SELF-SUPERVISED LEARNING BY ESTIMATING TWIN CLASS DISTRIBUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present TWIST, a novel self-supervised representation learning method by classifying large-scale unlabeled datasets in an end-to-end way. We employ a siamese network terminated by a softmax operation to produce twin class distributions of two augmented images. Without supervision, we enforce the class distributions of different augmentations to be consistent. In the meantime, we regularize the class distributions to make them sharp and diverse. Specifically, we minimize the entropy of the distribution for each sample to make the class prediction for each sample assertive and maximize the entropy of the mean distribution to make the predictions of different samples diverse. In this way, TWIST can naturally avoid the trivial solutions without specific designs such as asymmetric network, stop-gradient operation, or momentum encoder. Different from the clustering-based methods which alternate between clustering and learning, our method is a single learning process guided by a unified loss function. As a result, TWIST outperforms state-of-the-art methods on a wide range of tasks, including unsupervised classification, linear classification, semi-supervised learning, transfer learning, and some dense prediction tasks such as detection and segmentation.

1 INTRODUCTION

Deep neural networks learned from large-scale datasets have powered many aspects of machine learning. In computer vision, the neural networks trained on the ImageNet dataset (Deng et al., 2009) can perform better than or as well as humans in image classification (Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016; Huang et al., 2017). In addition, the obtained representations can also be adapted to other downstream tasks such as object detection (Ren et al., 2015; He et al., 2017; Redmon & Farhadi, 2017) and semantic segmentation (Long et al., 2015). However, learning from large-scale labeled data requires expensive data annotation, making it difficult to scale.

Recently, self-supervised learning has achieved remarkable performance and largely closed the gap with supervised learning. The contrastive learning approaches (Wu et al., 2018; He et al., 2020; Chen et al., 2020a;b; Grill et al., 2020; Chen & He, 2021) learn representations by maximizing the agreement of different augmentations and pushing away the representations of different images. BYOL (Grill et al., 2020) and SimSiam (Chen & He, 2021) propose to abandon the negative samples, and design the asymmetric architecture and momentum encoder (or stop-gradient) to avoid collapsed solutions. Clustering-based methods (Caron et al., 2018; Asano et al., 2019; Caron et al., 2020; 2021) usually employ the clustering algorithms to generate supervision signals from one augmentation and use them to guide the learning process for other augmentations.

In this paper, we present TWIST (**Tw**in Class D**ist**ributions Estimation), a novel self-supervised learning method to learn representations by classifying large-scale unlabeled datasets. Unlike the clustering-based methods that alternate between clustering and learning, our method is a single learning process guided by a unified loss function. We employ a siamese network terminated by a softmax operation to produce the class distributions of two augmented images. The whole network is trained by the proposed TWIST loss which embodies the following properties: (1) augmentation consistency, (2) distribution sharpness, and (3) prediction diversity. Specifically, optimizing the TWIST loss leads to force the two augmentations of the same image to be classified into the same class, which is achieved by minimizing the divergence between the twin class distributions. In the meantime, TWIST loss minimizes the entropy of the class distribution for each sample to make

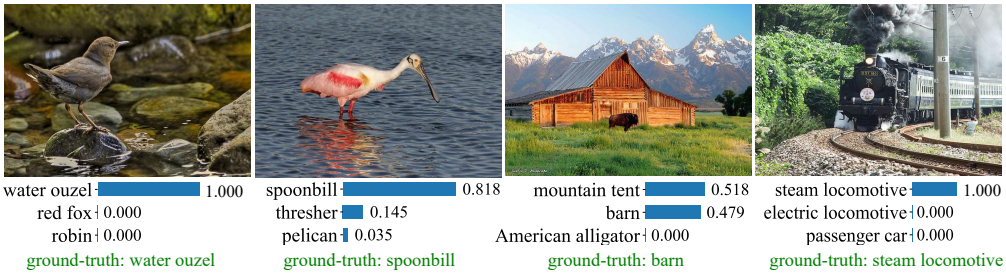

Figure 1: Examples of unsupervised top-3 predictions of TWIST. The predicted class indices are mapped to the labels in ImageNet by Kuhn–Munkres algorithm (Kuhn, 1955). Note that the labels are only used to map our predictions to the ImageNet labels, we do not use any label to participate in the training process. More examples are given in Appendix.

the class distribution sharp and maximize the entropy of the *mean class distribution* to make the predictions for different samples diverse. This is inspired by the fact that in supervised learning, the class distribution for each sample is sharp (low-entropy), and the class predictions for different samples are diverse. TWIST can naturally avoid the trivial solution problem (outputting the same class for all images). However, in practice, optimizing the TWIST loss is challenging. We discover that the output predictions are prone to be low-diverse, which causes the network to generate sub-optimal solutions. To address the issue, we add a batch normalization layer before the softmax function to ensure that the distributions of the samples in a mini-batch are more scattered. Detailed analysis is given in the Ablation section. With the TWIST loss and the design of batch normalization before softmax, we can successfully classify images without labels. Fig. 1 shows some examples.

TWIST is different from both contrastive learning methods and clustering-based methods. Compared with contrastive learning methods which learn instance-invariant features, TWIST takes into consideration the relation of different samples and learns category-invariant features. In contrast, TWIST is more similar to the clustering-based methods, which also learn the category-invariant features. However, TWIST is different from the clustering-based methods in the following aspects: **(1)** Clustering-based methods share a common two-stage paradigm. They first generate the pseudo-labels for images and then use the pseudo-labels as supervision to guide the learning process. TWIST has no concept of pseudo-labels. The TWIST loss will guide the network to automatically estimate the class distribution for each image. **(2)** TWIST does not require any labeling process, so we do not rely on any non-differentiable clustering technique such as K-means (Caron et al., 2018), Sinkhorn-Knopp algorithm (Asano et al., 2019; Caron et al., 2020), or momentum encoder (Caron et al., 2021).

We evaluate the efficacy of TWIST on both unsupervised classification and downstream representation learning tasks. For unsupervised classification on ImageNet, our model outperforms previous state-of-the-art methods by a large margin (+6.5 AMI). For representation learning, we surpass previous state-of-the-art methods on a wide range of downstream tasks. Specifically, with a ResNet-50 (He et al., 2016), **(1)** TWIST achieves 75.5% (+0.2) top-1 accuracy on ImageNet linear classification. **(2)** For semi-supervised learning, TWIST achieves 61.2% (+6.2) top-1 accuracy on 1% label semi-supervised classification. **(3)** For fine-tuning on small-scale datasets, we surpass supervised baseline and other methods on 7(/11) datasets. **(4)** We also achieve the state-of-the-art performance on dense predictive tasks, including object detection, instance segmentation, and semantic segmentation. Some of the results even surpass the densely designed self-supervised learning methods. These results show that TWIST successfully connects unsupervised classification and representation learning, and could serve as a strong baseline for both purposes.

2 RELATED WORK

Self-supervised Learning. Self-supervised Learning has been a promising paradigm to learn useful image representations. Many self-supervised methods try to design handcrafted auxiliary tasks to acquire common knowledge useful for other vision tasks. Examples include context prediction (Doersch et al., 2015), colorization (Zhang et al., 2016), context encoder (Pathak et al., 2016), jigsaw puzzle (Noroozi & Favaro, 2016), and rotation prediction (Gidaris et al., 2018). Although these

self-supervised methods have achieved remarkable performance, the design of pretext tasks depends on domain-specific knowledge, limiting the generality of the learned representations.

Recently, contrastive learning has drawn much attention and achieved state-of-the-art performances. Representative methods include Instance Discrimination (Wu et al., 2018), MoCo (He et al., 2020; Chen et al., 2020b), and SimCLR (Chen et al., 2020a). Contrastive methods learn an embedding space where features of different augmentations from the same image are attracted, and features of different images are separated. BYOL (Grill et al., 2020) and SimSiam (Chen & He, 2021) propose to abandon the negative samples and design some special techniques such as asymmetric architecture, momentum encoder and stop gradients to avoid the collapsed solution. Barlow Twins (Zbontar et al., 2021) and VICReg (Bardes et al., 2021) propose to learn informative representations by reducing the redundancy or covariance of different representation dimensions.

The clustering-based methods (Caron et al., 2018; Asano et al., 2019; Caron et al., 2019; 2020; 2021) have also exhibited remarkable performances. They first use a clustering tool to generate pseudo-labels for images and then classify the images with the generated pseudo-labels. The two processes alternate with each other. For example, DeepCluster (Caron et al., 2018) uses the K-means algorithm to generate pseudo-labels for every epoch. SwAV (Caron et al., 2020) uses the Sinkhorn-Knopp algorithm (Asano et al., 2019) to generate soft pseudo-labels for images and updates the pseudo-labels online for every iteration. The most recent DINO (Caron et al., 2021) updates pseudo-labels using the output of the momentum teacher together with the sharpening and centering operations.

Unsupervised Classification. There are many unsupervised classification methods developed. Some methods use an EM-like algorithm to optimize the pseudo-labels and the networks alternately. Examples like DEC (Xie et al., 2016), JULE (Yang et al., 2016), and SeLa (Asano et al., 2019). They use the clustering algorithms to generate the pseudo-labels to guide the learning of neural networks, which is similar with the clustering-based self-supervised learning methods, while they focus on the task of clustering or unsupervised classification. Some methods propose to maximize the mutual information of output features between an image and its augmentations. IIC (Ji et al., 2019) proposes a clustering objective to maximize the mutual information of different views. IMSAT (Hu et al., 2017) also uses data augmentations to impose the invariance on discrete representations, and in the mean time maximizes the information theoretic dependency between data and their predicted discrete representations. SCAN (Van Gansbeke et al., 2020) advocates a two-step approach to achieve this task. They use representation learning methods such as SimCLR (Chen et al., 2020a) to extract the high-level features and then clusters the features with their proposed loss function which takes into consideration of the nearest neighbors, achieving state-of-the-art performance. Our method is an end-to-end single process method with an effective objective for training, achieving significant advantages on large-scale unsupervised classification and representation learning.

3 METHOD

Our goal is to learn an end-to-end unsupervised classification network to make accurate predictions and learn good representations. Without labels, we design the TWIST loss to make the predictions of two augmented images be recognized as the same class. In the meantime, we regularize the class distribution to make it sharp and diverse, which helps the network avoid the trivial solution problem and learn category-level invariant representations. The motivation is based on the observation of supervised leaning that the class distributions in a mini-batch are both sharp and diverse. Next, we will give detailed descriptions and analyses.

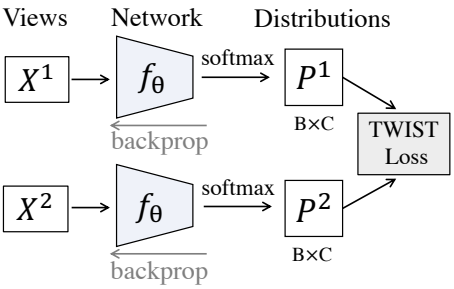

Figure 2: Network architecture of TWIST.

3.1 FORMULATION

Given an unlabeled dataset $\mathcal{S}$, we randomly sample a batch of images $X \subset \mathcal{S}$ with a batch-size of B, and generate two augmented version X^1 and X^2 according to a predefined set of image augmentations. The two augmented images are fed into a siamese neural network (terminated by

a softmax operation) $f_\theta(\cdot)$ with parameters θ. The outputs of the neural networks $f_\theta(\cdot)$ are two probability distributions over C categories $P^k = f_\theta(X^k)$ ($k \in \{1, 2\}$). The process is shown in Fig. 2. Note that $P^k \in \mathbb{R}^{B\times C}$, and P_i^k denotes the i-th row of P^k, i.e., probability distribution of the i-th sample.

With the probability distributions of two augmented views P^1 and P^2, we define the learning objective as follows:

$$\mathcal{L}(P^1, P^2) = \frac{1}{2}(\mathcal{L}(P^1||P^2) + \mathcal{L}(P^2||P^1)), \tag{1}$$

where

$$\mathcal{L}(P^1||P^2) = \underbrace{\frac{1}{B}\sum_{i=1}^{B} D_{KL}(P_i^1||P_i^2)}_{\text{consistency term}} + \underbrace{\frac{1}{B}\sum_{i=1}^{B} H(P_i^1)}_{\text{sharpness term}} - \underbrace{H(\frac{1}{B}\sum_{i=1}^{B} P_i^1)}_{\text{diversity term}}, \tag{2}$$

and $\mathcal{L}(P^2||P^1)$ is defined in the same way. $D_{KL}(\cdot||\cdot)$ denotes the Kullback–Leibler divergence (Kullback & Leibler, 1951) between two probability distributions. $H(\cdot)$ denotes the Entropy (Shannon, 1948) of a specific probability distribution.

We next give analysis to the above loss function. Specifically, minimizing the consistency term makes the predictions of different views consistent. Namely, different augmentations of the same image are required to be recognized as the same class. For the sharpness term, we minimize the entropy of class distribution for each sample to regularize the output distribution to be sharp, which makes each sample have a deterministic assignment (i.e., one-hot vector in the ideal case). Besides, features of samples assigned to the same category will be more compact. For the diversity term, we try to make the predictions for different samples be diversely distributed over C classes to avoid the network assigning all images to the same class. This is achieved by maximizing the entropy of the mean distribution across different samples $H(\frac{1}{B}\sum_{i=1}^{B} P_i^1)$.

Relations to previous methods: To give a comprehensive understanding of Eq. (2), we rewrite it as

$$\mathcal{L}(P^1||P^2) = \frac{1}{B}\sum_{i=1}^{B} CE(P_i^1, P_i^2) - H(\frac{1}{B}\sum_{i=1}^{B} P_i^1), \tag{3}$$

where CE denotes the cross entropy. The equivalence of Eq. (2) and Eq. (3) is because of the fact that $CE(p, q) = D_{KL}(p||q) + H(p)$, for two probability distributions p and q. The above loss function is similar to the cross-entropy loss in supervised learning. Without ground-truth label, we minimize the cross-entropy loss between the class distributions of two augmentations. In addition to the cross-entropy term, we have the additional diversity term to maximize the diversity of the network predictions.

Eq. (3) differentiates our method from previous clustering-based methods. Previous methods usually transform the probability distribution P^1 to a deterministic vector V^1 to create pseudo-labels. The transformations vary in different methods, hard or soft, e.g., K-Means in Deep Cluster (Caron et al., 2018), the Sinkhorn-Knopp algorithm in SwAV (Caron et al., 2020; Asano et al., 2019), and softmax over samples in Self-classifier (Amrani & Bronstein, 2021). In contrast, we provide a simpler and more unified solution, i.e., using a single loss function to simultaneously learn the probability distributions of P^1 and P^2.

3.2 Batch Normalization before Softmax

In practice, directly optimizing the TWIST loss will derive sub-optimal solutions. Specifically, we find that the consistency term and the sharpness term are easy to minimize while the diversity term is difficult to maximize. Furthermore, we visualize the standard deviations of each row and each column of the features before softmax. As illustrated in Fig. 3, we find that the column standard deviation keeps small during the training process, which makes the probability of each class tend to be similar across different samples in a mini-batch. This will cause the low diversity of classification results. Our solution is simple: adding a batch normalization (Ioffe & Szegedy, 2015) before the softmax to force the probabilities in the same column be separated. Actually, after adding the batch normalization (without affine parameters), the standard deviation for each column is forced to be 1. By adding the batch normalization layer, the diversity term is well optimized. As a result, the

batch normalization before softmax brings about 5% improvements in ImageNet linear classification. More analyses and experiments are shown in the Ablation section.

3.3 TRAINING STRATEGY

Multi-crop: We adopt the multi-crop augmentation strategy proposed in SwAV (Caron et al., 2020). For a comprehensive comparison, we also report the performance without multi-crop. For multi-crop, we generate global views and local views. For each combination of two different views, we calculate the TWIST loss and use the average as the final loss.

Self-labeling for CNN: For convolutional neural networks, the multi-crop strategy helps us improve the linear classification performance from 72.6% to 74.3%, shown in Tab. 10. However, compared with the performance improvements of SwAV (from 71.8% to 75.3%), the performance gain of TWIST is much smaller (3.5% v.s. 1.7%). Such phenomenon has also been observed by Caron et al. (2021). Specifically, SwAV uses the global crops to generate relatively accurate pseudo-labels as supervision to train the local crops. However, in our method, the global crops and local crops are regarded equally. Thus the noisy local crops can also affect the accurate predictions of global crops, which will not happen in self-labeling methods such as SwAV and DINO. To take full advantage of the multi-crop strategy, we add a self-labeling stage after the regular training stage. Specifically, we use the outputs of the global crops as supervision to train other crops. Different with SwAV: **(1)** we use the outputs of our network as supervision, instead of the outputs of the Sinkhorn-Knopp algorithm, **(2)** we only use the samples in a mini-batch whose confidences surpass a predefined threshold. With only 50 epochs of self-labeling after finishing the regular training, we have another 1.2% performance gains (from 74.3% to 75.5%). We empirically show that the proposed self-labeling could not improve the performance of SwAV (Caron et al., 2020) or DINO (Caron et al., 2021).

Momentum Encoder for ViT: For Vision Transformers (Dosovitskiy et al., 2020; Touvron et al., 2021), we do not use the self-labeling process. Instead, we adopt the momentum encoder design, which is widely adopted to train ViT-based self-supervised models (Caron et al., 2021; Chen et al., 2021). Specifically, one tower of our siamese network is updated by the exponential moving average of the parameters from the other tower, similar as He et al. (2020) and Grill et al. (2020). The whole network is updated by the TWIST loss. Although we use the momentum encoder as the default setting for ViT backbones, TWIST using ViT as backbone can also work without the momentum encoder and achieves 72.5% ImageNet Top-1 linear accuracy for Deit-S 300 epochs. The momentum encoder is only adopted in ViT-based models. We do not use it for CNN models.

4 MAIN RESULTS

We evaluate the performances of TWIST on unsupervised classification, as well as a wide range of downstream tasks. We set $C = 4096$ for the downstream tasks and $C = 1000$ for the unsupervised classification in accordance with the standard ImageNet class number. More implementation details can be found in Appendix. *For all downstream tasks, we strictly follow the common evaluation procedures*. All TWIST models are trained on the train set of ImageNet ILSVRC-2012 which has $\sim$1.28 million images (Deng et al., 2009).

Unsupervised Classification: The TWIST model can be regarded as a clustering function which takes images as input and outputs the assignments. Therefore, we adopt the measures for clustering in evaluation, including normalized mutual information (NMI), adjusted mutual information (AMI), and adjusted rand index (ARI). Besides, we map the predicted assignments to the class labels of ImageNet to evaluate the unsupervised classification accuracy. We use the Kuhn–Munkres algorithm (Kuhn, 1955) to find the best one-to-one permutation mapping, following the settings in IIC (Ji et al., 2019) and SCAN (Van Gansbeke et al., 2020). *Though*

Table 1: Unsupervised classification results on ImageNet. All numbers are reported on the validation set of ImageNet. Comparison methods include SCAN (Van Gansbeke et al., 2020), SeLa (Asano et al., 2019), and Self Classifier (Amrani & Bronstein, 2021).

Method	NMI	ARI	AMI	ACC
SCAN	72.0	27.5	51.2	39.9
SeLa	65.7	16.2	42.0	-
SelfClassifier	64.7	13.2	46.2	-
TWIST	**74.3**	**30.0**	**57.7**	**40.6**

this process uses the real labels, they do not participate in any training process and are only used to map the prediction to the meaningful ImageNet labels. Tab. 1 shows the results of TWIST and other state-of-the-art methods. We use ResNet-50 as backbone and set $C = 1000$ in accordance with ImageNet. Our method surpasses other methods by large margins in terms of the clustering measures and unsupervised classification accuracy. In Appendix, we show some randomly selected results for qualitative evaluation.

Table 2: Linear classification results on ResNet-50. We show results with and without multi-crop training.

Method	Epoch	Top1	Top5
without multi-crop			
MoCo v2	800	71.1	90.1
SimCLR	1000	69.3	89.0
BarlowT	1000	73.2	91.0
BYOL	1000	74.3	91.6
SwAV	800	71.8	-
TWIST	800	72.6	91.0
with multi-crop			
SwAV	800	75.3	-
DINO	800	75.3	92.5
TWIST	300	75.0	92.4
TWIST	800	**75.5**	**92.5**

Table 3: Linear classification results with the backbones of wider ResNet-50 and Vision Transformer.

Method	Network	Param	Epoch	Top1	Top5
Wider ResNet					
SimCLR	RN50w2	94M	1000	74.2	92.0
CMC	RN50w2	94M	-	70.6	89.7
SwAV	RN50w2	94M	800	77.3	-
BYOL	RN50w2	94M	1000	77.4	93.6
TWIST	RN50w2	94M	300	**77.7**	**93.9**
Vision Transformer					
MoCo-v3	Deit-S	21M	300	72.5	-
DINO	Deit-S	21M	300	75.9	-
TWIST	Deit-S	21M	300	75.6	92.3
MoCo-v3	ViT-B	86M	300	76.5	-
DINO	ViT-B	86M	800	78.2	93.9
TWIST	ViT-B	86M	300	77.3	93.3

Linear Classification: We evaluate the performance of linear classification on ImageNet. Specifically, we add a linear classifier on top of the frozen network and measure the top-1 and top-5 center-crop classification accuracies, following previous works (Zhang et al., 2016; He et al., 2020; Chen et al., 2020a). The results are shown in Tab. 2. TWIST outperforms other state-of-the-art methods, achieving 75.5% top-1 accuracy on ResNet-50.

Besides, we also train TWIST with other backbones of neural networks, including wider ResNet-50 and Vision Transformers, obtaining the results shown in Tab. 10. For wider ResNet-50, the superiority of TWIST becomes more apparent. TWIST outperforms SwAV and BYOL by 0.4% and 0.3% respectively using ResNet-50w2 and 300 training epochs. For Vision Transformers, TWIST is also comparable with other state-of-the-art methods.

Table 4: Semi-supervised classification results on ImageNet. We report top-1 and top-5 one-crop accuracies. Detailed settings can be found in Appendix.

Method	Backbone	Param	**1%** Labels		**10%** Labels		**100%** Labels	
			Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
SUP	RN50	24M	25.4	48.4	56.4	80.4	76.5	-
SimCLR	RN50	24M	48.3	75.5	65.6	87.8	76.5	93.5
BYOL	RN50	24M	53.2	78.4	68.8	89.0	77.7	93.9
SwAV	RN50	24M	53.9	78.5	70.2	89.9	-	-
DINO	RN50	24M	52.2	78.2	68.2	89.1	-	-
BarlowTwins	RN50	24M	55.0	79.2	69.7	89.3	-	-
TWIST	RN50	24M	**61.2**	**84.2**	**71.7**	**91.0**	**78.4**	**94.6**
SimCLR	RN50 $\times 2$	94M	58.5	83.0	71.7	91.2	-	-
BYOL	RN50 $\times 2$	94M	62.2	84.1	73.5	91.7	-	-
TWIST	RN50 $\times 2$	94M	**67.2**	**88.2**	**75.3**	**92.8**	**80.3**	**95.4**

Semi-supervised Classification: We fine-tune the pre-trained TWIST model on a subset of ImageNet, following the standard procedure (Chen et al., 2020a; Grill et al., 2020). From Tab. 4, we can see that TWIST outperforms all other state-of-the-art methods by large margins. With only 1% of labeled data, TWIST achieves 61.2% top-1 accuracy with ResNet-50, surpassing the previous best result by 6.2%. The trend is preserved when the backbone becomes larger. For example,

TWIST achieves 67.2% (+5.0%) top-1 accuracy with ResNet-50w2. With 10% of labeled data, it still achieves the best result.

We also fine-tune the pre-trained TWIST model on the full ImageNet. The results are shown in the last two columns of Tab. 4. With our pre-trained model as initialization, ResNet-50 can achieve 78.4% top-1 accuracy, surpassing ResNet-50 trained from scratch by a large margin (+1.9%).

Transfer Learning: We also evaluate the performances of fine-tuning TWIST model on eleven different datasets, including Food101 (Bossard et al., 2014), CIFAR10, CIFAR100 (Krizhevsky et al., 2009), SUN397 (Xiao et al., 2010), Cars (Krause et al., 2013), FGVC-Aircraft (Maji et al., 2013), Pascal VOC2007 (Everingham et al., 2009), Describable Textures Dataset (DTD) (Cimpoi et al., 2014), Oxford-IIIT Pet (Parkhi et al., 2012), Caltech101 (Li et al., 2004), and Flowers (Nilsback & Zisserman, 2008). Tab. 5 shows the results. We also report the results of linear classification models for a comprehensive comparison. We can observe that TWIST performs competitively on these datasets. In some datasets, TWIST achieves improvements over 1%. TWIST outperforms the supervised models on seven out of eleven datasets. We also observe that our method shows more advantages over other methods in the fine-tune setting.

Table 5: Transfer learning results on eleven datasets, including linear evaluation and fine-tuneing.

Method	Food	Cifar10	Cifar100	Sun397	Cars	Aircraft	VOC	DTD	Pets	Caltech	Flowers
Linear evaluation:											
SimCLR	68.4	90.6	71.6	58.8	50.3	50.3	80.5	74.5	83.6	90.3	91.2
BYOL	75.3	91.3	**78.4**	62.2	**67.8**	60.6	82.5	75.5	90.4	94.2	**96.1**
SUP	72.3	**93.6**	78.3	61.9	66.7	**61.0**	82.8	74.9	91.5	**94.5**	94.7
TWIST	**78.0**	91.2	74.4	**66.8**	55.2	53.6	**85.7**	**76.6**	**91.6**	91.1	93.4
Fine-tune:											
Random	86.9	95.9	80.2	53.6	91.4	85.9	67.3	64.8	81.5	72.6	92.0
SimCLR	87.5	97.4	85.3	63.9	91.4	87.6	84.5	75.4	89.4	91.7	96.6
BYOL	88.5	97.8	86.1	63.7	91.6	**88.1**	85.4	76.2	91.7	**93.8**	97.0
SUP	88.3	97.5	86.4	64.3	**92.1**	86.0	85.0	74.6	92.1	93.3	**97.6**
TWIST	**89.3**	**97.9**	**86.5**	**67.4**	91.9	85.7	**86.5**	**76.4**	**94.5**	93.5	97.1

Object Detection and Instance Segmentation: We evaluate the learned representations of TWIST on object detection and instance segmentation. We conduct experiments on Pascal VOC (Everingham et al., 2009) and MS COCO (Lin et al., 2014), following the procedure of Tian et al. (2020). We use the Feature Pyramid Network (FPN) (Lin et al., 2017) as the backbone architecture. For Pascal VOC, we use the Faster R-CNN (Ren et al., 2015) as the detector, and for MSCOCO, we follow the common practice to use the Mask R-CNN (He et al., 2017) as the detector. In implementation, we use Detectron2 (Wu et al., 2019), with the same configurations as Tian et al. (2020) and Wang et al. (2021). Tab. 11 shows the results. We can see that TWIST performs better on all three tasks, demonstrating the advantages of using TWIST as the pre-trained model in object detection and instance segmentation. Besides, we find that the FPN architecture is important for the category-level self-supervised learning methods to achieve good performance. Analysis is given in Appendix.

Table 6: Object detection and segmentation results. †means that we download the pre-trained models and conduct the experiments. For the VOC dataset, we run five trials and report the average. The performance is measured by Average Precision (AP). DC-v2 denotes the DeepCluster-v2.

Method	VOC07+12 detection			COCO detection			COCO instance seg		
	AP_{all}	AP_{50}	AP_{75}	AP^{bb}_{all}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}_{all}	AP^{mk}_{50}	AP^{mk}_{75}
Moco-v2	56.4	81.6	62.4	39.8	59.8	43.6	36.1	56.9	38.7
SimCLR †	**58.2**	83.8	65.1	41.6	61.8	45.6	37.6	59.0	40.5
SwAV	57.2	83.5	64.5	41.6	62.3	**45.7**	**37.9**	59.3	**40.8**
DC-v2 †	57.0	83.7	64.1	41.0	61.8	45.1	37.3	58.7	39.9
DINO †	57.2	83.5	63.7	41.4	62.2	45.3	37.5	58.8	40.2
DenseCL	56.9	82.0	63.0	40.3	59.9	44.3	36.4	57.0	39.2
TWIST	58.1	**84.2**	**65.4**	**41.9**	**62.6**	**45.7**	**37.9**	**59.7**	40.6

Semantic Segmentation: We also evaluate TWIST on semantic segmentation, using FCN (Long et al., 2015) and DeepLab v3+ (Chen et al., 2017) as architectures. We use the MMSegmentation (Contributors, 2020) to train the architectures. Tab. 7 shows the results on Pascal VOC (Everingham et al., 2009) and Cityscapes (Cordts et al., 2016). The results indicate that TWIST is competitive to other state-of-the-art methods.

Table 7: Results on semantic segmentation with different architectures. All results are averaged over five trials.

Method	FCN-FPN		DeepLab-v3	
	VOC	Cityscapes	VOC	Cityscapes
Rand init	37.9	62.9	49.5	68.3
Sup	67.7	**75.4**	76.6	**78.6**
Moco-v2	67.5	**75.4**	72.9	**78.6**
SimCLR	72.8	74.9	**78.5**	77.8
SwAV	71.9	74.4	77.2	77.0
DC-v2	72.1	73.8	76.0	76.2
DINO	71.9	73.8	76.4	76.2
TWIST	**73.3**	74.6	77.3	76.9

5 ABLATION STUDY

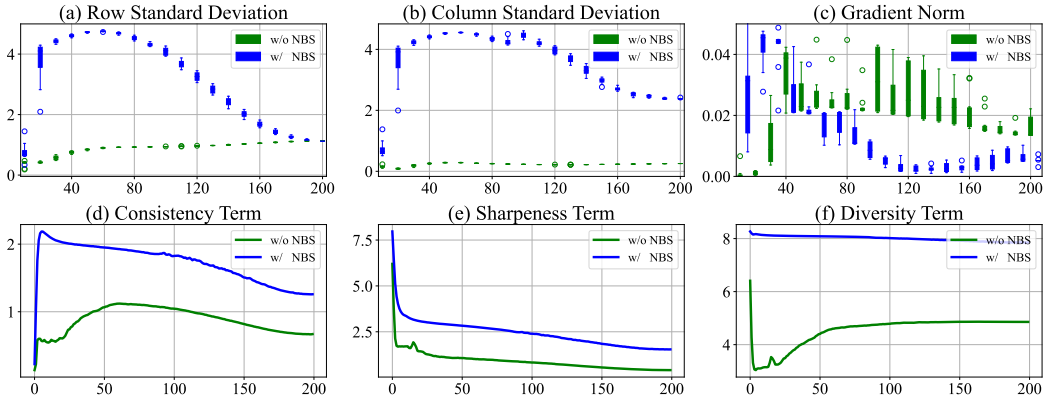

Figure 3: We show the statistical characteristics of the output before softmax operation with and without NBS in (a),(b) and (c), and the loss change for each term, in (d),(e) and (f).

Table 8: Ablation study on NBS.

NBS	Loss	ACC	NMI	std_c	std_r
✓	-5.05	70.6	59.0	2.37	1.12
✗	-3.78	65.5	47.0	0.26	1.14

Batch Normalization before Softmax: We study the impact of the batch normalization before softmax (abbreviated as NBS in Tab. 8). From the loss values, we can see that the model with NBS is optimized much better than the model without NBS. NBS brings 5.1% top-1 accuracy improvement and 12% NMI improvement on ImageNet, demonstrating the effectiveness of NBS. To better understand how NBS works, we look at the behaviors of the models with and without NBS. Fig. 3 (a) and (b) show the row and column standard deviations of the output before softmax (input of BN for the NBS model). Although the intermediate processes are different, the row standard deviations are closing when training is finished. For the column deviations, it is not the case. The column standard deviation of NBS model is much larger than the model without NBS at the end of the training, indicating that the samples in the same batch tend to give similar predictions. This is also reflected in Fig. 3 (f), from which we see that the diversity term of the model with NBS is better optimized than the model without NBS. The observation indicates that the model without NBS tends to be column-collapsed. Although the solution is not fully collapsed, it tends to output similar predictions for samples in a mini-batch. NBS can successfully avoid the degenerated solution because batch normalization will force the standard deviation of each column to be one. Fig. 3 (c) shows the magnitude and stability of the gradients from the optimization perspective.

Impact of Loss Terms: The loss of TWIST consists of three terms. We test the impact of these terms. As shown in Tab. 9, the models trained without the sharpness term generate collapsed solutions: The entropy for each sample is as large as the entropy of a uniform distribution, and the gradient magnitude rapidly decreases to 0. In contrast, the models trained without the diversity term do not generate collapsed solutions, but their performances deteriorate significantly. Theoretically, models trained without the diversity term will also lead to collapsed solutions, i.e., outputting the same one-hot distributions. However, the batch-normalization before the softmax operation helps avoid the

Table 9: Ablation study on the loss terms. Here L_s and L_d denote the sharpness and diversity term respectively. $|g|$ denotes the mean magnitude of gradients before the last batch normalization and "acc" is the linear accuracy.

L_s	L_d	value of L_s	value of L_d	$\|g\|$	acc
✗	✗	8.28	8.28	0	0.1
✗	✓	8.27	8.28	0	0.1
✓	✗	2.59	6.42	0.01	56.1
✓	✓	1.51	7.87	0.02	70.9

Table 10: Ablation study on multi-crop and self-labeling. We report the linear accuracy.

multi-crop	self-labeling	acc
✓	✓	75.5
✓	✗	74.0
✗	✓	73.8
✗	✗	72.6

problem because it can separate the probabilities in different columns and force them to have a unit standard deviation. Therefore, all three terms are indispensable for TWIST.

Multi-crop and Self-labeling: Tab. 10 shows the ablation results on multi-crop and self-labeling, where the models are trained for 800 epochs. We observe that the multi-crop and self-labeling can improve the performance respectively, and the best result comes from the combination of both two techniques.

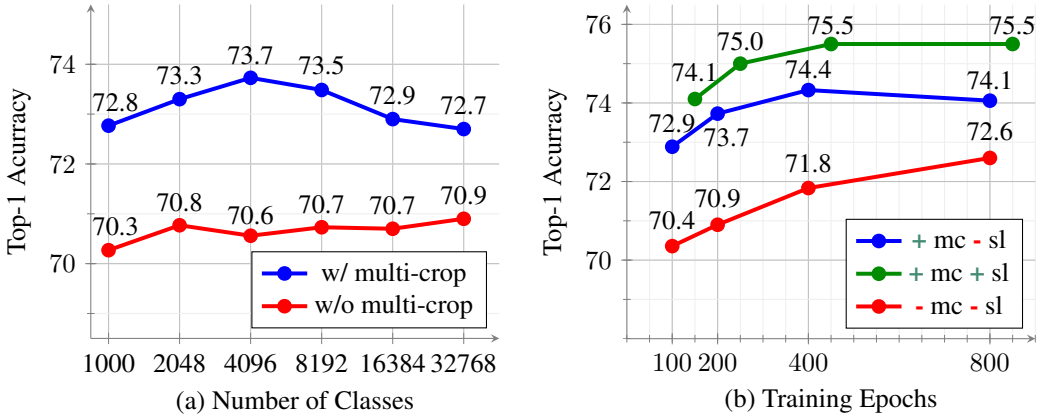

Figure 4: **(a)** Effect of different numbers of classes in TWIST. **(b)** Effect of different training epochs in TWIST, where "mc" denotes multi-crop and "sl" denotes self-labeling.

Number of Classes: We show the impact of class number C in Fig. 4 (a). To make a comprehensive evaluation, we show the results of TWIST with and without multi-crop. The models are trained by setting the number of classes from 1000 to 32768. With multi-crop, TWIST performs best when the number of classes is 4096. Overall, the performances are quite stable and fluctuate within the range of 1%, particularly when without multi-crop and $C >= 2048$.

Training Epochs: Fig. 4 (b) shows the performances of training TWIST with different epochs. Training longer improves the performance of TWIST model without multi-crop, while has less impact on the TWIST model with multi-crop (when the training epochs > 400).

6 CONCLUSION

In this paper, we have presented a novel self-supervised approach TWIST. With a single loss function, our method can learn to classify images without labels, reaching 40.6% top-1 accuracy on ImageNet. The learned representations can be used in a wide range of downstream tasks to achieve better results than existing state-of-the-art methods, including linear classification, semi-supervised classification, transfer learning, and dense prediction tasks such as object detection and instance segmentation. TWIST is very simple and do not rely on any clustering tools, making it easy to implement. There are many topics worth exploring in future work such as extensions to other modalities, and applications of TWIST to larger datasets.

REFERENCES

Elad Amrani and Alex Bronstein. Self-supervised classification network. *arXiv preprint arXiv:2103.10994*, 2021.

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

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, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018.

Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2959–2968, 2019.

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.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

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.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021.

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.

MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

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.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2009.

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

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, 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*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

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.

Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *International conference on machine learning*, pp. 1558–1567. PMLR, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Xu Ji, Joao F. Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Jonathan Krause, Jun Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Fei-Fei Li, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR Workshops*, 2004.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

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

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505, 2012.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28: 91–99, 2015.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

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*, 2020.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European Conference on Computer Vision*, pp. 268–285. Springer, 2020.

Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

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.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 478–487, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/xieb16.html.

Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5147–5156, 2016.

Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 6:12, 2017.

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.

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

A IMPLEMENTATION DETAIL

A.1 PRE-TRAINING

We add a non-linear projection head connected behind the backbone (Chen et al., 2020a; Grill et al., 2020). The projection head is an MLP consisting of three layers with dimensions of $4096-4096-C$. The first two layers of the projection head are followed by a batch normalization layer and rectified linear units. After the projection head, we add a batch-normalization layer without affine parameters and a softmax operation to calculate the final probability distributions. For multi-crop, we set the global scale to (0.4, 1.0) and local scale to (0.05, 0.4). For CNN backbones, we use 12 crops. For ViT backbones, we use 6 crops.

For self-labeling, we choose the samples in a mini-batch whose classification confidence (the maximum of softmax output) is larger than a predefined threshold. In practice, we use a cosine annealing schedule to choose the top 50% to 60% confident samples. We then use the chosen samples to generate labels for the subsequent fine-tuning process. For augmentation, we use multi-crop augmentations same as the regular training setting, except for that we change the global crop scale to (0.14, 0.4) and the local crop scale to (0.05, 0.14). We use the predictions of global crops as labels to guide the learning of local crops.

To train CNN backbones, we use LARS optimizer (You et al., 2017) with a cosine annealing learning rate schedule (Loshchilov & Hutter, 2016). We use a batch-size of 2048 splitting over 16 Tesla-V100 GPUs for ResNet50, and a batch-size of 1920 splitting over 32 Tesla-V100 GPUs for ResNet50$\times$2. The learning rate is set to $0.5\times$ BatchSize $/256$. The weight decay is set to 1.5e-6. The computation and memory costs are mainly from the multi-crop augmentations. For model without multi-crop, it can use 8 Tesla-V100 GPUs to achieve 72.6% top-1 linear accuracy on ImageNet.

For ViT backbones, we use the momentum encoder. The momentum is set to a variable value raised from 0.996 to 1 with a cosine annealing schedule. We change the weight of the three different terms of TWIST loss. Specifically, we set the coefficient as 1.0, 0.4, 1.0 for the consistency term, sharpness term and diversity term, respectively. The optimizer is AdamW (Loshchilov & Hutter, 2017). We use a batch-size of 2048 splitting over 32 Tesla-V100 GPUs. The learning rate is set to $0.0005\times$ BatchSize $/256$. The weight decay is set to 0.05.

B DISTRIBUTION VISUALIZATION

In this section, we will give visualizations to show how TWIST learns meaningful predictions. Fig. 6 shows the evolution of the softmax output when optimizing the TWIST loss (we set $B = C = 64$ for demonstration). We can see that the predictions become more deterministic and diverse, and the predictions of two views are more consistent.

To give a more clear demonstration, we only optimize the sharpness term and the diversity term, and set $B = C = 8$. Fig. 5 shows the visualization. We could clearly see the evolution of the predictions (The largest value is not 1 because of the batch normalization before softmax operation).

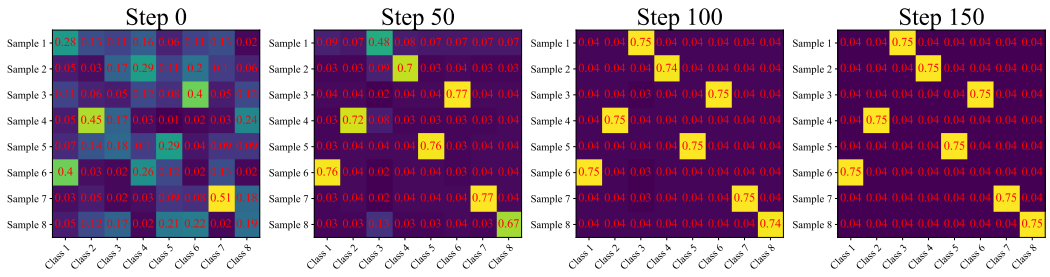

Figure 5: Evolution of the predictions for one view. We set $B = C = 8$ to give a clear demonstration.

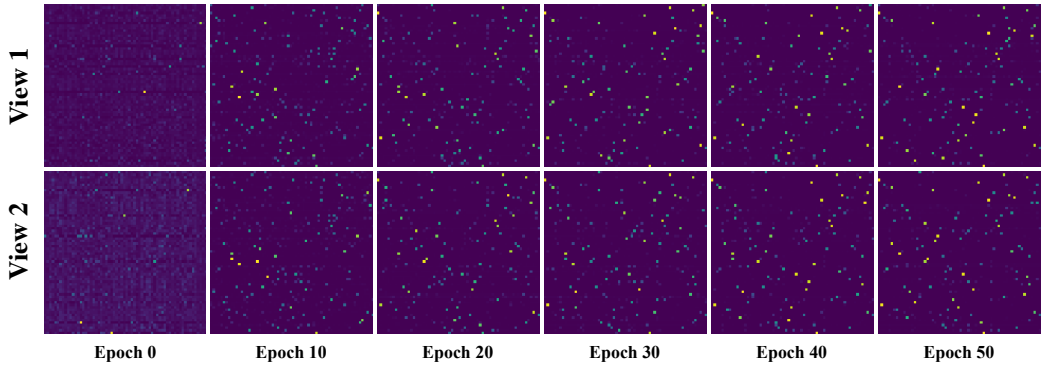

Figure 6: Evolution of the predictions for both two views. We set $B = C = 64$.

C PyTorch-like Pseudo Code

We give the PyTorch-like pseudo code, shown in Algorithm 1.

Algorithm 1 Pseudocode of TWIST in a PyTorch-like style.

```
# eps = 1e-7
# Definition of TWIST loss.
def twist_loss(p1, p2, alpha=1, beta=1):
    # calculate the consistency term: KL-divergency
    kl_div = ((p2 * p2.log()).sum(dim=1) - (p2 * p1.log()).sum(dim=1)).mean()
    # calculate the sharp term: mean_entropy
    mean_entropy = - (p1 * (p1.log() + eps)).sum(dim=1).mean()
    # calculate the uniform term: entropy_mean
    mean_prob = p1.mean(dim=0)
    entropy_mean = - (mean_prob * (mean_prob.log() + eps)).sum()
    return kl_div + alpha * mean_entropy - beta * entropy_mean
# f: encoder network.
# bn: batch normalization operation, with no affine parameter.
# B: batch-size.
# C: class number, hyper-parameter.
for  x in loader do:
    x1 = aug(x)
    x2 = aug(x)
    feat1 = f(x1)    # feat1: B × C
    feat2 = f(x2)    # feat2: B × C
    p1 = softmax(bn(feat1), dim=1)    # p1: B × C
    p2 = softmax(bn(feat2), dim=1)    # p2: B × C
    # symmetric twist_loss loss
    loss = (twist_loss(p1, p2) + twist_loss(p2, p1)) * 0.5
    loss.backward()
    update(f.params)
end for
```

D Visualization of similarity

Fig. 7 shows the similarities of features sampling from the same or different classes of ImageNet. Specifically, we collect the outputs of the last convolutional layer as features, and calculate the similarities. For positive samples, we sample two images from the same ImageNet class. For negative samples, we sample two images from different ImageNet classes. We then l2-normalize the features and calculate the similarities of positive/negative samples. The similarity distributions are shown in Fig. 7. We compare TWIST with SimCLR, SwAV and Supervised models. From Fig. 7, we observe

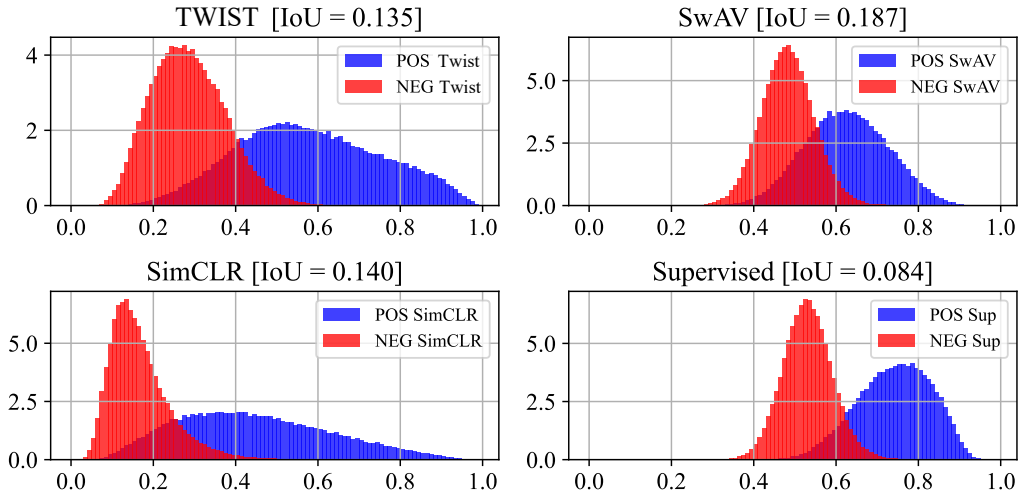

Figure 7: The distributions of positive/negative similarities. IoU is the area of overlap between positive and negative distributions.

that the postive distributions and the negative distributions of TWIST are more separable than other self-supervised methods.

E MORE RESULTS AND ANALYSES ON DENSE TASKS.

Table 11 shows the object detection performance on Pascal VOC and COCO datasets and the instance segmentation performance on COCO. We compare different architectures, namely C4 and FPN. From Table 11, we find some interesting phenomenons. **(1)** With architectures using feature pyramid network, TWIST achieves state-of-the-art results. Clustering-based methods also perform pretty well. **(2)** For C4 architectures, TWIST and clustering-based methods perform worse than contrastive learning methods like MoCo-v2.

We thought the reason of the above phenomenons is that the classification-based methods (TWIST and clustering-based methods) tend to capture category-level invariances instead of instance-level invariances, which makes the outputs of the last convolutional layer discard intra-class variations that is useful for dense predictive tasks. When using FPN-based detectors, features of different layers are combined to compensate for the discarded information from the last layer. Less work concentrates on the effect of the intermediate layers of self-supervised models, while we find the intermediate features may preserve useful information for dense tasks. When using the FPN detectors, TWIST even outperforms those self-supervised methods designed specifically for the dense tasks, such as DenseCL (Wang et al., 2021), i.e., +1.6 AP on COCO detection.

We find the similar phenomenon on semantic segmentation, shown in Table 12. We give results of semantic segmentation with different architectures. The FCN architecture has no fusion of different layers, while FCN-FPN and DeepLab v3+ have the fusion operation. From Table 12, we could observe that when combining with feature pyramid network, our method achieves best or competitive results. Without fusion of different layers of features, TWIST and other clustering-based models perform worse than contrastive learning methods.

Table 11: Object detection and instance segmentation results with different architectures. For methods marked with †, we download the pre-trained models and run the detection and segmentation by ourselves. We report results both with C4 architecture and FPN architecture. For VOC dataset, we run 5 times and report the average.

Method	VOC07+12 det			COCO det			COCO instance seg		
	AP_{all}	AP_{50}	AP_{75}	AP^{bb}_{all}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}_{all}	AP^{mk}_{50}	AP^{mk}_{75}
C4									
Sup	53.5	81.3	58.8	38.2	58.2	41.2	33.3	54.7	35.2
Moco-v2	57.4	82.5	64.0	39.3	58.9	42.5	34.4	55.8	36.5
SimCLR †	57.0	82.4	63.5	38.5	58.5	41.7	33.8	55.1	36.0
SwAV	56.1	82.6	62.7	38.4	58.6	41.3	33.8	55.2	35.9
DINO †	55.2	81.8	61.3	37.4	57.8	40.0	33.0	54.3	34.9
DC-v2 †	54.2	81.6	59.9	37.0	57.7	39.5	32.8	54.2	34.4
SimSiam	57	82.4	63.7	39.2	59.3	42.1	34.4	56.0	36.7
BarlowTwins	56.8	82.6	63.4	39.2	59.0	42.5	34.3	56.0	36.5
TWIST	55.3	82.2	61.2	38.0	58.4	40.8	33.5	54.9	35.5
FPN									
Moco-v2	56.4	81.6	62.4	39.8	59.8	43.6	36.1	56.9	38.7
SimCLR †	**58.2**	83.8	65.1	41.6	61.8	45.6	37.6	59.0	40.5
SwAV	57.2	83.5	64.5	41.6	62.3	**45.7**	**37.9**	59.3	**40.8**
DC-v2 †	57.0	83.7	64.1	41.0	61.8	45.1	37.3	58.7	39.9
DINO †	57.2	83.5	63.7	41.4	62.2	45.3	37.5	58.8	40.2
DenseCL	56.9	82.0	63.0	40.3	59.9	44.3	36.4	57.0	39.2
TWIST	58.1	**84.2**	**65.4**	**41.9**	**62.6**	**45.7**	**37.9**	**59.7**	40.6

Table 12: Semantic segmentation with different architectures. All results are averaged over 5 trials.

Method	FCN		FCN-FPN		DeepLab-v3	
	VOC	Cityscapes	VOC	Cityscapes	VOC	Cityscapes
Rand init	40.7	63.5	37.9	62.9	49.5	68.3
Sup	67.7	73.7	70.4	75.4	76.6	78.6
Moco-v2	67.5	74.5	67.5	75.4	72.9	78.6
SimCLR †	68.9	72.9	72.8	74.9	78.5	77.8
SwAV	66.4	71.4	71.9	74.4	77.2	77.0
DC-v2 †	65.6	70.8	72.1	73.8	76.0	76.2
DINO †	66.0	71.1	71.9	73.8	76.4	76.2
TWIST	66.7	71.5	73.3	74.6	77.3	76.9

Figure 8: Randomly chosen classes. We ***randomly*** choose 24 classes for visualization. For each class, we ***randomly*** choose 25 pictures to display. Note we did not make any selections on the pictures or the categories, all the categories and images are randomly chosen to give readers the accurate impression.

F EXAMPLES FOR UNSUPERVISED CLASSIFICATION

To give readers a qualitative impression on the performance of unsupervised classification, we display the top-5 predictions of some randomly selected pictures, shown as follows. Specifically, the labels are mapped to the labels in ImageNet by Kuhn–Munkres algorithm. **Note that the labels are only used to map our predictions to the meaningful ImageNet label descriptions, we do not use any label to participate in the training process.**


Border terrier 0.253
wire-haired fox terrier 0.164
Norfolk terrier 0.139
guenon 0.054
Tibetan mastiff 0.006
ground-truth: Lakeland terrier


vault 0.910
altar 0.048
organ 0.002
theater curtain 0.000
restaurant 0.000
ground-truth: church


indigo bunting 0.890
junco 0.029
birdhouse 0.001
vulture 0.000
chickadee 0.000
ground-truth: house finch


fireboat 0.989
steel arch bridge 0.000
liner 0.000
schooner 0.000
lifeboat 0.000
ground-truth: fireboat


yellow lady's slipper 0.994
chickadee 0.000
bell pepper 0.000
hip 0.000
jackfruit 0.000
ground-truth: yellow lady's slipper