C3: Cross-instance guided Contrastive Clustering

Mohammadreza Sadeghi[1,2]
mohammadreza.sadeghi@mcgill.ca
Hadi Hojjati[1,2]
hadi.hojjati@mcgill.ca
Narges Armanfard[1,2]
narges.armanfard@mcgill.ca

[1] Department of Electrical and Computer Engineering, McGill University
Montreal, QC, Canada

[2] Mila - Quebec AI Institute
Montreal, QC, Canada

Abstract

Clustering is the task of gathering similar data samples into clusters without using any predefined labels. It has been widely studied in machine learning literature, and recent advancements in deep learning have revived interest in this field. Contrastive clustering (CC) models are a staple of deep clustering in which positive and negative pairs of each data instance are generated through data augmentation. CC models aim to learn a feature space where instance-level and cluster-level representations of positive pairs are grouped together. Despite improving the SOTA, these algorithms ignore the cross-instance patterns, which carry essential information for improving clustering performance. This increases the false-negative-pair rate of the model while decreasing its true-positive-pair rate. In this paper, we propose a novel contrastive clustering method, Cross-instance guided Contrastive Clustering (C3), that considers the cross-sample relationships to increase the number of positive pairs and mitigate the impact of false negative, noise, and anomaly sample on the learned representation of data. In particular, we define a new loss function that identifies similar instances using the instance-level representation and encourages them to aggregate together. Moreover, we propose a novel weighting method to select negative samples in a more efficient way. Extensive experimental evaluations show that our proposed method can outperform state-of-the-art algorithms on benchmark computer vision datasets: we improve the clustering accuracy by 6.6%, 3.3%, 5.0%, 1.3% and 0.3% on CIFAR-10, CIFAR-100, ImageNet-10, ImageNet-Dogs, and Tiny-ImageNet.

1 Introduction

In the past few years, self-supervised learning [8, 23, 32], particularly contrastive learning, has established itself as a state-of-the-art representation learning algorithm [4]. They generate a transformed version of the input samples and attempt to learn a representation in which augmentations of the same data point are closer together and further away from other points. They managed to outperform other baselines, particularly in

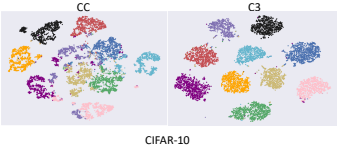

Figure 1: t-SNE visualization of Contrastive Clustering (CC) [22] and the proposed C3

tasks related to computer vision, such as image classification [6], image anomaly detection [14], and object recognition [36]. Encouraged by these results, several studies have attempted to apply contrastive learning to the clustering task. An early attempt by Li *et al*. [22] showed that contrastive clustering could significantly outperform other baselines on benchmark datasets. Despite these improvements, contrastive clustering and the majority of other deep clustering methods do not consider the interrelationship between data samples, which commonly leads to sub-optimal clustering [28]. However, it would be challenging to incorporate information about the cross-relationship of different instances in an unsupervised setting where we do not have access to data labels. As a result, existing unsupervised CC-based algorithms suffer from high false-negative-pair and low true-positive-pair rates because the model cannot effectively take advantage of the degree of similarity of samples.

In this paper, we propose a groundbreaking technique to discover similarities between samples. Then, we employ these similarities to identify positive and negative pairs more accurately. We design a soft-weighting scheme that allows focusing on the more challenging to cluster data points. To realize this goal, we create a pool of *weighted* sample pairs where higher weights are assigned to the samples closer to the data cluster boundaries. Such a weighting scheme mitigates the influence of easy-to-cluster data, noise, and, more importantly, the existing false-negative-pair issue in the CC-based methods. Also, by incorporating a larger number of positive samples, we improve the true-positive-pair rate. Overall, our method significantly improves the training efficiency of the model and leads to a cluster-friendly representation. In the remainder of this paper, we describe our idea more formally. Then, we carry out a series of extensive analyses to show that our scheme can significantly improve the clustering performance, and we try to explain how it can achieve such enhancement.

We summarize the contribution of this work as follows: **(1)** we propose a new contrastive loss function to incorporate the newly discovered positive pairs toward learning a more reliable representation space. **(2)** we propose a novel weighting scheme that aims to separate more challenging data samples, specifically those that are close to cluster boundaries. This improves the representation learning process and greatly impacts the false-negative-pair selection rate. **(3)** by carrying out extensive experiments, we show that our proposed scheme can significantly outperform current state-of-the-art by a significant margin in terms of several clustering criteria. This significant improvement results from considering the pairwise data similarities. **(4)** We offer insight into the behavior of our developed model, discuss the intuition behind how it improves the clustering performance and support them by conducting relevant experiments.

2 Related Works

Deep learning-based clustering methods can be categorized into two groups [43]: (I) Models that use deep networks for embedding the data into a lower-dimensional representation and apply a traditional clustering such as k-means to the new representation, and (II) Algorithms that jointly train the neural network for extracting features and optimizing the clustering results. In order to achieve a more clustering-friendly representation, previous studies have added regularization terms and constraints to the loss function of neural networks. For example, Huang *et al*. [16] proposed the Deep embedding network (DEN), which imposes a locality preserving and a group sparsity constraint to the latent representation of the autoencoder. These two constraints reduce the inner cluster and increase the inter-cluster distances to improve the clustering performance. In another work, Peng *et al*. [26] proposed deep subspace clustering with sparsity prior (PARTY) that enhances the clustering efficiency of the

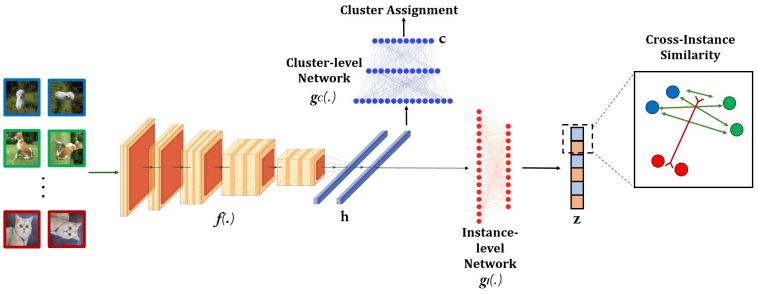

Figure 2: An overview of the training phase of our proposed C3 method.

autoencoder by incorporating the structure's prior in order to consider the relationship between different samples. Sadeghi and Armanfard [30] proposed Deep Multi-Representation Learning for Data Clustering (DML), which uses a general autoencoder for instances that are easily clustered along separate AEs for difficult-to-cluster data to improve the performance. More recent works jointly train the neural network with the clustering objective to further improve the clustering performance. For instance, Deep clustering network (DCN) [38] uses k-means objective as the clustering loss and jointly optimizes it with the loss of an autoencoder. Analogously, Deep embedded clustering (DEC) [37] first embeds the data into a lower-dimensional space by minimizing the reconstruction loss. Then, it iteratively updates the encoder part of the AE by optimizing a Kullback-Leiber (KL) divergence [20] loss between the soft assignments and adjusted target distributions. Following the success of DEC, a series of improved algorithms have been developed. For instance, improved deep embedded clustering with local structure preservation (IDEC) [10] jointly optimizes the clustering loss and AE loss to preserve the local structure of data, IDECF [29] adds a fuzzy c-mean network for improving the auxiliary cluster assignment of IDEC during training, and Deep embedded clustering with data augmentation (DEC-DA) [1] applies the DEC method along with the data augmentation strategy to improve the performance. Several other methods design auxiliary tasks for learning an efficient representation. E.g., JULE [39] applies agglomerative clustering to learn the data representation and cluster assignments. In another algorithm named invariant information clustering (IIC) [17], the mutual information between the cluster assignment of a pair is maximized. Recently, researchers have turned their attention to self-supervised learning (SSL) models for clustering. For example, MMDC (multi-modal deep clustering) [31] improves the clustering accuracy by solving the proxy task of predicting the rotation. SCAN (semantic clustering by adopting nearest neighbors) [33] first obtains a high-level feature representation using self-supervised learning and then improves the clustering performance by incorporating the nearest neighbor prior. Contrastive learning is a self-supervised learning paradigm that learns data representation by minimizing the distance between the augmentations of the same sample while pushing them away from other instances. SimCLR [6] is an example of a contrastive model for learning representation from images that can achieve performance on par with supervised methods. Researchers have increasingly utilized contrastive models for solving tasks such as clustering in the past couple of years. Zhong *et al*. proposed deep robust clustering (DRC) [42] in which a contrastive loss decreases the inter-class variance and another contrastive loss increases the intra-class distance. Contrastive clustering (CC) [22] improves the clustering performance by jointly performing the instance and cluster-level contrastive learning.

3 Method

Given an unlabelled dataset $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ and a predefined cluster number parameter M, the goal of the clustering problem is to partition $\mathcal{X}$ into M disjoint groups. Figure 2. shows the overall scheme of the C3 framework.

Like other contrastive learning methods, we apply two data augmentations $\mathcal{T}^a$ and $\mathcal{T}^b$, sampled randomly from a pool of transformations, $\mathcal{T}$, to form a *true positive pair* (x_i^a, x_i^b) for a sample x_i, where $x_i^a = \mathcal{T}^a(x_i)$ and $x_i^b = \mathcal{T}^b(x_i)$. In this paper, we used SimCLR transformation pool [6]. As is shown in Figure 2. , we use the encoder network $f(.)$ to extract features of augmented samples, i.e, $h_i^a = f(x_i^a)$ and $h_i^b = f(x_i^b)$. Inspired by CC [22], we devise instance-level and cluster-level contrastive networks, denoted by $g_I(.)$ and $g_C(.)$, respectively. The instance-level network maps the extracted feature of augmented samples to the latent representation (aka z-space), i.e. $z_i^a = g_I(h_i^a)$ and $z_i^b = g_I(h_i^b)$. The output of the cluster-level network is the cluster assignments of samples to different clusters, i.e. $c_i^a = g_C(h_i^a)$ and $c_i^b = g_C(h_i^b)$. We call the output of the cluster-level network c-space. We first initialize our networks, i.e. $f(.)$, $g_I(.)$, and $g_C(.)$ using the CC algorithm.

If we do such initialization process for a sufficient number of epochs, a partially reliable z-space will be obtained. However, the z-space obtained by minimizing the CC loss is sub-optimal for clustering, which results in a large false-negative-pair rate and a low true-positive-pair rate. To mitigate this issue, our method incorporates cross-sample similarities.

Since our framework is unsupervised and we do not have access to the data labels to identify samples belonging to the same cluster, we use a notion of self-supervision to refine clusters. We employ the cross-sample similarities in the partially trained z-space to realize the self-supervision concept. The cross-sample similarity is measured by the cosine distance of samples in the z-space. If, for a pair of instances, the similarity is greater than or equal to a threshold ζ, we consider those samples to be similar and pull them closer together by minimizing the loss function $\mathcal{L}_{C3}$ defined below:

$$\mathcal{L}_{C3} = \frac{1}{2N} \sum_{i=1}^{N} (\tilde{\ell}_i^a + \tilde{\ell}_i^b) \tag{1}$$

$$\tilde{\ell}_i^a = -\log \frac{\sum_{k\in\{a,b\}} \sum_{j=1}^{N} \mathbb{1}\{z_i^{a\intercal} z_j^k \geq \zeta\} \exp(z_i^{a\intercal} z_j^k)}{\sum_{k\in\{a,b\}} \sum_{j=1}^{N} w_{ij}^k \exp(z_i^{a\intercal} z_j^k)}, \tag{2}$$

where $\mathbb{1}\{.\}$ denotes the indicator function. Analogous to $\tilde{\ell}_i^a$, we define $\tilde{\ell}_i^b$ that considers similarity of z_i^b and other samples in the batch. Furthermore, in the denominator of the proposed loss function, we included w_{ij}^k to consider higher weights for the samples that are neither close together nor far from each other. In this way, we realize the goal of decreasing the false negative pair selection rate as, in the traditional CC-based methods, all augmented samples in the batch are equally considered when forming negative pairs, regardless of the possibility of them belonging to the same cluster and the difficulty of the samples to cluster.

We assume that the weight terms are given when minimizing the C3 loss defined in (2). To obtain an optimum value for a weight term w_{ij}^k, we propose solving the below optimization problem while the networks are frozen. The first term of the below optimization problem is defined based on our motivation to assign a very low weight to too close and too far away samples and, instead, let the remaining samples, which are more probably located on the cluster boundaries, take higher weights. The second term is to avoid the trivial solution of assigning a weight equal to one to the sample providing the maximum value of $1 - |z_i^{a\intercal} z_j^k|$. To avoid instability, we include the constraint by which the summation of all weights must be equal to 1.

$$\min_{w_{ij}^k} \sum_{k\in\{a,b\}} \sum_{j=1}^{N} -w_{ij}^k(1-|z_i^{a\intercal} z_j^k|) - \frac{1}{\Gamma} \mathrm{H}(W_i) \qquad s.t. \sum_{k\in\{a,b\}} \sum_{j=1}^{N} w_{ij}^k = 1 \tag{3}$$

In the above equation, Γ is a hyperparameter, $W_i = \{w_{ij}^k | j \in \{1,2,..,N\}, k \in (a,b)\}$ is the set of all weights, and $\mathrm{H}(.)$ is the entropy function. We solve the optimization problem defined in (3) using the Lagrange multiplier technique as below:

$$L = \sum_{k\in\{a,b\}} \sum_{j=1}^{N} -w_{ij}^k(1-|z_i^{a\intercal} z_j^k|) + \frac{1}{\Gamma} \sum_{k\in\{a,b\}} \sum_{j=1}^{N} w_{ij}^k \log(w_{ij}^k) + \lambda\Big(\sum_{k\in\{a,b\}} \sum_{j=1}^{N} w_{ij}^k - 1\Big) \tag{4}$$

$$\frac{\partial L}{\partial w_{ij}^k} = 0 \rightarrow -(1-|z_i^{a\intercal} z_j^k|) + \frac{1}{\Gamma}\log(w_{ij}^k) + \frac{1}{\Gamma} + \lambda = 0$$

$$w_{ij}^k = \exp(\Gamma(1-|z_i^{a\intercal} z_j^k|) - 1 - \Gamma\lambda) \tag{5}$$

Where λ is the Lagrange multiplier. By substituting (5) into the constraint of (3), we have:

$$\sum_{k\in\{a,b\}} \sum_{j=1}^{N} w_{ij}^k = 1 \rightarrow \sum_{k\in\{a,b\}} \sum_{j=1}^{N} \exp(\Gamma(1-|z_i^{a\intercal} z_j^k|) - 1 - \Gamma\lambda) = 1$$

$$\exp(-1-\Gamma\lambda) = \frac{1}{\sum_{k\in\{a,b\}} \sum_{j=1}^{N} \exp(\Gamma(1-|z_i^{a\intercal} z_j^k|))} \tag{6}$$

If we substitute (6) to (5), we have the final values for w_{ij}^k as below:

$$w_{ij}^k = \frac{\exp(\Gamma(1-|z_i^{a\intercal} z_j^k|))}{\sum_{k\in\{a,b\}} \sum_{j=1}^{N} \exp(\Gamma(1-|z_i^{a\intercal} z_j^k|))} \tag{7}$$

When comparing the loss function of other contrastive clustering algorithms, such as CC, with the loss of C3 (shown in Eq. (2)), several differences become apparent. Firstly, while other contrastive algorithms typically only allow one positive pair to appear in the numerator, C3 enables the networks to be trained by considering a much larger number of positive pairs. This leads to more efficient training and better performance. Secondly, C3 adopts a weighting scheme when creating the negative pairs; the scheme assigns lower weights to samples that are either too close or too far from each other while assigning higher weights to those that are more in the mixing cluster areas. This approach reduces the impact of false positive, noisy, and anomaly samples on the learning of representations.

4 Experiments and Discussions

In this section, we demonstrate the effectiveness of our proposed scheme by conducting rigorous experimental evaluations. We evaluated our method on five challenging computer vision benchmark datasets: CIFAR-10, CIFAR-100 [19], ImageNet-10, ImageNet-Dog [2], and Tiny-ImageNet [21]. For CIFAR-10 and CIFAR-100, we combined the training and test splits. Also, for CIFAR-100, instead of 100 classes, we used the 20 super-classes as the ground-truth. To evaluate the performance, we use three commonly-used metrics in clustering namely clustering accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI) [43].

For the sake of fair comparison, for all datasets, we used ResNet34 [13] as the backbone of our encoder $f(.)$, which is the same architecture that previous algorithms have adopted. We set the dimension of the output of the instance-level projection head g_I to 128, for all datasets. The output dimension of the cluster-level contrastive head is set to the number of

Table 1: Clustering performance of different methods.

Algorithm	CIFAR-10			CIFAR-100			ImageNet-10			ImageNet-Dogs			Tiny-ImageNet		
	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI
K-means [24]	0.087	0.229	0.049	0.084	0.130	0.028	0.119	0.241	0.057	0.055	0.105	0.020	0.065	0.025	0.005
SC [41]	0.103	0.247	0.085	0.090	0.136	0.022	0.151	0.274	0.076	0.038	0.111	0.013	0.063	0.022	0.004
AC [8]	0.105	0.228	0.065	0.098	0.138	0.034	0.138	0.242	0.067	0.037	0.139	0.021	0.069	0.027	0.005
NMF [8]	0.081	0.190	0.034	0.079	0.118	0.026	0.132	0.230	0.065	0.044	0.118	0.016	0.072	0.029	0.005
AE [8]	0.239	0.314	0.169	0.100	0.165	0.048	0.210	0.317	0.152	0.104	0.185	0.073	0.131	0.041	0.007
DAE [32]	0.251	0.297	0.163	0.111	0.151	0.046	0.206	0.304	0.138	0.104	0.190	0.078	0.127	0.039	0.007
DCGAN [27]	0.265	0.315	0.176	0.120	0.151	0.045	0.225	0.346	0.157	0.121	0.174	0.078	0.135	0.041	0.007
DeCNN [40]	0.240	0.282	0.174	0.092	0.133	0.038	0.186	0.313	0.142	0.098	0.175	0.073	0.111	0.035	0.006
VAE [13]	0.254	0.291	0.167	0.108	0.152	0.040	0.193	0.334	0.168	0.107	0.179	0.079	0.113	0.036	0.006
JULE [39]	0.192	0.272	0.138	0.103	0.137	0.033	0.175	0.300	0.138	0.054	0.138	0.028	0.102	0.033	0.006
DEC [34]	0.275	0.301	0.161	0.136	0.185	0.050	0.282	0.381	0.203	0.122	0.195	0.079	0.115	0.037	0.007
DAC [3]	0.396	0.522	0.306	0.185	0.238	0.088	0.394	0.527	0.302	0.219	0.275	0.111	0.190	0.066	0.017
ADC [12]	-	0.325	-	-	0.160	-	-	-	-	-	-	-	-	-	-
DDC [8]	0.424	0.524	0.329	-	-	-	0.433	0.577	0.345	-	-	-	-	-	-
DCCM [35]	0.496	0.623	0.408	0.285	0.327	0.173	0.608	0.710	0.555	0.321	0.038	0.182	0.224	0.108	0.038
IIC [17]	-	0.617	-	-	0.257	-	-	-	-	-	-	-	-	-	-
PICA [15]	0.591	0.696	0.512	0.310	0.337	0.171	0.802	0.870	0.761	0.352	0.352	0.201	0.277	0.098	0.040
GATCluster [25]	0.475	0.610	0.402	0.215	0.281	0.116	0.609	0.762	0.572	0.322	0.333	0.200	-	-	-
CC [22]	0.678*	0.770*	0.607*	0.421*	0.423*	0.261*	0.850*	0.893*	0.811*	0.436*	0.421*	0.268*	0.331*	0.137*	0.062*
EDESC [8]	0.627	0.464	-	0.385	0.370	-	-	-	-	-	-	-	-	-	-
C3 (Ours)	**0.743**	**0.836**	**0.703**	**0.435**	**0.456**	**0.274**	**0.905**	**0.943**	**0.860**	**0.447**	**0.434**	**0.280**	**0.335**	**0.140**	**0.064**

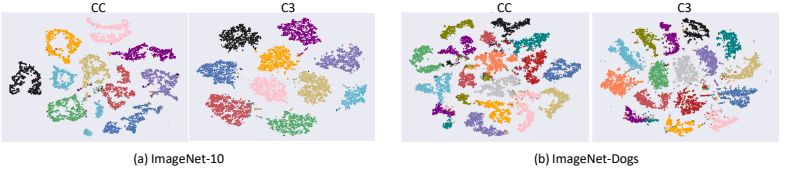

Figure 3: t-SNE visualization of clusters learned by the CC and C3 methods.

classes, M, in each dataset. All networks are initialized by the CC [22] algorithm with the hyperparameters suggested by its authors. The Adam optimizer with an initial learning rate of 0.00001 and batch size of 128 is used for C3. All networks are trained for 20 epochs. The experiments are run on NVIDIA TESLA V100 32G GPU.

4.1 Comparison with State-of-the-art

Table 1 shows the results of our proposed method on benchmark datasets, compared to state-of-the-art and some common traditional clustering methods. For CC, we run the code provided by its authors for all datasets, and the results are indicated by (*). As is evident in this table, our proposed method significantly outperforms all other baselines in all datasets. Quantitatively, comparing to the second best algorithm (i.e. CC), C3 improves the ACC by 6.6%, 3.3%, 5.0%, 1.3% and 0.3%), the NMI by 6.5%, 1.4%, 5.5%, 1.1% and 0.4%, and the ARI by 9.6%, 1.3%, 4.9%, 1.2% and 0.2%, respectively on CIFAR-10, CIFAR-100, ImageNet-10, ImageNet-Dogs, and Tiny-ImageNet. The main difference between our framework and other baselines, such as CC, is that we exploit an additional set of informa-

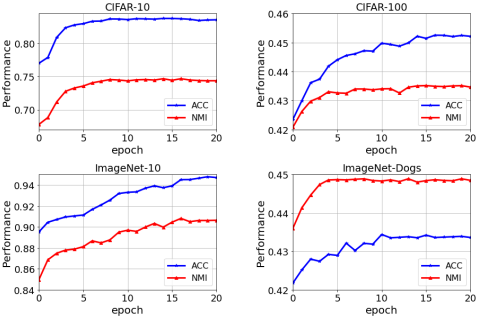

Figure 4: C3's performance vs epochs.

tion, i.e. the similarity between samples, to further enhance the learned representation for clustering. In our approach, we increase the number of positive samples, which improves the true-positive-pair rate, and put more weight on clustering the samples that are more probably located on the boundary of the clusters, which leads to a decrease in the false-negative-pair selection rate. We believe this is the main reason our method's performance is superior compared to the baselines.

As opposed to the CC method that misses the global patterns present in each data cluster, C3 correctly tries to consider such patterns (at least partially) by employing cross-instance data similarities. Looking at (2) and (3), one can infer that C3 implicitly reduces the intra-cluster distance while maximizing the inter-cluster distance, which is what an efficient grouping technique would do in the presence of the data labels. This can be confirmed by visualizing the clusters before and after applying the C3 loss. As Figures 1 and 3 show, the samples are fairly clustered after initialization with CC, i.e. before the start of training using the C3 loss. However, some of the difficult clusters are mixed in the boundaries. After initialization, clusters are expanded with a considerable number of miss-clustered data. However, after training with the proposed C3 method, we observe that the new cluster space is much more reliable, and individual clusters are densely populated while being distant from each other.

4.2 Convergence Analysis

Results of section 4.1 depict the superiority of our proposed scheme. Now, we analyze C3's convergence and the computational complexity to evaluate at what cost it makes such an improvement over other baselines. We plotted the trend of clustering accuracy and NMI for four datasets during the training epochs in Figure 4. We can readily confirm that although we are just training the C3 step for 20 epochs, the graphs quickly converge to a settling point, which corresponds to the peak performance. Also, we can observe that both ACC and NMI are improved throughout the C3 training phase in all datasets. The performance at epoch $= 0$ corresponds to the clustering performance after initialization with CC. These figures clearly show that C3 improves its clustering quality and justifies the qualitative results shown in Section 4.1.

One may naively think that the better performance of C3 is because it is being trained for 20 more epochs; note that in all our experiments, as suggested by the CC authors, we trained the CC algorithm networks for 1000 epochs. We train the C3 networks for 1020 epochs. We reject this argument and support it by training the CC networks for the same number of epochs as what the C3 is trained for, i.e. 1020 epochs. We observe that no improvement is obtained for CC when trained for an extra 20 epochs. The result of such an experiment on CIFAR-10 is shown in Figure 5.

4.3 How does C3 loss improve the clusters?

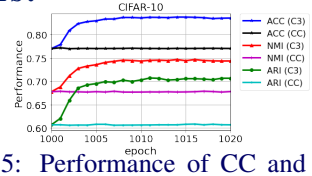

Figure 5: Performance of CC and C3 after 1020 epochs.

As we saw in Figure 3, the improvement that C3 achieves is mainly because it is able to reduce the distance between instances of the same cluster while repelling them from other clusters. We can justify this observation by considering the loss function of C3, i.e. Eq. (2). In this function, the term $\mathbb{1}\{z_i^{a\intercal} z_j^k \geq \zeta\}$ indicates that if the cosine similarity of two samples is greater than ζ, they should be moved further close to each other.

At the beginning of training with the C3 loss, since the networks are initiated with CC, we can assume that the points that are very similar to each other have a cosine distance, in the z-space, less than threshold ζ, so they will become further close by minimizing the C3 loss. For instance, take two points z_1 and z_2 with a cosine similarity larger than ζ, and assume that z_3 has a similarity greater than ζ with z_1, but its similarity with z_2 is smaller than ζ. Therefore, according to the loss function, z_1 and z_2, as well as z_1 and z_3, are forced to become closer, but it is not the case for z_2 and z_3. However, these two points will also implicitly move closer to each other because their distance to z_1 is reduced. As the training continues, at some point, the similarity of z_2 and z_3 also may pass the threshold ζ. Therefore, as the similar pairs move closer to each other during the training, a series of new connections will be formed, and the cluster will become denser. To support this hypothesis, we plotted the average of the loss function and the average number of positive pairs of each data sample in Figure 6-a and 6-b, respectively. We can observe that the number of positive pairs exponentially increases during the training until it settles to form the final clusters. Corresponding to this exponential increase, we can see that the loss is decreasing, and the network learns a representation in which clusters are distanced from each other while samples of each cluster are packed together.

We can also deduct from this experiment that the number of positive pairs is also related to the number of classes in each dataset. For example, if we have an augmented batch size of $N = 256$, for Tiny-ImageNet that has 200 classes, we expect to have $\frac{256}{200} = 1.28$ positive pairs per sample which is very close to 1 and it is the reason that we do not see the same sharp increasing trend as other datasets in Tiny-ImageNet.

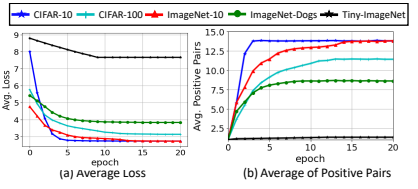

Figure 6: Plot of loss and number of positive pairs versus epoch.

4.4 Effect of Hyperparameter ζ and Γ

Our method, C3, introduces two new hyperparameter ζ and Γ. ζ is a threshold for identifying similar samples. Throughout the experiments, we fixed $\zeta = 0.6$, which yielded consistent results across datasets. Now, we carry out an experiment in which we change ζ and record the performance. Note that since $z_i^{a\intercal} z_j^k \in [-1, 1]$, we can technically change ζ from -1 to 1. Intuitively, for a small or negative value of ζ, most points in the z-space will be considered similar, and the resulting clusters will not be reliable. Therefore, in our experiment, we change ζ from 0.4 to 0.9 in 0.1 increments for CIFAR-10. For Tiny-ImageNet, as there are lots of clusters, we set $\zeta \in \{0.60, 0.85, 0.90, 0.95\}$. We then plot the accuracy, NMI, average loss, and the average of positive pairs per sample. The graphics are shown in Figure 7.

In CIFAR-10 experiments, in Figure 7-a and Figure 7-b, we see that for $\zeta = 0.4$, accuracy and NMI are indeed decreasing during the C3 training. This is because this value of ζ is too lenient and considers the points not in the same cluster to be similar. We can confirm this explanation by looking at Figure 7-d. We can see that for smaller ζs, we will have more average positive pairs per sample. As we increase the ζ, we can see that the performance begins to improve. For larger values such as $\zeta = 0.9$, we can see that the performance does not significantly change during the training. This is because $\zeta = 0.9$ is a strict threshold, and if we look at the number of positive pairs, only a few instances are identified as similar during the training.

Comparing the results of CIFAR-10 and Tiny-ImageNet experiments shows that the value

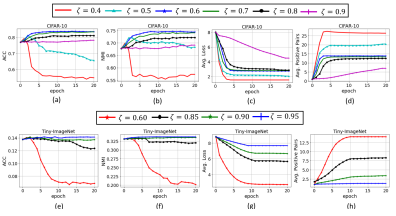

Figure 7: Performance and behavior of C3 for different values of ζ, for CIFAR 10 (top row) and Tiny-ImageNet (bottom row).

of ζ also depends on the number of clusters. Since we have 200 classes in Tiny-ImageNet, a smaller value of ζ might yield two or more clusters merging together and this would decrease the accuracy. Therefore, we should choose a more strict threshold such as $\zeta = 0.9$ or $\zeta = 0.95$ to improve. In Figure 7-c and Figure 7-g, the average loss plot also conveys interesting observations about the behaviour of ζ. We can see that for smaller values, the loss is exponentially converging to the minimum, but for larger ζ, the rate is much slower. This can be due to the fact that a smaller ζ considers most points to be similar and of the same class, and therefore, it can yield the trivial solution of considering all points to be similar and mapping them into one central point. In the extreme case of $\zeta \to 1$, C3 considers a few samples as positive pairs, and therefore, we will not have any major improvement. In contrast, if we set $\zeta \to -1$, the loss considers all points to be positive and the numerator and denominator of Eq. (2) become equal. Therefore, the loss function becomes zero and the network does not train. Following the above discussion, we suggest a value like $\zeta = 0.6$, which is a good balance. However, the choice of ζ might be influenced by the number of clusters in the data. If we have a large number of clusters, it would be better to choose a large ζ. On the other hand, if the data has a small number of clusters, a smaller ζ (but not too small) is preferred since it trains faster. In our experiments, we set $\zeta = 0.6$ unless in Tiny-ImageNet which has 200 classes where we used $\zeta = 0.95$.

Figure 8 illustrates the impact of hyper-parameter Γ on the performance C3. For very low values of Γ (i.e., $\Gamma \to 0$), all weights converge to the same value of $w_{ij}^k = \frac{1}{2N}$. Conversely, for very high values of Γ (i.e., $\Gamma \to \infty$), the effect of entropy in Eq. (3) is neglected, leading to a trivial solution where our weighting function in Eq. (7) selects one negative sample having the highest value of $1 - |z_i^{a\intercal} z_j^k|$ to minimize the first term in Eq. (3). In all our experiments for all the datasets, Γ is set to 0.1 though a better performance may be obtained if we fine-tune it per dataset. Overall, our results demonstrate the importance of selecting an appropriate value for Γ to optimize the performance of our proposed method.

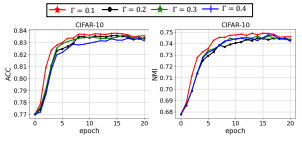

Figure 8: Performance of C3 for different values of Γ, for CIFAR 10.

5 Conclusion

In this paper, we proposed C3, an algorithm for contrastive data clustering that incorporates the similarity between different instances to form a better representation for clustering. We experimentally showed that our method could significantly outperform the state-of-the-art on five challenging computer vision datasets. In addition, through additional experiments, we evaluated different aspects of our algorithm and provided several intuitions on how and why our proposed scheme can help in learning a more cluster-friendly representation.

References

[1] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper/2006/file/5da713a690c067105aeb2fae32403405-Paper.pdf.

[2] Deng Cai, Xiaofei He, Xuanhui Wang, Hujun Bao, and Jiawei Han. Locality preserving nonnegative matrix factorization. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, page 1010–1015, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[3] Jinyu Cai, Jicong Fan, Wenzhong Guo, Shiping Wang, Yunhe Zhang, and Zhao Zhang. Efficient deep embedded subspace clustering. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21–30, 2022. doi: 10.1109/CVPR52688.2022.00012.

[4] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5880–5888, 2017. doi: 10.1109/ICCV.2017.626.

[5] Jianlong Chang, Yiwen Guo, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep discriminative clustering analysis. *ArXiv*, abs/1905.01681, 2019.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet full (fall 2011 release).

[8] Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning by rotation feature decoupling. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10356–10366, 2019. doi: 10.1109/CVPR.2019.01061.

[9] K. Chidananda Gowda and G. Krishna. Agglomerative clustering using the concept of mutual nearest neighbourhood. *Pattern Recognit.*, 10:105–112, 1978.

[10] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1753–1759. AAAI Press, 2017. ISBN 9780999241103.

[11] Xifeng Guo, En Zhu, Xinwang Liu, and Jianping Yin. Deep embedded clustering with data augmentation. In Jun Zhu and Ichiro Takeuchi, editors, *Proceedings of The 10th Asian Conference on Machine Learning*, volume 95 of *Proceedings of Machine Learning Research*, pages 550–565. PMLR, 14–16 Nov 2018. URL https://proceedings.mlr.press/v95/guo18b.html.

[12] Philip Haeusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, and Daniel Cremers. Associative deep clustering: Training a classification network with no labels. In Thomas Brox, Andrés Bruhn, and Mario Fritz, editors, *Pattern Recognition*, pages 18–32, Cham, 2019. Springer International Publishing. ISBN 978-3-030-12939-2.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Hadi Hojjati, Thi Kieu Khanh Ho, and Narges Armanfard. Self-supervised anomaly detection: A survey and outlook. *ArXiv*, abs/2205.05173, 2022.

[15] Jiabo Huang, Shaogang Gong, and Xiatian Zhu. Deep semantic clustering by partition confidence maximisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[16] Peihao Huang, Yan Huang, Wei Wang, and Liang Wang. Deep embedding network for clustering. In *2014 22nd International Conference on Pattern Recognition*, pages 1532–1537, 2014. doi: 10.1109/ICPR.2014.272.

[17] 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.

[18] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[19] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

[20] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22 (1):79–86, 1951.

[21] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge, 2015.

[22] Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8547–8555, May 2021. doi: 10.1609/aaai.v35i10.17037. URL https://ojs.aaai.org/index.php/AAAI/article/view/17037.

[23] Z Lin, S Qi, S Zhengyang, and W Changhu. Inter-intra variant dual representations for self-supervised video recognition. In *British Machine Vision Conference*, volume 2, page 7, 2021.

[24] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136, 1982.

[25] Chuang Niu, Jun Zhang, Ge Wang, and Jimin Liang. Gatcluster: Self-supervised gaussian-attention network for image clustering. In *European Conference on Computer Vision (ECCV)*, 2020.

[26] Xi Peng, Shijie Xiao, Jiashi Feng, Wei-Yun Yau, and Zhang Yi. Deep subspace clustering with sparsity prior. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1925–1931. AAAI Press, 2016. ISBN 9781577357704.

[27] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2016.

[28] Mohammadreza Sadeghi and Narges Armanfard. Deep Clustering with Self-supervision using Pairwise Data Similarities. *TechRxiv*, 6 2021. doi: 10.36227/techrxiv.14852652.v1. URL https://www.techrxiv.org/articles/preprint/Deep_Clustering_with_Self-supervision_using_Pairwise_Data_Similarities/14852652.

[29] Mohammadreza Sadeghi and Narges Armanfard. Idecf: Improved deep embedding clustering with deep fuzzy supervision. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1009–1013, 2021. doi: 10.1109/ICIP42928.2021.9506051.

[30] Mohammadreza Sadeghi and Narges Armanfard. Deep Multi-Representation Learning for Data Clustering. *TechRxiv*, 3 2022. doi: 10.36227/techrxiv.19357979.v1. URL https://www.techrxiv.org/articles/preprint/Deep_Multi-Representation_Learning_for_Data_Clustering/19357979.

[31] Guy Shiran and Daphna Weinshall. Multi-modal deep clustering: Unsupervised partitioning of images. In *International Conference on Pattern Recognition (ICPR)*, 2020.

[32] Hitika Tiwari, Min-Hung Chen, Yi-Min Tsai, Hsien-Kai Kuo, Hung-Jen Chen, Kevin Jou, KS Venkatesh, and Yong-Sheng Chen. Self-supervised robustifying guidance for monocular 3d face reconstruction. 2021.

[33] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *Proceedings of the European Conference on Computer Vision*, 2020.

[34] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(110):3371–3408, 2010. URL http://jmlr.org/papers/v11/vincent10a.html.

[35] Jianlong Wu, Keyu Long, Fei Wang, Chen Qian, Cheng Li, Zhouchen Lin, and Hongbin Zha. Deep comprehensive correlation mining for image clustering. In *International Conference on Computer Vision*, 2019.

[36] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8372–8381, 2021. doi: 10.1109/ICCV48922.2021.00828.

[37] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 478–487. JMLR.org, 2016.

[38] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3861–3870. JMLR.org, 2017.

[39] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5147–5156, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society. doi: 10.1109/CVPR.2016.556. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.556.

[40] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, 2010. doi: 10.1109/CVPR.2010.5539957.

[41] Lihi Zelnik-manor and Pietro Perona. Self-tuning spectral clustering. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004. URL https://proceedings.neurips.cc/paper/2004/file/40173ea48d9567f1f393b20c855bb40b-Paper.pdf.

[42] Huasong Zhong, C. Chen, Zhongming Jin, and Xiansheng Hua. Deep robust clustering by contrastive learning. *ArXiv*, abs/2008.03030, 2020.

[43] Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Zhao Li, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, and Martin Ester. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *ArXiv*, abs/2206.07579, 2022.