Jigsaw Clustering for Unsupervised Visual Representation Learning
Supplementary File

Pengguang Chen[1], Shu Liu[2], Jiaya Jia[1]

The Chinese University of Hong Kong[1]   SmartMore[2]

{pgchen, leojia}@cse.cuhk.edu.hk   liushuhust@gmail.com

1. Pseudo Code for Our Training Process

Algorithm 1 Training Process for Jigsaw Clustering

```
Input: Data Set $D$ without labels
Output: Model $F$
 1: Initialize model $F$
 2: for Every batch $\mathbf{x} \in D$ do    // Sample a batch of $n$ images from the dataset.
 3:     $\mathbf{x} = Split(\mathbf{x})$    // Split every image in the batch to $m \times m$ patches
 4:     $\mathbf{x} = Transform(\mathbf{x})$    // Apply data augmentation on every patch individually
 5:     $p = RandomPermute(n \times m \times m)$    // Generate a random permutation to permute the input patches
 6:     $\mathbf{x} = \mathbf{x}[p]$    // Permute input patches
 7:     $\mathbf{x} = Stitch(\mathbf{x})$    // Stitch patches into montage images
 8:     $\mathbf{y} = F(\mathbf{x})$    // Forward the montage images to the network and get features
 9:     $\mathbf{y} = Decouple(\mathbf{y})$    // Decouple the features into that of every patch
10:     $\mathbf{z} = MLP(\mathbf{y})$    // Embed features into low dimension logits for the clustering task
11:     $\mathcal{L}_{clu} = SupCluLoss(\mathbf{z}, p\%n)$    // Calculate supervised clustering loss with logits ($\mathbf{z}$) and labels ($p\%n$)
12:     $\mathbf{l} = FC(\mathbf{y})$    // Embed features into logits for location task
13:     $\mathcal{L}_{loc} = CrossEntropyLoss(\mathbf{l}, p/n)$    // Calculate location loss with logits ($\mathbf{l}$) and labels ($p/n$)
14:     $\mathcal{L} = \mathcal{L}_{clu} + \mathcal{L}_{loc}$
15:     $\mathcal{L}.backward()$
16:     $F.update()$
17: end for
```

This is the persudo code of our training process for the Jigsaw Clustering task. In line 11 of Alg. 1, we use $p\%n$ as the label for the clustering task because of the way we arrange the patches. In the split operation, suppose we split an image x_i into 4 patches $(x_i^0, x_i^1, ..., x_i^{m-1})$. Then the n images $x_1, x_2, ..., x_n$ are split into an order of $x_1^0, ...x_n^0, x_1^1, ..., x_n^1, ..., x_1^{m-1}, ..., x_n^{m-1}$. The stitch operation also stitches patches into images following this order. In this condition, the label of the supervised clustering loss and location loss are simply $p\%n$ and p/n respectively.

2. Transfer Learning For Semantic Segmentation

We transfer our models to the Pascal VOC segmentation task and summarize the results in Table 1. We use PSPNet-50 as our basic architecture. And we do not use deep-stem for the backbone to perfectly load our pretrained weights. All models are trained for 100 epochs with a batch size of 8 on 4 GPUs. The learning rate for the backbone and head are initialized as 1e-3 and 1e-2 respectively, and follow a poly decay policy. As we can conclude from the table, weights pretrained by our methods outperform both ImageNet-1k pretrained weights and MoCo v2 pretrained weights. The instance-level intra-image information and detailed location information benefit our methods for downstream tasks.

Model	mIoU (%)	mAcc (%)	allAcc (%)
Supervised	76.92	84.53	94.80
MoCo v2	76.89	84.83	94.97
JigClu (Ours)	**77.51**	**85.88**	**95.01**

Table 1. Results of PSPNet-50 models on Pascal VOC dataset. The backbone is initialized with different pretrained weights.

3. Implementation Details

3.1. Linear Evaluation Details

The training hyperparameters are following [1]. All models are training for 100 epochs with a batch size of 256 on 4 GPUs. The learning rate is initialized as 30, and decay by 0.1 at 60, 80 epochs. And we do not use any regularization methods including weight decay. The data augmentation is following standard ImageNet-1k training, which is composed of random resized crop and random flip. The backbone is frozed including the statics for batchnorm layer. And the newly added linear classifier only contains a fully connected layer, which is randomly initialized.

3.2. Semi-supervised Learning Details

For semi-supervised learning, we train all models for 100 epochs with a batch size of 256. The learning rate decay by 0.1 on 30, 60, 90 epochs. We do not use any regularization methods including weight decay. The data augmentation is following standard ImageNet-1k training. On 10% labels training, we initialize the learning rate for backbone and classifier as 0.01 and 0.2 respectively. On 1% labels training, the learning rate of backbone and classifier are initialized as 0.02 and 5.0 respectively.

3.3. COCO Finetune Details

We finetune our pretrained model on COCO following [1]. All models are trained for 12 epochs with a batch size of 8 on 4 GPUs. The learning rate is initialized as 0.01 and decay by 0.1 on 8 epochs and 10 epochs.

3.4. CIFAR Training Details

We train our models on CIFAR-10 and CIFAR-100 datasets for 100 epochs with a batch size of 128 on a single GPU. The data augmentation is composed of padding, random crop, and random flip. We do not change the stride of models for CIFAR training. For the finetune tasks, the learning rate is initialized as 0.1 and decay by 0.2 on 40 60 and 80 epochs. And we set the weight decay as 5e-4. For the linear classification tasks, the learning rate is initialized as 10 and decay by 0.2 on 40 60 and 80 epochs. We do not use any regularization including weight decay.

References

[1] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2