CONTRASTIVE LEARNING IS JUST META-LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Contrastive learning has recently taken off as a paradigm for learning from unlabeled data. In this paper, we discuss the close relationship between contrastive learning and meta-learning, and in fact show that contrastive learning can be interpreted as a special case of meta-learning with a certain task distribution. We complement this observation by showing that established meta-learning methods, such as Prototypical Networks, achieve comparable performance to SimCLR when paired with this task distribution. This close relationship can be leveraged by taking established techniques from the meta-learning literature, such as task-based data augmentation, and showing that they benefit contrastive learning as well. These tricks also benefit state-of-the-art self-supervised learners without using negative pairs such as BYOL, which achieves 94.6% accuracy on CIFAR-10 using a self-supervised ResNet-18 feature extractor trained with our meta-learning tricks.

1 INTRODUCTION

Self-supervised visual representation learning (SSL) has recently gathered attention due to its ability to learn good image features without manual supervision, thus allowing for efficient learning on downstream tasks such as detection and segmentation (Noroozi & Favaro, 2016; Zhang et al., 2016; Oord et al., 2018; Hjelm et al., 2018; Wu et al., 2018; Gidaris et al., 2018; He et al., 2020; Misra & Maaten, 2020; Tian et al., 2020a; Chen et al., 2020a; Kim et al., 2020; Grill et al., 2020; Caron et al., 2020; Kalantidis et al., 2020; Shen et al., 2020; Li et al., 2020; Zbontar et al., 2021; Chen & He, 2021; Caron et al., 2021). Among SSL approaches, contrastive learning based methods (Chen et al., 2020a; He et al., 2020; Chen et al., 2020b) show particularly strong potential and achieve promising results which are close to those of fully supervised methods on numerous computer vision benchmarks.

These methods rely on applying various data augmentations such as random crops, flips, color distortion, and Gaussian blur on the same training sample to create different views of an image. Two such example methods, SimCLR (Chen et al., 2020a) and MoCo (He et al., 2020), involve reducing the distance between features corresponding to positive pairs (different augmented views of the same image), and increasing the distance between features corresponding to negative pairs (augmented views of different images).

Meanwhile, meta-learning is an established popular framework for learning models that quickly adapt to on-the-fly tasks given a small number of examples (Hochreiter et al., 2001; Finn et al., 2017; Nichol et al., 2018; Bertinetto et al., 2018; Lee et al., 2019). The training loop for meta-learners typically involves (i) sampling a random batch of classes and (ii) updating a feature extractor to distinguish between these classes. This procedure mirrors that of contrastive learning, which proceeds by (i) sampling a batch of images and augmenting them to generate classes (each "class" is an image plus all of its views), and (ii) updating a feature extractor to distinguish between these classes. Conceptually, contrastive learning is merely a special case of meta-learning in which the training tasks are generated by computing views of individual images.

In this paper, we discuss the close relationship between contrastive learning and meta-learning. Concretely, we show that established meta-learning algorithms, originally designed for few-shot learning, can achieve the same performance as recent contrastive learning algorithms on standard SSL problems when paired with the same data sampling strategy. In addition, we explore techniques, originally designed for meta-learning, that can improve contrastive learning. Specifically, we first

explore ways by which we can adapt data augmentation strategies inspired by recent work in meta-learning (Su et al., 2020; Ni et al., 2021) to SSL and find that this approach can yield significant performance boosts. We further investigate the relationship between the number of negative pairs in a batch and the number of training tasks from a meta-learning perspective, and we thus improve the performance of contrastive learning for small batch sizes and memory constraints.

Our contributions can be summarized as follows:

- First, we propose a meta-learning based framework for understanding self-supervised learning, and we show that meta-learners can achieve comparable self-supervised performance to contrastive learning methods.
- Second, we propose a meta-specific task augmentation strategy which boosts the performance of self-supervised learning. This data augmentation method generalizes to methods with no negative pairs, such as BYOL (Grill et al., 2020), as well.
- Finally, inspired by meta-learning training tasks, we show that a simple technique can improve the performance of contrastive learning in the small batch regime by removing the reliance on massive numbers of negative pairs, thus allowing for gradient accumulation.

2 Related Work and Background

2.1 Meta-Learning

Meta-learning algorithms aim to learn a model that can quickly adapt to new tasks with limited data and generalize to unseen examples. To achieve this, the adaptation and evaluation procedures are both simulated during meta-training. During each episode of meta-learning, we sample a task, $\mathcal{T}_i$, from a distribution of tasks, often corresponding to combinations of training classes formed into classification problems. Each task consists of *support* data $\mathcal{T}_i^s$ and *query* data $\mathcal{T}_i^q$, so that $\mathcal{T}_i = \{\mathcal{T}_i^s, \mathcal{T}_i^q\}$. When applied to few-shot classification, this task is called a k-shot, N-way classification problem, where k denotes the number of training samples per category in the support data. Then, support data will be used to simulate few-shot training data, while query data will be used to simulate novel testing samples.

A meta-learning model F, in this setting, contains a feature extractor and a classification strategy, $\mathcal{A}$. This classification strategy can take various forms, such as adding a linear classifier on top of the feature extractor and fine-tuning either the linear layer or the whole network end-to-end, or this strategy may simply classify samples by selecting the nearest class prototype. Meta-learning training algorithms have an *inner loop* and an *outer loop* in each parameter update. In the inner loop, the model is first fine-tuned on support data $\mathcal{T}_i^s$. Then, in the outer loop, the updated model is used to predict on query data $\mathcal{T}_i^q$, and a loss is minimized with respect to the model's parameters before fine-tuning. Intuitively, we update parameters so that the feature extractor extracts better features for the classification strategy, often resulting in tightly clustered features corresponding to each class (Goldblum et al., 2020). Existing works apply various methods for fine-tuning on support data during the inner loop. In a line of algorithms, such as MAML and Reptile (Finn et al., 2017; Nichol et al., 2018), all the parameters in the model are updated using gradient descent during fine-tuning on support data. Other algorithms, such as MetaOptNet and R2-D2 (Lee et al., 2019; Bertinetto et al., 2018), keep the feature extractor frozen during fine-tuning; MetaOptNet uses SVM, and R2-D2 uses ridge regression on top of the feature extractor. Similarly, metric learning approaches, such as Prototypical Networks (ProtoNet) (Snell et al., 2017; Kye et al., 2020), freeze the feature extractor as well, and create class centroids from the support data in the inner loop. In this paper, we primarily focus on the latter algorithms due to their efficiency and performance as well as the similarity of contrastive learning to metric learning.

2.2 Self-Supervised Learning

Contrastive SSL. Contrastive methods (Oord et al., 2018; Wu et al., 2018; Tian et al., 2020a; Chen et al., 2020a; He et al., 2020; Chen et al., 2020b) achieve promising performance in self-supervised learning. As mentioned previously, contrastive learning maximizes agreement on different augmented views of the same image (called *positive pairs*) while ensuring disagreement on samples

generated by different base images (called *negative pairs*). Given a batch of input images $\mathbf{x}$, m random data augmentations are applied on the same batch, generating a set of training samples $\{\tilde{\mathbf{x}}^i\}_{i=1}^m$. These samples are fed into a backbone network $f(\cdot)$ to obtain the feature representations $\{\mathbf{h}^i\}_{i=1}^m$. Then, a small neural network $g(\cdot)$, usually a non-linear MLP, is applied to project $\{\mathbf{h}^i\}_{i=1}^m$ to the latent representations $\{\mathbf{z}^i\}_{i=1}^m$ in the space where a contrastive loss $l(\cdot)$ is applied, ensuring that latent representations of positive pairs are similar while latent representations of negative pairs are different. In this paper, we mainly focus on this type of self-supervised learning and show its close relation with meta-learning. The batch sampling procedure of contrastive learning can be viewed as sampling a new classification problem with a number of classes equal to the number of base images used to generate augmented views. We will see that this on-the-fly sampling of classification problems closely mirrors a common meta-learning setup.

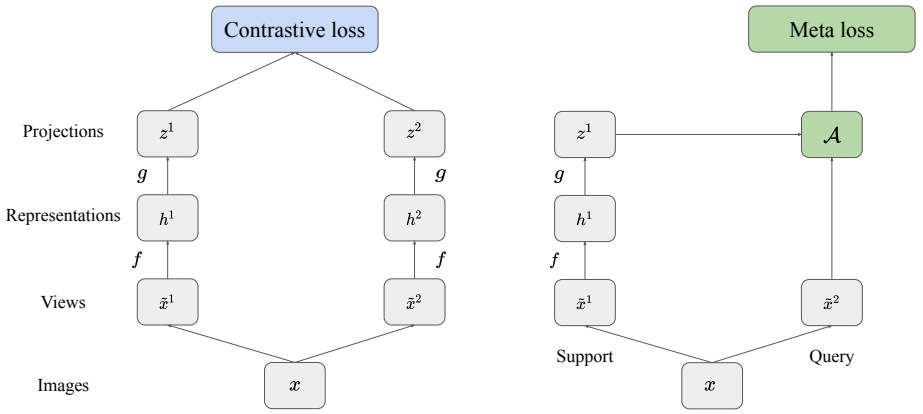
Non-Contrastive SSL. Some non-contrastive methods are generative approaches, such as auto-encoders (Vincent et al., 2008; 2010; Kingma & Welling, 2013), and adversarial learning (Goodfellow et al., 2014), where a distribution is learned over data and a latent embedding. These methods usually require operations in pixel space, which is computationally expensive. Other non-contrastive methods rely on using heuristic designed pretext tasks (Doersch et al., 2015; Zhang et al., 2016; Noroozi & Favaro, 2016; Gidaris et al., 2018) to learn the representation. More recently, BYOL (Grill et al., 2020) showed that by bootstrapping a target representation prediction, feature representations can be learned without negative pairs. However, BYOL still adopts the data augmentation procedure from contrastive learning, where different augmented views are used as training samples. In section 5, we show that BYOL still benefits from our proposed task augmentation strategy.

Data Augmentation in Meta-Learning and SSL. Data augmentations play an essential role in both meta-learning and self-supervised learning. In meta-learning, Su et al. (2020); Liu et al. (2020); Ni et al. (2021) show that proper data augmentation and meta-specific task augmentations dramatically improve few-shot learning performance by expanding the number of classes available for sampling. In self-supervised learning, Tian et al. (2020b) show contrastive learners can find better feature representations when views contain less mutual information. In addition, Kim et al. (2020); Shen et al. (2020); Li et al. (2020) show that adding harder examples such as cut-mixed samples into the training pipeline can improve self-supervised performance. In this paper, we show that similarly to meta-learning, self-supervised learners can benefit from carefully applied data augmentation techniques which mirror task augmentations from the meta-learning literature.

3 Experimental Setup

Datasets and Evaluation We conduct self-supervised training on both the CIFAR-10 and ImageNet datasets (Krizhevsky et al., 2009; Deng et al., 2009). CIFAR-10 contains 10 classes of images, and each class contains 5000 training samples and 1000 testing samples of size 32×32. The ImageNet dataset consists of 1.28 million training images and 50000 testing images. Following Chen et al. (2020a), we evaluate the pre-trained representation in a linear evaluation setting, where the feature extractor is frozen, and a classification head is stacked on top and tuned. In addition, we test the pre-trained feature extractor's performance on other downstream tasks such as transfer learning on various datasets and semi-supervised learning with 1% and 10% of labeled data. Evaluation details can be found in Appendix A.1.

Pre-Training Details We use a ResNet-18 backbone for all experiments on CIFAR-10 and ResNet-50 for those on ImageNet. We train the model on CIFAR-10 with the LARS optimizer (You et al., 2019) and batch size 1024 for 1000 epochs (with 4 GPUs). On ImageNet, we use the same optimizer and batch size of 256, and we train for 100 epochs (with 8 GPUs). For ImageNet pre-training, we follow the hyperparameter setting in Chen et al. (2020a), including baseline data augmentation methods, dimension of the latent space, and learning rate decay schedule. Unless otherwise stated, we accumulate the gradients and update our model every 8 iterations, which will be further discussed in Section 6. For CIFAR-10 pre-training, we use the same CIFAR-10 specific hyperparameters as SimCLR again, except that we pre-train with learning rate 4. We use temperature 0.5 for both meta-learning and contrastive learning based methods. For BYOL, we use the same learning rate schedule as meta-learners and start with learning rate 4. In addition, both the projector and predictor in BYOL are two-layer MLPs with hidden dimension 2048 and output dimension 256.



(a) Training procedure of contrastive learning

(b) Meta-Learning framework for SSL

Figure 1: (a) Training procedure of contrastive learning. Two augmented views are generated by applying random transformations to the same input batch. A backbone $f(\cdot)$ and a projector $g(\cdot)$ is learned through contrastive prediction tasks. (b) Meta-Learning framework for SSL. We adopt the same data augmentation operations as contrastive learning. We generate a b-way classification problem, b is the batch size, by treating each image itself as a class. Two views are separated as *support* data, on which the network is fine-tuned and the classification strategy is learned by $\mathcal{A}$; *query* data, on which we applied the updated model and calculate the meta-loss. At the end of training, everything but $f(\cdot)$ is discarded, and $\mathbf{h}$ is used as the image representation.

4 A META-LEARNING FRAMEWORK FOR SSL

Meta-learners, designed for few-shot learning, and contrastive learners for SSL are built on similar intuitions. Both approaches learn to solve new tasks on-the-fly with each batch – new classification problems in the case of meta-learning and differentiating a new batch of images in the case of contrastive learning. Furthermore, both approaches hold a goal of learning invariances which generalize to novel problems at inference; meta-learners should extract similar features for each instance of a novel test class, and contrastive learners should extract similar features for each view of an image sample. In this section, we show how one can construct a meta-learning framework for SSL which closely mirrors the strategy adopted by recent contrastive learning methods.

We now describe how to generate meta-learning task distributions $p(\mathcal{T})$ for self-supervised learning. We adopt the data augmentation operations from contrastive learning, where different random augmentations are applied to the input batch to generate alternative views. In general, given m random augmented views of a batch of b input images $\mathbf{x}$, we can create a b-way classification problem by treating all images generated by the same base image as a class. Then, we can divide the data from each class into m_1 support and m_2 query samples so that $m_1 + m_2 = m$. This framework for sampling a batch containing augmented views of base images and dividing them into support and query samples yields a task distribution $p(\mathcal{T})$. In few-shot learning terminology, each training task $\mathcal{T}_i$ is a m_1-shot-b-way classification problem, since we have b base images which generate classes, and for each such image, we have m_1 support samples. Viewed in this way, SimCLR sets $m = 2$ and $m_1 = m_2 = 1$, but SimCLR has important differences from common meta-learners.

Unlike typical meta-learning methods, SimCLR compares every sample with every other sample, while methods like ProtoNet only compare each query sample with each support sample and not with each other. Moreover, SimCLR samples a single large batch of samples for each episode of training which corresponds to sampling a single task, while meta-learners typically sample a batch of many tasks, i.e., classification problems, during each episode. We will see that this distinction can be leveraged to improve efficiency in Section 6.

Now that we have established a framework for sampling tasks, we can directly apply various meta-learning algorithms, such as R2-D2 and ProtoNet described in Section 2.1, in order to learn the parameters θ of the base model F. Recall that the base model contains a classification strategy and a feature extractor, which is the learning target of SSL. We use the same feature extractor here used

Algorithm 1: Meta-Learning Framework for Self-Supervised Learning

Require: Base model F_θ, classification strategy $\mathcal{A}$, learning rate γ, and distribution of data augmentations $\mathcal{D}$.
Initialize θ, the weights of F;
while *not done* **do**
  for $j = 1, ..., n$ **do**
    Sample a batch of base images $\mathbf{x}$;
    Sample m random data augmentations from $\mathcal{D}$ to obtain augmented views $\{\tilde{\mathbf{x}}^{\mathbf{i}}\}_{i=1}^{m}$;
    Separate $\{\tilde{\mathbf{x}}^{\mathbf{i}}\}_{i=1}^{m}$ into support set $\mathcal{T}_j^s = \{\tilde{\mathbf{x}}^{\mathbf{i}}\}_{i=1}^{m_1}$ and query set $\mathcal{T}_j^q = \{\tilde{\mathbf{x}}^{\mathbf{i}}\}_{i=m_1+1}^{m}$;
    Fine-tune model on $\mathcal{T}_j$: $\theta_j = \mathcal{A}(\theta, \mathcal{T}_j^s)$;
    Compute gradient $g_j = \nabla_\theta \mathcal{L}(F_{\theta_j}, \mathcal{T}_j^q)$;
  end
  Update base model parameters: $\theta \leftarrow \theta - \frac{\gamma}{n} \sum_j g_j$.
end

for contrastive learning in Section 2.2, which consists of a backbone $f(\cdot)$ followed by a projection head $g(\cdot)$. Formally, we solve the meta-learning optimization problem,

$$\min_\theta \mathbb{E}_{\mathcal{T}} \Big[\mathcal{L}_{SSL} \Big], \ \mathcal{L}_{SSL} = l(F_{\theta'}, \mathcal{T}^q),$$

where $\theta' = \mathcal{A}(\theta, \mathcal{T}^s)$ are parameters updated by training on support tasks, and l is the loss function used in the outer loop of training. After pre-training, only the backbone $f(\cdot)$ will be kept for self-supervised evaluation. This meta-learning framework for self-supervised learning is summarized in Algorithm 1 and Figure 1.

We compare the performance of representations learned by meta-learners with one learned by contrastive learning, namely, SimCLR under the default setting described in Section 3. Table 1 and Table 2 show the linear evaluation top-1 accuracy for the feature representations trained and tested on the CIFAR-10 and ImageNet datasets, respectively. We observe that representations learned via meta-learning (R2-D2 and ProtoNet) can achieve performance on par with SimCLR on CIFAR-10 but worse on ImageNet. Note that during training, we use the same exact hyperparameter as SimCLR due to computational constraints, which may not be optimal specifically for our proposed method. We will see in the following experiments that although R2-D2 achieves worse linear evaluation on ImageNet with this hyperparameter setting, it actually performs better than SimCLR on downstream tasks, such as semi-supervised learning and transfer learning, other popular (and plausibly more realistic) evaluation scenarios for SSL methods.

Following Chen et al. (2020a), we first evaluate the pre-trained model by semi-supervised learning, where we fine-tune the pre-trained model with only a fraction of labeled ImageNet data (1% and 10%). As we can see from Table 3, with the same fine-tune setting (See Appendix A.1), model pre-trained by R2-D2 can achieve $\sim 5\%$ higher top-1 accuracy than those pre-trained by SimCLR after fine-tuning with few labeled data. To be noticed, the supervised baseline from Zhai et al. (2019) is strong due to exhaustive hyperparameters searching and stronger data augmentations used during the training.

To further compare the feature representations learned by different methods, we apply the pre-trained weights to transfer learning. We consider 8 datasets with natural images of various categories (Nils-

Table 1: Linear evaluation on CIFAR-10 for representations learned via contrastive learning and our meta-learning framework.

Method	Backbone	Top-1 Acc(%)
SimCLR	ResNet-18	91.4
R2-D2	ResNet-18	91.6
ProtoNet	ResNet-18	91.8

Table 2: Linear evaluation on ImageNet for representations learned via contrastive learning and our meta-learning framework.

Method	Backbone	Top-1 Acc(%)
SimCLR	ResNet-50	58.8
R2-D2	ResNet-50	55.5
ProtoNet	ResNet-50	51.1

Table 3: ImageNet Top-1 accuracy (%) of models fine-tuned with few labels.

Method	Backbone	Label fraction	
		1%	10%
Supervised baseline	ResNet-50	25.4	56.4
SimCLR	ResNet-50	32.4	53.6
R2-D2	ResNet-50	**37.9**	**58.8**

Table 4: Transfer learning using ImageNet pre-trained weights. We report mean per-class accuracy (%) on the Flowers and Aircraft datasets, mean average precision (mAP) on the VOC2007 classification dataset, and Top-1 accuracy on the remaining datasets.

	Flowers102	DTD	VOC2007	Aircraft	Food101	SUN397	CIFAR-10	CIFAR-100
Baseline	92.0	64.8	67.3	85.9	**86.9**	53.6	95.9	80.2
SimCLR	92.4	72.7	66.0	83.7	86.3	57.4	94.8	79.1
R2-D2	**94.5**	**73.8**	**69.9**	**86.2**	**86.9**	**59.7**	**96.7**	**82.8**

back & Zisserman, 2008; Cimpoi et al., 2014; Everingham et al.; Maji et al., 2013; Bossard et al., 2014; Xiao et al., 2010; Krizhevsky et al., 2009). For each dataset, we use the backbone (ResNet-50) pre-trained on ImageNet as an initialization for the feature extractor of the downstream classification model. In contrast to linear evaluation, we fine-tune the entire model on the given dataset for 20,000 iterations with the best hyperparameter setting selected on its validation split. Details of our hyperparameter selection are included in Appendix A.1. Both SimCLR and R2-D2 are pre-trained on ImageNet for 100 epochs. We also include a baseline model provided in Chen et al. (2020a) which does not use pre-trained weight as an initialization. Note that the baseline model is tuned to achieve comparable performance with a larger search space for hyperparameters, and it is trained for a longer duration. From the results in Table 4, we find that R2-D2 initialized model consistently outperforms its contrastive counterpart on all 8 datasets. These results suggest that for the same number of epochs, a model trained with R2-D2 works better as an initialization for downstream tasks than one trained with SimCLR. We speculate that this property is connected to R2-D2's few-shot learning driven design and simulation of adapting to new tasks during its inner loop.

5 Boosting SSL with Meta-Specific Augmentation

Now that we have established a relationship between contrastive learning and meta-learning, we will apply tools developed in the latter discipline to enhance contrastive learners. Previous work has shown that data augmentations such as crops and colorizing play an important role in both contrastive learning and meta-learning. We focus on a particular augmentation strategy from the meta-learning literature, termed *task augmentation*, which aims to expand the number of classes available for sampling rather than expanding the number of samples per class. Liu et al. (2020); Ni et al. (2021) show empirically that data augmentations work best when applied to carefully chosen parts of the meta-learning batch, and large rotations can only work as a task augmentation, in which rotation by a chosen degree is applied to all images in an entire class, and we then treat them as a new class. Large rotations, and other dramatic transformations used for task augmentation, actually decreases performance when instead applied independently on support and query samples (as a way to increase data within a class) rather than uniformly on an entire class (therefore defining a new class). Augmentations that exhibit this task augmentation behavior are typically those which transform an image so much that its semantic content looks different to a human. By keeping images with very large augmentations in the same class, we may accidentally encourage models to learn overly strong invariances which do not naturally exist in the data.

We observe that this phenomenon occurs in contrastive learning as well. Namely, the same large rotation data augmentation, which can improve the performance of meta-learners via task augmentation, also degrades the performance of contrastive learners when applied to samples independently (instead of uniformly to an entire class). Table 5 shows linear evaluation accuracy on CIFAR-10

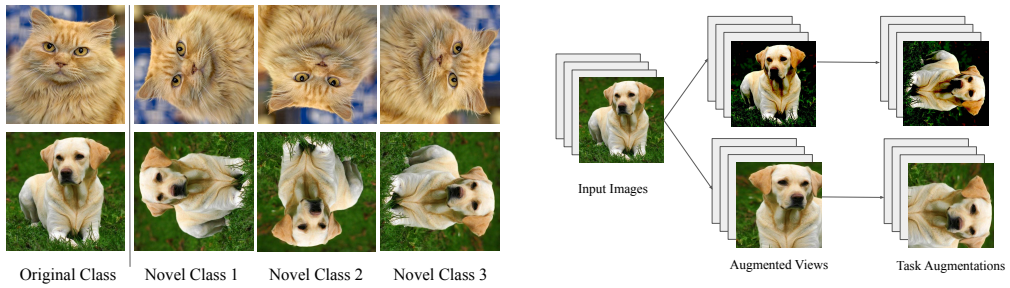


Figure 2: (a) Examples of novel classes created by rotation by $\{90°, 180°, 270°\}$. (b) Adding task augmentation into the existing data augmentation pipeline. We first apply random transformations on the input batch, then for each augmented view from the same image, we rotate them by the same degree.

when we add this augmentation to the training pipeline of SimCLR. In this experiment, we randomly rotate every augmented view by $\{90°, 180°, 270°\}$ with probability 0.25 each. In Table 5, we see that the accuracy of SimCLR drops by $\sim 2\%$, and this degradation also occurs in self-supervised learning methods without negative pairs such as BYOL, which adopts the same augmentation pipeline as SimCLR. Driven by the observation and insights from the meta-learning literature, we are motivated to apply strong augmentations, such as large rotations, to contrastive learning at the task level so that models can benefit from the additional augmentation without learning overly strong and harmful invariances.

In the literature on meta-learning literature for few-shot classification, large rotations can be used either as a task augmentation or an auxiliary loss (Su et al., 2020; Liu et al., 2020; Gidaris et al., 2019). We adopt both methods into our meta-learning inspired pipeline for self-supervised learning and describe the details below. This procedure is illustrated in Figure 2 and Figure 3.

Large Rotation as Task Augmentation. Instead of randomly rotating each training sample independently, we rotate all images from the same class (different augmented views of the same original base image) by the same degree (chosen randomly from $\{0°, 90°, 180°, 270°\}$) in both the support and the query data. In such a way, the number of potential classes is enlarged by 3 times. We combine large rotation augmentation with the basic augmentations used in contrastive learning during the sampling stage and keep other components of the training procedure unchanged.

Large Rotation as Auxiliary Loss. In addition to task augmentation, large rotations can be used as an auxiliary prediction task for the original self-supervised problem, where the angle of a rotated image is used as the target label. To this end, we spin the input batch $\mathbf{x}$ by an angle to generate the rotated training samples, $\{\mathbf{x}_d, \mathbf{y}\}$, where $d \in \{0°, 90°, 180°, 270°\}, \mathbf{y} = \{d/90\}$. Then, we stack a 4-way classification head r on top of the shared backbone f and projection head g, to predict the angle of the rotations evaluated with cross-entropy loss l_{LR}:

$$\mathcal{L}_{LR} = \sum_{(x,y)\in\{\mathbf{x}_d,\mathbf{y}\}} l_{LR}(r(g(f(x))), y). \tag{1}$$

During training, we sum this loss with the original self-supervised loss to achieve the final objective:

$$\mathcal{L} = \mathcal{L}_{SSL} + \lambda\mathcal{L}_{LR}, \tag{2}$$

Table 5: Linear evaluation (Top-1 accuracy (%)) on CIFAR-10 with feature representations learned by SimCLR and BYOL in default setting (see Section 3). Simply adding large rotations to data augmentation hurts the performance of self-supervised learning.

Rotation	SimCLR	BYOL
No	91.4	92.1
Yes	89.7	90.6

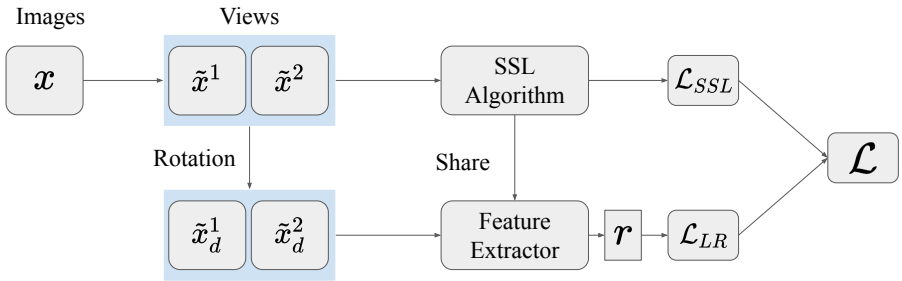


Figure 3: Training procedure of adding Large Rotation augmentation as an auxiliary loss. We randomly rotate each augmented view by a degree in $\{0°, 90°, 180°, 270°\}$. We share the feature extractor used in the specific SSL algorithm and apply it along with a 4-way linear head $r(\cdot)$ to do the rotation angle prediction. We sum up the SSL loss, $\mathcal{L}_{SSL}$, with the rotation angle prediction loss, $\mathcal{L}_{LR}$, as our ultimate objective, $\mathcal{L}$.

where λ is a coefficient that controls the influence of our prediction task.

In contrast to few-shot classification problems, the batch size on self-supervised tasks is often very large. If we rotate all images individually by four different degrees, we will triple the batch size which will consume valuable memory. Instead, we proposed to only sample a single rotation for each image. We conduct an ablation study on the number of rotations used to augment each batch on CIFAR-10 pre-raining, and we show that when the batch size is large, rotating each augmented view in the batch once is enough (see Appendix A.2)

Table 6 highlights the effectiveness of treating large rotations as a task augmentation or as an auxiliary loss on pre-trained CIFAR-10 representation in the default setting. Notably, our method enables BYOL to achieve 94.6% accuracy with a backbone trained entirely on unlabeled data, just as high as models of the same architecture trained in a fully supervised setting (but without our augmentations) (Kuang et al., 2017). For task augmentation, we apply large rotations with the probability of 0.25 each on top of the baseline augmentations. For the rotation angle predictor loss, we weight the additional loss term with coefficient $\lambda = 1$ for all experiments except for pre-training with R2-D2, where we set $\lambda = 0.01$. In Table 6, we see that by using large rotations as a task augmentation, we can avoid the massive accuracy drop observed in Table 5. In addition, when we add the angle prediction loss for large rotation as an auxiliary loss, we boost the linear evaluation of all the pre-trained representations by at least 1%. We further show that our proposed method can be combined with existing data augmentation methods by adding harder examples with mix-up or cut-mix in Table 6. Following Shen et al. (2020), we mix one augmented view from each base image with an augmented view from a second image where the indices used for the second augmented view are reverse ordered. In order to mix images, we randomly apply mix-up or cut-mix with equal probability. Our proposed large rotation prediction loss consistently improves performance on top of the mixing augmentation by $\sim 1\%$ on all pre-training methods, thus leading to more than 2% improvement from the baseline augmentations. On ImageNet, Table 7 displays the top-1 linear evaluation accuracy for methods with and without our proposed rotation angle prediction loss. With the proposed auxiliary loss, we achieve 0.8% improvement on the representation trained for 100 epochs.

6 Meta-Learning Batch Strategies for SSL: More Negative Pairs or Batches of Tasks?

The training cost of contrastive learning methods may be prohibitive to practitioners who do not possess vast compute resources with sufficient memory to accommodate the enormous batch sizes used in competitive algorithms. Large batch sizes contain more negative pairs, and it is widely believed that these negative pairs are essential for contrastive learners such as SimCLR. With smaller batch sizes, methods such as SimCLR suffer a large performance drop (around 3.5% when dropping the batch size from 2048 to 256). In addition, there is no natural way to split these large batches across multiple sub-batches and perform gradient accumulation since every image must be compared to every other sample in the batch during the loss calculation. As a result, achieving comparable results

Table 6: Linear evaluation (Top-1 accuracy (%)) on CIFAR-10 with representations learned via different augmentations."Baseline" denotes the augmentations used in SimCLR, "+ TA" denotes adding large rotation into the baseline pipeline as task augmentations, "+ $\mathcal{L}_{LR}$" denotes adding large rotation as an auxiliary loss, and "+ Mix" denotes adding image mixing augmentation (Shen et al., 2020).

Augmentations	SimCLR	R2-D2	ProtoNet	BYOL
Baseline	91.4	91.6	91.8	92.1
+ TA (ours)	91.6	91.3	92.0	92.2
+ $\mathcal{L}_{LR}$ (ours)	93.0	92.9	93.0	93.1
+ Mix	92.8	93.2	93.2	93.8
+ Mix + $\mathcal{L}_{LR}$ (ours)	**94.0**	**93.8**	**94.1**	**94.6**

Table 7: Linear evaluation on ImageNet with representations learned by SimCLR and with proposed augmentations.

Model	Top-1 (%)
SimCLR	58.8
SimCLR + $\mathcal{L}_{LR}$ (ours)	**59.6**

Table 8: Effects of gradient accumulation. Performance is the Top-1 accuracy (%) on ImageNet evaluated after 100/200 epochs.

Model	100 ep	200 ep
SimCLR	57.5	61.9
SimCLR + grad acc.	**58.8**	**63.2**

without massive computation resources may appear unattainable. In meta-learning, we instead typically sample a large batch of tasks (i.e. 1-shot-b-way classification problems) instead of one single huge classification problem with many classes. This strategy can be directly applied to contrastive learning and allows us to split large batches of tasks into individual tasks and accumulate gradients. For example, instead of training on 1-shot-2048-way tasks in the large batch setting, we can train the feature extractor with 8 batches of 1-shot-256-way tasks. In such a way, we can accumulate the gradients across batches of training tasks instead of foregoing contrastive learning due to memory constraints.

We present linear evaluation results for representations learned by SimCLR with and without gradient accumulation in Table 8. We use the default setting in Section 3 where the batch size is 256 and the accumulation step contains 8 sub-batches when used. In addition, we extend the training length to 200 epochs to further demonstrate the effectiveness of the proposed method. With gradient accumulation, we can consistently achieve improvement by more than 1% on both representations pre-trained with 100 and 200 epochs, growing between these epochs, which shows that in addition to negative pairs, more tasks in a training batch can be helpful as well. This trick offers a simple but effective way to improve the pre-trained representation performance when GPU memory is limited.

7 Conclusion

In this work, we discuss the close relationship between contrastive learning and meta-learning. In doing so, we proposed a new meta-learning based framework for self-supervised learning by converting the contrastive setup into on-the-fly image classification tasks. We show that feature extractors pre-trained via meta-learning can achieve comparable results to contrastive learning based methods and, in fact, they can transfer better to downstream classification in our testing cases. In addition, we leverage data augmentation ideas from meta-learning and incorporate them into contrastive learning. We demonstrate that the proposed meta-specific data augmentation consistently improves the performance of popular self-supervised learning algorithms. Finally, from the perspective of meta-learning, we show that we can improve the performance of contrastively learned representations trained with a limited memory footprint by accumulating gradients from a batch of training tasks during training, which further supports the close relationship between contrastive learning and meta-learning.

REPRODUCIBILITY STATEMENT

Our work is fully reproducible. We include the code in the supplementary materials, and we discuss the training details in Section 3 and evaluation details in Appendix A.1.

REFERENCES

Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750–15758, 2021.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pp. 1422–1430, 2015.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8059–8068, 2019.

Micah Goldblum, Steven Reich, Liam Fowl, Renkun Ni, Valeriia Cherepanova, and Tom Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. *arXiv preprint arXiv:2002.06753*, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pp. 87–94. Springer, 2001.

Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *arXiv preprint arXiv:2010.01028*, 2020.

Sungnyun Kim, Gihun Lee, Sangmin Bae, and Se-Young Yun. Mixco: Mix-up contrastive learning for visual representation. *arXiv preprint arXiv:2010.06300*, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Liu Kuang, Yang Wei, Yang Peiwen, and Ducau Felipe. Train cifar10 with pytorch, 2017. URL `https://github.com/kuangliu/pytorch-cifar`. https://github.com/kuangliu/pytorch-cifar.

Seong Min Kye, Hae Beom Lee, Hoirin Kim, and Sung Ju Hwang. Transductive few-shot learning with meta-learned confidence. *arXiv preprint arXiv:2002.12017*, 2020.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Chunyuan Li, Xiujun Li, Lei Zhang, Baolin Peng, Mingyuan Zhou, and Jianfeng Gao. Self-supervised pre-training with hard examples improves visual representations. *arXiv preprint arXiv:2012.13493*, 2020.

Jialin Liu, Fei Chao, and Chih-Min Lin. Task augmentation by rotating for meta-learning. *arXiv preprint arXiv:2003.00804*, 2020.

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6707–6717, 2020.

Renkun Ni, Micah Goldblum, Amr Sharaf, Kezhi Kong, and Tom Goldstein. Data augmentation for meta-learning. In *International Conference on Machine Learning*, pp. 8152–8161. PMLR, 2021.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pp. 69–84. Springer, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Zhiqiang Shen, Zechun Liu, Zhuang Liu, Marios Savvides, Trevor Darrell, and Eric Xing. Unmix: Rethinking image mixtures for unsupervised visual representation learning. *arXiv preprint arXiv:2003.05438*, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017.

Jong-Chyi Su, Subhransu Maji, and Bharath Hariharan. When does self-supervision improve few-shot learning? In *European Conference on Computer Vision*, pp. 645–666. Springer, 2020.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 776–794. Springer, 2020a.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *arXiv preprint arXiv:2005.10243*, 2020b.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010. doi: 10.1109/CVPR.2010.5539970.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1476–1485, 2019.

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pp. 649–666. Springer, 2016.

A APPENDIX

A.1 EVALUATION DETAILS

Linear Evaluation. For ImageNet, we follow similar procedure as Chen et al. (2020a), where we use a batch size 4096, a larger learning rate of 1.6 (following LR = 0.1 × BatchSize/256) and longer training of 90 epochs. For CIFAR-10, we train the linear classifier for 200 epochs with batch size 1024, and a linear decay learning rate schedule starts with rate 0.1. We use SGD as the optimizer for linear evaluation on both datasets.

Semi-supervised Learning. Following Chen et al. (2020a), we fine-tune the pre-trained model with 1% and 10% labeled ImageNet data. Due to the computational limit, instead of using batch size 4096, we use the SGD optimizer with a batch size of 256 and momentum of 0.9. We use a linear decay learning rate schedule starts with rate 0.05. We only use the regular data augmentation for ImageNet training (random crop and resize). We do not use any regularization such as weight decay. For both 1% and 10% of the labeled data, we fine-tune for 60 epochs. The supervised baseline results are imported from Zhai et al. (2019), where they trained several thousand models for hyperparameter tuning and using strong data augmentations.

Transfer Learning. We evaluate the performance of models in transfer learning through fine-tuning. We use SSL pre-trained feature extractors with a linear classification head as our model in transfer learning, and we update the entire network during fine-tuning. In this experiment, we use an SGD optimizer with Nesterov momentum (momentum=0.9). We set the batch size to be 256 and train our models for 20,000 iterations. For each dataset, we follow the training/validation/testing split in Chen et al. (2020a) and perform the hyperparameter search on the validation set. We search through different learning rates (ranging from 10^{-2} to 10^{-1}) and weight decays (ranging from 10^{-6} to 10^{-3}, and 0), and choose the best setting by comparing the performance on the validation set. Results in Table 4 are evaluated on the test split, and our models are trained on the union of training and validation set, using the best hyperparameter settings we have found.

A.2 THE SAMPLE SIZE FOR LARGE ROTATION AUXILIARY LOSS

Table 9 shows the linear evaluation accuracy for representations learned by different self-supervised learning algorithms with different rotation methods. A full rotation method rotates all the augmented views by 3 times, a half rotation method rotates half of the augmented views by 3 times, and the random rotation only rotates all the augmented views by an angle randomly chosen from $\{0°, 90°, 180°, 270°\}$. With different rotation methods, the number of training examples for rotation angle prediction is different. Among all the experiments, we use the same batch size 1024, and the full rotation will generate $1024 \times 2 \times 4$ training examples, the half rotation will generate $512 \times 2 \times 4$ examples and random rotation will only generate 1024×2 samples, where 2 represents for the number of the random augmentations and 4 is the length of the degree set. From Table 9, we can see that even if we have much fewer training samples when applying random rotation, we can still achieve comparable results. As a result, we use random rotation in all the experiments as it makes the training more memory efficient.

Table 9: Linear evaluation (top-1 accuracy (%)) on CIFAR-10 with representations learned by various SSL methods with different rotation methods.

Method	SimCLR	R2-D2	PN
Random Rotation	93.0	92.9	93.0
Full Rotation	92.6	92.8	92.9
Half Rotation	92.9	92.6	92.7