Boosting Few-Shot Visual Learning with Self-Supervision

Spyros Gidaris[1], Andrei Bursuc[1], Nikos Komodakis[2], Patrick Pérez[1], Matthieu Cord[1,3]
[1]valeo.ai [2]LIGM, Ecole des Pont ParisTech [3]Sorbonne Université

Abstract

Few-shot learning and self-supervised learning address different facets of the same problem: how to train a model with little or no labeled data. Few-shot learning aims for optimization methods and models that can learn efficiently to recognize patterns in the low data regime. Self-supervised learning focuses instead on unlabeled data and looks into it for the supervisory signal to feed high capacity deep neural networks. In this work we exploit the complementarity of these two domains and propose an approach for improving few-shot learning through self-supervision. We use self-supervision as an auxiliary task in a few-shot learning pipeline, enabling feature extractors to learn richer and more transferable visual representations while still using few annotated samples. Through self-supervision, our approach can be naturally extended towards using diverse unlabeled data from other datasets in the few-shot setting. We report consistent improvements across an array of architectures, datasets and self-supervision techniques.

1. Introduction

Deep learning based methods have achieved impressive results on various image understanding tasks, such as image classification [21, 29, 51, 53], object detection [46], or semantic segmentation [4]. However, in order to successfully learn these tasks, such models need to access large volumes of manually labeled training data. If not, the trained models might suffer from poor generalization performance on the test data. In image classification for instance, learning to recognize reliably a set of categories with convolutional neural networks (convnets) requires hundreds or thousands of training examples per class. In contrast, humans are perfectly capable of learning new visual concepts from only one or few examples, generalizing without difficulty to new data. The aim of *few-shot learning* [9, 10, 26, 30, 54] is to endow artificial perception systems with a similar ability, especially with the help of modern deep learning.

Hence, the goal of few-shot visual learning is to devise recognition models that are capable of efficiently learning to recognize a set of classes despite the fact that there are available very few training examples for them (*e.g.*, only 1 or 5 examples per class). In order to avoid overfitting due to data scarcity, few-shot learning algorithms rely on transfer learning techniques and have two learning stages. During a first stage, the model is usually trained using a different set of classes, called *base classes*, which is associated to a large set of annotated training examples. The goal of this stage is to let the few-shot model acquire transferable visual analysis abilities, typically in the form of learned representations, that are mobilized in the second stage. In this subsequent step, the model indeed learns to recognize *novel classes*, unseen during the first learning stage, using only a few training examples per class.

Few-shot learning relates with self-supervised representation learning [6, 7, 14, 31, 39, 60]. The latter is a form of unsupervised learning that trains a model on an annotation-free pretext task defined using only the visual information present in images. The purpose of this self-supervised task is to make the model learn image representations that would be useful when transferred to other image understanding tasks. For instance, in the seminal work of Doersch *et al*. [6], a network, by being trained on the self-supervised task of predicting the relative location of image patches, manages to learn image representations that are successfully transferred to the vision tasks of object recognition, object detection, and semantic segmentation. Therefore, as in few-shot learning, self-supervised methods also have two learning stages, the first that learns image representations with a pretext self-supervised task, and the second that adapts those representations to the actual task of interest. Also, both learning approaches try to limit the dependence of deep learning methods on the availability of large amounts of labeled data.

Inspired by the connection between few-shot learning and self-supervised learning, we propose to combine the two methods to improve the transfer learning abilities of few-shot models. More specifically, we propose to add a self-supervised loss to the training loss that a few-shot model minimizes during its first learning stage (see Figure 1). Hence, we artificially augment the training task(s) that a few-shot model solves during its first learning stage. We argue that this task augmentation forces the model to learn a more diversified set of image features, and this in

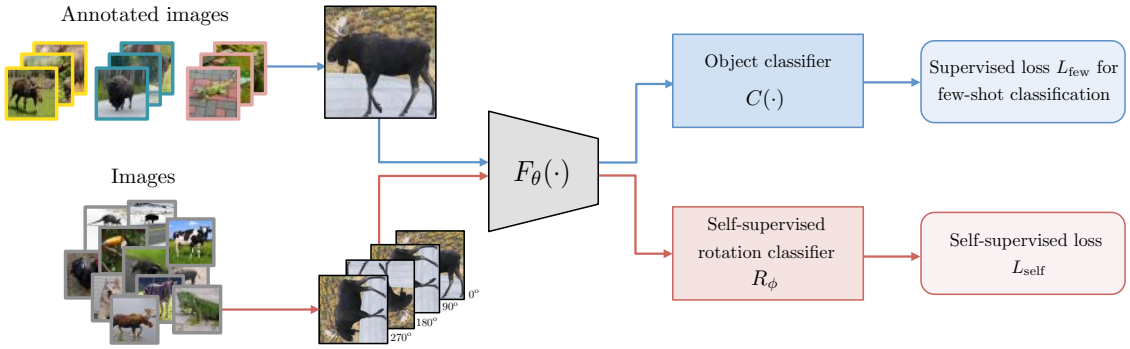

Figure 1: **Combining self-supervised image rotation prediction and supervised base class recognition in first learning stage of a few-shot system.** We train the feature extractor $F_\theta(\cdot)$ with both annotated (top branch) and non-annotated (bottom branch) data in a multi-task setting. We use the annotated data to train the object classifier $C(\cdot)$ with the few-shot classification loss L_{few}. For the self-supervised task, we sample images from the annotated set (and optionally from a different set of non-annotated images). Here, we generate four rotations for each input image, process them with $F_\theta(\cdot)$ and train the rotation classifier R_ϕ with the self-supervised loss L_{self}. The pipeline for relative patch location self-supervision is analogue to this one.

turn improves its ability to adapt to novel classes with few training data. Moreover, since self-supervision does not require data annotations, one can include extra unlabeled data to the first learning stage. By extending the size and variety of training data in this manner, one might expect to learn richer image features and to get further performance gain in few-shot learning. At the extreme, using only unlabeled data in the first learning stage, thus removing the use of base classes altogether, is also appealing. We will show that both these semi-supervised and unsupervised regimes can be indeed put at work for few-shot recognition thanks to self-supervised tasks.

In summary, the contributions of our work are: **(1)** We propose to weave self-supervision into the training objective of few-shot learning algorithms. The goal is to boost the ability of the latter to adapt to novel classes with few training data. **(2)** We study the impact of the added self-supervised loss by performing exhaustive quantitative experiments on MiniImagenet, CIFAR-FS, and tiered-MiniImagenet few-shot datasets. In all of them self-supervision improves the few-shot learning performance leading to state-of-the-art results. **(3)** Finally, we extend the proposed few-shot recognition framework to semi-supervised and unsupervised setups, getting further performance gain in the former, and showing with the latter that our framework can be used for evaluating and comparing self-supervised representation learning approaches on few-shot object recognition.

2. Related work

Few-shot learning. There is a broad array of few-shot learning approaches, including, among many: gradient descent-based approaches [1, 11, 38, 44], which learn how to rapidly adapt a model to a given few-shot recognition task via a small number of gradient descent iterations; metric learning based approaches that learn a distance metric between a query, *i.e.*, test image, and a set of support images, *i.e.*, training images, of a few-shot task [26, 52, 54, 56, 58]; methods learning to map a test example to a class label by accessing memory modules that store training examples for that task [12, 25, 34, 37, 49]; approaches that learn how to generate the weights of a classifier [13, 16, 42, 43] or of a multi-layer neural network [3, 18, 19, 57] for the new classes given the few available training data for each of them; methods that "hallucinate" additional examples of a class from a reduced amount of data [20, 56].

In our work we consider two approaches from the metric learning category, namely *Prototypical Networks* [52] and *Cosine Classifiers* [13, 42] for their simplicity and flexibility. Nevertheless, the proposed auxiliary self-supervision is compatible with several other few-shot classification solutions.

Self-supervised learning. This is a recent paradigm which is mid-way between unsupervised and supervised learning, and aims to mitigate the challenging need for large amounts of annotated data. Self-supervised learning defines an annotation-free pretext task, in order to provide a surrogate supervision signal for feature learning. Predicting the colors of images [31, 60], the relative position of image patches [6, 39], the random rotation that has been applied to an image [14], or the missing central part of an image [41], are some of the many methods [15, 32, 35, 55, 61] for self-supervised feature learning. The intuition is that, by solving such tasks, the trained model extracts semantic features that can be useful for other downstream tasks. In our case, we consider a multi-task setting where we train the backbone convnet using joint supervision from the supervised end-task and an auxiliary self-supervised pretext task. Unlike most multi-task settings aiming at good results on all tasks simultaneously [27], we are interested in improving performance on the main task only by leveraging supervision from the surrogate task, as also shown in [36]. We expect that, in a

few-shot setting where squeezing out generalizable features from the available data is highly important, the use of self-supervision as an auxiliary task will bring improvements. Also, related to our work, Chen *et al.* [5] recently added rotation prediction self-supervision to generative adversarial networks [17] leading to significant quality improvements of the synthesized images.

3. Methodology

As already explained, few-shot learning algorithms have two learning stages and two corresponding sets of classes. Here, we define as $D_b = \{(\mathbf{x}, y)\} \subset I \times Y_b$ the training set of *base classes* used during the first learning stage, where $\mathbf{x} \in I$ is an image with label y in label set Y_b of size N_b. Also, we define as $D_n = \{(\mathbf{x}, y)\} \subset I \times Y_n$ the training set of N_n *novel classes* used during the second learning stage, where each class has K samples ($K = 1$ or 5 in benchmarks). One talks about N_n-way K-shot learning. Importantly, the label sets Y_n and Y_b are disjoint.

In the remainder of this section, we first describe in §3.1 the two standard few-shot learning methods that we consider and introduce in §3.2 the proposed method to boost their performance with self-supervision.

3.1. Explored few-shot learning methods

The main component of all few-shot algorithms is a feature extractor $F_\theta(\cdot)$, which is a convnet with parameters θ. Given an image $\mathbf{x}$, the feature extractor will output a d-dimensional feature $F_\theta(\mathbf{x})$. In this work we consider two few-shot algorithms, Prototypical Networks (PN) [52] and Cosine Classifiers (CC) [13, 42], described below. They are fairly similar, with their main difference lying in the first learning stage: only CC learns actual base classifiers along with the feature extractor, while PN simply relies on class-level averages.

Prototypical Networks (PN) [52]. During the first stage of this approach, the feature extractor $F_\theta(\cdot)$ is learned on sampled few-shot classification sub-problems that are analogue to the targeted few-shot classification problem. In each training episode of this learning stage, a subset $Y_* \subset Y_b$ of N_* base classes are sampled (they are called "support classes") and, for each of them, K training examples are randomly picked from within D_b. This yields a training set D_*. Given current feature extractor F_θ, the average feature for each class $j \in Y_*$, its "prototype", is computed as

$$\mathbf{p}_j = \frac{1}{K} \sum_{\mathbf{x} \in X_*^j} F_\theta(\mathbf{x}), \text{ with } X_*^j = \{\, \mathbf{x} \mid (\mathbf{x}, y) \in D_*,\, y = j \,\} \tag{1}$$

and used to build a simple similarity-based classifier. Then, given a new image $\mathbf{x}_q$ from a support class but different from samples in D_*, the classifier outputs for each class j the normalized classification score

$$C^j(F_\theta(\mathbf{x}_q); D_*) = \text{softmax}_j \Big[\text{sim}\big(F_\theta(\mathbf{x}_q), \mathbf{p}_i\big)_{i \in Y_*} \Big], \tag{2}$$

where $\text{sim}(\cdot, \cdot)$ is a similarity function, which may be cosine similarity or negative squared Euclidean distance. So, in practice, the image $\mathbf{x}_q$ will be classified to its closest prototype. Note that the classifier is conditioned on D_* in order to compute the class prototypes. The first learning stage finally amounts to iteratively minimizing the following loss w.r.t. θ:

$$L_{\text{few}}(\theta; D_b) = \mathop{\mathbb{E}}_{\substack{D_* \sim D_b \\ (\mathbf{x}_q,\, y_q)}} \Big[-\log C^{y_q}(F_\theta(\mathbf{x}_q); D_*) \Big], \tag{3}$$

where $(\mathbf{x}_q, y_q)$ is a training sample from a support class defined in D_* but different from images in D_*.

In the second learning stage, the feature extractor F_θ is frozen and the classifier of novel classes is simply defined as $C(\cdot; D_n)$, with prototypes defined as in (1) with $D_* = D_n$.

Cosine Classifiers (CC) [13, 42]. In CC few-shot learning, the first stage trains the feature extractor F_θ together with a cosine-similarity based classifier on the (standard) supervised task of classifying the base classes. Denoting $W_b = [\mathbf{w}_1, ..., \mathbf{w}_{N_b}]$ the matrix of the d-dimensional classification weight vectors, the normalized score for an input image $\mathbf{x}$ reads

$$C^j(F_\theta(\mathbf{x}); W_b) = \text{softmax}_j \Big[\gamma \cos\big(F_\theta(\mathbf{x}), \mathbf{w}_i\big)_{i \in Y_b} \Big], \tag{4}$$

where $\cos(\cdot, \cdot)$ is the cosine operation between two vectors, and the scalar γ is the inverse temperature parameter of the softmax operator.[1]

The first learning stage aims at minimizing w.r.t. θ and W_b the negative log-likelihood loss:

$$L_{\text{few}}(\theta, W_b; D_b) = \mathop{\mathbb{E}}_{(\mathbf{x}, y) \sim D_b} \big[-\log C^y(F_\theta(\mathbf{x}); W_b) \big]. \tag{5}$$

One of the reasons for using the cosine-similarity based classifier instead of the standard dot-product based one, is that the former learns feature extractors that reduce intra-class variations and thus can generalize better on novel classes. By analogy with PN, the weight vectors $\mathbf{w}_j$'s can be interpreted as learned prototypes for the base classes, to which input image features are compared for classification.

As with PN, the second stage boils down to computing one representative feature $\mathbf{w}_j$ for each new class by simple averaging of associated K samples in D_n, and to define the final classifier $C(.; [\mathbf{w}_1 \cdots \mathbf{w}_{N_n}])$ the same way as in (4).

[1] Specifically, γ controls the peakiness of the probability distribution generated by the softmax operator [22].

3.2. Boosting few-shot learning via self-supervision

A major challenge in few-shot learning is encountered during the first stage of learning. How to make the feature extractor learn image features that can be readily exploited for novel classes with few training data during the second stage? With this goal in mind, we propose to leverage the recent progress in self-supervised feature learning to further improve current few-shot learning approaches.

Through solving a non-trivial pretext task that can be trivially supervised, such as recovering the colors of images from their intensities, a network is encouraged to learn rich and generic image features that are transferable to other downstream tasks such as image classification. In the first stage of few-shot learning, we propose to extend the training of the feature extractor $F_\theta(.)$ by including such a self-supervised task besides the main task of recognizing base classes.

We consider two ways for incorporating self-supervision into few-shot learning algorithms: (1) by using an auxiliary loss function based on a self-supervised task, and (2) by exploiting unlabeled data in a semi-supervised way during training. In the following we will describe the two techniques.

3.2.1 Auxiliary loss based on self-supervision

We incorporate self-supervision to a few-shot learning algorithm by adding an auxiliary self-supervised loss during its first learning stage. More formally, let $L_{\text{self}}(\theta, \phi; X_b)$ be the self-supervised loss applied to the set $X_b = \{\, \mathbf{x} \mid (\mathbf{x}, y) \in D_b \,\}$ of training examples in D_b deprived of their class labels. The loss $L_{\text{self}}(\theta, \phi; X_b)$ is a function of the parameters θ of the feature extractor and of the parameters ϕ of a network only dedicated to the self-supervised task. The first training stage of few-shot learning now reads

$$\min_{\theta, [W_b], \phi} L_{\text{few}}(\theta, [W_b]; D_b) + \alpha L_{\text{self}}(\theta, \phi; X_b) \,, \quad (6)$$

where L_{few} stands either for the PN few-shot loss (3) or for the CC one (5), with additional argument W_b in the latter case (hence bracket notation). The positive hyper-parameter α controls the importance of the self-supervised term[2]. An illustration of the approach is provided in Figure 1.

For the self-supervised loss, we consider two tasks in the present work: predicting the rotation incurred by an image, [14], which is simple and readily incorporated into a few-shot learning algorithm; predicting the relative location of two patches from the same image [6], a seminal task in self-supervised learning. In a recent study, both methods have been shown to achieve state-of-the-art results [28].

[2]In our experiments, we use $\alpha = 1.0$.

Image rotations. In this task, the convnet must recognize among four possible 2D rotations in $\mathcal{R} = \{0°, 90°, 180°, 270°\}$ the one applied to an image (see Figure 1). Specifically, given an image $\mathbf{x}$, we first create its four rotated copies $\{\, \mathbf{x}^r \mid r \in \mathcal{R} \,\}$, where $\mathbf{x}^r$ is the image $\mathbf{x}$ rotated by r degrees. Based on the features $F_\theta(\mathbf{x}^r)$ extracted from such a rotated image, the new network R_ϕ attempts to predict the rotation class r. Accordingly, the self-supervised loss of this task is defined as:

$$L_{\text{self}}(\theta, \phi; X) = \mathop{\mathbb{E}}_{\mathbf{x} \sim X} \Big[\sum_{\forall r \in \mathcal{R}} - \log R^r_\phi\big(F_\theta(\mathbf{x}^r)\big) \Big] \,, \quad (7)$$

where X is the original training set of non-rotated images and $R^r_\phi(\cdot)$ is the predicted normalized score for rotation r. Intuitively, in order to do well for this task the model should reduce the bias towards *up-right* oriented images, typical for ImageNet-like datasets, and learn more diverse features to disentangle classes in the low-data regime.

Relative patch location. Here, we create random pairs of patches from an image and then predict, among eight possible positions, the location of the second patch w.r.t. to the first, *e.g.*, "on the left and above" or "on the right and below". Specifically, given an image $\mathbf{x}$, we first divide it into 9 regions over a 3×3 grid and sample a patch within each region. Let's denote $\bar{\mathbf{x}}^0$ the central image patch, and $\bar{\mathbf{x}}^1 \cdots \bar{\mathbf{x}}^8$ its eight neighbors lexicographically ordered. We compute the representation of each patch[3] and then generate patch feature pairs $\big(F_\theta(\bar{\mathbf{x}}^0), F_\theta(\bar{\mathbf{x}}^p)\big)$ by concatenation. We train a fully-connected network $P_\phi(\cdot, \cdot)$ to predict the position of $\bar{\mathbf{x}}^p$ from each pair.

The self-supervised loss of this task is defined as:

$$L_{\text{self}}(\theta, \phi; X) = \mathop{\mathbb{E}}_{\mathbf{x} \sim X} \Big[\sum_{p=1}^{8} - \log P^p_\phi\big(F_\theta(\bar{\mathbf{x}}^0), F_\theta(\bar{\mathbf{x}}^p)\big) \Big] \,, \quad (8)$$

where X is a set of images and P^p_ϕ is the predicted normalized score for the relative location p. Intuitively, a good model on this task should somewhat recognize objects and parts, even in presence of occlusions and background clutter. Note that, in order to prevent the model from learning low-level image statistics such as chromatic aberration [6], the patches are preprocessed with aggressive color augmentation (*i.e.*, converting to grayscale with probability 0.66 and normalizing the pixels of each patch individually to have zero mean and unit standard deviation).

3.2.2 Semi-supervised few-shot learning

The self-supervised term L_{self} in the training loss (6) does not depend on class labels. We can easily extend it to learn

[3]If the architecture of $F_\theta(\cdot)$ is fully convolutional, we can apply it to both big images and smaller patches.

as well from additional unlabeled data. Indeed, if a set X_u of unlabeled images is available besides D_b, we can make the self-supervised task benefit from them by redefining the first learning stage as:

$$\min_{\theta,[W_b],\phi} L_{\text{few}}(\theta,[W_b];D_b)+\alpha\cdot L_{\text{self}}(\theta,\phi;X_b\cup X_u)\ . \quad (9)$$

By training the feature extractor F_θ to also minimize the self-supervised loss on these extra unlabeled images, we open up its visual scope with the hope that this will further improve its ability to accommodate novel classes with scarce data. This can be seen as a semi-supervised training approach for few-shot algorithms. An interesting aspect of this semi-supervised training approach is that it does not require the extra unlabeled data to be from the same (base) classes as those in labeled dataset D_b. Thus, it is much more flexible w.r.t. the source of the unlabeled data than standard semi-supervised approaches.

4. Experimental Results

In this section we evaluate self-supervision as auxiliary loss function in §4.2 and in §4.3 as a way of exploiting unlabeled data in semi-supervised training. Finally, in §4.4 we use the few-shot object recognition task for evaluating self-supervised methods.

Datasets. We perform experiments on three few-shot datasets, MiniImageNet [54], *tiered*-MiniImageNet [45] and CIFAR-FS [2]. **MiniImageNet** consists of 100 classes randomly picked from the ImageNet dataset [47] (*i.e.*, 64 base classes, 16 validation classes, and 20 novel test classes); each class has 600 images with size 84×84 pixels. ***tiered*-MiniImageNet** consists of 608 classes randomly picked from ImageNet [47] (*i.e.*, 351 base classes, 97 validation classes, and 160 novel test classes); in total there are $779,165$ images again with size 84×84. Finally, **CIFAR-FS** is a few-shot dataset created by dividing the 100 classes of CIFAR-100 into 64 base classes, 16 validation classes, and 20 novel test classes. The images in this dataset have size 32×32 pixels.

Evaluation metrics. Few-shot classification algorithms are evaluated based on the classification performance in their second learning stage (when the learned classifier is applied to test images from the novel classes). More specifically, a large number of N_n-way K-shot tasks are sampled from the available set of novel classes. Each task is created by randomly selecting N_n novel classes from the available test (validation) classes and then within the selected images randomly selecting K training images and M test images per class (making sure that train and test images do not overlap). The classification performance is measured on the $N_n\times M$ test images and is averaged over all the sampled few-shot tasks. Except otherwise stated, for all experiments we used $M=15$, $N_n=5$, and $K=1$ or $K=5$ (1-shot and 5-shot settings respectively).

4.1. Implementation details

Network architectures. We conduct experiments with 3 different feature extractor architectures F_θ, Conv-4-64, Conv-4-512, and WRN-28-10. **Conv-4-64** [54] consists of 4 convolutional blocks each implemented with a 3×3 convolutional layer followed by BatchNorm [24] + ReLU + 2×2 max-pooling units [54]. All blocks of Conv-4-64 have 64 feature channels. The final feature map has size $5\times 5\times 64$ and is flattened into a final 1600-dimensional feature vector. **Conv-4-512** is derived from Conv-4-64 by gradually increasing its width across layers leading to 96, 128, 256, and 512 feature channels for its 4 convolutional blocks respectively. Therefore, its output feature vector has $5\times 5\times 512=12,800$ dimensions after flattening. Finally, **WRN-28-10** is a 28-layer Wide Residual Network [59] with width factor 10. Its output feature map has size $10\times 10\times 640$ which after global average pooling creates a 640-dimensional feature vector.

The network $R_\phi(\cdot)$ specific to the rotation prediction task gets as input the output feature maps of F_θ and is implemented as a convnet. Given two patches, the network $P_\phi(\cdot,\cdot)$ specific to the relative patch location task gets the concatenation of their feature vectors extracted with F_θ as input, and forwards it to two fully connected layers.

For further architecture details see Appendix B.1.

Training optimization routine for fist learning stage. The training loss is optimized with mini-batch stochastic gradient descent (SGD). For the labeled data we apply both recognition L_{few} and self-supervised L_{self} losses. For the semi-supervised training, at each step we sample mini-batches that consist of labeled data, for which we use both losses, and unlabeled data, in which case we apply only L_{self}. Learning rates, number of epochs, and batches sizes, were cross-validated on the validation sets. For further implementation details see Appendices B.2 and B.3.

Implementation of relative patch location task. Due to the aggressive color augmentation of the patches in the patch localization task, and the fact that the patches are around 9 times smaller than the original images, the data distribution that the feature extractor "sees" from them is very different from that of the images. To overcome this problem we apply an extra auxiliary classification loss to the features extracted from the patches. Specifically, during the first learning stage of CC we merge the features $F_\theta(\bar{\mathbf{x}}^p)$ of the 9 patches of an image (*e.g.*, with concatenation or averaging) and then apply the cosine classifier (4) to the resulting feature (this classifier does not share its weight vectors with the classifier

Model	Backbone	1-shot	5-shot
CC	Conv-4-64	54.31 ± 0.42%	70.89 ± 0.34%
CC+rot		**54.83** ± 0.43%	**71.86** ± 0.33%
CC	Conv-4-512	55.68 ± 0.43%	73.19 ± 0.33%
CC+rot		**56.27** ± 0.43%	**74.30** ± 0.33%
CC	WRN-28-10	61.09 ± 0.44%	78.43 ± 0.33%
CC+rot		**62.93** ± 0.45%	**79.87** ± 0.33%
PN	Conv-4-64	52.20 ± 0.46%	69.98 ± 0.36%
PN+rot		**53.63** ± 0.43%	**71.70** ± 0.36%
PN	Conv-4-512	54.60 ± 0.46%	71.59 ± 0.36%
PN+rot		**56.02** ± 0.46%	**74.00** ± 0.35%
PN	WRN-28-10	55.85 ± 0.48%	68.72 ± 0.36%
PN+rot		**58.28** ± 0.49%	**72.13** ± 0.38%

Table 1: Rotation prediction as auxiliary loss on MiniImageNet. Average 5-way classification accuracies for the novel classes on the test set of MiniImageNet with 95% confidence intervals (using 2000 test episodes).

Models	Backbone	1-shot	5-shot
CC	Conv-4-64	61.80 ± 0.30%	78.02 ± 0.24%
CC+rot		**63.45** ± 0.31%	**79.79** ± 0.24%
CC	Conv-4-512	65.26 ± 0.31%	81.14 ± 0.23%
CC+rot		**65.87** ± 0.30%	**81.92** ± 0.23%
CC	WRN-28-10	71.83 ± 0.31%	84.63 ± 0.23%
CC+rot		**73.62** ± 0.31%	**86.05** ± 0.22%
PN	Conv-4-64	62.82 ± 0.32%	79.59 ± 0.24%
PN+rot		**64.69** ± 0.32%	**80.82** ± 0.24%
PN	Conv-4-512	66.48 ± 0.32%	80.28 ± 0.23%
PN+rot		**67.94** ± 0.31%	**82.20** ± 0.23%
PN	WRN-28-10	68.35 ± 0.34%	**81.79** ± 0.23%
PN+rot		**68.60** ± 0.34%	81.25 ± 0.24%

Table 2: Rotation prediction as auxiliary loss on CIFAR-FS. Average 5-way classification accuracies for the novel classes on the test set of CIFAR-FS with 95% confidence intervals (using 5000 test episodes).

applied to the original images features). Note that this patch based auxiliary classification loss has the same weight as the original classification loss L_{few}. Also, during the second learning stage we do not use the patch based classifier.

4.2. Self-supervision as auxiliary loss function

Rotation prediction as auxiliary loss function. We first study the impact of adding rotation prediction as self-supervision to the few-shot learning algorithms of Cosine Classifiers (CC) and Prototypical Networks (PN). We perform this study using the MiniImageNet and CIFAR-FS datasets and report results in Tables 1 and 2 respectively. For the CC case, we use as strong baselines, CC models without self-supervision but trained to recognize all the 4 rotated versions of an image. The reason for using this baseline is that during the first learning stage, the model "sees" the same type of data, *i.e.*, rotated images, as the model with rotation prediction self-supervision. Note that despite the rotation augmentations of the first learning stage, during the second stage the model *uses as training examples for the novel classes only the up-right versions of the images*. Still however, using rotation augmentations improves the classification performance of the baseline models when adapted to the novel classes. Therefore, for fair comparison, we also apply rotation augmentations to the CC models with rotation prediction self-supervision. For the PN case, we do not use rotation augmentation since in our experiments this lead to performance degradation.

The results in Tables 1 and 2 demonstrate that **(1)** indeed, adding rotation prediction self-supervision improves the few-shot classification performance, and **(2)** the performance improvement is more significant for high capacity architectures, *e.g.*, WRN-28-10.

Model	Backbone	1-shot	5-shot
CC	Conv-4-64	53.72 ± 0.42%	70.96 ± 0.33%
CC+loc		**54.30** ± 0.42%	**71.58** ± 0.33%
CC	Conv-4-512	55.58 ± 0.42%	73.52 ± 0.33%
CC+loc		**56.87** ± 0.42%	**74.84** ± 0.33%
CC	WRN-28-10	58.43 ± 0.46%	75.45 ± 0.34%
CC+loc		**60.71** ± 0.46%	**77.64** ± 0.34%

Table 3: Relative patch location as auxiliary loss on MiniImageNet. Average 5-way classification accuracies for the novel classes on the test set of MiniImageNet with 95% confidence intervals (using 2000 test episodes).

Relative patch location prediction as auxiliary loss function. As explained in §3.2.1, we consider a second self-supervised task, namely relative patch location prediction. For simplicity, we restrict its assessment to the CC few-shot algorithm, which in our experiments proved to perform better than PN and to be simpler to train. Also, for this study we consider only the MiniImageNet dataset and not CIFAR-FS since the latter contains thumbnail images of size 32×32 from which it does not make sense to extract patches: their size would have to be less than 8×8 pixels, which is too small for any of the evaluated architectures. We report results on MiniImageNet in Table 3. As a strong baseline we used CC models without self-supervision but with the auxiliary patch based classification loss described in §4.1.

Based on the results of Table 3 we observe that: **(1)** relative patch location also manages to improve the few-shot classification performance and, as in the rotation prediction case, the improvement is more significant for high capacity network architectures. **(2)** Also, comparing to the rotation prediction case, the relative patch location offers smaller

Models	Backbone	1-shot	5-shot
MAML [11]	Conv-4-64	48.70 ± 1.84%	63.10 ± 0.92%
Prototypical Nets [52]	Conv-4-64	49.42 ± 0.78%	68.20 ± 0.66%
LwoF [13]	Conv-4-64	56.20 ± 0.86%	72.81 ± 0.62%
RelationNet [58]	Conv-4-64	50.40 ± 0.80%	65.30 ± 0.70%
GNN [12]	Conv-4-64	50.30%	66.40%
R2-D2 [2]	Conv-4-64	48.70 ± 0.60%	65.50 ± 0.60%
R2-D2 [2]	Conv-4-512	51.20 ± 0.60%	68.20 ± 0.60%
TADAM [40]	ResNet-12	58.50 ± 0.30%	76.70 ± 0.30%
Munkhdalai *et al*. [37]	ResNet-12	57.10 ± 0.70%	70.04 ± 0.63%
SNAIL [50]	ResNet-12	55.71 ± 0.99%	68.88 ± 0.92%
Qiao *et al*. [43]*	WRN-28-10	59.60 ± 0.41%	73.74 ± 0.19%
LEO [48]*	WRN-28-10	61.76 ± 0.08%	77.59 ± 0.12%
CC+rot	Conv-4-64	54.83 ± 0.43%	71.86 ± 0.33%
CC+rot	Conv-4-512	56.27 ± 0.43%	74.30 ± 0.34%
CC+rot	WRN-28-10	**62.93** ± 0.45%	**79.87** ± 0.33%
CC+rot+unlabeled	WRN-28-10	**63.77** ± 0.45%	**80.70** ± 0.33%

Table 4: Comparison with prior work on MiniImageNet. Average 5-way classification accuracies for the novel classes on the test set of MiniImageNet with 95% confidence intervals (using 2000 test episodes). *: using also the validation classes for training. For the description of the CC+rot+unlabeled model see §4.3.

performance improvement.

Comparison with prior work. In Tables 4, 5, and 6, we compare our approach with prior few-shot methods on the MiniImageNet, CIFAR-FS, and *tiered*-MiniImageNet datasets respectively. For our approach we used CC and rotation prediction self-supervision, which before gave the best results. In all cases we achieve state-of-the-art results surpassing prior methods with a significant margin. For instance, in the 1-shot and 5-shot settings of MiniImageNet we outperform the previous leading method LEO [48] by around 1.3 and 2.3 percentage points respectively.

More detailed results are provided in Appendix A.

4.3. Semi-supervised few-shot learning

Next, we evaluate the proposed semi-supervised training approach. In these experiments we use CC models with rotation prediction self-supervision. We perform two types of semi-supervised experiments: (1) training with unlabeled data from the same base classes, and (2) training with unlabeled data that are not from the base classes.

Training with unlabeled data from the same base classes. From the base classes of MiniImageNet, we use only a small percentage of the training images (*e.g.*, 5% of images per class) as annotated training data, while the rest of the images (*e.g.*, 95%) are used as the unlabeled data in the semi-supervised training. We provide results in the first two sections of Table 7. The proposed semi-supervised

Models	Backbone	1-shot	5-shot
PN [52]†	Conv-4-64	55.50 ± 0.70%	72.00 ± 0.60%
PN [52]†	Conv-4-512	57.90 ± 0.80%	76.70 ± 0.60%
PN [52]‡	Conv-4-64	62.82 ± 0.32%	79.59 ± 0.24%
PN [52]‡	Conv-4-512	66.48 ± 0.32%	80.28 ± 0.23%
MAML [11]†	Conv-4-64	58.90 ± 1.90%	71.50 ± 1.00%
MAML [11]†	Conv-4-512	53.80 ± 1.80%	67.60 ± 1.00%
RelationNet [58]†	Conv-4-64	55.00 ± 1.00%	69.30 ± 0.80%
GNN [12]†	Conv-4-64	61.90%	75.30%
GNN [12]†	Conv-4-512	56.00%	72.50%
R2-D2 [2]	Conv-4-64	60.00 ± 0.70%	76.10 ± 0.60%
R2-D2 [2]	Conv-4-512	64.00 ± 0.80%	78.90 ± 0.60%
CC+rot	Conv-4-64	63.45 ± 0.31%	79.79 ± 0.24%
CC+rot	Conv-4-512	65.87 ± 0.30%	81.92 ± 0.23%
CC+rot	WRN-28-10	**73.62** ± 0.31%	**86.05** ± 0.22%

Table 5: Comparison with prior work in CIFAR-FS. Average 5-way classification accuracies for the novel classes on the test set of CIFAR-FS with 95% confidence (using 5000 test episodes). †: results from [2]. ‡: our implementation.

Models	Backbone	1-shot	5-shot
MAML [11]†	Conv-4-64	51.67 ± 1.81%	70.30 ± 0.08%
Prototypical Nets [52]	Conv-4-64	53.31 ± 0.89%	72.69 ± 0.74 %
RelationNet [58]†	Conv-4-64	54.48 ± 0.93%	71.32 ± 0.78%
Liu *et al*. [33]	Conv-4-64	57.41 ± 0.94%	71.55 ± 0.74
LEO [48]	WRN-28-10	66.33 ± 0.05%	81.44 ± 0.09 %
CC	WRN-28-10	70.04 ± 0.51%	84.14 ± 0.37%
CC+rot	WRN-28-10	**70.53** ± 0.51%	**84.98** ± 0.36%

Table 6: Rotation prediction as auxiliary loss on *tiered*-MiniImageNet. Average 5-way classification accuracies for the novel classes on the test set of *tiered*-MiniImageNet with 95% confidence (using 2000 test episodes for our entries). †: results from [33].

training approach is compared with a CC model without self-supervision and with a CC model with self-supervision but no recourse to the unlabeled data. The results demonstrate that indeed, our method manages to improve few-shot classification performance by exploiting unlabeled images. Compared to Ren *et al*. [45], that also propose a semi-supervised method, our method with Conv-4-64 and 20% annotations achieves better results than their method with Conv-4-64 and 40% annotations (*i.e.*, our **51.21**% and **68.89**% MiniImageNet accuracies vs. their 50.41% and 64.39% for the 1-shot and 5-shot settings respectively).

Training with unlabeled data not from the base classes. This is a more realistic setting, since it is hard to constrain the unlabeled images to be from the same classes as the base classes. For this experiment, we used as unlabeled data the training images of the *tiered*-MiniImageNet base classes *minus the classes that are common with the base, validation, or test classes of MiniImageNet*. In total, 408, 726

Rot	M	T	$\mu = 5\%$		$\mu = 10\%$		$\mu = 20\%$	
			1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Conv-4-64								
			41.87%	57.76%	45.63%	62.29%	49.34%	66.48%
✓			43.26%	58.88%	47.57%	64.03%	50.48%	67.92%
✓	✓		**45.41%**	**62.14%**	**47.75%**	**64.93%**	**51.21%**	**68.89%**
WRN-28-10								
			39.05%	54.89%	44.59%	61.42%	49.52%	67.81%
✓			43.60%	60.52%	49.05%	67.35%	53.66%	72.69%
✓	✓		**47.25%**	**65.07%**	**52.90%**	**71.48%**	**56.79%**	**74.67%**
✓		✓	46.95%	64.93%	52.66%	71.13%	55.37%	73.66%

Table 7: Semi-supervised training with rotation prediction as self-supervision on MiniImageNet. Average 5-way classification accuracies on the test set of MiniImageNet. μ is the percentage of the base class training images of MiniImageNet that are used as annotated data during training. *Rot* indicates adding self-supervision, *M* indicates using as unlabeled data the (rest of) MiniImageNet training dataset, and *T* using as unlabeled data the *tiered*-MiniImageNet training dataset.

Models	Backbone	1-shot	5-shot	20-shot	50-shot
CACTUs [23]	Conv-4-64	39.90%	53.97%	63.84%	69.64%
CC	Conv-4-64	53.63%	70.74%	80.03%	82.61%
Rot		41.70%	58.64%	68.61%	71.86%
Loc		37.75%	53.02%	61.38%	64.15%
CC	WRN-28-10	58.59%	76.59%	82.70%	84.27%
Rot		43.43%	60.86%	69.82%	72.20%
Loc		41.78%	59.10%	67.86%	70.32%

Table 8: Evaluating self-supervised representation learning methods on few-shot recognition. Average 5-way classification accuracies on the test set of MiniImageNet. *Rot* refers to the rotation prediction task, *Loc* to the relative patch location task, and *CC* to the supervised method of Cosine Classifiers.

unlabeled images are used from *tiered*-MiniImageNet. We report results in the last row of Table 7. Indeed, even in this difficult setting, our semi-supervised approach is still able to exploit unlabeled data and improve the classification performance. Furthermore, we did an extra experiment in which we trained a WRN-28-10 based model using 100% of MiniImageNet training images and unlabeled data from *tiered*-MiniImageNet. This model achieved **63.77%** and **80.70%** accuracies for the 1-shot and 5-shot settings respectively on MiniImageNet (see entry CC+rot+unlabeled of Table 4), which improves over the already very strong CC+rot model (see Table 4).

4.4. Few-shot object recognition to assess self-supervised representations

Given that our framework allows the easy combination of any type of self-supervised learning approach with the adopted few-shot learning algorithms, we also propose to use it as an alternative way for comparing/evaluating the effectiveness of different self-supervised approaches. To this end, the only required change to our framework is to use uniquely the self-learning loss (*i.e.*, no labeled data is now used) in the first learning stage (for implementation details see Appendix B.2). The performance of the few-shot model resulting from the second learning stage can then be used for evaluating the self-supervised method under consideration.

Comparing competing self-supervised techniques is not straightforward since it must be done by setting up another, somewhat contrived task that exploits the learned representations [6, 28]. Instead, given the very similar goals of few-shot and self-supervised learning, we argue that the proposed comparison method could be more meaningful for assessing different self-supervised features. Furthermore, it is quite simple and fast to perform when compared to some alternatives such as fine-tuning the learned representations on the PASCAL [8] detection task [6], with the benefit of obtaining more robust statistics aggregated over evaluations of thousands of episodes with multiple different configurations of classes and training/testing samples.

To illustrate our point, we provide in Table 8 quantitative results of this type of evaluation on the MiniImageNet dataset, for the self-supervision methods of rotation prediction and relative patch location prediction. For self-supervised training we used the training images of the base classes of MiniImageNet and for the few-shot classification step we used the test classes of MiniImageNet. We observe that the explored self-supervised approaches achieve relatively competitive classification performance when compared to the supervised method of CC and obtain results that are on par or better than other, more complex, unsupervised systems. We leave as future work a more detailed and thorough comparison of self-learned representations in this evaluation setting.

5. Conclusions

Inspired by the close connection between few-shot and self-supervised learning, we propose to add an auxiliary loss based on self-supervision during the training of few-shot recognition models. The goal is to boost the ability of the latter to recognize novel classes using only few training data. Our detailed experiments on MiniImagenet, CIFAR-FS, and tiered-MiniImagenet few-shot datasets reveal that indeed adding self-supervision leads to significant improvements on the few-shot classification performance, which makes the employed few-shot models achieve state-of-the-art results. Furthermore, the annotation-free nature of the self-supervised loss allows us to exploit diverse unlabeled data in a semi-supervised manner, which further improves the classification performance. Finally, we show that the proposed framework can also be used for evaluating self-supervised or unsupervised methods based on few-shot object recognition.

References

[1] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016. 2

[2] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019. 5, 7

[3] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, 2016. 2

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. PAMI*, 40(4):834–848, 2018. 1

[5] T. Chen, X. Zhai, M. Ritter, M. Lucic, and N. Houlsby. Self-supervised generative adversarial networks. *arXiv preprint arXiv:1811.11212*, 2018. 3

[6] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 1, 2, 4, 8

[7] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, pages 766–774, 2014. 1

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 8

[9] L. Fei-Fei et al. A bayesian approach to unsupervised one-shot learning of object categories. In *ICCV*, 2003. 1

[10] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. PAMI*, 28(4):594–611, 2006. 1

[11] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017. 2, 7

[12] V. Garcia and J. Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017. 2, 7

[13] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018. 2, 3, 7

[14] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 1, 2, 4

[15] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. 2

[16] F. Gomez and J. Schmidhuber. Evolving modular fast-weight networks for control. In *ICANN*, 2005. 2

[17] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 3

[18] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2

[19] C. Han, S. Shan, M. Kan, S. Wu, and X. Chen. Face recognition with contrastive convolution. In *ECCV*, 2018. 2

[20] B. Hariharan and R. B. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017. 2

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[22] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[23] K. Hsu, S. Levine, and C. Finn. Unsupervised learning via meta-learning. In *ICLR*, 2019. 8

[24] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 5

[25] Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017. 2

[26] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015. 1, 2

[27] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017. 2

[28] A. Kolesnikov, X. Zhai, and L. Beyer. Revisiting self-supervised visual representation learning. *arXiv preprint arXiv:1901.09005*, 2019. 4, 8, 12

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[30] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *An. Meeting of the Cognitive Science Society*, 2011. 1

[31] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016. 1, 2

[32] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang. Unsupervised representation learning by sorting sequences. In *ICCV*, 2017. 2

[33] Y. Liu, J. Lee, M. Park, S. Kim, and Y. Yang. Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018. 7

[34] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018. 2

[35] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016. 2

[36] T. Mordan, N. Thome, G. Henaff, and M. Cord. Revisiting multi-task learning with rock: a deep residual auxiliary block for visual detection. In *NIPS*, 2018. 2

[37] T. Munkhdalai and H. Yu. Meta networks. *arXiv preprint arXiv:1703.00837*, 2017. 2, 7

[38] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. 2

[39] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 1, 2

[40] B. N. Oreshkin, A. Lacoste, and P. Rodriguez. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018. 7

[41] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

[42] H. Qi, M. Brown, and D. G. Lowe. Low-shot learning with imprinted weights. In *CVPR*, 2018. 2, 3
[43] S. Qiao, C. Liu, W. Shen, and A. Yuille. Few-shot image recognition by predicting parameters from activations. *arXiv preprint arXiv:1706.03466*, 2, 2017. 2, 7
[44] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. *ICLR*, 2017. 2
[45] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018. 5, 7
[46] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1
[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5
[48] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018. 7
[49] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016. 2
[50] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017. 7
[51] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1
[52] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017. 2, 3, 7
[53] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1
[54] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, 2016. 1, 2, 5, 10
[55] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos. In *ECCV*, 2018. 2
[56] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan. Low-shot learning from imaginary data. *arXiv preprint arXiv:1801.05401*, 2018. 2
[57] Y.-X. Wang, D. Ramanan, and M. Hebert. Learning to model the tail. In *NIPS*, 2017. 2
[58] F. S. Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 2, 7
[59] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016. 5, 10
[60] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016. 1, 2
[61] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 2

A. Extra experimental results

A.1. Rotation prediction self-supervision: Impact of rotation augmentation

In the experiments reported in Section 4, we use rotation augmentation when training the baselines to compare against the CC-models with self-supervised rotation prediction. In Table 9 of this Appendix we also provide results without using rotation augmentation. The purpose is to examine what is the impact of this augmentation technique. We observe that (1) the improvements yielded by rotation prediction self-supervision are more significant, and (2) in some cases rotation augmentation actually reduces the few-shot classification performance.

A.2. Relative patch location self-supervision: impact of patch based object classification loss

When we study the impact of adding relative patch location self-supervision to CC-based models in Section 4, we use an auxiliary patch based object classification loss. In Table 10 we also provide results without using this auxiliary loss when training CC models. The purpose is to examine what is the impact of this auxiliary loss. We observe that the improvement brought by this auxiliary loss is small (or no-existing) when compared to the performance improvement thanks to the relative patch location self-supervision.

B. Additional implementation details

B.1. Network architectures

Conv-4-64 [54]. It consists of 4 convolutional blocks each implemented with a 3×3 convolutional layer with 64 channels followed by BatchNorm + ReLU + 2×2 max-pooling units. In the MiniImageNet experiments for which the image size is 84×84 pixels, its output feature map has size $5 \times 5 \times 64$ and is flattened into a final 1600-dimensional feature vector. For the CIFAR-FS experiments, the image size is 32×32 pixels, the output feature map has size $2 \times 2 \times 64$ and is flattened into a 256-dimensional feature vector.

Conv-4-512. It is derived from Conv-4-64 by gradually increasing its width across layers leading to 96, 128, 256, and 512 feature channels for its 4 convolutional blocks respectively. Therefore, for a 84×84 sized image (*i.e.*, MiniImageNet experiments) its output feature map has size $5 \times 5 \times 512$ and is flattened into a final 12800-dimensional feature vector, while for a 32×32 sized image (*i.e.*, CIFAR-FS experiments) its output feature map has size $2 \times 2 \times 512$ and is flattened into a final 2048-dimensional feature vector.

WRN-28-10 [59]. It is a Wide Residual Network with 28 convolutional layers and width factor 10. The 12 residual layers of this architecture are grouped into 3 residual blocks

Model	Rot. Aug.	Backbone	MiniImageNet		CIFAR-FS	
			1-shot	5-shot	1-shot	5-shot
CC		Conv-4-64	53.94 ± 0.42%	71.13 ± 0.34%	62.83 ± 0.31%	79.14 ± 0.24%
CC+rot			**55.41** ± 0.43%	**72.98** ± 0.33%	**63.98** ± 0.31%	**80.44** ± 0.23%
CC	✓		54.31 ± 0.42%	70.89 ± 0.34%	61.80 ± 0.30%	78.02 ± 0.24%
CC+rot	✓		54.83 ± 0.43%	71.86 ± 0.33%	63.45 ± 0.31%	79.79 ± 0.24%
CC		Conv-4-512	54.51 ± 0.42%	72.52 ± 0.34%	65.64 ± 0.31%	81.10 ± 0.23%
CC+rot			**56.59** ± 0.43%	**74.67** ± 0.34%	**67.00** ± 0.30%	**82.55** ± 0.23%
CC	✓		55.68 ± 0.43%	73.19 ± 0.33%	65.26 ± 0.31%	81.14 ± 0.23%
CC+rot	✓		56.27 ± 0.43%	74.30 ± 0.33%	65.87 ± 0.30%	81.92 ± 0.23%
CC		WRN-28-10	58.59 ± 0.45%	76.59 ± 0.33%	70.43 ± 0.31%	83.84 ± 0.23%
CC+rot			60.10 ± 0.45%	77.40 ± 0.33%	72.49 ± 0.31%	84.77 ± 0.22%
CC	✓		61.09 ± 0.44%	78.43 ± 0.33%	71.83 ± 0.31%	84.63 ± 0.23%
CC+rot	✓		**62.93** ± 0.45%	**79.87** ± 0.33%	**73.62** ± 0.31%	**86.05** ± 0.22%

Table 9: Impact of rotation augmentation. Average 5-way classification accuracies for the novel classes on the test sets of MiniImageNet and CIFAR-FS with 95% confidence intervals. *Rot. Aug.* indicates using rotation augmentation during the first learning stage.

Model	Patch Cls.	Backbone	1-shot	5-shot
CC		Conv-4-64	53.63 ± 0.42%	70.74 ± 0.34%
CC	✓		53.72 ± 0.42%	70.96 ± 0.33%
CC+loc	✓		**54.30** ± 0.42%	**71.58** ± 0.33%
CC		Conv-4-512	54.51 ± 0.42%	72.52 ± 0.34%
CC	✓		55.58 ± 0.42%	73.52 ± 0.33%
CC+loc	✓		**56.87** ± 0.42%	**74.84** ± 0.33%
CC		WRN-28-10	58.59 ± 0.45%	76.59 ± 0.33%
CC	✓		58.43 ± 0.46%	75.45 ± 0.34%
CC+loc	✓		**60.71** ± 0.46%	**77.64** ± 0.34%

Table 10: Impact of auxiliary patch based object classification loss. Average 5-way classification accuracies for the novel classes on the test set of MiniImageNet with 95% confidence intervals. *Patch Cls.* indicates using an auxiliary patch based object classification loss during the first learning stage.

(4 residual layers per block). In the MiniImageNet and *tiered*-MiniImageNet experiments, the network gets as input images of size 80×80 (rescaled from 84×84), and during feature extraction each residual block downsamples by a factor of 2 the processed feature maps. Therefore, the output feature map has size $10 \times 10 \times 640$ which, after global average pooling, creates a 640-dimensional feature vector. In the CIFAR-FS experiments, the input images have size 32×32 and during feature extraction only the last two residual blocks downsample the processed feature maps. Therefore, in the CIFAR-FS experiments, the output feature map has size $8 \times 8 \times 640$ which again after global average pooling creates a 640-dimensional feature vector.

Rotation prediction network, $R_\phi(\cdot)$. This network gets as input the output feature maps of F_θ and is implemented as a convnet. More specifically, for the Conv-4-64 and Conv-4-512 feature extractor architectures (regardless of the dataset), R_ϕ consists of two 3×3 convolutional layers with BatchNorm + ReLU units, followed by a fully connected classification layer. For Conv-4-64, those two convolutional layers have 128 and 256 feature channels respectively, while for Conv-4-512 both convolutional layers have 512 feature channels. In the WRN-28-10 case, R_ϕ consists of a 4-residual-layer residual block that actually replicates the last (3rd) residual block of WRN-28-10. This residual block is followed by global average pooling plus a fully connected classification layer.

Relative patch location network, $P_\phi(\cdot, \cdot)$. Given two patches, $P_\phi(\cdot, \cdot)$ gets the concatenation of their feature vectors extracted with F_θ as input, and forwards it to two fully connected layers. The single hidden layer, which includes BatchNorm + ReLU units, has 256, 1024, and 1280 channels for the Conv-4-64, Conv-4-512, and WRN-28-10 architectures respectively.

B.2. Incorporating self-supervision during training

Here we provide more implementation details regarding how we incorporate self-supervision during the fist learning

stage.

Training with rotation prediction self-supervision. During training for each image of a mini-batch we create its 4 rotated copies and apply to them the rotation prediction task (*i.e.*, L_{self} loss). When training the object classifier with rotation augmentation (*e.g.*, CC-based models) the object classification task (*i.e.*, L_{few} loss) is applied to all rotated versions of the images. Otherwise, only the upright images (*i.e.*, the 0 degrees images) are used for the object classification task. Note that in the PN-based models, we apply the rotation task to both the support and the query images of a training episode, and also we do not use rotation augmentation for the object classification task.

Training with relative patch location self-supervision. In this case during training each mini-batch includes two types of visual data, images and patches. Similar to [28], in order to create patches, an image is: (1) resized to 96×96 pixels (from 84×84), (2) converted to grayscale with probability 0.66, and then (3) divided into 9 regions of size 32×32 with a 3×3 regular grid. From each 32×32 sized region we (4) randomly sample a 24×24 patch, and then (5) normalize the pixels of the patch individually to have zero mean and unit standard deviation. The object classification task is applied to the image data of the mini-batch while the relative patch location task to the patch data of the mini-batch. Also, as already explained, we also apply an extra auxiliary object classification loss to the patch data.

B.3. Training routine for first learning stage

To optimize the training loss we use mini-batch SGD optimizer with momentum 0.9 and weight decay $5\text{e}-4$. In the MiniImageNet and CIFAR-FS experiments, we train the models for 60 epochs (each with 1000 SGD iterations), starting with a learning rate of 0.1 which is decreased by a factor of 10 every 20 epochs. In the *tiered*-MiniImageNet experiments we train for 100 epochs (each with 2000 SGD iterations), starting with a learning rate of 0.1 which is decreased by a factor of 10 every 40 epochs. The mini-batch sizes were cross-validated on the validation split. For instance, the models based on CC and Conv-4-64, Conv-4-512, or WRN-28-10 architectures are trained with mini-batch sizes equal to 128, 128, or 64 respectively. Finally, we perform early stopping w.r.t. the few-shot classification accuracy on the validation novel classes (for the CC-based models we use the 1-shot classification accuracy).

Semi-supervised training. Here each mini-batch consists of both labeled and unlabeled data. Specifically, for the experiments that use the Conv-4-64 network architecture, and 5%, 10%, or 20% of MiniImageNet as labeled data, each mini-batch consists of 64 labeled images and 64 unlabeled images. For the experiments that use the WRN-28-10 network architecture, and 5%, 10%, or 20% of MiniImageNet as labeled data, each mini-batch consists of 16 labeled images and 48 unlabeled images. For the experiment that uses 100% of MiniImageNet as labeled data and *tiered*-MiniImageNet for unlabeled data, then each mini-batch consists of 32 labeled images and 32 unlabeled images.

B.4. Assessing self-supervised representations based on the few-shot object recognition task

Here we provide implementation details for the experiments in §4.4, which assess the self-supervised representations using the few-shot object recognition task. Except from the fact that during the first learning stage, (1) there is no object based supervision (*i.e.*, no L_{few} loss), and (2) *no early stopping based on a validation set*, the rest of the implementation details remain the same as in the other CC-based experiments. A minor difference is that, when evaluating the WRN-28-10 and relative patch location based model, we create the representation of each image by averaging the extracted feature vectors of its 9 patches (similar to [28]).