Prototypical Contrastive Learning of Unsupervised Representations

Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, Steven C.H. Hoi
Salesforce Research

Abstract

This paper presents Prototypical Contrastive Learning (PCL), an unsupervised representation learning method that addresses the fundamental limitations of instance-wise contrastive learning. PCL not only learns low-level features for the task of instance discrimination, but more importantly, it implicitly encodes semantic structures of the data into the learned embedding space. Specifically, we introduce prototypes as latent variables to help find the maximum-likelihood estimation of the network parameters in an Expectation-Maximization framework. We iteratively perform E-step as finding the distribution of prototypes via clustering and M-step as optimizing the network via contrastive learning. We propose ProtoNCE loss, a generalized version of the InfoNCE loss for contrastive learning, which encourages representations to be closer to their assigned prototypes. PCL achieves state-of-the-art results on multiple unsupervised representation learning benchmarks, with >10% accuracy improvements in low-resource transfer tasks[1].

1 Introduction

Unsupervised visual representation learning aims to learn image representations from pixels themselves without relying on semantic annotations, and recent advances are largely driven by instance discrimination tasks [1, 2, 3, 4, 5, 6, 7]. These methods usually consist of two key components: image transformation and contrastive loss. Image transformation aims to generate multiple embeddings that represent the same image, by data augmentation [2, 8, 9], patch perturbation [4], or using momentum features [3]. The contrastive loss, in the form of a noise contrastive estimator [10], aims to bring closer samples from the same instance and separate samples from different instances. Essentially, instance-wise contrastive learning leads to an embedding space where all instances are well-separated, and each instance is locally smooth (*i.e.* input with perturbations have similar representations).

Despite their improved performance, instance discrimination based methods share a common fundamental weakness: semantic structure of data is not encoded by the learned representations. This problem arises because instance-wise contrastive learning treats two samples as a negative pair as long as they are from different instances, regardless of their semantic similarity. This is magnified by the fact that thousands of negative samples are generated to form the contrastive loss, leading to many negative pairs that share similar semantics but are undesirably pushed apart in the embedding space.

In this paper, we propose *prototypical contrastive learning* (PCL), a new framework for unsupervised representation learning that implicitly encodes the semantic structure of data into the embedding space. Figure 1 shows an illustration of PCL. A prototype is defined as "a representative embedding for a group of semantically similar instances". We assign several prototypes of different granularity to each instance, and construct a contrastive loss which enforces the embedding of a sample to be more similar to its corresponding prototypes compared to other prototypes. In practice, we can find prototypes by performing standard clustering on the embeddings.

[1]Code is available at: https://github.com/salesforce/PCL

Preprint. Under review.

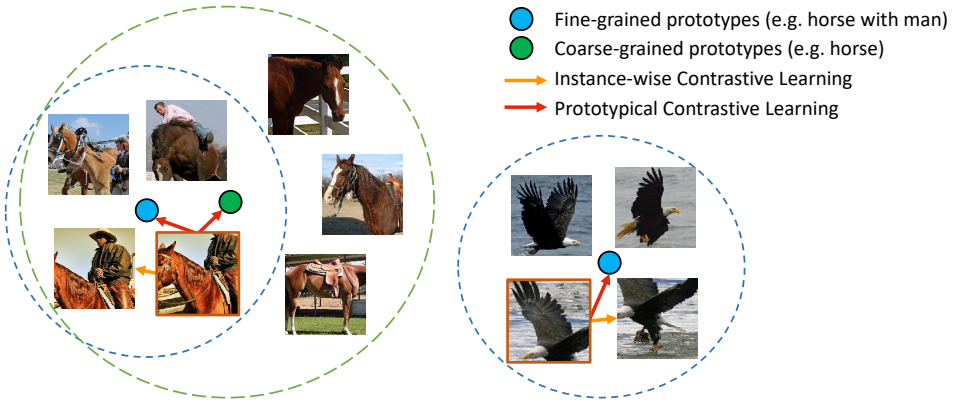


Figure 1: Illustration of Prototypical Contrastive Learning. Each instance is assigned to multiple prototypes with different granularity. PCL learns an embedding space which encodes the semantic structure of data.

We formulate prototypical contrastive learning as an Expectation-Maximization (EM) algorithm, where the goal is to find the parameters of a Deep Neural Network (DNN) that best describes the data distribution, by iteratively approximating and maximizing the log-likelihood function. Specifically, we introduce prototypes as additional latent variables, and estimate their probability in the E-step by performing k-means clustering. In the M-step, we update the network parameters by minimizing our proposed contrastive loss, namely *ProtoNCE*. We show that minimizing ProtoNCE is equivalent to maximizing the estimated log-likelihood, under the assumption that the data distribution around each prototype is isotropic Gaussian. Under the EM framework, the widely used instance discrimination task can be explained as a special case of prototypical contrastive learning, where the prototype for each instance is its augmented feature, and the Gaussian distribution around each prototype has the same fixed variance. The contributions of this paper can be summarized as follows:

- We propose prototypical contrastive learning, a novel framework for unsupervised representation learning. The learned representation not only preserves the local smoothness of each image instance, but also captures the hierarchical semantic structure of the global dataset.
- We give a theoretical framework that formulates PCL as an Expectation-Maximization (EM) based algorithm. The iterative steps of clustering and representation learning can be interpreted as approximating and maximizing the log-likelihood function. The previous methods based on instance discrimination form a special case in the proposed EM framework.
- We propose ProtoNCE, a new contrastive loss which improves the widely used InfoNCE [1, 3, 6, 9] by dynamically estimating the concentration for the feature distribution around each prototype. We provide explanations for PCL from an information theory perspective, by showing that the learned prototypes contain more information about the image classes.
- PCL sets a new state-of-the-art for unsupervised representation learning on multiple benchmarks.

2 Related work

Our work is closely related to two main branches of studies in unsupervised/self-supervised learning: instance-wise contrastive learning and deep unsupervised clustering.

Instance-wise contrastive learning. At the core of state-of-the-art unsupervised representation learning algorithms [1, 2, 3, 4, 11, 5, 6, 7, 9], instance-wise contrastive learning aims to learn an embedding space where samples (*e.g.* crops) from the same instance (*e.g.* an image) are pulled closer and samples from different instances are pushed apart. To construct the contrastive loss for a mini-batch of samples, positive instance features and negative instance features are generated for each sample. Different contrastive learning methods vary in their strategy to generate instance features. The *memory bank* approach [1] stores the features of all samples calculated in the previous step, and selects from the memory bank to form positive and negative pairs. The *end-to-end* approach [2, 7, 9] generates instance features using all samples within the current mini-batch, and apply the same encoder to both the original samples and their augmented version. Recently, the *momentum encoder* (MoCo) approach [3] is proposed, which encodes samples on-the-fly by a momentum-updated encoder, and maintains a queue of instance features.

Despite their improved performance, the existing methods based on instance-wise contrastive learning have the following two major limitations, which can be addressed by the proposed PCL framework.

- The task of instance discrimination could be solved by exploiting low-level image differences, thus the learned embeddings do not necessarily capture high-level semantics. This is supported by the fact that the accuracy of instance classification often rapidly rises to a high level (>90% within 10 epochs) and further training gives limited informative signals. A recent study also shows that better performance of instance discrimination could worsen the performance on downstream tasks [12].
- A sufficiently large number of negative instances need to be sampled, which inevitably yields negative pairs that share similar semantic meaning and should be closer in the embedding space. However, they are undesirably pushed apart by the contrastive loss. Such problem is defined as class collision in [13] and is shown to hurt representation learning. Essentially, instance discrimination learns an embedding space that only preserves the local smoothness around each instance but largely ignores the global semantic structure of the dataset.

Deep unsupervised clustering. Clustering based methods have been proposed for deep unsupervised learning. Approaches in [14, 15, 16, 17, 18, 19] jointly learn image embeddings and cluster assignments, but they have not shown the ability to learn transferable representations from a large scale of images. Closer to our work, DeepCluster [20] learns from millions of images by performing iterative clustering and unsupervised representation learning. However, our method is conceptually different from DeepCluster. In DeepCluster, the cluster assignments are considered as pseudo-labels and a classification objective is optimized, which results in two weaknesses: (1) the high-dimensional features from the penultimate layer of a ConvNet are not optimal for clustering and need to be PCA-reduced; (2) an additional linear classification layer is frequently re-initialized which interferes with representation learning. In our method, representation learning happens directly in a low-dimensional embedding space, by optimizing a contrastive loss on the prototypes (cluster centroids). This frees our method from the computationally expensive linear layer, and enables a much larger number of clusters where each instance is assigned to multiple clusters of different granularity.

Self-supervised pretext tasks. Another line of self-supervised learning methods focus on training DNNs to solve pretext tasks that lead to good image representations being learned. These tasks usually involve hiding certain information about the input and training the network to recover those missing information. Examples include image inpainting [21], colorization [22, 23], prediction of patch orderings [24, 25] and image transformations [26, 27, 28, 29]. However, most of these pretext tasks exploit specific structures of visual data, making them harder to generalize to other domains. The proposed PCL is a more general learning framework with better theoretical justification. Furthermore, PCL can incorporate the pretext tasks (*e.g.* Jigsaw [25] or Rotation [27]) as a form of image transformation, which could potentially lead to improved performance.

3 Prototypical Contrastive Learning

3.1 Preliminaries

Given a training set $X = \{x_1, x_2, ..., x_n\}$ of n images, unsupervised visual representation learning aims to learn an embedding function f_θ (realized via a DNN) that maps X to $V = \{v_1, v_2, ..., v_n\}$ with $v_i = f_\theta(x_i)$, such that v_i best describes x_i. Instance-wise contrastive learning achieves this objective by optimizing a contrastive loss function, such as InfoNCE [6, 3], defined as:

$$\mathcal{L}_{\text{InfoNCE}} = \sum_{i=1}^{n} -\log \frac{\exp(v_i \cdot v_i'/\tau)}{\sum_{j=0}^{r} \exp(v_i \cdot v_j'/\tau)}, \tag{1}$$

where v_i' is a positive embedding for instance i, and v_j' includes one positive embedding and r negative embeddings for other instances, and τ is a temperature hyper-parameter. In MoCo [3], these embeddings are obtained by feeding x_i to a momentum encoder parametrized by θ', $v_i' = f_{\theta'}(x_i)$, where θ' is a moving average of θ.

In prototypical contrastive learning, we use prototypes c instead of v', and replace the fixed temperature τ with a per-prototype concentration estimation ϕ. An overview of our training framework is shown in Figure 2, where clustering and representation learning are performed iteratively at each epoch. Next, we will delineate the theoretical framework of PCL based on EM. A pseudo-code of our algorithm is given in appendix B.

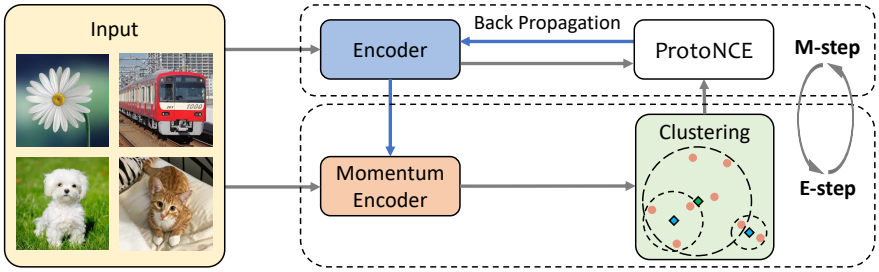


Figure 2: Training framework of Prototypical Contrastive Learning.

3.2 PCL as Expectation-Maximization

Our objective is to find the network parameters θ that maximizes the log-likelihood function of the observed n samples:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{n} \log p(x_i; \theta) \tag{2}$$

We assume that the observed data $\{x_i\}_{i=1}^n$ are related to latent variable $C = \{c_i\}_{i=1}^k$ which denotes the prototypes of the data. In this way, we can re-write the log-likelihood function as:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{n} \log p(x_i; \theta) = \arg\max_{\theta} \sum_{i=1}^{n} \log \sum_{c_i \in C} p(x_i, c_i; \theta) \tag{3}$$

It is hard to optimize this function directly, so we use a surrogate function to lower-bound it:

$$\sum_{i=1}^{n} \log \sum_{c_i \in C} p(x_i, c_i; \theta) = \sum_{i=1}^{n} \log \sum_{c_i \in C} Q(c_i) \frac{p(x_i, c_i; \theta)}{Q(c_i)} \geq \sum_{i=1}^{n} \sum_{c_i \in C} Q(c_i) \log \frac{p(x_i, c_i; \theta)}{Q(c_i)}, \tag{4}$$

where $Q(c_i)$ denotes some distribution over c's ($\sum_{c_i \in C} Q(c_i) = 1$), and the last step of derivation uses Jensen's inequality. To make the inequality hold with equality, we require $\frac{p(x_i, c_i; \theta)}{Q(c_i)}$ to be a constant. Therefore, we have:

$$Q(c_i) = \frac{p(x_i, c_i; \theta)}{\sum_{c_i \in C} p(x_i, c_i; \theta)} = \frac{p(x_i, c_i; \theta)}{p(x_i; \theta)} = p(c_i; x_i, \theta) \tag{5}$$

By ignoring the constant $-\sum_{i=1}^{n} \sum_{c_i \in C} Q(c_i) \log Q(c_i)$ in Equation 4, we should maximize:

$$\sum_{i=1}^{n} \sum_{c_i \in C} Q(c_i) \log p(x_i, c_i; \theta) \tag{6}$$

E-step. In this step, we aim to estimate $p(c_i; x_i, \theta)$. To this end, we perform k-means on the features $v_i' = f_{\theta'}(x_i)$ given by the momentum encoder to obtain k clusters. We define prototype c_i as the centroid for the i-th cluster. Then, we compute $p(c_i; x_i, \theta) = \mathbb{1}(x_i \in c_i)$, where $\mathbb{1}(x_i \in c_i) = 1$ if x_i belongs to the cluster represented by c_i; otherwise $\mathbb{1}(x_i \in c_i) = 0$. Similar to MoCo [3], we found features from the momentum encoder yield more consistent clusters.

M-step. Based on the E-step, we are ready to maximize the lower-bound in Equation 6.

$$\begin{aligned} \sum_{i=1}^{n} \sum_{c_i \in C} Q(c_i) \log p(x_i, c_i; \theta) &= \sum_{i=1}^{n} \sum_{c_i \in C} p(c_i; x_i, \theta) \log p(x_i, c_i; \theta) \\ &= \sum_{i=1}^{n} \sum_{c_i \in C} \mathbb{1}(x_i \in c_i) \log p(x_i, c_i; \theta) \end{aligned} \tag{7}$$

Under the assumption of a uniform prior over cluster centroids, we have:

$$p(x_i, c_i; \theta) = p(x_i; c_i, \theta) p(c_i; \theta) = \frac{1}{k} \cdot p(x_i; c_i, \theta), \tag{8}$$

where we set the prior probability $p(c_i; \theta)$ for each c_i as $1/k$ since we are not provided any samples.

We assume that the distribution around each prototype is an isotropic Gaussian, which leads to:

$$p(x_i; c_i, \theta) = \exp\left(\frac{-(v_i - c_s)^2}{2\sigma_s^2}\right) \Big/ \sum_{j=1}^{k} \exp\left(\frac{-(v_i - c_j)^2}{2\sigma_j^2}\right), \tag{9}$$

where $v_i = f_\theta(x_i)$ and $x_i \in c_s$. If we apply ℓ_2-normalization to both v and c, then $(v-c)^2 = 2-2v\cdot c$. Combining this with Equations 3, 4, 6, 7, 8, 9, we can write maximum log-likelihood estimation as

$$\theta^* = \arg\min_\theta \sum_{i=1}^{n} -\log \frac{\exp(v_i \cdot c_s/\phi_s)}{\sum_{j=1}^{k} \exp(v_i \cdot c_j/\phi_j)}, \tag{10}$$

where $\phi \propto \sigma^2$ denotes the concentration level of the feature distribution around a prototype and will be introduced later. Note that Equation 10 has a similar form as the InfoNCE loss in Equation 1. Therefore, InfoNCE can be interpreted as a special case of the maximum log-likelihood estimation, where the prototype for a feature v_i is the augmented feature v_i' from the same instance (*i.e.* $c = v'$), and the concentration of the feature distribution around each instance is fixed (*i.e.* $\phi = \tau$).

In practice, we take the same approach as NCE and sample r negative prototypes to calculate the normalization term. We also cluster the samples M times with different number of clusters $K = \{k_m\}_{m=1}^{M}$, which enjoys a more robust probability estimation of prototypes that encode the hierarchical structure. Furthermore, we add the InfoNCE loss to retain the property of local smoothness and help bootstrap clustering. Our overall objective, namely **ProtoNCE**, is defined as

$$\mathcal{L}_{\text{ProtoNCE}} = \sum_{i=1}^{n} -\left(\log \frac{\exp(v_i \cdot v_i'/\tau)}{\sum_{j=0}^{r} \exp(v_i \cdot v_j'/\tau)} + \frac{1}{M} \sum_{m=1}^{M} \log \frac{\exp(v_i \cdot c_s^m/\phi_s^m)}{\sum_{j=0}^{r} \exp(v_i \cdot c_j^m/\phi_j^m)} \right). \tag{11}$$

3.3 Concentration estimation

The distribution of embeddings around each prototype has different level of concentration. We use ϕ to denote the concentration estimation, where a smaller ϕ indicates larger concentration. Here we calculate ϕ using the momentum features $\{v_z'\}_{z=1}^{Z}$ that are within the same cluster as a prototype c. The desired ϕ should be small (high concentration) if (1) the average distance between v_z' and c is small, and (2) the cluster contains more feature points (*i.e.* Z is large). Therefore, we define ϕ as:

$$\phi = \frac{\sum_{z=1}^{Z} \|v_z' - c\|_2}{Z \log(Z + \alpha)}, \tag{12}$$

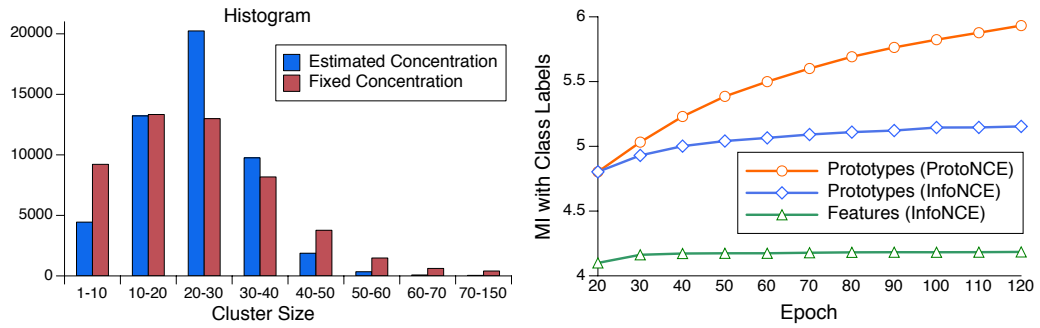
where α is a smooth parameter to ensure that small clusters do not have an overly-large ϕ. We normalize ϕ for each set of prototypes C^m such that they have a mean of τ.

In the ProtoNCE loss (Equation 11), ϕ_s^m acts as a scaling factor on the similarity between an embedding v_i and its prototype c_s^m. With the proposed ϕ, the similarity in a loose cluster (larger ϕ) are down-scaled, pulling embeddings closer to the prototype. On the contrary, embeddings in a tight cluster (smaller ϕ) have an up-scaled similarity, thus less encouraged to approach the prototype. Therefore, learning with ProtoNCE yields more balanced clusters with similar concentration, as shown in Figure 3(a). It prevents a trivial solution where most embeddings collapse to a single cluster, a problem that could only be heuristically addressed by data-resampling in DeepCluster [20].

3.4 Mutual information analysis

It has been shown that minimizing InfoNCE is maximizing a lower bound on the mutual information (MI) between representations V and V' [6]. Similarly, minimizing the proposed ProtoNCE can be considered as simultaneously maximizing the mutual information between V and all the prototypes $\{V', C^1, ..., C^M\}$. This leads to better representation learning, for two reasons.

First, the encoder would learn the *shared* information among prototypes, and ignore the individual noise that exists in each prototype. The shared information is more likely to capture higher-level semantic knowledge. Second, we show that *compared to instance features, prototypes have a larger mutual information with the class labels*. We estimate the mutual information between the instance features (or their assigned prototypes) and the ground-truth class labels for all images in ImageNet [30] training set, following the method in [31]. We compare the obtained MI of our method



(a) (b)

Figure 3: (a) Histogram of cluster size for PCL (#clusters k=50000) with fixed or estimated concentration. Using a different ϕ for each prototype yields more balanced clusters with similar sizes, which leads to better representation learning. (b) Mutual info between instance features (or their assigned prototypes) and class labels of all images in ImageNet. Compared to InfoNCE, our ProtoNCE learns better prototypes with more semantics.

(ProtoNCE) and that of MoCo [3] (InfoNCE). As shown in Figure 3(b), compared to instance features, the prototypes have a larger MI with the class labels due to the effect of clustering. Furthermore, compared to InfoNCE, training on ProtoNCE can increase the MI of prototypes as training proceeds, indicating that better representations are learned to form more semantically-meaningful clusters.

3.5 Prototypes as linear classifier

Another interpretation of PCL can provide more insights into the nature of the learned prototypes. The optimization in Equation 10 is similar to optimizing the cluster-assignment probability $p(s; x_i, \theta)$ using the cross-entropy loss, where the prototypes c represent weights for a linear classifier. With k-means clustering, the linear classifier has a fixed set of weights as the mean vectors for the representations in each cluster, $c = \frac{1}{Z}\sum_{z=1}^{Z} v'_z$. A similar idea has been used for few-shot learning [32], where a non-parametric prototypical classifier performs better than a parametric linear classifier.

3.6 Implementation details

It has been shown that the performance of unsupervised learned representations can be improved by using a larger network, adopting stronger data augmentation, training for more epochs, or using a larger batchsize [33, 9, 34, 4]. However, these improvements usually come at the cost of more computation resources. Therefore, to enable a *fair and direct* comparison with previous methods, we follow the *same setting* for unsupervised training as MoCo [3]. We perform training on the ImageNet-1M dataset with 1000 classes. A ResNet-50 [35] is adopted as the encoder, whose last fully-connected layer outputs a 128-D and L2-normalized feature [1]. We perform additional experiments using a non-linear projection head (a 2-layer MLP), which has been shown to improve representation learning [9, 33]. For efficient clustering, we adopt the GPU k-means implementation in faiss [36]. More details about our method (pseudo-code of our algorithm, convergence proof, experimental settings, cluster analysis, and representation visualization) are given in appendices.

4 Experiments

Following common practice in self-supervised learning [37], we evaluate PCL on transfer learning tasks, based on the principle that a good representation should transfer with limited supervision and limited fine-tuning. The most important baseline is MoCo [3], the recent SOTA instance-wise contrastive learning method. To enable fair and direct comparisons, we follow the settings in [3].

4.1 Image classification with limited training data

Low-shot classification. We evaluate the learned representation on image classification tasks with few training samples per-category. We follow the setup in [37] and train linear SVMs using fixed representations on two datasets: Places205 [38] for scene recognition and PASCAL VOC2007 [39] for object classification. We vary the number k of samples per-class and report the average result across 5 independent runs. Table 1 shows the results, in which our method substantially outperforms both MoCo [3] and SimCLR [9].

Method	architecture	VOC07					Places205				
		k=1	k=2	k=4	k=8	k=16	k=1	k=2	k=4	k=8	k=16
Random	ResNet-50	8.0	8.2	8.2	8.2	8.5	0.7	0.7	0.7	0.7	0.7
Supervised		54.3	67.8	73.9	79.6	82.3	14.9	21.0	26.9	32.1	36.0
Jigsaw [25, 37]	ResNet-50	26.5	31.1	40.0	46.7	51.8	4.6	6.4	9.4	12.9	17.4
MoCo [3]		31.4	42.0	49.5	60.0	65.9	8.8	13.2	18.2	23.2	28.0
PCL (ours)		**46.9**	**56.4**	**62.8**	**70.2**	**74.3**	**11.3**	**15.7**	**19.5**	**24.1**	**28.4**
SimCLR [9]	ResNet-50-MLP	32.7	43.1	52.5	61.0	67.1	9.4	14.2	19.3	23.7	28.3
PCL v2 (ours)		**47.9**	**59.6**	**66.2**	**74.5**	**78.3**	**12.5**	**17.5**	**23.2**	**28.1**	**32.3**

Table 1: **Low-shot image classification** on both VOC07 and Places205 datasets using linear SVMs trained on fixed representations. All methods were pretrained on ImageNet-1M dataset (except for Jigsaw [25, 37] trained on ImageNet-14M). We vary the number of labeled examples k and report the mAP (for VOC) and accuracy (for Places) across 5 runs. Results for Jigsaw were taken from [37]. We use the released pretrained model for MoCo, and re-implement SimCLR. MoCo, SimCLR, and PCL are trained for the same number of epochs (200 epochs).

Semi-supervised image classification. We perform semi-supervised learning experiments to evaluate whether the learned representation can provide a good basis for fine-tuning. Following the setup from [1, 4], we randomly select a subset (1% or 10%) of ImageNet training data (with labels), and fine-tune the self-supervised trained model on these subsets. Table 2 reports the top-5 accuracy on ImageNet validation set. Our method sets a new state-of-the-art under 200 training epochs, outperforming both self-supervised learning methods and semi-supervised learning methods.

Method	architecture	#pretrain epochs	Top-5 Accuracy 1%	10%
Random [1]	ResNet-50	-	22.0	59.0
Supervised baseline [40]	ResNet-50	-	48.4	80.4
Semi-supervised learning methods:				
Pseudolabels [40]	ResNet-50v2	-	51.6	82.4
VAT + Entropy Min. [41, 40]	ResNet-50v2	-	47.0	83.4
S^4L Exemplar [40]	ResNet-50v2	-	47.0	83.7
S^4L Rotation [40]	ResNet-50v2	-	53.4	83.8
Self-supervised learning methods:				
Instance Discrimination [1]	ResNet-50	200	39.2	77.4
Jigsaw [25, 4]	ResNet-50	90	45.3	79.3
SimCLR [9]	ResNet-50-MLP	200	56.5	82.7
MoCo [3]	ResNet-50	200	56.9	83.0
PIRL [4]	ResNet-50	800	57.2	83.8
PCL (ours)	ResNet-50	200	**75.3**	**85.6**

Table 2: **Semi-supervised learning** on ImageNet. We report top-5 accuracy on the ImageNet validation set of self-supervised models that are finetuned on 1% or 10% of labeled data. We use the released pretrained model for MoCo, and re-implement SimCLR; all other numbers are adopted from corresponding papers.

4.2 Image classification benchmarks

Linear classifiers. Next, we train linear classifiers on fixed image representations using the entire labeled training data. We follow previous setup [37, 4] and evaluate the performance of such linear classifiers on three datasets, including ImageNet, VOC07, and Places205. Table 3 reports the results. PCL achieves the highest single-crop top-1 accuracy of all self-supervised methods that use a ResNet-50 model with no more than 200 pretrain epochs.

KNN classifiers. Following [1, 11], we perform k-nearest neighbor (kNN) classification on ImageNet using the learned representations. For a query image with feature v, we take its top k nearest neighbors from the momentum features, and perform weighted-combination of their labels where the weights are calculated by $\exp(v \cdot v'_i/\tau)$. Table 4 reports the accuracy. Our method significantly outperforms previous methods while requiring fewer number of neighbors (20 neighbors as compared to 200 in [1, 11]).

Method	architecture (#params)	#pretrain epochs	Dataset		
			ImageNet	VOC07	Places205
Colorization [22, 37]	R50 (24M)	28	39.6	55.6	37.5
Jigsaw [25, 37]	R50 (24M)	90	45.7	64.5	41.2
Rotation [27, 4]	R50 (24M)	–	48.9	63.9	41.4
DeepCluster [20, 28]	VGG(15M)	100	48.4	71.9	37.9
BigBiGAN [42]	R50 (24M)	–	56.6	–	–
InstDisc [1]	R50 (24M)	200	54.0	–	45.5
MoCo [3]	R50 (24M)	200	60.6	79.2*	48.9*
PCL (ours)	R50 (24M)	200	**61.5**	**82.3**	**49.2**
SimCLR [9]	R50-MLP (28M)	200	61.9	–	–
PCL v2 (ours)	R50-MLP (28M)	200	**67.6**	**85.4**	**50.3**
LocalAgg [11]	R50 (24M)	200	60.2†	–	50.1†
SelfLabel [43]	R50 (24M)	400	61.5	–	–
CPC [6]	R101 (28M)	–	48.7	–	–
CPCv2 [34]	R170$_w$ (303M)	$\sim$200	65.9	–	–
CMC [7]	R50$_{L+ab}$ (47M)	280	64.0	–	–
PIRL [4]	R50 (24M)	800	63.6	81.1	49.8
AMDIM [8]	Custom (626M)	150	68.1†	–	55.0†
SimCLR [9]	R50-MLP (28M)	1000	69.3†	80.5†	–

Table 3: **Image classification with linear models.** We report top-1 accuracy. Numbers with * are from released pretrained model; all other numbers are adopted from corresponding papers.
†: LocalAgg uses 10-crop evaluation. CMC and ADMIM uses FastAutoAugment [44] that is supervised by ImageNet labels. SimCLR requires a large batch size of 4096 allocated on 128 TPUs.

Method	Instance Disc. [1]	MoCo [3]	Local Agg. [11]	PCL (ours)
Accuracy	46.5	47.1	49.4	**54.5**

Table 4: **Image classification with kNN classifiers** using ResNet-50 features on ImageNet. We report top-1 accuracy. Results for [1, 11] are taken from corresponding papers. Result for MoCo is from released model.

4.3 Object detection

We further assess the generalization capacity of the learned representation on object detection. Following [37], we train a Faster R-CNN [45] model on VOC07 or VOC07+12, and evaluate on the test set of VOC07. We keep the pretrained backbone frozen to better evaluate the learned representation, and use the same training schedule for both the supervised and self-supervised methods. Table 5 reports the average mAP across three independent runs. Our method substantially closes the gap between self-supervised methods and supervised training.

In the appendices, we show the results for fine-tuning the pretrained model for object detection and instance segmentation on COCO [46]. PCL outperforms both MoCo and supervised training.

Method	Pretrain Dataset	Architecture	Training data	
			VOC07	VOC07+12
Supervised	ImageNet-1M	Resnet-50-FPN	72.8	79.3
Jigsaw [25, 37]	ImageNet-14M	Resnet-50-C4	62.7	64.8
MoCo [3]	ImageNet-1M	Resnet-50-FPN	66.4	73.5
PCL (ours)	ImageNet-1M	Resnet-50-FPN	**71.7**	**78.5**

Table 5: **Object detection** for frozen conv body on VOC using Faster R-CNN. We measure the average mAP@0.5 on VOC07 test set across three runs.

5 Conclusion

This paper proposed Prototypical Contrastive Learning, a generic unsupervised representation learning framework that finds network parameters to maximize the log-likelihood of the observed data. We introduce prototypes as latent variables, and perform iterative clustering and representation learning in an EM-based framework. PCL learns an embedding space which encodes the semantic structure of data, by training on the proposed ProtoNCE loss. Our extensive experiments on multiple benchmarks demonstrate the state-of-the-art performance of PCL for unsupervised representation learning.

Broader impacts

Our research advances unsupervised representation learning especially for computer vision, which alleviates the need for expensive human annotation when training deep neural network models. By utilizing the enormous amount of unlabeled images available on the web, smarter AI systems can be built with stronger visual representation abilities. However, unsupervised representation learning puts heavy requirement on computational resource during the pretraining stage, which could be costly both financially and environmentally. Therefore, more efforts are needed towards reducing the computational cost for unsupervised learning. As part of the efforts, we will release our pretrained models to facilitate future research in downstream applications without the expensive retraining.

References

[1] Wu, Z., Y. Xiong, S. X. Yu, et al. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, pages 3733–3742. 2018.

[2] Ye, M., X. Zhang, P. C. Yuen, et al. Unsupervised embedding learning via invariant and spreading instance feature. In *CVPR*, pages 6210–6219. 2019.

[3] He, K., H. Fan, Y. Wu, et al. Momentum contrast for unsupervised visual representation learning. In *CVPR*. 2020.

[4] Misra, I., L. van der Maaten. Self-supervised learning of pretext-invariant representations. *arXiv preprint arXiv:1912.01991*, 2019.

[5] Hjelm, R. D., A. Fedorov, S. Lavoie-Marchildon, et al. Learning deep representations by mutual information estimation and maximization. In *ICLR*. 2019.

[6] Oord, A. v. d., Y. Li, O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[7] Tian, Y., D. Krishnan, P. Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[8] Bachman, P., R. D. Hjelm, W. Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

[9] Chen, T., S. Kornblith, M. Norouzi, et al. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[10] Gutmann, M., A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, pages 297–304. 2010.

[11] Zhuang, C., A. L. Zhai, D. Yamins. Local aggregation for unsupervised learning of visual embeddings. In *ICCV*, pages 6002–6012. 2019.

[12] Tschannen, M., J. Djolonga, P. K. Rubenstein, et al. On mutual information maximization for representation learning. In *ICLR*. 2020.

[13] Saunshi, N., O. Plevrakis, S. Arora, et al. A theoretical analysis of contrastive unsupervised representation learning. In *ICML*, pages 5628–5637. 2019.

[14] Xie, J., R. B. Girshick, A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487. 2016.

[15] Yang, J., D. Parikh, D. Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, pages 5147–5156. 2016.

[16] Liao, R., A. G. Schwing, R. S. Zemel, et al. Learning deep parsimonious representations. In *NIPS*, pages 5076–5084. 2016.

[17] Yang, B., X. Fu, N. D. Sidiropoulos, et al. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, pages 3861–3870. 2017.

[18] Chang, J., L. Wang, G. Meng, et al. Deep adaptive image clustering. In *ICCV*, pages 5880–5888. 2017.

[19] Ji, X., J. F. Henriques, A. Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, pages 9865–9874. 2019.

[20] Caron, M., P. Bojanowski, A. Joulin, et al. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 139–156. 2018.

[21] Pathak, D., P. Krähenbühl, J. Donahue, et al. Context encoders: Feature learning by inpainting. In *CVPR*, pages 2536–2544. 2016.

[22] Zhang, R., P. Isola, A. A. Efros. Colorful image colorization. In *ECCV*, pages 649–666. 2016.

[23] Zhang, R., P. Isola, A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, pages 1058–1067. 2017.

[24] Doersch, C., A. Gupta, A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, pages 1422–1430. 2015.

[25] Noroozi, M., P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, pages 69–84. 2016.

[26] Dosovitskiy, A., J. T. Springenberg, M. A. Riedmiller, et al. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, pages 766–774. 2014.

[27] Gidaris, S., P. Singh, N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*. 2018.

[28] Caron, M., P. Bojanowski, J. Mairal, et al. Unsupervised pre-training of image features on non-curated data. In *ICCV*, pages 2959–2968. 2019.

[29] Zhang, L., G. Qi, L. Wang, et al. AET vs. AED: unsupervised representation learning by auto-encoding transformations rather than data. In *CVPR*. 2019.

[30] Deng, J., W. Dong, R. Socher, et al. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. 2009.

[31] Ross, B. C. Mutual information between discrete and continuous data sets. *PloS one*, 9(2), 2014.

[32] Snell, J., K. Swersky, R. S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087. 2017.

[33] Chen, X., H. Fan, R. Girshick, et al. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[34] Hénaff, O. J., A. Razavi, C. Doersch, et al. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

[35] He, K., X. Zhang, S. Ren, et al. Deep residual learning for image recognition. In *CVPR*, pages 770–778. 2016.

[36] Johnson, J., M. Douze, H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[37] Goyal, P., D. Mahajan, A. Gupta, et al. Scaling and benchmarking self-supervised visual representation learning. In *ICCV*, pages 6391–6400. 2019.

[38] Zhou, B., À. Lapedriza, J. Xiao, et al. Learning deep features for scene recognition using places database. In *NIPS*, pages 487–495. 2014.

[39] Everingham, M., L. V. Gool, C. K. I. Williams, et al. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[40] Zhai, X., A. Oliver, A. Kolesnikov, et al. S4l: Self-supervised semi-supervised learning. In *ICCV*, pages 1476–1485. 2019.

[41] Miyato, T., S. Maeda, M. Koyama, et al. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(8):1979–1993, 2019.

[42] Donahue, J., K. Simonyan. Large scale adversarial representation learning. In *NeurIPS*, pages 10541–10551. 2019.

[43] Asano, Y. M., C. Rupprecht, A. Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*. 2020.

[44] Lim, S., I. Kim, T. Kim, et al. Fast autoaugment. In *NeurIPS*, pages 6662–6672. 2019.

[45] Ren, S., K. He, R. B. Girshick, et al. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99. 2015.

[46] Lin, T., M. Maire, S. J. Belongie, et al. Microsoft COCO: common objects in context. In *ECCV*, pages 740–755. 2014.

[47] He, K., G. Gkioxari, P. Dollár, et al. Mask R-CNN. In *ICCV*, pages 2980–2988. 2017.

[48] Girshick, R., I. Radosavovic, G. Gkioxari, et al. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[49] Fan, R., K. Chang, C. Hsieh, et al. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[50] Chen, K., J. Wang, J. Pang, et al. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[51] Nguyen, X. V., J. Epps, J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, 2010.

[52] Maaten, L. v. d., G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9:2579–2605, 2008.

A Training details for unsupervised learning

For the unsupervised learning experiment, We follow previous works [3, 1] and perform data augmentation with random crop, random color jittering, random horizontal flip, and random grayscale conversion. We use SGD as our optimizer, with a weight decay of 0.0001, a momentum of 0.9, and a batch size of 256. We train for 200 epochs, where we warm-up the network in the first 20 epochs by only using the InfoNCE loss. The initial learning rate is 0.03, and is multiplied by 0.1 at 120 and 160 epochs. In terms of the hyper-parameters, we set $\tau = 0.1$, $\alpha = 10$, $r = 16000$, and number of clusters $K = \{25000, 50000, 100000\}$. For PCL v2, we follow [9, 33] and use a MLP projection layer, stronger data augmentation with additional Gaussian blur, and temperature $\tau = 0.2$. The clustering is performed per-epoch on center-cropped images. We find over-clustering to be beneficial. We use the GPU k-means implementation in faiss [36] which takes less than 20 seconds. Overall, PCL introduces $\sim 1/3$ computational overhead compared to MoCo.

B Pseudo-code for Prototypical Contrastive Learning

Algorithm 1: Prototypical Contrastive Learning.

Input: encoder f_θ, training dataset X, number of clusters $K = \{k_m\}_{m=1}^M$
$\theta' = \theta$ // initialize momentum encoder as the encoder
while not MaxEpoch **do**
  /* E-step */
  $V' = f_{\theta'}(X)$ // get momentum features for all training data
  for $m = 1$ **to** M **do**
    $C^m = k\text{-means}(V', k_m)$ // cluster V' into k_m clusters, return prototypes
    $\phi_m = \text{Concentration}(C^m, V')$ // estimate the distribution concentration around each prototype with Equation 12
  end
  /* M-step */
  for x **in** Dataloader(X) **do** // load a minibatch x
    $v = f_\theta(x), v' = f_{\theta'}(x)$ // forward pass through encoder and momentum encoder
    $\mathcal{L}_{\text{ProtoNCE}}(v, v', \{C^m\}_{m=1}^M, \{\phi_m\}_{m=1}^M)$ // calculate loss with Equation 11
    $\theta = \text{SGD}(\mathcal{L}_{\text{ProtoNCE}}, \theta)$ // update encoder parameters
    $\theta' = 0.999 * \theta' + 0.001 * \theta$ // update momentum encoder
  end
end

C Convergence proof

Here we provide the proof that the proposed PCL would converge. Suppose let

$$\begin{aligned} F(\theta) = \sum_{i=1}^{n} \log p(x_i; \theta) = \sum_{i=1}^{n} \log \sum_{c_i \in C} p(x_i, c_i; \theta) = \sum_{i=1}^{n} \log \sum_{c_i \in C} Q(c_i) \frac{p(x_i, c_i; \theta)}{Q(c_i)} \\ \geq \sum_{i=1}^{n} \sum_{c_i \in C} Q(c_i) \log \frac{p(x_i, c_i; \theta)}{Q(c_i)}. \end{aligned} \tag{13}$$

We have shown in Section 3.2 that the above inequality holds with equality when $Q(c_i) = p(c_i; x_i, \theta)$.

At the t-th E-step, we have estimated $Q^t(c_i) = p(c_i; x_i, \theta^t)$. Therefore we have:

$$F(\theta^t) = \sum_{i=1}^{n} \sum_{c_i \in C} Q^t(c_i) \log \frac{p(x_i, c_i; \theta^t)}{Q^t(c_i)}. \tag{14}$$

At the t-th M-step, we fix $Q^t(c_i) = p(c_i; x_i, \theta^t)$ and train parameter θ to maximize Equation 14. Therefore we always have:

$$F(\theta^{t+1}) \geq \sum_{i=1}^{n} \sum_{c_i \in C} Q^t(c_i) \log \frac{p(x_i, c_i; \theta^{t+1})}{Q^t(c_i)} \geq \sum_{i=1}^{n} \sum_{c_i \in C} Q^t(c_i) \log \frac{p(x_i, c_i; \theta^t)}{Q^t(c_i)} = F(\theta^t). \tag{15}$$

The above result suggests that $F(\theta^t)$ monotonously increase along with more iterations. Hence the algorithm will converge.

Method	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
Supervised	40.0	59.9	43.1	34.7	56.5	36.9
MoCo [3]	40.7	60.5	44.1	35.4	57.3	37.6
PCL (ours)	**41.0**	**60.8**	**44.2**	**35.6**	**57.4**	**37.8**

Table 6: Object detection and instance segmentation fine-tuned on COCO. We evaluate bounding-box AP (AP^{bb}) and mask AP (AP^{mk}) on val2017.

D COCO object detection and segmentation

Following the experiment setting in [3], we use Mask R-CNN [47] with C4 backbone. We finetune all layers end-to-end on the COCO train2017 set and evaluate on val2017. The schedule is the default $2\times$ in [48]. PCL outperforms both MoCo [3] and supervised pre-training in all metrics.

E Training details for transfer learning experiments

For training linear SVMs on Places and VOC, we follow the procedure in [37] and use the LIBLINEAR [49] package. We preprocess all images by resizing to 256 pixels along the shorter side and taking a 224×224 center crop. The linear SVMs are trained on the global average pooling features of ResNet-50.

For image classification with linear models, we use the pretrained representations from the global average pooling features (2048-D) for ImageNet and VOC, and the conv5 features (averaged pooled to $\sim$9000-D) for Places. We train a linear SVM for VOC, and a logistic regression classifier (a fully-connected layer followed by softmax) for ImageNet and Places. The logistic regression classifier is trained using SGD with a momentum of 0.9. For ImageNet, we train for 100 epochs with an initial learning rate of 10 and a weight decay of 0. Similar hyper-parameters are used by [3]. For Places, we train for 40 epochs with an initial learning rate of 0.3 and a weight decay of 0.

For semi-supervised learning, we finetune ResNet-50 with pretrained weights on a subset of ImageNet with labels. We optimize the model with SGD, using a batch size of 256, a momentum of 0.9, and a weight decay of 0.0005. We apply different learning rate to the ConvNet and the linear classifier. The learning rate for the ConvNet is 0.01, and the learning rate for the classifier is 0.1 (for 10% labels) or 1 (for 1% labels). We train for 20 epochs, and drop the learning rate by 0.2 at 12 and 16 epochs.

For object detection on VOC, We use the R50-FPN backbone for the Faster R-CNN detector available in the `MMdetection` [50] codebase. We freeze all the `conv` layers and also fix the BatchNorm parameters. The model is optimized with SGD, using a batch size of 8, a momentum of 0.9, and a weight decay of 0.0001. The initial learning rate is set as 0.05. We finetune the models for 15 epochs, and drop the learning rate by 0.1 at 12 epochs.

F Adjusted mutual information

In order to evaluate the quality of the clusters produced by PCL, we compute the adjusted mutual information score (AMI) [51] between the clusterings and the ground-truth labels for ImageNet training data. AMI is adjusted for chance which accounts for the bias in MI to give high values to clusterings with a larger number of clusters. AMI has a value of 1 when two partitions are identical, and an expected value of 0 for random (independent) partitions. In Figure 4, we show the AMI scores for three clusterings obtained by PCL, with number of clusters $K = \{25000, 50000, 100000\}$. In Table 7, we show that compared to DeepCluster [20] and MoCo [3], PCL produces clusters of substantially higher quality.

Method	DeepCluster [20]	MoCo [3]	PCL (ours)
AMI	0.281	0.285	**0.410**

Table 7: Adjusted mutual information score for k-means clustering ($k = 25000$) on ImageNet representation learned with different methods. PCL produces clusters of higher quality.

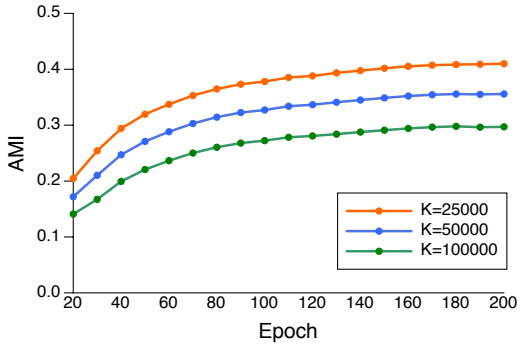


Figure 4: Adjusted mutual information score between the clusterings generated by PCL and the ground-truth labels for ImageNet training data.

G Visualization of learned representation

In Figure 5, we visualize the unsupervised learned representation of ImageNet training images using t-SNE [52]. Compared to the representation learned by MoCo, the representation learned by the proposed PCL forms more separated clusters, which also suggests representation of lower entropy.

H Visualization of clusters

In Figure 6, we show ImageNet training images that are randomly chosen from clusters generated by the proposed PCL. PCL not only clusters images from the same class together, but also finds fine-grained patterns that distinguish sub-classes, demonstrating its capability to learn useful semantic representations.

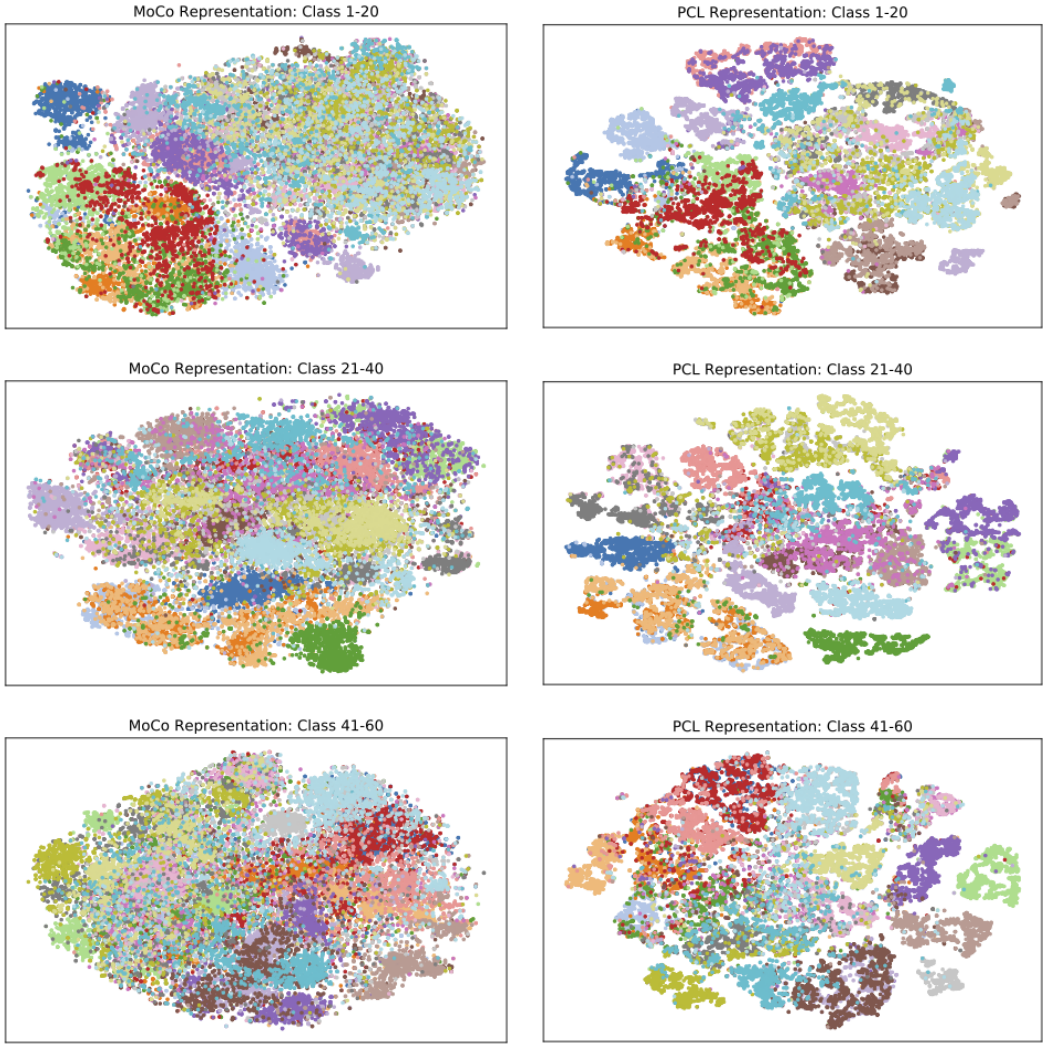


Figure 5: T-SNE visualization of the unsupervised learned representation for ImageNet training images from the first 60 classes. Left: MoCo; Right: PCL (ours). Colors represent classes.

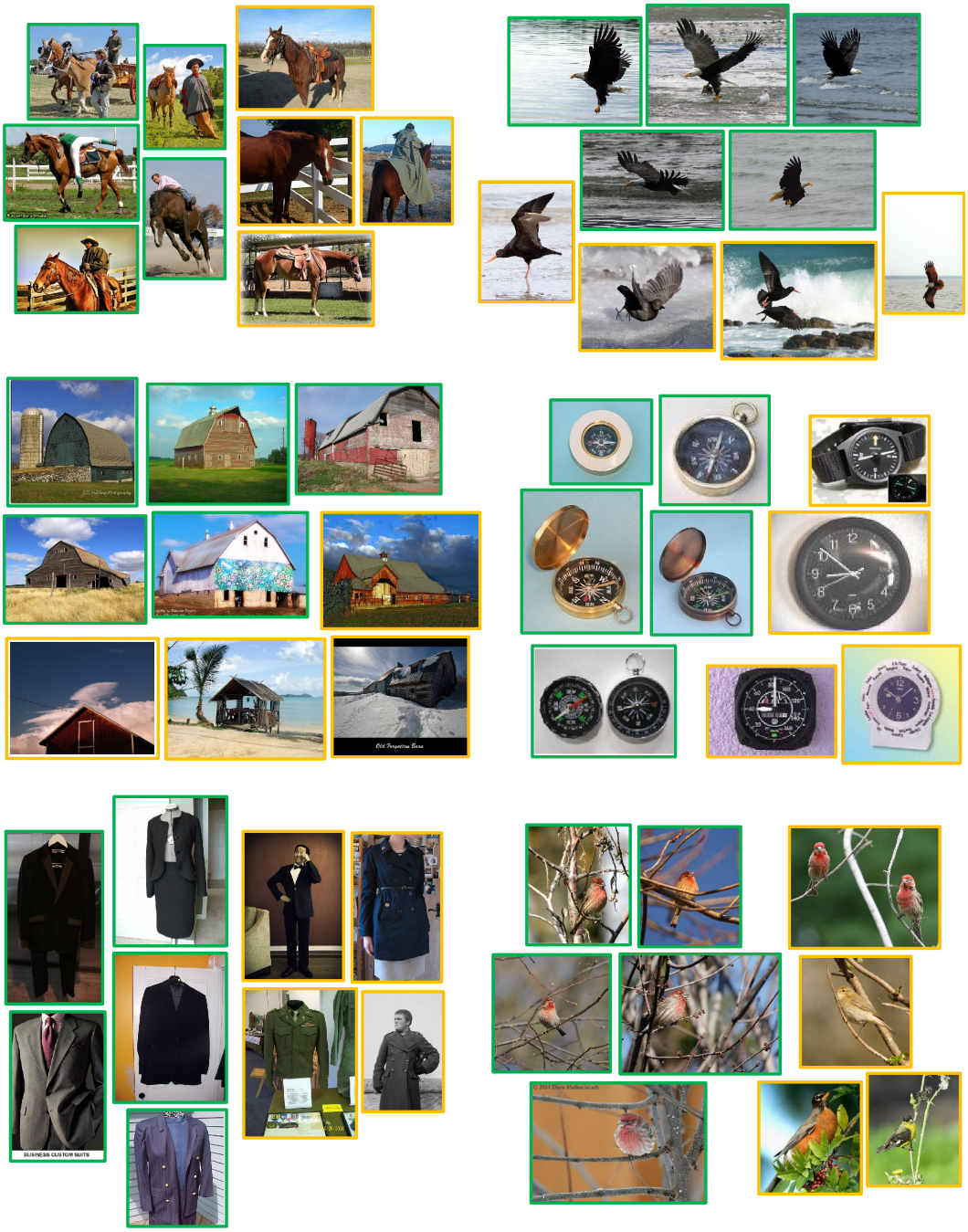


Figure 6: Visualization of randomly chosen clusters generated by PCL. Green boarder marks top-5 images that are closest to fine-grained prototypes ($K = 100k$). Orange boarder marks images randomly chosen from coarse-grained clusters ($K = 50k$) that also cover the same green images. PCL can discover hierarchical semantic structures within the data (*e.g.* images with horse and man form a fine-grained cluster within the coarse-grained horse cluster.)