What Should Be Equivariant In Self-Supervised Learning

Yuyang Xie[1], Jianhong Wen[1,2]*, Kin Wai Lau[1,3], Yasar Abbas Ur Rehman[1], Jiajun Shen[1]
[1]TCL AI Lab, [2]Fuzhou University, [3]City University of Hong Kong
{yuyang.xie, jianhong1.wen, stevenlau, yasar, sjj} @tcl.com

Abstract

Self-supervised learning (SSL) aims to learn feature representation without human-annotated data. Existing methods approach this goal by encouraging the feature representations to be invariant under a set of task-irrelevant transformations and distortions defined a priori. However, multiple studies have shown that such an assumption often limits the expressive power of the representations and model would perform poorly when downstream tasks violate this assumption. For example, being invariant to rotations would prevent features from retaining enough information to estimate object rotation angles. This suggests additional manual work and domain knowledge are required for selecting augmentation types during SSL. In this work, we relax the transformation-invariance assumption by introducing a SSL framework that encourages the feature representations to preserve the order of transformation scale in embedding space for some transformations while maintaining invariance to other transformations. This allows the learned feature representations to retain information about task-relevant transformations. In addition, this framework gives rise to a handy mechanism to determine the augmentation types to which the features representations should be invariant and equivariant during SSL. We demonstrate the effectiveness of our method on various datasets such as Fruits 360, Caltech-UCSD Birds 200, and Blood cells dataset.

1. Introduction

There is an emerging interest in using self-supervised learning (SSL) approaches for learning feature representations with unlabeled data. Such SSL approaches include contrastive learning methods such as SimCLR [2], MOCO [13], MOCO-V2 [3] and PIRL [20], where representations are often learned by minimizing embedding distances between the different augmented views of same images and maximizing the embedding distances between different images. Another category of SSL approach consists of non-contrastive methods such as BYOL [11], SimSiam [4] and Barlow-twins [35] manage to learn feature representations without the use of negative samples while avoiding undesired trivial solutions. Regardless of the category of SSL, a careful choice of transformations is nevertheless required to generate different views of the input data and the trained feature representations are encouraged to be invariant to these transformations.

However, a few recent work [7, 33] show that it may be harmful to enforce transformation invariance for some downstream tasks. For example, as shown in [33], adding color jitters in the augmentation may harm the performance of the model designed to classify objects with similar appearances but different colors. Similarly, [9] points out the issue of dismissing intra-class variance, proposing a new framework that preserves the local structure change of intra-class samples in the embedding space. However, their method still requires negative samples to learn inter-class variance and therefore existing non-contrastive approach cannot naïvely apply this approach. Moreoever, SSL training often relies on choosing transformation types and scales to generate image views, which would take considerable amount of trials and errors [27]. Most SSL methods have been adopting a pre-defined set of transformations to generate image views during training and their works have been based on often implicit assumption that such a selection would be generic to any downstream tasks. [16, 27, 29, 31] propose different ways of automatically generating optimal views to help the model learn more efficiently. However, these methods focus on acquiring feature representations that are invariant to the transformations. As a result, they may not capture the important information when applied to a downstream task that requires transformation-equivariant feature representations, i.e., feature representations that varies according to the transformation via a mapping function [17].

In this paper, we present a SSL framework that learns both transformation-invariant and transformation-equivariant representations by correlating the feature embeddings of different image views with the corresponding

*This work was done during Jianhong Wen's internship at TCL

transformation scales orders. To avoid trying out different transformation types manually, we then leverage the proposed framework and further design a procedure to determine the transformation type to which the feature representations should be equivariant and invariant for a specific downstream task without training. The main contributions of this work can be stated as follows:

1. Unlike existing SSL methods which can only learn transformation-invariant feature representations, we extend these methods by developing a SSL framework that learns both transformation-invariant and transformation-equivariant features.
2. Based on the framework above, we provide a procedure to determine the transformation types to which the feature representations are invariant and equivariant during the training process.
3. We validate the effectiveness of our proposed approach on several benchmark datasets and show significant performance improvement over baseline methods.

The code to our methodology will be released.

2. Related Works

2.1. Self-Supervised Learning

Self-supervised learning methods aim to learn supervision information from unlabeled data. One promising direction is to learn feature representations by maximizing agreement between differently augmented views of the same data example via a contrastive or non-contrastive loss in the latent space [2, 20]. The contrastive learning based SSL techniques have been successful in learning view-invariant representations - by bringing feature representations of the positive pairs closer and repelling apart the feature representations of the negative pairs [1, 2, 28, 30]. Non-contrastive learning methods, on the other hand, manage to learn useful feature representations with only positive pairs. These methods typically employ a dual pair of siamese networks composed of an online network and a target network which requires no gradient. For example, BYOL [11] employs a momentum encoder that slowly follows the online network in a delayed fashion through an exponential moving average while SimSiam [4] uses a direct copy of the online network to form the target network. Barlow Twins [35] enforces the cross-correlation matrix between outputs of a positive pair close to the identity matrix.

2.2. Selecting the Appropriate Transformation Types

Regardless of which SSL approaches mentioned above, as [2] points out, the choice of the composition of multiple data augmentation operations in SSL is crucial for producing effective representations. Yet we often lack efficient ways to determine which data augmentation operation to choose and it often requires domain expertise and well understanding of the downstream tasks to make such a choice. InfoMin [29] proposes a way of selecting good views for a given task by minimizing the mutual information between views while keeping task-relevant information. Nevertheless, this method requires transfer-task knowledge to select appropriate augmentation. Viewmaker [27] proposes a generative model to learn the appropriate views without domain knowledge. However, this method cannot generate views with structural changes such as cropping and rotation, so the feature representations may lack generalization capacity. AutoAugment [6], PBA [15], Fast AutoAugment [19] and Faster AutoAugment [12] propose algorithms that find the optimal augmentation policy from a self-designed search space contains various transformations, but these methods all require training additional networks to learn the strategy.

2.3. Transformation Invariance and Equivariance

The SSL methods mentioned above ignore the variance among the positive samples by enforcing the features to be invariant to the augmentation applied. In [33], the authors show that training a set of transformation-invariant feature representations can cause poor performance of the model in the downstream task. They present a framework that helps capture transformation variant and invariant visual representations by constructing separate embedding spaces, each of which is invariant to all but one transformation. Note that in their work, although the features are variant to transformation, they are not equivariant to the transformation as they cannot express the similarity between different augmented views. In [9], the authors present a novel framework which can capture the features that are sensitive to the local structure change of the input image by designing a new loss function. However, they simply composite different types of transformations with ordered scales, and it may not be the optimal choice for every task. Some other works like [5, 7, 10, 18, 25, 26] have shown the advantages of transformation-equivariant feature representations in some computer vision tasks. However, they do not provide a method to determine which transformations should be invariant or equivariant when training the network.

3. Method

Our objective is to learn feature representations that are equivariant to only a specific set of transformations and invariant to others. To achieve such kind of property, we correlate the feature embeddings of augmented views with the corresponding transformation scales. The model architecture and the loss design are described in Section 3.1 and Section 3.2. In addition, since the optimal strategy of choosing transformations is task-dependent [8, 24, 34], in Section 3.3 we propose a procedure to determine the transfor-

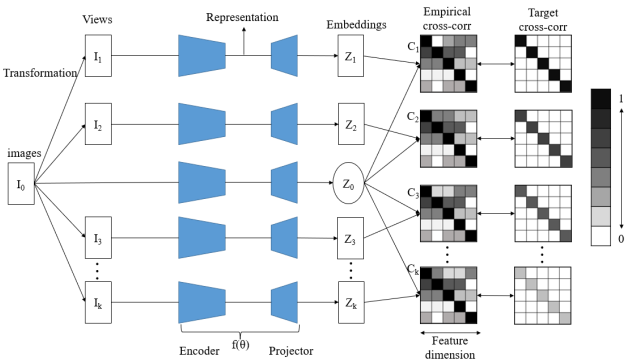

Figure 1. Overview of model structure: The structure consists of an ordered scales of transformation modules, followed by encoders, and projection heads. The model learns the transformation-equivariant features by correlating mean of diagonal elements of correlation matrix with transformation scales.

mation type to which the features should be variant for a given downstream task without trying all the types of transformations during training. By following this procedure, we can find the optimal setting to train our neural network without consuming lots of computation resources.

3.1. Model Architectures

For each augmentation type, we generate a series of distinct views of each data sample using a set of scales of an augmentation that are ordered by their relevant strength. This is in contrast to the conventional random augmentations to generate different views of the data. For each input image denoted as I_0, we generate a series of k transformed views $(I_1, I_2, \ldots, I_k)$ ordered by the k scales of an augmentation as shown in the Figure 1. Together with an input image I_0, these k transformed views of the input image are mapped to their corresponding feature representations through an encoder followed by the projector to generate the feature embeddings. The projector consists of three linear layers that reduces the dimensionality of the feature representations. Similar to Barlow-Twins, we calculate the correlation matrix between the feature embeddings of each transformed view against the original view to get a series of k correlation matrices, $C^1, C^2, \ldots, C^k$. To decorrelate the feature components, the off-diagonal elements of these metrices are encouraged to be zero, which are mostly considered as the redundancy between the components of these features. However, unlike Barlow Twins, which equates the diagonal elements of all correlation matrices to be one, here we want the mean of diagonal elements of the corresponding correlation matrix to follow the order of the transformation scale n, i.e:

$$\frac{1}{D}\sum_{i=1}^{D} C^{n}_{i,i} \geq \frac{1}{D}\sum_{i=1}^{D} C^{n+1}_{i,i} \tag{1}$$

where i, i indexes the diagonal elements, and D represents the feature dimensions.

3.1.1 Generating Transformations

Following the above procedure, we now discuss the procedure to generate a series of ordered scales of transformation for each augmentation type that are applied to the input images. We define T_{ordered} to represent the ordered scale of transformations for an augmentation that results in equivariant feature representations when applied to the input images. The T_{random} in contrast represents the scale of transformations for an augmentation that results in invariant feature representations when applied to the input images.

For T_{ordered}, we randomly divide the total scale range into k non-overlapping sub-ranges. The degree of separation between the sub-ranges is controlled by the hyper-parameter d. We randomly pick the scales from each sub-range with a truncated Gaussian sampling centering around the mid-point of each sub-range to maintain the order of sampled scales and a considerably large distance between two adjacent scales. For instance, if we learn the feature representations that are equivariant to cropping, we generate k different cropping ratio (i.e., $T_{\text{ordered}} = \{c_1, c_2, \ldots, c_k\}$ where c_i is the cropping ratio for i-th view and $c_1 > c_2 > \cdots > c_k$). For the augmentation types that are used to learn invariant feature representations, we generate the set T_{random} of transformations using a random scale. All the transformations can be applied together to generate the views I_i.

Extending to the case where we need to learn multiple types of transformation-equivariant feature representations, we simply composite the series of ordered scales of different transformations together. Suppose we have n types of transformation with k ordered scales for each, we can form n sets of T_{ordered}. We then randomly sample k values from each T_{ordered} and rank the lists in ascending order. We finally concatenate these n scale lists element-wisely. Concretely, if we have two types of transformations such as cropping c and rotation r, we firstly generate a T_{ordered} (e.g., $\{c_1, c_2, \ldots, c_k\}$ and $\{r_1, r_2, \ldots, r_k\}$) for each type of transformation, and we randomly pick n elements from it with replacement to form a new scale list (e.g., $c_2, c_2, c_3, ..., c_n$ and $r_1, r_2, r_2, ..., r_{n-1}$). Lastly, we concatenate both scale lists element-wise to form a final transformation set (i.e., $c_2r_1, c_2r_2, c_3r_2, ..., c_nr_{n-1}$ in this example). By combining different transformations this way, the model can benefit from varies of views. We further perform this study in Section 5.3.

3.2. Loss function

Inspired by Barlow-twins, our loss function operates on the correlation matrices of the embeddings obtained from

different transformed views of the input image. Unlike Barlow-twins, it encourages the values of the diagonal elements of the correlation matrix to inversely correlate to the scale of the transformation, so that feature embeddings of a less transformed image will be more correlated with the feature embeddings of the original image. Since we have k correlation matrices, obtained by correlating the embeddings of the input image with the embeddings of its each k transformed views, it implies that the mean of diagonal values of $k-1$ correlation matrix should be higher than the mean of the diagonal values k correlation matrix, i.e.,

$$\frac{1}{D}\sum_{i=1}^{D} C_{i,i}^{n} \geq \frac{1}{D}\sum_{i=1}^{D} C_{i,i}^{n+1} + \alpha \tag{2}$$

where, n $\in$ 1, 2, 3, . . . , k-1, and α is a hyperparameter that controls the degree of the margin between the two mean values. Taking such prospect of the loss function into consideration, the final loss is a combination of three loss components, viz., L_{ranking}, L_{positive}, and L_{offdiag}. The ranking loss L_{ranking} is defined as:

$$L_{\text{ranking}} = \sum_{n=1}^{k} \text{Relu}(\alpha - d_{n-1} + d_n) \tag{3}$$

where $d_n = \frac{1}{D}\sum_i C_{i,i}^{n}$.

Since all the transformed views of the input image represent positive samples, the feature representations of the transformed views should remain positively correlated with the feature representations of the original image. We design an additional loss term L_{positive} to reflect this constraint:

$$L_{\text{positive}} = \text{Relu}(\beta - d_n) \tag{4}$$

where β is a hyper-parameter which control the lower bound of the correlation between the input image and the n transformed view of the input image.

Also as suggested in Barlow-Twins, we further try to equate the off-diagonal elements of the correlation matrices to zero to avoid the redundancy among feature components. Therefore, we include another loss term L_{offdiag} specified below:

$$L_{\text{offdiag}} = \frac{1}{D(D-1)} \sum_{n=0}^{k} \sum_{i,j,i\neq j} |C_{i,j}^{n}| \tag{5}$$

The final loss is a linear combination of L_{ranking}, L_{positive}, and L_{offdiag} as:

$$L = L_{\text{ranking}} + L_{\text{positive}} + L_{\text{offdiag}} \tag{6}$$

3.3. Selection of transformation

In this section, we consider the problem of selecting the optimal transformation types during SSL. As suggested in the previous works [8, 34], it is important to select the optimal set of transformation types in SSL pretraining for maximum performance for a particular downstream task. However this often requires an exhaustive search over the possible sets of all transformations types, which is computationally expensive and infeasible in practice, particularly when performing SSL with large-scale unlabeled datasets. To alleviate this problem, we study the transformation-equivariant feature representations obtained by our proposed approach (Section 3.1).

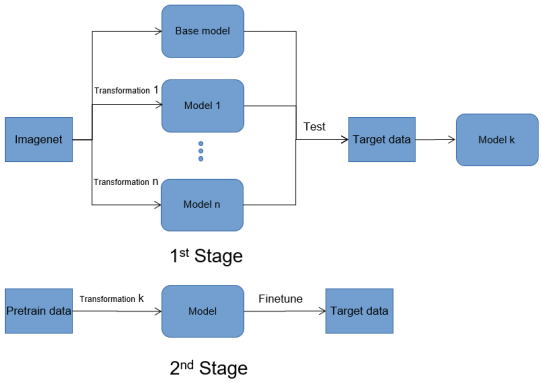

Figure 2. Overview of our procedure of how to select the appropriate types of transformation without training.

We show our selection procedure in Figure 2. Assume we have n collection of models that are pretrained on a reference dataset (e.g., ImageNet) using our proposed approach in Section 3.1, i.e., each model learns feature representations that are equivariant to one of the transformation types and invariant to other transformation types. Additionally, we have a base model which learns feature representations that are invariant to all the transformation types. We use image retrieval to compare the performance of the each individual model in our collection against the base model. For this purpose, we use the feature representations of the test set to query similar feature representations in the training set. The class label of each image in the test set can then be determined by the majority voting of K nearest feature representations in the training set. We hypothesize that if a certain model achieves better performance compared to the base model, then the transformation type to which the feature representations are equivariant would be task-relevant. Eventually, we can determine a subset of transformation types to which the features should be equivariant for a given downstream task and these choices can be used when we perform SSL on a different large-scale unlabeled dataset for the downstream task. We validate our hypothesis and procedure above in Section 4.3.

3.4. Implementation Details

We use PyTorch to implement our method on a GTX 3090Ti GPU with 24GB memory. All the input images are resized to 64 $\times$ 64 during training and testing due to the limitation of our computation resource. In our experiments, we pick three transformations including cropping, rotation and color jittering, and examine how the pretrained model would behave when we apply these transformations with ordered scales. In addition, we randomly apply horizontal flipping, Gaussian blurring and grayscale conversion to the input. It should be noted that our approach can be used with any transformation types. We set the number of ordered scale $k = 4$, which implies that each transformation type will be applied 4 times to every image to generate the distorted views for training. The training time of our method is around 2.5$\times$ that of Barlow-Twins due to multiple views generated for training. The computation time for the fine-tuning and inference stage is the same as Barlow-Twins.

Throughout our experiments, we use ResNet50 [14] as the feature encoder, followed by a projector network with the output dimensions of 128. For the pretraining stage, we follow the optimization protocol in Barlow-twins. We use the Adam optimizer and train for 1000 epochs with batch size of 128. The learning rate is set to 0.001 with a weight decay of 10^{-6}, and a learning rate warm-up period is set to 10 epochs. For the fine-tuning stage, we train a linear classifier for 200 epochs with batch size of 512. The hyper-parameters α is set to 0.05 and β is set to 0.8. The scale ranges of the transformations are set to be the same as those used in Barlow-twins.

4. Experiments

4.1. Datasets

We evaluate our models on four datasets, including the Fruits 360 dataset [22], Caltech-UCSD Birds 200 (CUB-200-2011) [32], Oxford 102 Flower [23] and the Blood Cells dataset [21]. Due to the relatively small-scale of the later three datasets, we use the whole training set for both SSL pretraining and fine-tuning to ensure enough data is available for obtaining expressive feature representations during SSL pretraining.

The Fruits 360 dataset [22] It consists of 90483 images of 131 different types of fruits. Unlike the rest of the other datasets, here we split the dataset into two parts, one for SSL and one for downstream task. We select images from 63 classes to form a dataset for downstream task evaluation and use the images from the other 68 classes for SSL pretraining.

Caltech-UCSD Birds 200 (CUB-200-2011) [32] It contains 11,788 images of 200 species of birds. Each class consists of 50 to 60 images.

Oxford 102 Flower [23] This dataset is an image classification dataset consisting of 102 categories of flowers. There are around 7000 images and each class consists of between 40 and 258 images.

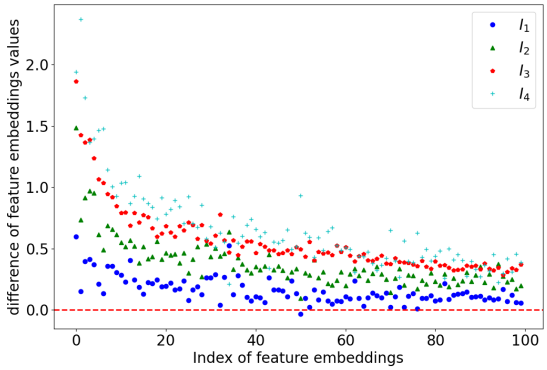

Figure 3. We display the difference of the averaged feature values of each augmented view and the input image. $I_1, ..., I_4$ are generated by applying color jitters with 4 ordered scales to the each input image from the Fruit 360 dataset.

The Blood Cells dataset [21] It contains 12,500 augmented images of blood cells. There are approximately 3,000 images for each of 4 different cell types which are Eosinophil, Lymphocyte, Monocyte, and Neutrophil.

4.2. Evaluation on transformation-equivariant features

First, we conduct an experiment on the Fruit 360 dataset to verify that the model can be trained to extract features that are equivariant to a specific type of transformation by following our method described in Section 3.1. We choose color-jitters as the transformation type, and follow the steps in Section 3.1 to learn a transformation-equivariant feature extractor. We randomly select 1000 images from the Fruit 360 dataset and apply color-jitters with four different ordered scales to each image followed by extracting the corresponding feature representations. We then calculate the average embedding values across the image batch for each transformation scale. Figure 3 shows how these average embedding values change for different transformation scale. Here, the y-axis displays the difference between the averaged feature values across each batch of four augmented views and those across the batch of original images. We can see from the Figure 3 that as the augmentation scale increases, the differences of the feature values of the augmented views and the original input images become larger. This demonstrates that the feature representations extracted from this model are indeed equivariant to color-jitters.

Secondly, we compare the performances of our methodology against the baseline SSL method, Barlow-Twins, that only learns transformation-invariant feature representations. We notice that Barlow-Twins [35] does not learn rotation-invariant feature representation in the original setting. To demonstrate the effectiveness of rotation-

Pretraining Datasets	barlow-twins	barlow-twins (w/ rotation)	cropping	color-jitter	rotation
CUB w/ finetune	19.70	15.30	22.75	**25.50**	21.90
Fruits 360 w/ finetune	92.51	96.19	94.57	**96.74**	96.58
Blood cells w/ finetune	58.30	49.30	70.89	74.47	**81.75**
Oxford 102 Flowers w/ finetune	**78.38**	77.65	77.75	74.5	77.88

Table 1. Evaluation results on the Caltech-UCSD Birds, the Fruits 360 dataset, the Oxford Flowers and the Blood cells dataset.

equivariance, we also include random rotation when generating views when training the Barlow-Twins baseline for a fair comparison, and the results are shown in table 1 column 3. Since the rotation-invariant feature deteriorates all datasets' performance except Fruits 360, we does not include it when we study the cropping and color-jitter equivariant feature representation. For our method, we use color-jitters, crop and rotation as the transformation types to which the feature representations are equivariant and the rest as transformation types to which the feature representations are invariant. These models are pretrained on the pretraining dataset specified in Section 4.1 and fine-tuned on the corresponding fine-tuning dataset. As shown in Table 1, each column shows a type of transformation that the model is equivariant to, and the Barlow-Twins is used as the baseline that is invariant to all these transformations. The evaluations on the CUB, the Blood cells and the Fruits 360 datasets demonstrate that the transformation-equivariant feature representations can perform better than the transformation-invariant representations in some tasks. However, transformation-equivariant feature representations are less expressive for the Oxford 102 Flower dataset since the classes of this dataset cannot be discriminated by the features equivariant to these transformations. For example, the same type of flowers may have multiple colors, and the flowers with the same color may belong to different categories.

Based on these evaluations, we conclude that having transformation-equivariant features and transformation-invariant features could benefit different cases. It is crucial to determine the optimal choice of transformation types before pretraining, and such a selection is task-dependent. For this reason, we need a method to efficiently determine the appropriate choice of transformations for a given task, as described in Section 4.3, and we conduct experiments to demonstrate the efficacy of our method.

4.3. Evaluation on the selection of transformations

In this section, we first want to validate the hypotheses made in Section 3.3, i.e., for a particular transformation type, if we find using transformation-equivariant feature representations perform better for a specific downstream task, we deem this particular transformation type crucial to be included in pretraining, even if the model is trained on a different dataset. We then follow the procedure in Section 3.3 and show how one can select the optimal transformation types effectively.

We use the Tiny-ImageNet as the reference dataset on which we train the SSL models. We first train three models following the method described in Section 3.1. Each model is trained to be equivariant to one of the transformations, i.e., cropping, color jitters, and rotation. We also train a baseline Bawlow-Twins model for reference purpose. As shown in the first row of Table 2, we pretrain different models on Tiny-ImageNet and evaluate the downstream task performance by KNN, following the steps in Section 3.3. Since all these models outperform the baseline, we believe it is beneficial to include all of the transformation types when training for transformation-equivariant features, as verified in Table 6. Indeed we show that when pretraining on Fruits 360, it is also beneficial to include all transformation types when training transformation-equivariant features. Moreover, we find the performance of the models pretrained on the Fruits 360 is positively related to that of the models pretrained on Tiny-ImageNet, which verifies our hypothesis. The same conclusion stands for the scenario when the labeled data is limited as well. As we can see from the Table 2, even when we conduct our experiments on the few-shot learning cases where we only have 10 images per class for fine-tuning the pretrained model, the performance of the fine-tuned model on the target set are still positively related to the performances of models pretrained on the reference dataset. We conduct the same experiments on Blood cells dataset and reach the same conclusion.

Similarly with the Oxford 102 Flowers dataset, we conduct the above analysis and find that the transformation-invariant features are more effective than the transformation-equivariant features when pretrained on reference dataset. Therefore we derive the conclusion that we should encourage the models to learn transformation-invariance representations for the downstream dataset, which is also confirmed in Table 3 as we show that the baseline Barlow-Twins model without transformation equivariance performs the best.

Test on Fruit-360				
Pretrained Datasets	barlow-twins	cropping	color-jitter	rotation
Tiny-ImageNet w/o finetune	82.16	83.39	**88.59**	84.43
Fruits 360 w/o finetune	75.79	84.10	**92.90**	89.49
Fruits 360 finetuned	92.51	94.57	**96.74**	96.58
Fruits 360-finetuned-10 shots	72.50	75.11	**88.17**	86.20
Test on Blood-cells				
Tiny-ImageNet pretrained	37.60	41.82	43.71	**49.14**
Blood cells-finetune	58.30	70.89	74.47	**81.75**

Table 2. Comparison of the evaluation results of different models on Fruits 360 and Blood cells dataset with Tiny-ImageNet as reference dataset.

Pretraining Datasets	Barlow-Twins	cropping	color-jitters	rotation
Tiny-ImageNet (w/o finetune)	**33.1**	30.0	30.0	28.25
Tiny-ImageNet (w/ finetune)	62.63	**63.75**	58.75	59.63
Oxford Flowers (w/ finetunue)	**78.38**	77.75	74.50	77.88

Table 3. Comparison of the evaluation results of different models on Oxford 102 Flowers dataset.

5. Ablation Study

In this section, we conduct a few ablation studies to analyze how different numbers of ordered scales, transformation scale sampling methods, transformation composition methods and cropping methods would affect the performance of our model.

5.1. Number of ordered transformation scales

Firstly, we conduct an experiment to explore the effect of the number of ordered scales of transformation, denote by k as described in section 3.1, on the performance of the model. In this experiment, we apply cropping with different values of k and follow the method described in section 3.1 for SSL pretraining. We vary the numbers of cropping ratios during training and see how that would affect the performance. Note that since our method require at least two correlation matrices for comparison, the minimal value of k is 2. In addition, for a fair comparison, the ranges of cropping ratio are kept the same regardless of the number of ratios we use during training. We train and evaluate on the Fruit 360 dataset following the setup described in Section 4.1 and compare the accuracy between the models trained with different values of k in Table 4. Surprisingly, we find the model achieves the best performance when k equals to 2 and 3, and the performance starts to drop as k increases in the training process. We conclude that even though transformation-equivariant features are beneficial for a specific task, simply increasing the number ordered scales of transformation may hurt the performance. We conjecture that as the number of transformation scales increases, the difference between the generated views will decrease, and the model trained this way will extract features that are too sensitive to the transformation. This may cause a large variation of feature embeddings of the samples within the same class, and it will require more labeled data to learn the in-class variance.

Number of ordered scales	2	3	4	6
pretrained	**89.12**	88.56	84.10	84.46
finetuned	96.40	**96.42**	94.57	93.14

Table 4. Comparison of the performance of models trained with different number of augmented views.

5.2. Comparison of different ways of sampling scales for transformations

We also compare different ways of choosing the scales of the transformation. Firstly, the range of the scale is the same for transformation with ordered scales and random scales. The first method is to set a fixed range for each transformation with ordered scales and sample the scale within the range uniformly for each iteration during training. The second way is to generate the range randomly and sample the scale uniformly for each image. The third way is to generate the range randomly and sample the scale using Gaussian distributed sampling. We generate the augmented views with these 3 ways while training the models on Fruits-360 and compare their results in Table 5. The model trained with applying transformations using Gaussian distributed sampled scales achieve slightly better performance than the other models. This result demonstrates that increasing the of variety of augmented views in the training stage can improve the expressiveness of the transformation equivariant features.

5.3. Combination of transformations

We compare two ways of composing different types of transformation when generating image views. Given two types of transformations and their corresponding series of scales $c_1, c_2, ...c_k$ and $r_1, r_2, ..., r_k$. A simple way to generate a series of combined transformations is to concatenate the two series of transformation scales element-wise to form a series of transformation scales of the two types $c_1r_1, c_2r_2, ..., c_kr_k$. We describe a different method in Section 3.1, where we randomly generated a composite ordered series in which not all transformation scales are strictly increasing. We train our models on Fruits 360 by applying color jitters, cropping and rotation together with ordered scales and show the performance comparison of the two approaches in Table 6. Column 5 shows the results of the model trained by combining the three transformations with element-wise composition and Column 6 shows the results by combining the transformations with random non-decreasing scales. Both methods outperform the model trained with one transformation using $T_{ordered}$, while the performance of the model trained by combining the transformations with non-decreasing random order is

Pretraining Datasets	color-fixed	color-random	color-Gaussian
Fruits 360-w/o finetune	91.99	91.84	**92.21**
Fruits 360-finetuned	96.87	96.46	**96.99**

Table 5. Comparison of three ways of sampling transformation scales. We apply color jitters with ordered scales to generate augmented views during training and compare the performances of the three models on the Fruits 360 dataset.

Pretraining Datasets	color-jitter	rotation	cropping	Combined (e.w.c.)	Combined (n.d.r.)
Fruits 360 w/o finetune	92.90	89.49	84.10	93.33	**94.46**
Fruits 360 w/ finetune	96.74	96.58	94.57	98.01	**98.91**

Table 6. Comparison of the performances of model trained with multiple transformations with ordered scales. Combined means using color-jitter,cropping, and rotation simultaneously. Here e.w.c. stands for element-wise composition and n.d.r. stands for non-decreasing randomization.

slightly better than that by combining transformations with fixed order. We conclude that the model is more robust when trained by combining transformations with random orders during training. We conjecture that generating more varied views can help the model to capture more expressive features.

5.4. Comparison of two ways of applying cropping with ordered scales

Unlike other types of transformations that we study, cropping operation is often defined by the four coordinates of cropped regions, and therefore it is not straightforward how we define the "scales" for cropping. Here we compare two ways of applying cropping with ordered scales on the Blood Cell dataset. The first method, as shown by the first row of Figure 4, is applying cropping on the input image with ordered ratios to generate the augmented views, yet it is not required the subsequent view region need to be fully covered by the previous view region. The second method, shown by the second row of Figure 4, is to generate a view by applying cropping on its previous view so that the subsequent image will always be a subimage of its previous view. Table 7 shows the performances of the models trained with these two methods of cropping as well as the performance of the baseline Barlow-Twins model for reference. As we can see from the results, although it is clear that learning cropping-equivariant feature representations is beneficial for downstream tasks, we are seeing mixed results when it comes to which way to use when generating ordered cropping scales. Essentially, we find the second approach works better for the Blood Cell dataset while the first one works better for the Tiny-ImageNet dataset. We look into it and conjecture such a phenomenon is caused by the misalignment of the semantic information correlation and the transformation scale orders. For example, the object of interest in the Blood Cell dataset is the purple region which indicates the main portion of the cell. As we can see from the sample images and cropping regions produced by the first method, even though the fourth cropping ratio is larger than the third one, it contains more area of the object of interest and therefore it should have higher correlation with the original image compared to the third image view in terms of semantic information. The loss function, on the contrary, is encouraging the correlation terms to follow the scale order, which would in turn affect the performance. The second cropping approach, as shown in Figure 4, avoids this problem by limiting the subsequent image being a subimage of the previous one, hence containing less semantic information in subsequent images. We do not observe similar behavior when pretraining on Tiny-ImageNet primarily because of much larger and centered object of interest in those images, and the model benefits from the large variety of views when applied with the first cropping method.

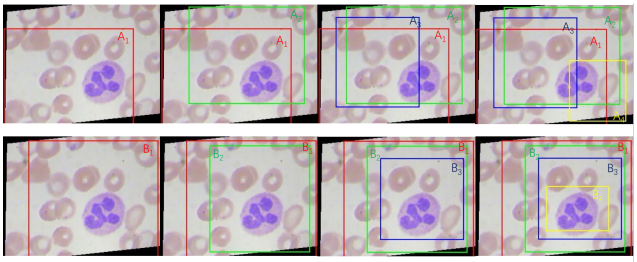

Figure 4. Examples of 2 ways of applying cropping with ordered scales on the Blood cells dataset. First row shows the example of applying cropping on the input with different ratios. Second row shows the examples of generating views from applying cropping on the previous view.

Pretraining datasets	Barlow-Twins	cropping 1	cropping 2
Tiny-ImageNet w/o finetune	37.60	**45.24**	41.82
Tiny-ImageNet fintune	49.38	**50.99**	49.05
blood cells w/o finetune	55.49	53.96	**58.06**
blood cells-finetune	58.30	60.96	**70.89**

Table 7. Comparison of the results of different transformation-equivariant models on Tiny-ImageNet and Blood cell dataset.

6. Conclusion

This work presents a new framework that is able to extract both transformation-equivariant and transformation-invariant feature representations by extending existing SSL method with a new way of view generation approach, a new methodology as well as the corresponding novel design of the loss function. Beyond that it provides a procedure to determine the appropriate types of transformations to which the features should be trained to be invariant and equivariant. Experiments show that our new framework outperforms the baseline when the appropriate types of transformations are applied, and the efficacy of our procedure of transformation selection is also validated by the experiments. In the future, we will explore more effective view generation method that does not require hand-crafted designs, so that it can be easier to generalize to domains such as text and audio.

References

[1] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019. 2

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1, 2

[3] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1

[4] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 1, 2

[5] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016. 2

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019. 2

[7] Rumen Dangovski, Li Jing, Charlotte Loh, Seungwook Han, Akash Srivastava, Brian Cheung, Pulkit Agrawal, and Marin Soljacic. Equivariant self-supervised learning: Encouraging equivariance in representations. In *International Conference on Learning Representations*, 2021. 1, 2

[8] Jian Ding, Enze Xie, Hang Xu, Chenhan Jiang, Zhenguo Li, Ping Luo, and Gui-Song Xia. Unsupervised pretraining for object detection by patch reidentification. *arXiv preprint arXiv:2103.04814*, 2021. 2, 4

[9] Zheren Fu, Yan Li, Zhendong Mao, Quan Wang, and Yongdong Zhang. Deep metric learning with self-supervised ranking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1370–1378, 2021. 1, 2

[10] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. 2

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020. 1, 2

[12] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster autoaugment: Learning augmentation strategies using backpropagation. *CoRR*, abs/1911.06987, 2019. 2

[13] 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*, pages 9729–9738, 2020. 1

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[15] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pages 2731–2741. PMLR, 2019. 2

[16] Jongheon Jeong and Jinwoo Shin. Training gans with stronger augmentations via contrastive discriminator. In *International Conference on Learning Representations*, 2021. 1

[17] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015. 1

[18] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015. 2

[19] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *CoRR*, abs/1905.00397, 2019. 2

[20] 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*, pages 6707–6717, 2020. 1, 2

[21] Paul Mooney. Blood cell images. retrieved from https://www.kaggle.com/paultimothymooney/blood-cells, 2018. 5

[22] Horea Mureșan and Mihai Oltean. Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, 10:26–42, 06 2018. 5

[23] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008. 5

[24] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *Advances in Neural Information Processing Systems*, 33:3407–3418, 2020. 2

[25] Guo-Jun Qi, Liheng Zhang, Feng Lin, and Xiao Wang. Learning generalized transformation equivariant representations via autoencoding transformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2

[26] Uwe Schmidt and Stefan Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2050–2057. IEEE, 2012. 2

[27] Alex Tamkin, Mike Wu, and Noah Goodman. Viewmaker networks: Learning views for unsupervised representation learning. In *International Conference on Learning Representations*, 2020. 1, 2

[28] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *European conference on computer vision*, pages 776–794. Springer, 2020. 2

[29] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in Neural Information Processing Systems*, 33:6827–6839, 2020. 1, 2

[30] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807, 2018. 2

[31] Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *arXiv preprint arXiv:2104.07713*, 2021. 1

[32] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 5

[33] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. *arXiv preprint arXiv:2008.05659*, 2020. 1, 2

[34] Ceyuan Yang, Zhirong Wu, Bolei Zhou, and Stephen Lin. Instance localization for self-supervised detection pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3987–3996, 2021. 2, 4

[35] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021. 1, 2, 5