Supplementary Material: 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

A. Extra experimental results

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

In the experiments reported in Section 4 of main paper, we use rotation augmentation when training the baselines to compare against the CC-models with self-supervised rotation prediction. In Table 1 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 of main paper, we use an auxiliary patch based object classification loss. In Table 2 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.

A.3. Evaluation with a VGG-based feature extractor.

Here we evaluate self-supervision as auxiliary loss function for another high-capacity feature extractor network architecture, one based on the VGG-16 [3] network. For simplicity, we restrict the assessment on the CC few-shot algorithm, the rotation prediction self-supervised task, and the MiniImageNet dataset. We provide results in Table 3. As in the main paper, we observe that for such high-capacity feature extractor architectures the added self-supervision offers quite significant performance improvements.

A.4. Impact of the loss weight α of the auxiliary self-supervised task.

In our experiments, for simplicity, we always set $\alpha = 1.0$ for the loss weight of the auxiliary self-supervised task[1] (see equations (6) and (9) of main paper). Here we study the impact of this hyper-parameter on the CC few-shot algorithm, the rotation prediction self-supervised task, and the high-capacity WRN-28-10 network architecture. We provide results in Table 4. We observe that by more careful tuning of this hyper-parameter we can achieve even higher few-shot classification performance.

B. Additional implementation details

B.1. Network architectures

Conv-4-64 [4]. 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 $32{\times}32$ 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 [5]. It is a Wide Residual Network with 28 convolutional layers and width factor 10. The 12 residual

[1]The only exception is for the 1-shot PN+rot model with WRN-28-10 on CIFAR-FS where, for achieving good performance, it was necessary to set $\alpha = 0.3$.

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%	72.95 ± 0.31%	85.50 ± 0.22%
CC+rot			60.10 ± 0.45%	77.40 ± 0.33%	74.72 ± 0.31%	86.43 ± 0.21%
CC	✓		61.09 ± 0.44%	78.43 ± 0.33%	74.51 ± 0.31%	86.45 ± 0.21%
CC+rot	✓		**62.93** ± 0.45%	**79.87** ± 0.33%	**75.38** ± 0.31%	**87.25** ± 0.21%

Table 1: 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 2: 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.

Model	Backbone	1-shot	5-shot
CC	VGG	58.20 ± 0.45%	74.81 ± 0.36%
CC+rot		**59.89** ± 0.46%	**76.58** ± 0.35%

Table 3: Rotation prediction as auxiliary loss on MiniImageNet with VGG-based backbone and the CC algorithm. Average 5-way classification accuracies for the novel classes with 95% confidence intervals (using 2000 episodes). Both models were trained with rotation augmentations.

layers of this architecture are grouped into 3 residual blocks (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.

ResNet10 [1]. It is a typical Residual Network with 10 convolutional layers. It gets as input images of size 224×224 and outputs feature maps of size $7 \times 7 \times 512$, which, after global average pooling, creates a 512-dimensional feature vector.

VGG[3]-based feature extractor. In this case we used the convolutional part of the VGG-16 architecture with BatchNorm units after the convolutional layers. It gets as input images of size 80×80 and outputs feature maps of size $5 \times 5 \times 512$, which, after global average pooling, creates a 512-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-4-65, Conv-4-512, and VGG feature extractor architectures (regardless of the

Model	Backbone	α	MiniImageNet		CIFAR-FS	
			1-shot	5-shot	1-shot	5-shot
CC+rot	WRN-28-10	0.3	62.35 $\pm$ 0.45%	79.53 $\pm$ 0.34%	76.01 $\pm$ 0.31%	**87.83** $\pm$ 0.21%
		0.5	**63.78** $\pm$ 0.46%	**80.77** $\pm$ 0.33%	**76.09** $\pm$ 0.30%	87.59 $\pm$ 0.22%
		1.0	62.93 $\pm$ 0.45%	79.87 $\pm$ 0.33%	75.38 $\pm$ 0.31%	87.25 $\pm$ 0.21%
		2.0	62.56 $\pm$ 0.46%	79.89 $\pm$ 0.34%	75.74 $\pm$ 0.30%	87.42 $\pm$ 0.22%

Table 4: Impact of the loss weight α of the auxiliary self-supervised task. Average 5-way classification accuracies for the novel classes with 95% confidence intervals.

dataset), R_ϕ consists of two 3×3 convolutional layers with BatchNorm + ReLU units, followed by a fully connected cosine similarity-based classification layer. For Conv-4-64, those two convolutional layers have 128 and 256 feature channels respectively, while for Conv-4-512 and VGG both convolutional layers have 512 feature channels. In the WRN-28-10 case, R_ϕ consists of a 4-residual-layer residual block, similar to the last (3rd) residual block of WRN-28-10, with 640 feature channels as input and output. This residual block is followed by global average pooling plus a fully connected cosine similarity-based classification layer. In the ResNet10 case, R_ϕ consists of 2-residual-layer residual block with 512 feature channels as input and output.

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 degree 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 [2], 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 that is decreased by a factor of 10 when the validation error plateaus. In the *tiered*-MiniImageNet experiments we train for 100 epochs (each with 2000 SGD iterations), starting with a learning rate of 0.1 that is decreased by a factor of 10 every 40 epochs. In the ImageNet-FS experiments we train for 130 epochs (each with 4000 SGD iterations) starting with a learning rate of 0.1 that is decreased by a factor of 10 after 60, 90, and 120 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, WRN-28-10, or VGG architectures are trained with mini-batch sizes equal to 128, 128, 64, or 64 respectively. For the experiments in ImageNet-FS with the ResNet10 architecture, the mini-batch size is 128. 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 16 labeled images and 48 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 of the main paper, 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 [2]).

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[2] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. *arXiv preprint arXiv:1901.09005*, 2019. 3, 4

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2

[4] Oriol Vinyals, Charles Blundell, Tim Lillicrap, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016. 1

[5] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 1