APPENDIX A TRAINING DETAILS FOR UNSUPERVISED LEARNING

For the unsupervised learning experiment, We follow previous works (He et al., 2020; Wu et al., 2018) 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 Chen et al. (2020a;b) 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 (Johnson et al., 2017) which takes less than 20 seconds. Overall, PCL introduces $\sim 1/3$ computational overhead compared to MoCo.

APPENDIX B ABLATION ON PROTONCE

The proposed loss in eqn.(11) contains two terms: the instance-wise contrastive loss and the proposed prototypical contrastive loss. Here we study the effect of each term on representation learning. Table 7 reports the results for low-resource fine-tuning and linear classification on ImageNet. The prototypical term plays an important role, especially in the fine-tuning experiment. The warm-up also improves the result by bootstrapping the clustering with better representations.

Method	1% fine-tuning (top-5 acc.)	linear classification (top-1 acc.)
instance only	56.9	60.6
proto only (w/o warm-up)	60.7	60.4
proto only (w/ warm-up)	72.3	60.9
instance + proto (w/o warm-up)	74.6	61.3
instance + proto (w/ warm-up)	**75.3**	**61.5**

Table 7: Effect of instance-wise contrastive loss and prototypical contrastive loss.

APPENDIX C 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

APPENDIX D STANDARD DEVIATION FOR LOW-SHOT CLASSIFICATION

In Table 7, we report the standard deviation for the low-shot classification experiments in Table 1.

Method	VOC07					Places205				
	k=1	k=2	k=4	k=8	k=16	k=1	k=2	k=4	k=8	k=16
PCL	4.06	2.65	2.21	0.49	0.39	0.24	0.23	0.13	0.07	0.05
PCL v2	4.12	2.70	2.17	0.54	0.38	0.26	0.23	0.12	0.08	0.04

Table 8: Standard deviation across 5 runs for low-shot image classification experiments.

APPENDIX E COCO OBJECT DETECTION AND SEGMENTATION

Following the experiment setting in (He et al., 2020), we use Mask R-CNN (He et al., 2017) 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 (Girshick et al., 2018). PCL outperforms both MoCo (He et al., 2020) and supervised pre-training in all metrics.

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 (He et al., 2020)	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 9: Object detection and instance segmentation fine-tuned on COCO. We evaluate bounding-box AP (AP^{bb}) and mask AP (AP^{mk}) on val2017.

APPENDIX F TRAINING DETAILS FOR TRANSFER LEARNING EXPERIMENTS

For training linear SVMs on Places and VOC, we follow the procedure in (Goyal et al., 2019) and use the LIBLINEAR (Fan et al., 2008) 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 (He et al., 2020). 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` (Chen et al., 2019) 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.

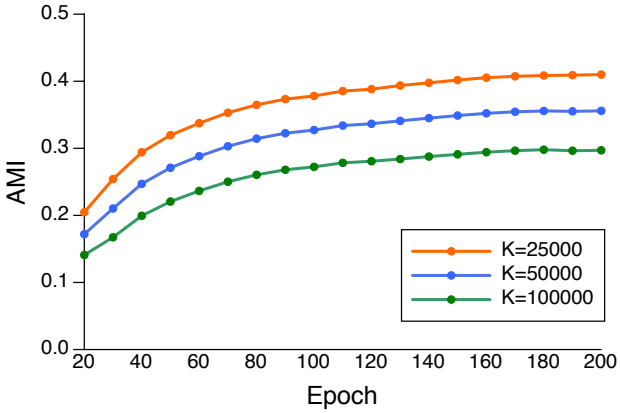

Figure 4: Adjusted mutual information score between the clusterings generated by PCL and the ground-truth labels for ImageNet training data.

APPENDIX G EVALUATION OF CLUSTERING

In order to evaluate the quality of the clusters produced by PCL, we compute the adjusted mutual information score (AMI) (Nguyen et al., 2010) 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 5, we show that compared to DeepCluster (Caron et al., 2018) and MoCo (He et al., 2020), PCL produces clusters of substantially higher quality.

APPENDIX H 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.

APPENDIX I VISUALIZATION OF LEARNED REPRESENTATION

In Figure 5, we visualize the unsupervised learned representation of ImageNet training images using t-SNE (Maaten & Hinton, 2008). 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.

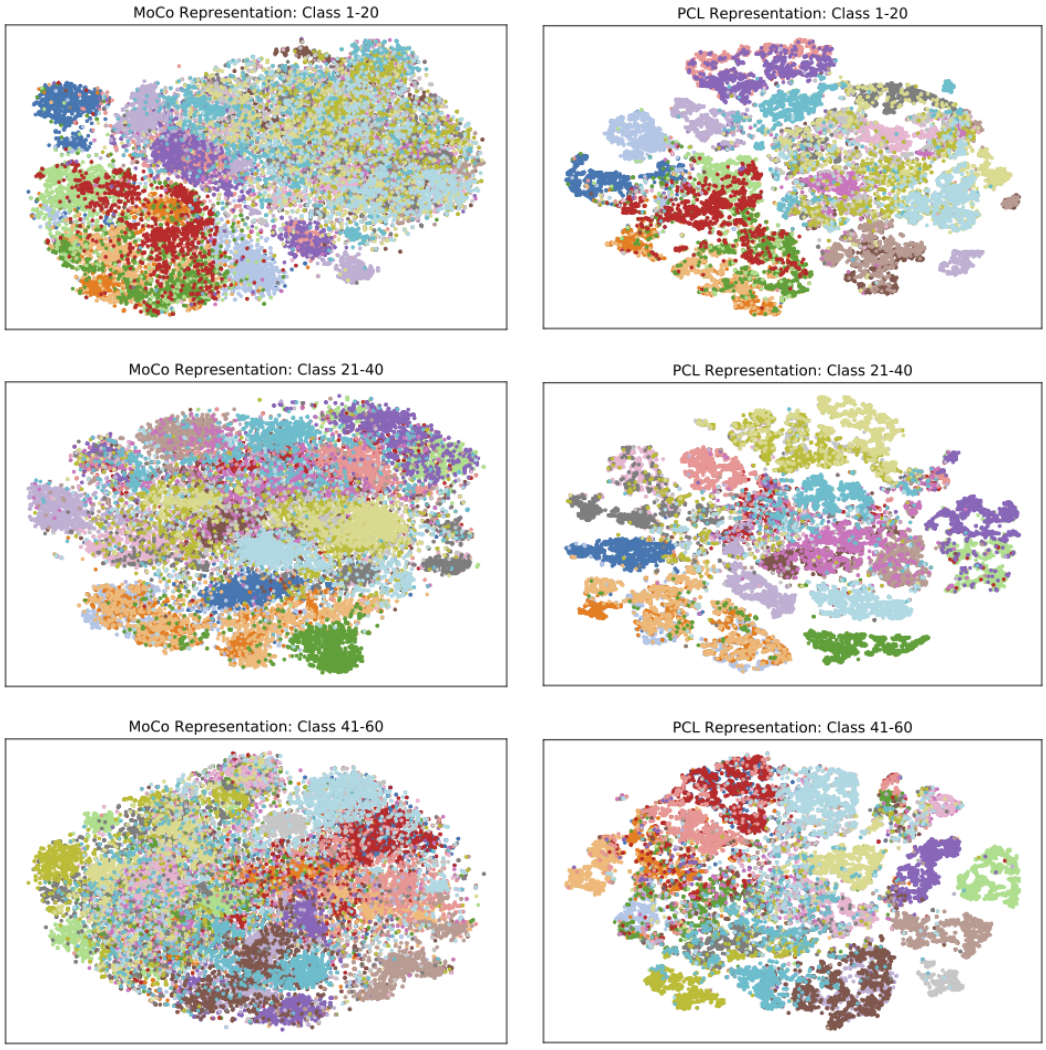

Figure 5: T-SNE visualization of the unsupervised learned representation for ImageNet training images from the first 60 classes. Left: MoCo; Right: PCL (ours). Colors represent classes.

APPENDIX J 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.

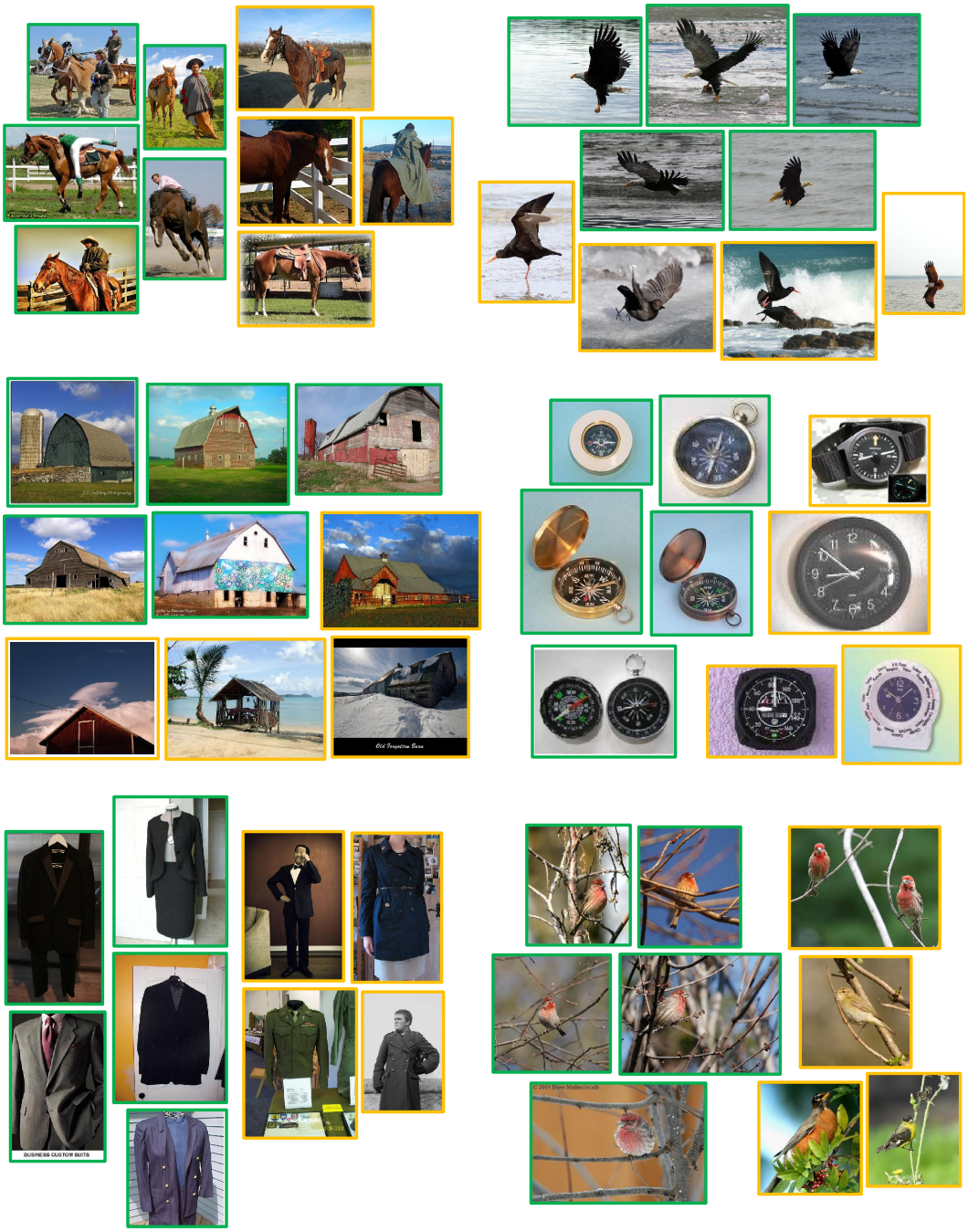

Figure 6: Visualization of randomly chosen clusters generated by PCL. Green boarder marks top-5 images that are closest to fine-grained prototypes ($K = 100k$). Orange boarder marks images randomly chosen from coarse-grained clusters ($K = 50k$) that also cover the same green images. PCL can discover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