Center-wise Local Image Mixture For Contrastive Representation Learning

Hao Li[1]
hao_li_96@163.com
Xiaopeng Zhang[2]
zxphistory@gmail.com
Hongkai Xiong[1]
xionghongkai@sjtu.edu.cn

[1] Institute of M.I.N
Shanghai Jiao Tong University
Shanghai, China

[2] Huawei, Inc.
Shanghai, China

1 Implementation details

1.1 Implementation details for contrastive pretraining

Architecture and Optimization. We follow the setting in MoCo v2, which relies on two encoders, one for training and the other one for momentum update ($m = 0.999$) to store negative keys. Following SimCLR, we replace the fc head with a 2-layer MLP to project the output of the final pooling layer to 128-d. We use SGD as optimizer, with weight decay setting as 0.0001 and the momentum as 0.9. We use a mini-batch size of 512 on 16 V100 GPUs with a cosine learning rate schedule decayed from 0.06. We train the model for 1200 epochs, as we introducing data mixing augmentation, and usually requires more epochs for better performance as in supervised learning. It is hard for fair comparison w.r.t. training epochs, since different methods make use of different epochs and batchsize. *e.g.,* BYOL and SimCLR report results on 1000 epochs, while MoCo and SwaV are 800 epochs. We empirically find that for MoCo, more training epochs do not improve the performance further.

Image Augmentations. We combine the proposed augmentations with previous widely used basic augmentation strategies, following the settings in SimCLR and MoCo v2. The basic augmentations are listed below, as well as the corresponding parameters.

- **RandomResizedCrop**: A crop of random size (from 0.2 to 1.0) of the original size and a random aspect ratio (from 3/4 to 4/3) of the original aspect ratio is made.
- **RandomFlip**: Randomly horizontally flip the image with a probability of 0.5.
- **ColorJitter**: Randomly change the brightness, contrast and saturation of an image.
- **RandomGrayscale**: Randomly convert RGB image to grayscale with a probability of 0.2.
- **RandomGaussianBlur**: Randomly blur the image with a probability of 0.5. The radius is randomly sampled from 0.1 to 2.0.

1.2 Details of Positive ample selection

We implement K-means and KNN on 8 v100 GPUs by faiss. For efficiency, we perform clustering and KNN computation every 5 epochs. Since each iteration can be finished within 2 minutes (1 minute for Kmeans and 1 minute for KNN), the extra computation cost is marginal comparing with the budget for model training. The number of clusters is set as $10K$, and we select the top 40 nearest neighbors in KNN. In order to balance the contribution of each image, we randomly select selected positive samples for the following cutmix augmentations. For situations where there remained no selected examples (*e.g.,* the anchor is already around the cluster center), we simply select the most nearest samples among the remained top-40 KNN samples.

2 More Ablation Studies

This section gives more detailed analysis w.r.t. some hyperparameters. Unless specified, we train the model for 200 epochs over the ImageNet-1000 and report the top-1 classification accuracy under linear evaluation protocol.

Number of Clusters (m)	5000			**10000**			20000		
KNN (k)	20	40	60	20	**40**	60	20	40	60
Accuracy (%)	69.1	69.5	69.4	70.0	**70.1**	69.7	69.6	69.9	69.5

Table 1: Impact of the number of clusters m and k of KNN

Frequencies	1 epochs	**5** epochs	10 epochs
Accuracy (%)	70.0	**70.1**	69.8

Table 2: Impact of update frequencies in K-means and KNN

Number of Clusters m and k in KNN. Here we inspect the impact of the number of clusters m in P-means ad the k in KNN to analyze their effect on the performance. In order to ensure local similarity, we restrict the nearest neighbors within a range from 20 to 60. The results for different clusters and top-k neighbors are shown in Table.1. We observe that CLIM consistently improves the performance comparing the baseline Moco 67.5%, and is relatively robust to different m and k. Notably, the best performance is achieved when $m = 10000$, $k = 40$.

We also analyze the update frequency of clustering and KNN. The result for different frequencies are shown in Table.2.

Hyperparameters α in Cutmix. The combination λ in cutmix is sampled from the beta distribution Beta(α, α), where α plays an important role in data mixing augmentation, which controls the strength of interpolation between the anchor and its positive pair. Here we inspect how different $\alpha \in \{1, 1.5, 2, 2.5\}$ affect the representation. As shown in Table 3.

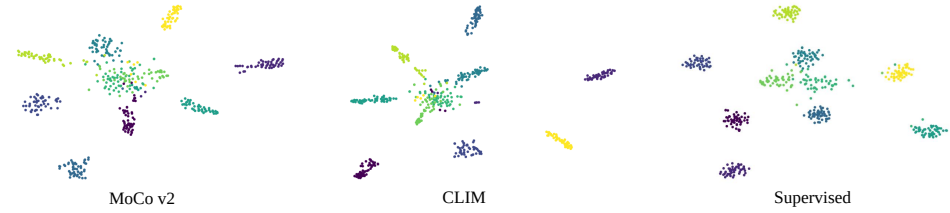

Figure 1: *t-sne* visualization of representation learned by MoCo, CLIM and supervised learning.

We find that the performance is relatively robust to different α, and the best performance is achieved when α is set as 2.

α	1.0	1.5	**2.0**	2.5
Accuracy (%)	69.7	69.9	**70.1**	69.8

Table 3: Impact of α in cutmix

Ablation Study on the Mixing Methods. Our method targets at generating new samples that expanding the neighborhood of an anchor. Here we compare performances of using different data mixing augmentation methods. We try different choices of in beta distribution for Mixup and choose the best one ($\alpha = 0.2$) for comparison. Table.4 shows that Cutmix performs better than Mixup.

Method	Accuracy (%)
Mixup	69.5
Cutmix	**70.1**

Table 4: Ablation study on the mixing methods

Extra Ablation Study on Longer Training Schedule. We compare the improvements brought by different components of our proposed method on the longer training schedule (800 epochs). Table.5 shows the top-1 accuracies under linear evaluation.

3 More Experimental Results

Visualization of Feature Representation. We visualize the feature space to better understand how CLIM augmentation pulls similar samples. Specifically, we randomly choose 10 classes from the validation set and provide the *t-sne* visualization of feature representation generated by CLIM, supervised training and MoCo v2. As shown in Fig. 1, the same color

Method	Accuracy (%)
MoCo v2	71.1
Center-wise + cutmix	73.7
Center-wise + cutmix + Multi-reso	**75.2**

Table 5: Ablation study on the longer training schedule

denotes features with the same label. It can be shown that CLIM takes on higher aggregation property comparing with MoCo, and the fully supervised learned representation reveals the highest aggregation due to it makes use of image labels. Furthermore, we compute the intra-class similarity as the average cosine distance among all intra-class pairwise samples, and report the average similarity across 1000 classes, as shown in Table.6, CLIM achieves an intra-class similarity of 0.65, which is much higher than that in MoCo v2 with similarity of only 0.58. As comparison, we also list the result of supervised learning, with a similarity metric of 0.75.

Method	Intra-class Similarity
Supervised	0.75
MoCo v2	0.58
CLIM	**0.65**

Table 6: Intra-class similarity for different models

Results of Different Training Epochs. In Table.7, we compare CLIM trained with different epochs. Our method achieves an accuracy of 72.3% with only 200 epochs, 75.2% with 800 epochs, and can be further improved to 75.5% when training with 1200 epochs.

Epochs	Accuracy (%)
200	72.3
800	75.2
1200	**75.5**

Table 7: Results of different training epochs