++</td>
<td>-</td>
<td>79.16 ± 0.46</td>
<td>80.00 ± 0.44</td>
<td>80.25 ± 0.42</td>
<td>81.11 ± 0.43</td>
</tr>
</tbody>
</table>

**1-shot**

<table>
<thead>
<tr>
<th>Ensemble type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>59.02 ± 0.63</td>
<td>60.07 ± 0.62</td>
<td>60.58 ± 0.61</td>
<td>61.24 ± 0.63</td>
<td>62.05 ± 0.61</td>
</tr>
<tr>
<td>Diversity</td>
<td>59.02 ± 0.63</td>
<td>58.87 ± 0.62</td>
<td>60.63 ± 0.61</td>
<td>61.30 ± 0.62</td>
<td>62.28 ± 0.61</td>
</tr>
<tr>
<td>Cooperation</td>
<td>59.02 ± 0.63</td>
<td>60.22 ± 0.62</td>
<td>61.03 ± 0.61</td>
<td>62.07 ± 0.61</td>
<td>62.42 ± 0.61</td>
</tr>
<tr>
<td>Robust</td>
<td>59.02 ± 0.63</td>
<td>60.92 ± 0.62</td>
<td>62.03 ± 0.62</td>
<td>62.78 ± 0.61</td>
<td>63.39 ± 0.62</td>
</tr>
</tbody>
</table>

**Distilled Ensembles**

<table>
<thead>
<tr>
<th>Ensemble type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robust-dist</td>
<td>-</td>
<td>61.07 ± 0.62</td>
<td>61.57 ± 0.61</td>
<td>62.24 ± 0.61</td>
<td>62.80 ± 0.62</td>
</tr>
<tr>
<td>Robust-dist++</td>
<td>-</td>
<td>61.37 ± 0.62</td>
<td>62.01 ± 0.60</td>
<td>62.45 ± 0.62</td>
<td>63.25 ± 0.62</td>
</tr>
</tbody>
</table>

Table A3: Few-shot classification accuracy on *Mini*ImageNet, using WideResNet28 and 80x80 image size. The first column gives the strategy, the top row indicates the number $N$ of networks in an ensemble. Here, dist means that an ensemble was distilled into a single network, and ‘++’ indicates that extra unannotated images were used for distillation. We performed 1 000 independent experiments on *Mini*ImageNet-test and report the average with 95% confidence interval. All networks are trained on *Mini*ImageNet-train set.

Figure A1: Dependency of ensemble accuracy on network architecture and input size for different ensemble strategies (one for each color) and various numbers of networks on *Mini*ImageNet 5-shots classification. Solid lines give the ensemble accuracy after aggregating predictions. The average performance of single models from the ensemble is plotted with a dashed line. Best viewed in color.