ProFeat: Unsupervised Image Clustering via Progressive Feature Refinement

Jeonghoon Kim
DGIST
jeonghoon@dgist.ac.kr

Sunghoon Im
DGIST
sunghoonim@dgist.ac.kr

Sunghyun Cho
POSTECH CSE & GSAI
s.cho@postech.ac.kr

Abstract

Unsupervised image clustering is a chicken-and-egg problem that involves representation learning and clustering. To resolve the inter-dependency between them, many previous approaches that iteratively perform the two tasks have been proposed. However, their quality can be easily degraded by inaccurate intermediate representations and clusters. To overcome this, we propose ProFeat*, a novel iterative approach to unsupervised image clustering based on progressive feature refinement. Our approach strictly divides representation learning and clustering, and maintains a dedicated network for the representation, which is progressively refined using only confident samples from intermediate clusters. The strict separation between the two tasks allows us to more easily maintain the discriminative power of features and to control the influence of intermediate clusters in the iterative learning process. We also propose ensemble-based approach for more robust clustering. Our experiments demonstrate that* ProFeat *achieves superior clustering results compared to previous methods.*

1. Introduction

Unsupervised image classification, or image clustering, infers semantic clusters of images without supervision. As it can eliminate the burden of manual annotation, unsupervised clustering has recently gained attention as an alternative to supervised image classification [19, 22, 2, 3, 11, 26, 4]. Unsupervised image clustering involves representation learning and clustering. High-quality clustering requires good representations, or features, that reflect the semantic meaning of data. On the other hand, to learn good representations, it is essential to exploit semantic information such as semantic clusters. This inter-dependency makes unsupervised image clustering a chicken-and-egg problem.

To resolve the inter-dependency between representation learning and clustering, various approaches have been proposed. A popular approach is to alternatingly refine clusters and representations [22, 4, 2, 3]. Using initial representations, images are clustered by assigning cluster labels to images, e.g., using K-means [15, 14], or by sampling image pairs that are likely to belong to the same clusters [4]. Then, representations are refined using either all or confident samples from the refined clusters as self-supervision.

Despite simultaneous learning of representations and clusters, previous methods still suffer from the inherent difficulty of unsupervised image clustering. First, inaccurate representations may result in semantically dissimilar images close to each other in the feature space, and eventually lead to inaccurate clusters. Second, inaccurate intermediate clusters may adversely affect the representation learning. Once semantically different images are accidentally clustered together, the following learning process easily fails to learn to discriminate between them.

In this paper, we present *ProFeat*, a novel approach to unsupervised image clustering based on progressive feature refinement. Similarly to previous iterative methods [22, 4, 2, 3], our approach alternates between representation learning and clustering to improve both. To overcome the aforementioned limitations, our approach is designed with two key ideas. First, unlike previous approaches that use a single network for both representation learning and clustering [7, 22], our approach strictly divides the two tasks, and maintains a network dedicated for image representations. Second, our approach progressively refines representations using a small set of confident samples from intermediate clustering results. To sample confident samples, we also propose an ensemble approach that samples confident samples based on the agreement between different models.

Our strategy brings us the following benefits. First, it allows us to use higher dimensional representations, so we can more easily maintain the discriminative power of representations between different images. Second, it enables the representation learning to selectively reflect the intermediate clustering results, while avoiding adversarial influence. Thanks to these, we can learn high-quality representations, and achieve state-of-the-art clustering results.

2. Method

In this section, we first present an overview of our iterative framework, and give detailed introduction to each component. Finally, we explain our ensemble-based approach. In the following, we denote our non-ensemble and ensemble models by *ProFeat*$_s$ and *ProFeat*$_e$, respectively.

2.1. Overview

Given a set of unlabelled images $\mathcal{X} = \{x_i | i = 1, \cdots, N\}$, our goal is to cluster the images into K clusters so that each cluster consists of semantically close images, i.e., we want to learn a function $f(x_i) = y_i$ where y_i is a K-dim. cluster prediction vector such that $y_{i,k} \geq 0\ \forall k$ and $\sum_k y_{i,k} = 1$ where $y_{i,k}$ is the k-th element of y_i. Our framework models $f(x)$ as a composition of an embedding function g and a clustering function h, i.e., $f(x) = h(g(x))$ where g embeds images into the feature space, while h clusters the embedded features. Our framework iteratively learns both g and h by alternatingly performing the representation learning and clustering steps. After the iterations, the final g and h are used for generating final clusters.

2.2. Representation Learning Step

The representation learning step of *ProFeat*$_s$ is built upon the contrastive learning method of SimCLR [5]. This step learns image representations by maximizing the similarity between images belonging to the same cluster in the feature space, while minimizing the similarity between images belonging to different clusters. As clusters are initially unknown, at the first iteration, we assume that all images belong to different clusters, and that different augmentations of a single image belong to the same cluster.

Specifically, at the first iteration, we randomly sample a minibatch of B images, and generate two augmented images of each image in the minibatch, resulting in $2B$ images. Pairs of augmented images from the same source images are treated as positive pairs, while all the other pairs are treated as negative pairs. We denote the sets of positive and negative pairs as $\mathcal{P}$ and $\mathcal{N}$, respectively, and their union by $\mathcal{Q}$ such that $\mathcal{Q} = \mathcal{P} \cup \mathcal{N}$. Then, the embedding function g is trained using a contrastive loss defined as:

$$L_g = -\sum_{(i,j)\in\mathcal{P}} \log \frac{\exp(\mathrm{sim}(z_i, z_j)/\tau)}{\sum_{(k,l)\in\mathcal{Q}_i} \exp(\mathrm{sim}(z_k, z_l)/\tau)} \quad (1)$$

where $\mathrm{sim}(\cdot)$ is a normalized cosine similarity defined as $\mathrm{sim}(u, v) = u^\top v / \|u\| \|v\|$. $\mathcal{Q}_i$ is a subset of $\mathcal{Q}$, which is defined as $\mathcal{Q}_i = \{(i,k) | (i,k) \in \mathcal{Q}\ \ \forall k\}$. As suggested in [5], we compute similarity between projected features, i.e., $z_i = p(g(x_i))$ where p is a projection function that maps $g(x_i)$ to the space where the contrastive loss is applied. p is defined as a multi-layer perceptron with one hidden layer. τ is a temperature parameter.

From the second iteration, we refine both g and p by updating $\mathcal{P}$ and $\mathcal{N}$ using confident samples from the clustering result from the previous iteration. Specifically, as done in the first iteration, we randomly sample a minibatch of B images. From the minibatch, we discard less confident images whose maximum prediction scores $y_i = \max_k y_{i,k}$ are smaller than a pre-defined threshold t_s. We denote the set of the remaining confident images by $\mathcal{C}$. Then, we find all the image pairs (x_i, x_j) from $\mathcal{C}$ such that $y_{i,k} > t_s \wedge y_{j,k} > t_s$ for any k, and include all the pairs of their augmented versions as positive pairs into $\mathcal{P}$. Using all the other remaining pairs derived from $\mathcal{C}$, we generate $\mathcal{N}$. $\mathcal{P}$ also includes pairs of different augmentations of the same images from $\mathcal{C}$. Using the updated $\mathcal{P}$ and $\mathcal{N}$, we fine-tune g and p by minimizing Eq. (1). Note that $\mathcal{P}$ and $\mathcal{N}$ consist of only confident pairs. Thanks to this, g and p can be updated more accurately as will be shown in Sec. 3.

2.3. Clustering Step

The clustering step learns the clustering function h based on the representations from the previous step. This step is built upon SCAN [19], and consists of a semantic clustering step and an optional fine-tuning step. For higher accuracy of intermediate clustering results, this step trains not only h but also an auxiliary embedding function g', which is initialized with g from the previous step. g' is used only for generating intermediate clusters at the current iteration to avoid adversarial effect from incorrect clusters.

The semantic clustering step exploits nearest neighbors to train h. For each x_i in $\mathcal{X}$, we find the M nearest neighbors in the feature space learned at the representation learning step. In our experiments, we use the projected feature space to find nearest neighbors. We denote the set of the M nearest neighbors of x_i as $\mathcal{M}_i$. Then, h and g' are trained to predict the same cluster prediction vectors for the image x_i and its neighbors using a loss defined as:

$$\begin{aligned} L_h &= -\frac{1}{N} \sum_{x_i\in\mathcal{X}} \sum_{x_j\in\mathcal{M}_i} \log \langle h(g'(x_i)), h(g'(x_j)) \rangle \quad (2) \\ &\quad + \lambda \sum_{k=1}^{K} h'_k \log h'_k, \end{aligned}$$

where $\langle\cdot\rangle$ is dot product. The second term on the right-hand side is an entropy term that encourages the predicted clusters to be uniformly distributed, which is defined as $h'_k = \frac{1}{N}\sum_{x_i\in\mathcal{X}} h_k(g'(x_i))$ where $h_k(g'(x_i))$ is the k-th element of the output vector of $h(g'(x_i))$. λ in Eq. (2) is the weight for the entropy term.

After the semantic clustering step, we optionally conduct a fine-tuning step, which uses self-supervision. In this step, we sample confident samples x_i from $\mathcal{X}$ whose $h_k(x_i)$ is larger than a threshold t_f for any k. Then, we generate pseudo-labels y'_i whose every element is zero except for $y'_{i,k} = 1$. We then fine-tune h using the pseudo-labels with the cross-entropy loss. We repeat the sampling of confident samples and fine-tuning until convergence following [19].

Table 1. Performance Comparison with Other Methods.

	CIFAR10			CIFAR100-20			STL10		
Methods	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means [21]	22.9	8.7	4.9	13.0	8.4	2.8	19.2	12.5	6.1
SC [24]	24.7	10.3	8.5	13.6	9.0	2.2	15.9	9.8	4.8
Triplets [18]	20.5	-	-	9.9	-	-	24.4	-	-
JULE [23]	27.2	19.2	13.8	13.7	10.3	3.3	27.7	18.2	16.4
AEVB [12]	29.1	24.5	16.8	15.2	10.8	4.0	28.2	20.0	14.6
SAE [16]	29.7	24.7	15.6	15.7	10.9	4.4	32.0	25.2	16.1
DAE [20]	29.7	25.1	16.3	15.1	11.1	4.6	30.2	22.4	15.2
SWWAE [25]	28.4	23.3	16.4	14.7	10.3	3.9	27.0	19.6	13.6
AE [1]	31.4	23.4	16.9	16.5	10.0	4.7	30.3	25.0	16.1
GAN [17]	31.5	26.5	17.6	15.1	12.0	4.5	29.8	21.0	13.9
DEC [22]	30.1	25.7	16.1	18.5	13.6	5.0	35.9	27.6	18.6
ADC [8]	32.5	-	-	16.0	-	-	53.0	-	-
DeepCluster [2]	37.4	-	-	18.9	-	-	33.4	-	-
DAC [4]	52.2	40.0	30.1	23.8	18.5	8.8	47.0	36.6	25.6
IIC [11]	61.7	51.1	41.1	25.7	22.5	11.7	59.6	49.6	39.7
Han et al. [9]	81.0	-	-	35.3	-	-	66.5	-	-
SCAN [19](Avg)	87.6 ± 0.4	78.7 ± 0.5	75.8 ± 0.7	45.9 ± 2.7	46.8 ± 1.3	30.1 ± 2.1	76.7 ± 1.9	68.0 ± 1.2	61.6 ± 1.8
SCAN [19](Best)	88.3	79.7	77.2	50.7	48.6	33.3	80.9	69.8	64.6
ProFeat$_s$(Avg)	88.7 ± 0.3	80.1 ± 0.5	77.6 ± 0.8	50.0 ± 1.2	48.4 ± 0.9	32.5 ± 1.0	82.4 ± 0.7	72.8 ± 0.6	68.5 ± 0.7
ProFeat$_s$(Best)	89.0	80.7	78.3	51.2	49.7	33.8	83.2	73.4	69.4
ProFeat$_e$(Avg)	92.4 ± 0.3	83.5 ± 0.4	83.1 ± 0.6	52.6 ± 1.1	51.3 ± 0.9	35.5 ± 1.2	84.8 ± 0.5	74.6 ± 0.5	71.2 ± 0.8
ProFeat$_e$(Best)	**92.7**	**84.3**	**84.2**	**54.4**	**52.1**	**37.1**	**85.5**	**75.2**	**72.3**

2.4. Ensemble Learning

To further improve the clustering quality, we present *ProFeat$_e$* that adopts the ensemble learning. *ProFeat$_e$* samples confident samples based on the agreement of different models. Specifically, at each iteration, we train multiple embedding and clustering functions $\{(g_e, h_e)|e = 1, \cdots, E\}$. Then, in the following representation learning step, we sample confident samples gathering all the clustering results from the previous iteration, and generate the set of positive and negative pairs $\mathcal{P}$ and $\mathcal{N}$. We then update the embedding functions using $\mathcal{P}$ and $\mathcal{N}$. Below we describe the sampling process of $\mathcal{P}$ and $\mathcal{N}$ in more detail.

First, during the training of each g_e, we randomly sample a minibatch of B images. For each pair of images (x_i, x_j) in the minibatch, we inspect whether the pair is likely to belong to the same cluster by counting the number of models that agree that they belong to the same cluster. Specifically, we define an ensemble-based confidence $c(x_i, x_j)$ as:

$$c(x_i, x_j) = \sum_{e=1}^{E} \delta_{k_{e,i} k_{e,j}} \tag{3}$$

where δ is a kronecker delta function evaluating to 1 if $k_{e,i} = k_{e,j}$ and 0 otherwise. $k_{e,i}$ is the cluster index with the largest probability for x_i predicted by the e-th model, defined as $k_{e,i} = \operatorname{argmax}_k h_{e,k}(g_e(x_i))$. If $c(x_i, x_j) > t_e$ where t_e is a threshold, we consider them as *connected*, or likely to belong to the same cluster. We denote the set of images with connections by $\mathcal{C}_x$, and the set of connected pairs by $\mathcal{C}_p$.

We may simply consider all the pairs in $\mathcal{C}_p$ as positive pairs and generate $\mathcal{P}$ and $\mathcal{N}$. However, we further prune away less confident samples for more robust sampling. To this end, we introduce connection-based thresholding. In our preliminary experiments, we observed that confident samples are usually connected with a large number of other samples. Based on this, for each x in $\mathcal{C}_x$, we count the number of connections with other images in $\mathcal{C}_p$. If the count is smaller than a pre-defined threshold t_c, we consider x less confident, and discard x and all the pairs with x from $\mathcal{C}_x$ and $\mathcal{C}_p$, respectively. Using the resulting $\mathcal{C}_x$ and $\mathcal{C}_p$, we generate $\mathcal{P}$ and $\mathcal{N}$. Specifically, we augment the images in $\mathcal{C}_x$ as done in Sec. 2.2. Then, we derive positive pairs of the augmented images from $\mathcal{C}_p$, and negative pairs using the augmentations of image pairs (x_i, x_j) such that $x_i \in \mathcal{C}_x$ and $x_j \in \mathcal{C}_x$ but $(x_i, x_j) \notin \mathcal{C}_p$.

The models (g_e, h_e) initially predict different clusters for the same images due to different initialization of the parameters. However, as the iteration goes, they converge to predict the same cluster for a given image even though the indices of the clusters are different. Therefore, after the final iteration, we simply keep only one model.

3. Experiments

We implemented both representation learning and clustering steps based on the implementation of SCAN[1] [19]. Below, we briefly describe important implementation details. More details and the source code can be found in our project webpage[2]. The embedding function g is modeled using the ResNet18 network [10] with a d-dim. output layer after average pooling, where we set $d = 512$. For p, we use

[1]https://github.com/wvangansbeke/Unsupervised-Classification

[2]Will be available after the acceptance of the paper

Table 2. Ablation Study of Our Approaches

	Iterative update	Confident samples	w/ g'	w/ t_c	ACC
Baseline (SCAN)					87.6
(a)	✓				86.9
(b)	✓	✓			87.7
(c)	✓		✓		88.1
ProFeat$_s$	✓	✓	✓		88.7
(d)	✓	✓	✓		89.3
ProFeat$_e$	✓	✓	✓	✓	92.4

two fully-connected layers with 512 and 128 nodes, respectively. We use $\tau = 0.1$. The representation learning step is implemented using the stochastic gradient descent (SGD) method. We set the number of SGD iterations to 500 epochs for the first iteration of *ProFeat*. We set the learning rate to 0.4 and gradually decrease it to 0.0004 through iterations. From the second iteration of *ProFeat*, we run 50 epochs. We use batch size 512. For *ProFeat*$_e$, we set $E = 5$ and $t_e = 5$ in all our experiments. We set t_c to around half the average number of images per class in a minibatch.

We evaluate *ProFeat* using standard datasets: CIFAR10, CIFAR100-20, and STL10 [13, 6]. For STL10, following [19], we use both training set and unlabelled image set for the representation learning at the first iteration. For the other steps, we use only the training set for training. We set t_s to 0.99, 0.99 and 0.95, and t_c to 20, 10, and 1 for CIFAR10, CIFAR100-20, and STL10, respectively. For CIFAR10 and CIFAR100-20, we set the number of iterations to 10 for both *ProFeat*$_s$ and *ProFeat*$_e$. For STL10, we set the number of iterations to 100 and 300 for *ProFeat*$_s$ and *ProFeat*$_e$, respectively. We use the optional fine-tuning step in the clustering step only for CIFAR10 and CIFAR100-20.

Comparison Table 1 compares *ProFeat* against previous methods using clustering accuracy (ACC), normalized mutual information (NMI), and adjusted rand index (ARI). All the values of the previous methods are from their original papers. The performance of a clustering algorithm depends on its initialization. Thus, for fairness, for the previous state-of-the-art method, SCAN [19], we report the average and best performances and the standard deviation of five differently initialized models. We also report the average and best performances of our models measured using five differently initialized models. Table 1 shows that *ProFeat*$_s$ consistently outperforms all the other methods, which proves the effectiveness of our progressive feature refinement. *ProFeat*$_e$ substantially improves the clustering quality thanks to the more accurate sampling of confident samples.

Ablation study In Table 2, 'Iterative update' means our iterative update scheme, i.e., no 'Iterative update' performs only a single iteration of the algorithm. 'Confident samples' indicates whether only confident samples or the entire images are used for updating image representations. 'w/ g'' indicates using the auxiliary embedding function g' instead of g. Not using g' means that the embedding function g is updated both in the representation learning and clustering steps. 'w/ t_c' means using the connection-based thresholding for *ProFeat*$_e$. The baseline performs only the first iteration of *ProFeat*$_s$, which is equivalent to SCAN [19].

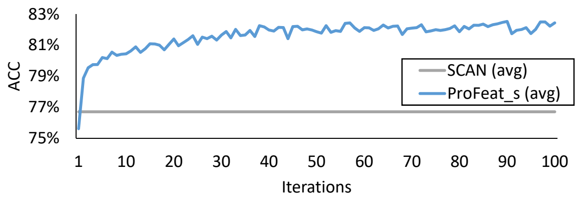

Figure 1. Clustering accuracy with respect to iterations on the STL10 test set.

Table 2(a) shows that naïve iteration between the representation learning and clustering steps does not improve the clustering accuracy. Table 2(b) and (c) show that each of using only confident samples and the auxiliary embedding function improves the accuracy. This result proves that restricting the influence of intermediate clusters to the representation learning is essential for high-quality results. *ProFeat*$_s$ that uses both confident samples and the auxiliary embedding function achieves the best performance. Finally, Table 2(d) and *ProFeat*$_e$ validate the effectiveness of the ensemble learning and connection-based thresholding.

Accuracy w.r.t. iterations Fig. 1 shows the clustering accuracy of *ProFeat*$_s$ with respect to iterations evaluated on the STL10 test set. The accuracy continuously increases as the iteration goes despite some fluctuations, proving that representation learning and clustering help improve each other. In the figure, we also plot the average performance of SCAN [19] for comparison. As SCAN is not an iterative algorithm, we plot its performance as a line. As shown in the figure, although the initial performance of *ProFeat*$_s$ is slightly lower due to the randomness of the initialization, *ProFeat*$_s$ starts to outperform SCAN within only a couple of iterations.

4. Conclusion

This paper presented *ProFeat*, a novel unsupervised image clustering algorithm, which iteratively performs the representation learning and clustering steps. To avoid adversarial effect from incorrect intermediate clusters, our algorithm strictly divides the representation learning and clustering, and progressively refines representations using only confident samples. To robustly sample confident samples, we also presented an ensemble-based strategy. We experimentally validated each component of *ProFeat* and showed that *ProFeat* achieves the state-of-the-art performance.

Acknowledgement This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1C1C1014863, No. 2020R1C1C1013210), Institute of Information communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH)), and the DGIST R&D Program of the Ministry of Science and ICT (20-CoE-IT-01).

References

[1] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[3] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[4] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of Machine Learning and Systems 2020*, pages 10719–10729, 2020.

[6] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[7] Divam Gupta, Ramachandran Ramjee, Nipun Kwatra, and Muthian Sivathanu. Unsupervised clustering using pseudo-semi-supervised learning. In *International Conference on Learning Representations*, 2020.

[8] Philip Haeusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, and Daniel Cremers. Associative deep clustering: Training a classification network with no labels. In *German Conference on Pattern Recognition*, pages 18–32. Springer, 2018.

[9] Sungwon Han, Sungwon Park, Sungkyu Park, Sundong Kim, and Meeyoung Cha. Mitigating embedding and class assignment mismatch in unsupervised image classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[11] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9865–9874, 2019.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[14] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[15] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[16] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[17] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[18] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems*, pages 41–48, 2004.

[19] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European Conference on Computer Vision (ECCV)*, volume 6, 2020.

[20] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[21] Jianfeng Wang, Jingdong Wang, Jingkuan Song, Xin-Shun Xu, Heng Tao Shen, and Shipeng Li. Optimized cartesian k-means. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):180–192, 2014.

[22] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. volume 48 of *Proceedings of Machine Learning Research*, pages 478–487, New York, New York, USA, 20–22 Jun 2016. PMLR.

[23] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

[24] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608, 2005.

[25] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015.

[26] Huasong Zhong, Chong Chen, Zhongming Jin, and Xian-Sheng Hua. Deep robust clustering by contrastive learning, 2020.