We provide further details on Kitchen-HC construction, implementation details, and various choices and experiments we have explored to validate our approach.

1. Kitchen-HC Dataset Construction

The original multi-view RGB-D kitchen dataset [7] is comprised of densely sampled views of several kitchen counter-top scenes with annotations in both 2D and 3D. The viewpoints of the scenes are densely sampled and objects in the scenes are annotated with bounding boxes and in the 3D point cloud. Kitchen-HC is constructed from multi-view RGB-D dataset Kitchen by extracting objects in their 2D bounding boxes. The customized Kitchen-HC dataset has 11 categories with highly correlated samples (from different viewing angles) and 20.8K / 4K / 14.4K instances for training / validation / testing. Fig. A.1 shows sample images in the original RGB-D Kitchen dataset from which our Kitchen-HC data are constructed (See samples used in Fig. 1).

![Figure A.1: Samples of multi-view RGB-D dataset Kitchen [7]. Instances of the same category captured from different perspectives are highly correlated. The high-correlation dataset Kitchen-HC is constructed from Kitchen by extracting objects in their bounding boxes.](image_url)

2. Implementation Details

We use SGD as our optimizer, with weight decay 0.0001 and momentum 0.9. We follow MoCo and NPID [17, 9] and use only standard data augmentation methods for experiments on NPID+CLD and MoCo+CLD: random cropping, resizing, horizontal flipping, color and grayscale transformation, unless otherwise noticed.

1. **ImageNet-{100 [15], ILSVRC-2012 [5], Long-tail [12]}.** For ILSVRC-2012 and ImageNet-LT, we use mini-batch size 256, initial learning rate 0.03, on 8 RTX 2080Ti GPUs. For ImageNet-100, we use batch size 512 and a larger initial learning rate of 0.8 on 8 GPUs, and apply the same setting to baselines and our methods. Training images are randomly cropped and resized to $224 \times 224$. For experiments on MoCov2+CLD with an MLP projection head, we extend the original augmentation in [9] by including the blur augmentation and apply cosine learning rate scheduler to further improve the performance on recognition as in [2]. BYOL+CLD is implemented based on OpenSelfSup [21] benchmark. For experiments on InfoMin+CLD and BYOL+CLD, we follow the same training recipe with InfoMin and BYOL [16, 8] for fair comparisons.

2. **CIFAR-{10, 100, 10-LT, 100-LT}, Kitchen-HC.** As [17], we use mini-batch size 256, initial learning rate 0.03 on 1 GPU for CIFAR [11] and Kitchen-HC. The number of epochs is 200 for CIFAR and 80 for Kitchen-HC. Training images are randomly cropped and resized to $32 \times 32$.

3. **STL-10 [3].** Following [18], we use mini-batch size 256, initial learning rate 0.03, on 2 GPUs. Baseline models and baselines with CLD are trained on "train+unlabelled" split (105k samples), and tested on "test" split (5k samples). Training images are randomly cropped and resized to $96 \times 96$.

4. **Transfer learning on object detection.** We use Faster R-CNN with a backbone of R50-C4, with tuned syn-
chronized batch normalization layers [13] as the detector. As in [9], the detector is fine-tuned for 24k iterations for the experiment on Pascal VOC trainval07+12 and 9k iterations for the experiment on Pascal VOC trainval07. The image scale is [480, 800] pixels during training and 800 at inference. NPID+CLD and MoCo+CLD use the same hyper-parameters as in MoCo [9]. The VOC-style evaluation metric [6] AP50 at IoU threshold is 50% and COCO-style evaluation metric AP are used.

5. Semi-supervised learning. To make fair comparisons with baseline methods, we use OpenSelfSup [21] benchmark to implement baseline results and ours. We follow [20] and fine-tune the pre-trained model on two subsets for semi-supervised learning experiments, i.e. 1% and 10% of the labeled ImageNet-1k training datasets in a class-balanced way. The necks or heads are removed and only the backbone CNN is evaluated by appending a linear classification head.

We apply greedy search on a list of hyper-parameter settings with the base learning rate from \{0.001, 0.01, 0.1\} and the learning rate multiplier for the head from \{1, 10, 100\}. We choose the optimal hyper-parameter setting for each method. Empirically, all baselines and their alternatives with CLD obtain the best performance with a learning rate of 0.01 and a learning rate multiplier for the head of 100. We train the network for 20 epochs using SGD with weight decay 0.0001 and a momentum of 0.9, and a mini-batch of 256 on 4 GPUs. The learning rate is decayed by 5 times at epoch 12 and 16 respectively.

3. Which Clustering Method to Use?

We have tried two popular clustering methods: k-Means clustering and spectral clustering, both implemented in Pytorch for fast performance on GPUs.

k-Means clustering [1, 10] aims to partition \( n \) representations into \( k \) groups, each representation belongs to the cluster with the nearest cluster centroid, serving as a prototype of the cluster. We use spherical k-Means clustering which minimizes: \( \sum (1 - \cos(f_i, u_{c(i)}) \) over all assignments \( c \) of objects \( i \) to cluster ids \( c(i) \in \{1, \ldots, k\} \) and over all prototypes \( u_1, \ldots, u_k \) in the same feature space as the feature vector \( f_i \) representing the objects. We use binary cluster assignment, where the cluster membership \( m_{ij} = 1 \) if item \( i \) is assigned to cluster \( j \) and 0 otherwise. The following k-means objective can be solved using the standard

<table>
<thead>
<tr>
<th>group</th>
<th>spectral</th>
<th>k-Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>77.1%</td>
<td>78.9%</td>
</tr>
<tr>
<td>64</td>
<td>74.5%</td>
<td>76.3%</td>
</tr>
<tr>
<td>128</td>
<td>72.6%</td>
<td>73.4%</td>
</tr>
<tr>
<td>256</td>
<td>70.5%</td>
<td>70.8%</td>
</tr>
</tbody>
</table>

Table A.1: Top-1 kNN accuracies on Kitchen-HC under different group numbers for different clustering methods.

<table>
<thead>
<tr>
<th>NPID+CLD Subspace Cross-augmentation</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>80.8%</td>
<td>51.6%</td>
</tr>
<tr>
<td>✓ share</td>
<td>82.7%</td>
<td>53.3%</td>
</tr>
<tr>
<td>✓ separate</td>
<td>84.2%</td>
<td>55.0%</td>
</tr>
<tr>
<td>✓ separate</td>
<td>86.5%</td>
<td>57.5%</td>
</tr>
</tbody>
</table>

Table A.2: Ablation study on various components of our method, i.e. adding the cross-level discrimination, projecting the representation to two different spaces, and using cross-augmentation comparison between \( x_i \) and \( x'_i \). kNN top-1 accuracy is reported here.

Expectation-Maximization algorithm [4]:

\[
\Phi(M, \{u_1, \ldots, u_C\}) = \sum_{i,j} m_{ij} \left(1 - \cos(f_i, u_{c(i)})\right) \]

\[
= \sum_{i,j} m_{ij} \left(1 - \frac{f_i \cdot u_{c(i)}}{||f_i|| \cdot ||u_{c(i)}||}\right). \tag{1}
\]

Spectral clustering [14, 19] treats data points as nodes of a graph.

1. For feature \( f_i \in \mathbb{R}^{d \times 1} \) of \( N \) samples, we build a weighted graph \( G = (V, E) \), with weight measuring pairwise feature similarity: \( w_{ij} = \frac{f_i \cdot f_j}{||f_i|| \cdot ||f_j||} \).

2. Let \( D \) be the \( N \times N \) diagonal degree matrix with \( d_{ii} = \sum_{j=1}^{n} w_{ij} \) and \( L \) be the normalized Laplacian matrix:

\[
L = D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} \tag{2}
\]

3. We compute the \( k \) largest eigenvalues of \( L \) and use the corresponding \( k \) row-normalized eigenvectors \( E_i \) as the globalized new feature [14, 19]. We apply EM to find the cluster centroids.

Table A.1 shows that k-means clustering achieves better performance on Kitchen-HC and outperforms optimal spectral clustering result by 1.8% when the group number is 10. However, as the group number increases, the performance difference becomes negligible.

4. Are Separate Feature and Group Branches Necessary?

Intuitively, instance grouping and instance discrimination are at odds with each other. Our solution is to formulate
the feature learning on a common representation, forking off two branches where we can impose grouping and discrimination separately. Table A.2 shows that projecting the representation to different spaces and jointly optimize the two losses increase top-1 kNN accuracy by 1.5% and 1.8% on CIFAR-10 and CIFAR-100 respectively.

5. How Effective Is Cross-Augmentation Comparisons?

Instance-level discrimination assumes each instance is its own class and any other instance is a negative. The groups needed for any group-level discrimination have to be built upon local clustering results extracted from the current feature in training, which are fluid and unreliable.

Our solution is to seek the most certainty among all the uncertainties: We presume stable grouping between one instance and its augmented version, and our cross-level discrimination compares the former with the groups derived from the latter. We roll the three processes: instance grouping, invariant mapping, and instance-group discrimination all into one CLD loss.

Table A.2 shows that our cross-augmentation comparison increases the top-1 accuracy by more than 2% on recognition task. It demands the feature not only to be invariant to data augmentation, but also to be respectful of natural grouping between individual instances, often aligning better with downstream semantic classification.

6. How Sensitive Are Hyper-parameters Weight $\lambda$ and Temperature $T$?

$\lambda$ controls the relative importance of CLD with respect to instance-level discrimination, and helps strike a balance between the caveats of noisy initial grouping and the benefits it brings with coarse-grained repulsion between instances and local groups. Table A.3 shows that, at a fixed group number, $\lambda = 0.25$ achieves optimal performance, and a larger $\lambda$ generally leads to worse performance and even decreases top-1 accuracy by 3.1% at $\lambda = 3$. 

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>NPID+CLD</th>
<th>MoCo+CLD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>top-1 (%)</td>
<td>top-5 (%)</td>
</tr>
<tr>
<td>$\lambda = 0$</td>
<td>75.3</td>
<td>92.4</td>
</tr>
<tr>
<td>$\lambda = 0.1$</td>
<td>78.8</td>
<td>94.4</td>
</tr>
<tr>
<td>$\lambda = 0.25$</td>
<td>79.7</td>
<td>95.1</td>
</tr>
<tr>
<td>$\lambda = 0.50$</td>
<td>78.9</td>
<td>94.4</td>
</tr>
<tr>
<td>$\lambda = 1.0$</td>
<td>78.8</td>
<td>94.5</td>
</tr>
<tr>
<td>$\lambda = 3.0$</td>
<td>76.6</td>
<td>93.2</td>
</tr>
</tbody>
</table>

Table A.3: Top-1 and top-5 linear classification accuracies (%) on ImageNet-100 with different $\lambda$’s. The backbone network is ResNet-50.

$T$ is known to critical for discriminative learning and can be sometimes tricky to choose. Table A.4 shows that the best performance is achieved at $T = 0.2$ for both CIFAR and ImageNet-100. With local grouping built into our CLD method, we find the sensitivity of $T$ is greatly reduced.

<table>
<thead>
<tr>
<th>$T(T_f = T_G)$</th>
<th>0.07</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-100</td>
<td>57.9%</td>
<td>57.8%</td>
<td>58.1%</td>
<td>58.1%</td>
<td>57.6%</td>
<td>57.2%</td>
</tr>
<tr>
<td>ImageNet-100</td>
<td>79.3%</td>
<td>79.6%</td>
<td>81.7%</td>
<td>80.7%</td>
<td>79.4%</td>
<td>79.0%</td>
</tr>
</tbody>
</table>

Table A.4: Linear (ImageNet-100) and kNN (CIFAR-100) evaluations for models trained with different choices of temperature $T$, $T_f = T_G$ for simplicity.

7. Is A Larger Memory Bank Always Better for Discriminative Learning?

A larger memory bank includes more negatives and is known to deliver a better discriminator. However, we cannot simply adjust the memory bank size according to NMI or retrieval accuracy in order to deliver the best performance on downstream classification.

Fig. A.2 compares NMI and retrieval accuracies under different negative prototype numbers. If there are too many negatives, the model would focus on repelling negative instances, ignoring the commonality between instances; if there are too few negatives, the model would be subject to random fluctuations from batch to batch, affecting optimization and convergence. However, neither the number of negatives (i.e. infoNCE-k) to obtain the best retrieval accuracy nor the number of negatives to achieve the best NMI score can deliver the best downstream classification task. To deliver optimal performance at downstream classification task, there is a trade-off between local mutual information (evaluated by retrieval task) and global mutual information (evaluated by Normalized Mutual Information).

8. Sample Retrievals

Fig. A.3 shows our near-perfect sample retrievals on ImageNet-100 using $f_I(x)$ in our NPID + CLD model. On the contrary, NPID seems to be much more sensitive to textural appearance (e.g., Rows 1,4,6,7), first retrieve those with similar textures or colors. CLD is able to retrieve semantically similar samples. Our conjecture is that by gathering similar textures into groups, CLD can actually find more informative feature that contrasts between groups. For example, the 5th query image is a Chocolate sauce, which has similar texture with Grouper fish. NPID incorrectly retrieves many images from the Grouper Fish class, but CLD successfully captures the semantic information of the query image, and retrieves instances with the same semantic information.
Figure A.2: MoCo trained with different memory bank sizes are evaluated with NMI, retrieval and kNN accuracy. While a larger memory bank improves the retrieval performance, the classification accuracy and NMI score do not always increase: the NMI score drops sharply due to a large negative/positive ratio. There is a trade-off for best performance at downstream classification.

Figure A.3: Comparisons of top retrieves by NPID (Columns 2-9) and NPID+CLD (Columns 10-17) according to $f_I$ for the query images (Column 1) from the ImageNet validation set. The results are sorted by NPID’s performance: Retrievals with the same category as the query are outlined in green and otherwise in red. NPID seems to be much more sensitive to textural appearance (e.g., Rows 1,4,5,7), first retrieve those with similar textures or colors. Integrated with CLD, NPID+CLD is able to retrieve semantically similar samples. (Zoom in for details)
References


