Collaborative Online Deep Clustering for Unsupervised Representation Learning

Xiaohang Zhan[1], Jiahao Xie[2], Ziwei Liu[1], Yew Soon Ong[2], and Chen Change Loy[2]

[1]Multimedia Laboratory, The Chinese University of Hong Kong
[2]Nanyang Technological University

[1]{zx017, zwliu}@ie.cuhk.edu.hk
[2]{jiahao003, asysong, ccloy}@ntu.edu.sg

Abstract

Joint clustering and feature learning have shown remarkable potential in unsupervised representation learning. However, the efficiency of such methods drops drastically when we need to scale it up to millions of unlabeled data. To overcome this challenge, we propose Online Deep Clustering (ODC) that performs clustering and CNN updating simultaneously rather than alternatively. ODC significantly boosts computational efficiency and learns better representations. Furthermore, it is easy to be extended into a multi-entry framework that allows different models to participate collaboratively. Collaborative Online Deep Clustering (CODC) achieves substantial improvements on both efficiency and performance compared to previous self-supervised learning methods. With CODC framework, we win the first places in multiple tracks of Facebook AI Self-Supervision Challenge 2019.

1. Introduction

Unsupervised representation learning [3, 12, 17, 11, 4, 10, 8, 5, 15, 16] aims at learning meaningful image or video representations that are useful for downstream tasks without manual annotations. Clustering-based representation learning methods [13, 14, 1, 2] arise and show great potential in this area. Conventional clustering is typically performed on fixed features, while these works jointly optimize clustering and feature learning. It is a non-trivial problem; self-training a CNN with clustering would easily lead to shortcut solutions where all samples are assigned to a single cluster [1].

While early works [13, 14] validate their approaches on small datasets, Deep Clustering [1] proposed by Caron *et al*. is the first attempt to scale up clustering-based representation learning. Deep Clustering alternates between clustering and CNN updating with pseudo labels for each epoch. Though good representations emerge during the process, the learning efficiency is compromised. First, before clustering, it has to extract features of the whole dataset. It results in extra computational overhead. Second, the classifier has to be randomly initialized before each epoch, because assigned labels are permuted in the clustering procedure. Hence, it takes long iterations to converge.

To learn good representations with high efficiency, we propose Online Deep Clustering (ODC). Specifically, K-Means clustering alternates between cluster assignment and centroids updating. ODC integrates K-Means clustering into each training step seamlessly. An ODC step contains four steps: 1) CNN forward, 2) CNN updating, 3) label re-assignment and 4) centroids updating. Clustering is performed simultaneously along with CNN updating.

Our Online Deep Clustering (ODC) has several appealing properties. First, with ODC, the labels are instantly reassigned in each iteration rather than in each epoch. It avoids unnecessary back-propagation at the start when the assigned labels are noisy. Second, it avoids costly feature extraction on the entire dataset by maintaining and reusing features in each step. Third, for Deep Clustering [1], the assigned labels are permuted in each epoch; while with ODC, the centroids and assigned labels are updated smoothly, hence the classifier in the CNN also evolves more steadily. In our experiments, ODC surpasses Deep Clustering both in efficiency and performance.

Furthermore, a notable issue of joint clustering and feature learning is drifting, *i.e*., the system gradually exploits trivial shortcut rather than learns meaningful representations. Hence, we further propose Collaborative Online Deep Clustering (CODC), a collaborative framework that allows multiple self-supervised pre-trained models to participate. Different models provide multi-view information in label re-assignment and centroids updating. In this way,

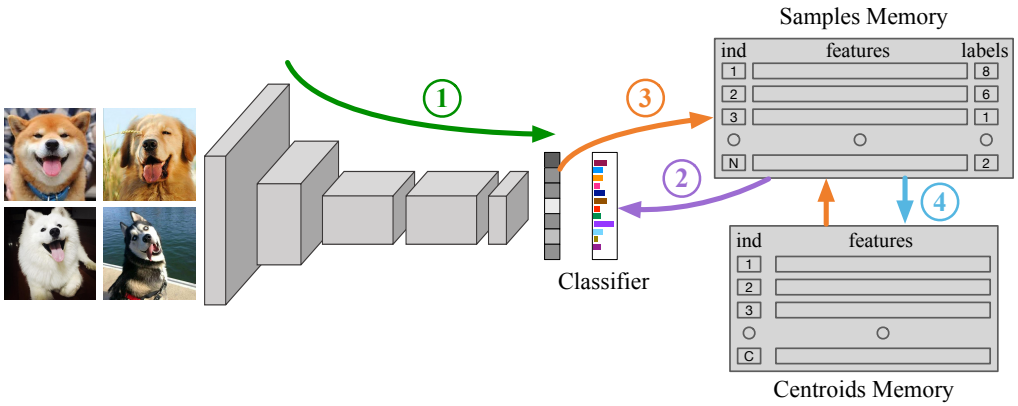

Figure 1. Each ODC iteration mainly contains four steps as the numbers indicating: 1. forward to obtain a compact feature vector; 2. read labels from samples memory and perform back-propagation to update the CNN; 3. update samples memory by updating features and assigning new labels; 4. update centroids memory by recomputing involved centroids.

the models are constrained by each other from drifting into trivial solutions during online deep clustering. With CODC, we achieve much higher performances and win the first place in all tracks in Facebook AI Self-Supervision Challenge 2019.

2. Collaborative Online Deep Clustering

Online Deep Clustering. We first evolve the basic idea of Deep Clustering (DC) to Online Deep Clustering (ODC). Given a random initialized network $f_\theta(*)$ along with a linear classifier $g_w(*)$, our goal is to learn good backbone parameters θ. Apart from the network and the classifier, we maintain a samples memory and a centroids memory, as shown in Figure 1. The former stores features and pseudo labels of the whole dataset, while the latter stores centroids' features. The samples and centroids memories are initialized via an initial clustering procedure.

An ODC iteration mainly contains four steps. First, given a batch of input images x, the network maps the images into compact feature vectors $F = f_\theta(x)$. Second, we read pseudo labels for this batch from the samples memory. Then we update the network with stochastic gradient descent to solve the following problem:

$$\min_{\theta,w} \frac{1}{B} \sum_{n=1}^{B} l\left(g_w\left(f_\theta\left(x_n\right)\right), y_n\right), \tag{1}$$

where y_n is the pseudo label from the samples memory, B denotes the size of each mini-batch. Third, $f_\theta(x)$ is reused to update the samples memory. Specifically, the involved batch of features in the samples memory are directly replaced by the new features. Simultaneously, the involved labels are re-assigned by finding the nearest centroid following $\min_{y\in\{1,..,C\}} \|f_\theta(x) - C_y\|_2^2$. Finally, the involved centroids are updated by averaging the features of samples belonging to the corresponding centroid.

Loss Re-weighting. To avoid ODC from drifting into a few huge clusters, we re-weight the loss according to the number of samples in each class. The weight $w_c \propto \frac{1}{\sqrt{N_c}}$, where N_c denotes the number of samples in class c. In this way, samples in smaller clusters contribute more to the loss.

Dealing with Small Clusters. Even though loss re-weighting has been introduced, small clusters are still under a high risk of collapsing into empty clusters. Denoting normal clusters as C_n whose sizes are larger than a threshold, and small clusters as C_s whose sizes are not, for $c \in C_s$, we first disperse samples in c to the nearest centroids in C_n to make c empty. Next, we split the largest cluster $c_{max} \in C_n$ into two parts by K-Means and randomly choose one part as the new c. We repeat the process until all clusters belong to C_n.

Collaborative Online Deep Clustering. CODC coordinates multiple backbone models with different architectures or initialization methods. The features are concatenated to update the memories and the labels are broadcasted to update each backbone.

Pre-training Details. We use ImageNet containing 128k images for training. Data augmentation includes random cropping (224x224), flipping, rotation ($\pm 2°$) and color jittering. The ODC model (ResNet-50) is trained from scratch for 400K iterations with a constant learning rate 0.01. The CODC model contains three ResNet-50 backbones respectively initialized with Deep Clustering (DC) [1], Rotation Prediction (ROT) [5] and Mixup Image Classification [7] with clustering results (CLS). It is trained for 100K iterations with a constant learning rate 0.01. A head layer of {fc-bn-relu-fc} is added for each backbone to map features in 2048 dimensions into 256 for clustering. The head layers are removed in downstream tasks.

method	devices	time	VOC07 (SVM)
DC [1] (AlexNet)	P100 (x1)	12 days	-
DC [1] (R50)	GTX1080TI (x8)	10 days	69.12
ODC (R50)	GTX1080TI (x8)	**2.7** days	**69.79**

Table 1. Efficiency comparison between DC and ODC.

	CODC (2 models)		CODC (3 models)		
components	DC	ROT	DC	ROT	CLS
before CODC	69.12	67.35	69.12	67.35	77.7
after CODC	75.79	74.54	76.48	72.94	78.05
concatenate	76.33		**79.82**		

Table 2. Ablation study on VOC2007 SVM classification.

3. Implementation Details in Challenge

We trained two CODC models with different numbers of classes, respectively 1K and 4K, and averaged the network parameters of them to get our final CODC model. We did not use magic init [9] to initialize our CODC models. The final CODC model contains three backbone models, *i.e.*, $\{M_I^{R50}|I \in \{DC, ROT, CLS\}\}$. We used the same CODC model for all four tracks. No extra parameters were added to downstream tasks. We provide more details of each track as follows.

Places-205 Classification. The training data was Places-205 train split. We used a single model M_{CLS}^{R50} as initialization. We kept the backbone including all convolution and batch normalization layers frozen and reported the results of layer4 defined in [6]. The hyper-parameters and test setting exactly followed the benchmark evaluation configuration. We reported 1-crop results.

VOC2007 Object Detection. The training data was VOC2007 train+val split. We used a single model M_{CLS}^{R50} as initialization. Following the official instruction, we used Fast R-CNN and froze the backbone in fine-tuning. The proposal files, hyper-parameters exactly followed the official configuration file. No model ensemble or other tricks was adopted.

VOC2007 and VOC2007 Low-shot SVM Classification. The training data was VOC2007 train+val split. We concatenated the output features of the three ResNet-50 models in the final CODC model to train SVMs. We followed the official instruction to train and test SVMs. We reported the results of layer5 defined in [6]. For low-shot classification, the number of samples included {1,2,4,8,16,32,64,96}. The results are shown in Table 2. CODC improves each initial model by a large margin. With a stronger CLS model participating, we achieve our full result 79.82% on VOC2007 classification.

Acknowledgement. We thank Yue Zhao for his participation in discussion of the idea.

References

[1] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 132–149, 2018. 1, 2, 3

[2] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Deepercluster: Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019. 1

[3] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 1

[4] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2017. 1

[5] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 1, 2

[6] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. *arXiv preprint arXiv:1905.01235*, 2019. 3

[7] Yann N. Dauphin David Lopez-Paz Hongyi Zhang, Moustapha Cisse. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 2

[8] Simon Jenni and Paolo Favaro. Self-supervised feature learning by learning to spot artifacts. In *CVPR*, pages 2733–2742, 2018. 1

[9] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. In *ICLR*, 2016. 3

[10] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017. 1

[11] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*. Springer, 2016. 1

[12] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 1

[13] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016. 1

[14] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, pages 5147–5156, 2016. 1

[15] Xiaohang Zhan, Ziwei Liu, Ping Luo, Xiaoou Tang, and Chen Change Loy. Mix-and-match tuning for self-supervised semantic segmentation. In *AAAI*, 2018. 1

[16] Xiaohang Zhan, Xingang Pan, Ziwei Liu, Dahua Lin, and Chen Change Loy. Self-supervised learning via conditional motion propagation. In *CVPR*, June 2019. 1

[17] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*. Springer, 2016. 1