CONSENSUS CLUSTERING WITH UNSUPERVISED REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in deep clustering and unsupervised representation learning are based on the idea that different views of an input image (generated through data augmentation techniques) must either be closer in the representation space, or have a similar cluster assignment. In this work, we leverage this idea together with ensemble learning to perform clustering and representation learning. Ensemble learning is widely used in the supervised learning setting but has not yet been practical in deep clustering. Previous works on ensemble learning for clustering neither work on the feature space nor learn features. We propose a novel ensemble learning algorithm dubbed Consensus Clustering with Unsupervised Representation Learning (ConCURL) which learns representations by creating a consensus on multiple clustering outputs. Specifically, we generate a cluster ensemble using random transformations on the embedding space, and define a consensus loss function that measures the disagreement among the constituents of the ensemble. Thus, diverse ensembles minimize this loss function in a synergistic way, which leads to better representations that work with all cluster ensemble constituents. Our proposed method ConCURL is easy to implement and integrate into any representation learning or deep clustering block. ConCURL outperforms all state of the art methods on various computer vision datasets. Specifically, we beat the closest state of the art method by 5.9 percent on the ImageNet-10 dataset, and by 18 percent on the ImageNet-Dogs dataset in terms of clustering accuracy. We further shed some light on the under-studied overfitting issue in clustering and show that our method does not overfit as much as existing methods, and thereby generalizes better for new data samples.

1 INTRODUCTION

Supervised learning algorithms have shown great progress recently, but generally require a lot of labeled data. However, in many domains (e.g., advertising, social platforms, etc.), most of the available data are not labeled and manually labeling it is a very labor, time, and cost intensive task (Xiao et al., 2015; Deshmukh, 2019; Mintz et al., 2009; Blum & Mitchell, 1998). On the other hand, clustering algorithms do not need labeled data to group similar data points into clusters. Some popular clustering algorithms include k-means, hierarchical clustering, DBSCAN (Ester et al., 1996), spectral clustering, etc., and the usefulness of each algorithm varies with the application. In this work, we deal with the clustering of images. Traditional clustering approaches focus on hand crafted features on which out of the box clustering algorithms are applied. However, hand crafted features are not optimal, and are not scalable to large scale real word datasets (Wu et al., 2019). Advancements in deep learning techniques have enabled learning rich representations end-to-end in the supervised setting, but combining clustering algorithms with representation learning methods is not straight forward. Since clustering methods are developed to work with linear models on top of fixed features, simultaneously learning the feature spaces while clustering leads to degenerate solutions, which until recently limited end to end implementations of clustering approaches (Caron et al., 2018). Recent deep clustering works take several approaches to address this issue such as alternating pseudo cluster assignments and pseudo supervised training, comparing the predictions with their own high confidence assignments (Caron et al., 2018; Asano et al., 2019; Xie et al., 2016; Wu et al., 2019), and maximizing mutual information between predictions of positive pairs (Ji et al., 2019). None of these works take advantage of rich ideas in ensemble learning for clustering with

representation learning and improving performance with respect to various evaluation metrics. In this work, we propose an algorithm that performs clustering tasks by leveraging the strength of an ensemble of clusterings.

In ensemble methods for supervised learning, weak learners are trained on labelled data and then combined, resulting in a strong learner that is better than its constituent learners (Freund et al., 1996; Breiman, 1996). The idea of ensembles was extended to clustering by Strehl & Ghosh (2002) and dubbed as knowledge reuse framework. A consensus algorithm is applied on constituent cluster partitions to generate an updated partition that clusters the data better than any component partition individually. However, the knowledge reuse framework and much of the consensus clustering literature that followed (Fern & Brodley, 2003; Fred & Jain, 2005; Topchy et al., 2005) do not make use of the underlying features used to generate the ensemble. We propose the use of consensus clustering as a way to extend ensemble methods to unsupervised representation learning. In particular, we define a 'disagreement' measure among the constituents of the ensemble. The key motivation for this is that the diversity of the ensemble drives the minimization of the disagreement measure in a synergistic way, thereby leading to better representations. We propose Consensus Clustering with Unsupervised Representation Learning (ConCURL) and following are our main contributions:

1. A novel ensemble learning algorithm which learns representations by creating a consensus on multiple clustering outputs generated by applying random transformations on the embeddings.
2. Our method outperforms the current state of the art clustering algorithms on popular computer vision datasets. We use cluster accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI) as proxies for evaluating the quality of clustering.
3. Even though there is no labeled data available while learning representations, clustering may still be prone to be overfitting to the "training data." As stated in Bubeck & Von Luxburg (2007), in clustering, we generally assume that the finite data set has been sampled from some underlying space and the goal is to find the true approximate partition of the underlying space rather than the best partition in a given finite data set. Hence, to check generalizability of the method proposed we also evaluate our models on the "test data" - data that was not available during training/representation learning. Our method is more generalizable compared to state of the art methods (i.e. it outperforms the other algorithms when evaluated on the test set).

2 Related work

Clustering is a ubiquitous task and it has been actively used in many different scientific and practical pursuits such as detecting genes from microarray data (Frey & Dueck, 2007), clustering faces (Rodriguez & Laio, 2014), and segmentation in medical imaging to support diagnosis (Masulli & Schenone, 1999). We refer interested readers to these excellent sources for a survey of these uses Jain et al. (1999); Liao (2005); Xu & Wunsch (2005); Nugent & Meila (2010).

Clustering with Deep Learning: In their influential work, Caron et al. (2018) show that it is possible to train deep convolutional neural networks with pseudo labels that are generated by a clustering algorithm (DeepCluster). More precisely, in DeepCluster, previous versions of representations are used to assign pseudo labels to the data using an out of the box clustering algorithm such as k-means. These pseudo labels are used to improve the learned representation of the data by minimizing a supervised loss. Along the same lines, several more methods have been proposed. For example, Gaussian ATtention network for image clustering (GATCluster) (Niu et al., 2020) comprises four self-learning tasks with the constraints of transformation invariance, separability maximization, entropy analysis and attention mapping. Training is performed in two distinct steps, similar to Caron et al. (2018) where the first step is to compute pseudo targets for a large batch of data and the second step is to train the model in a supervised way using the pseudo targets. Both DeepCluster and GATCluster use k-means to generate pseudo labels which may not scale well. Wu et al. (2019) propose Deep Comprehensive Correlation Mining (DCCM), where discriminative features are learned by taking advantage of the correlations of the data using pseudo-label supervision and triplet mutual information among features. However, DCCM may be susceptible to trivial solutions (Niu et al., 2020). Invariant Information Clustering (IIC) (Ji et al., 2019) maximizes mutual information between the class assignments of two different views of the same image (paired samples) in order to

learn representations that preserve what is common between the views while discarding instance specific details. Ji et al. (2019) argue that the presence of an entropy term in mutual information plays an important role in avoiding the degenerate solutions. However a large batch size is needed for the computation of mutual information in IIC, which may not be scalable for larger image sizes which are common in popular datasets (Ji et al., 2019; Niu et al., 2020). Huang et al. (2020) extend the traditional maximal margin clustering idea to the deep learning paradigm, by learning the most semantically plausible clustering through minimizing a proposed partition uncertainty index. Their algorithm PICA uses a stochastic version of the index, thereby facilitating mini-batch training. PICA fails to assign a sample-correct cluster when that sample either has high foreground or background similarity to samples in other clusters. Our proposed approach ConCURL is scalable to large datasets, does not suffer from trivial solutions and shows superior performance on a challenging set of image data sets. As shown in the experimental results, our proposed method also generalizes well to data points that were not available during training, when compared to the above approaches.

Self-supervised Representation Learning: Self-supervised learning is a sub-field of unsupervised learning in which the main goal is to learn general purpose representations by exploiting user-defined sub-tasks such as the relationship between different views of the same data. Although self-supervised learning methods show impressive performance on a variety of problems, it is not clear whether learned representations are good for clustering. There are many different flavors of self supervised learning; among these, contrastive learning-based methods comprise a major class of techniques. In contrastive learning (Tian et al., 2019; He et al., 2020; Hénaff et al., 2019; Hjelm et al., 2018; Chen et al., 2020), representations are learned by maximizing agreement between different augmented views of the same data example (known as positive pairs) and minimizing agreement between augmented views of different examples (known as negative pairs). Simple framework for Contrastive Learning of visual Representation (SimCLR) (Chen et al., 2020) achieves state of the art results without specialized architectures or a memory bank of negative pairs (which are usually required by contrastive learning techniques). The major disadvantage of the method proposed by Chen et al. (2020) is that it requires negative examples and, as it applies to instance classification, it has to compare every pair of images. These issues are addressed by Bootstrap your own latent (BYOL) (Grill et al., 2020) and Swapping Assignments between multiple Views (SwAV) (Caron et al., 2020). BYOL does not require negative examples and SwAV does not need to compare every pair of images. BYOL is motivated from the observation that it is possible to bootstrap available representations to learn potentially enhanced representations by predicting the target representations. In particular, BYOL has an online network and a target network. Given two views of the same image, the loss function is the normalized distance between the online representation of view 1 and the target representation of view 2, and vice versa. The online network is updated by the gradients, but the target network is a moving average of the weights of the online network. It is not clear if any of the above methods work for clustering as they work on linear evaluation of downstream tasks. For the main study in this paper we use BYOL as a backbone and adapt the soft clustering loss which was used in SwAV to learn prototypes, thereby addressing both the issues of negative samples and need for comparing every pair of images. Note that our proposed algorithm can use any backbone for representation learning like SimCLR, BYOL, & SwAV and can also use other soft clustering loss formulations.

Consensus Clustering: Analogous to ensemble methods in supervised learning, Strehl & Ghosh (2002) propose a framework for extending ensemble learning to clustering. Strehl & Ghosh (2002) focus on building an unsupervised ensemble by using several distinct clusterings of the same data and assuming that the underlying features that are used to compute the clustering are not available and fixed. Fern & Brodley (2003) build on the cluster ensemble framework based on random projections, and is the most relevant to our work. In this framework, Fern & Brodley (2003) show that a single run of clustering (random projection + Expectation Maximization) is highly unstable. Each run of clustering after EM results in a similarity matrix where each entry corresponds to the probability of two points belonging to the same cluster. They perform multiple runs of clustering and compute an aggregated similarity matrix which is used to cluster the data using an agglomerative clustering algorithm. Fred & Jain (2005) propose a voting approach to map the data partitions, i.e. cluster assignments, in the ensemble into a new similarity measure between clusterings induced by inter-pattern relationships present in the ensemble. The co-association matrix thus formed can be used with any clustering algorithm to result in a new data partition. It is not clear how any of these methods can be adapted when one needs to do representation learning along with clustering. It is also not clear if one can come up with end-to-end learning architecture with any of the above

methods. In contrast, our proposed consensus clustering method ConCURL can easily be integrated into any deep learning architecture for clustering and trained end-to-end.

3 PROPOSED METHOD

There are different ways by which one can obtain an ensemble of clusterings. Fred & Jain (2005) discuss different ways to generate an ensemble of clusterings which can be broadly classified into i) choosing a data representation, and ii) choosing a clustering algorithm. Ways of choosing a data representation include using different pre-processing techniques, subsets of features, or different transformations of the data/features. Ways of choosing a clustering algorithm include using multiple clustering algorithms, or using the same algorithm with different parameters/initializations. Any combination of these approaches results in an ensemble of clusterings. In our proposed algorithm, we focus on choosing of data representation to generate cluster ensembles. In addition to the widely-used method of generating multiple views of the image with random data augmentation techniques, we also generate multiple random transformations of the embeddings output by the network. In particular, we use random projections and diagonal transformations of the embeddings (see section 4.2). In the rest of this paper, we focus on explaining the algorithm for getting consensus on multiple data representations.

3.1 ALGORITHM

Given a set of observations $\mathcal{X} = \{x_i\}_{i=1}^{N}$, the goal is to learn a representation f, cluster representations C (called as prototypes henceforth) and partition N observations into K disjoint clusters. To check the generalization of the algorithm, the final goal is to partition N_T observations $\mathcal{X}_{\mathcal{T}} = \{x_i\}_{i=N+1}^{N+N_T}$ into K disjoint clusters given learnt representation f and prototypes C.

3.1.1 UNSUPERVISED REPRESENTATION LEARNING

We use an unsupervised representation learning block in our algorithm (for example Grill et al. (2020)). We assume that the unsupervised representation learning algorithm outputs features $\mathbf{f}$ for given data points by optimizing loss L_1. This generally results in good representations but may not be best for a clustering task. We add further loss functions to improve on clustering evaluation metrics.

3.1.2 SOFT CLUSTERING

Given an input batch of images $\mathcal{X}_b$, we use different data augmentations in the image space, such as random horizontal flip, random colorization, random Gaussian blur, etc, to generate two different views $\mathcal{X}_b^1, \mathcal{X}_b^2$ of the input batch. Performing a forward pass on the two views results in feature vectors $\mathbf{f}^1, \mathbf{f}^2$. The superscript is used to identify the view of the input image. It's usage will be made clear from context, and ignored otherwise.

We use a multi-layer perceptron g to project the features $\mathbf{f}^1$ and $\mathbf{f}^2$ to a lesser dimensional space. The MLP consists of a hidden layer of dimension 2048, followed by a $1d$ batch norm, ReLu, and an output layer of dimension 256. The output of this MLP (referred to as embeddings) is denoted using $Z^1 = \{\mathbf{z}_1^1, \ldots, \mathbf{z}_B^1\}$ and $Z^2 = \{\mathbf{z}_1^2, \ldots, \mathbf{z}_B^2\}$ for view 1 and view 2 respectively.

We store a set of randomly initialized prototypes $C = \{\mathbf{c}_1, \cdots, \mathbf{c}_K\} \in \mathbb{R}^{d \times K}$, where K is the number of clusters. Soft clustering approaches based on centroids/prototypes often requires one to compute a measure of similarity between each data point and each centroid (Xie et al., 2016; Caron et al., 2020). We compute the probability of assigning a cluster j to image i using the normalized vectors $\bar{\mathbf{z}}_i^1 = \frac{\mathbf{z}_i^1}{||\mathbf{z}_i^1||}$, $\bar{\mathbf{z}}_i^2 = \frac{\mathbf{z}_i^2}{||\mathbf{z_i}^2||}$ and $\bar{\mathbf{c}}_j = \frac{\mathbf{c_j}}{||\mathbf{c_j}||}$ as

$$\mathbf{p}_{i,j}^1 = \frac{\exp(\frac{1}{\tau}\langle \bar{\mathbf{z}}_i^1, \bar{\mathbf{c}}_j \rangle)}{\sum_{j'} \exp(\frac{1}{\tau}\langle \bar{\mathbf{z}}_i^1, \bar{\mathbf{c}}_{j'} \rangle)}, \quad \mathbf{p}_{i,j}^2 = \frac{\exp(\frac{1}{\tau}\langle \bar{\mathbf{z}}_i^2, \bar{\mathbf{c}}_j \rangle)}{\sum_{j'} \exp(\frac{1}{\tau}\langle \bar{\mathbf{z}}_i^2, \bar{\mathbf{c}}_{j'} \rangle)}. \tag{1}$$

We concisely write $\mathbf{p}_i^1 = \{\mathbf{p}_{i,j}^1\}_{j=1}^K$ and $\mathbf{p}_i^2 = \{\mathbf{p}_{i,j}^2\}_{j=1}^K$. Here, τ is a temperature parameter. Note that, we use $\mathbf{p}_i$ to denote the predicted cluster assignment probabilities for image i (when not referring to a particular view), and a shorthand $\mathbf{p}$ is used when i is clear from context.

We further use the Sinkhorn algorithm for computing codes which will be used as pseudo targets for the predicted assignments. The idea of predicting assignments $\mathbf{p}$, and then comparing them with high-confidence estimates $\mathbf{q}$ (referred to as codes henceforth) of the predictions is not new (Xie et al., 2016). While Xie et al. (2016) uses pretrained features (from autoencoder) to compute the predicted assignments and the codes, doing this in an end to end unsupervised manner might lead to degenerate solutions. Asano et al. (2019) avoids such degenerate solutions by enforcing an equipartition constraint (the prototypes equally partition the data) during code computation. Caron et al. (2018) follow the same formulation but compute the codes in an online manner for each mini-batch. The assignment codes are computed by solving the following optimization problem

$$Q^1 = \arg\max_{Q \in \mathcal{Q}} \text{Tr}(Q^T C^T Z^1) + \epsilon H(Q) \text{ and } Q^2 = \arg\max_{Q \in \mathcal{Q}} \text{Tr}(Q^T C^T Z^2) + \epsilon H(Q), \quad (2)$$

where $\mathcal{Q}$ is the transportation polytope, $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_B\}$, and $H(Q) = -\sum_{i,j} Q_{i,j} \log Q_{i,j}$. The above optimization is computed using a fast version of the Sinkhorn-Knopp algorithm (Cuturi, 2013) as described in Caron et al. (2020).

After computing the codes Q^1 and Q^2, we compute the loss using the probabilities $\mathbf{p}_{ij}$ and the assigned codes $\mathbf{q}_{ij}$ by comparing the probabilities of view 1 with the assigned codes of view 2 and vice versa, given as

$$L_{2,1} = -\frac{1}{2B} \sum_{i=1}^{B} \sum_{j=1}^{K} \mathbf{q}_{ij}^2 \log \mathbf{p}_{ij}^1, \quad L_{2,2} = -\frac{1}{2B} \sum_{i=1}^{B} \sum_{j=1}^{K} \mathbf{q}_{ij}^1 \log \mathbf{p}_{ij}^2, \quad L_2 = L_{2,1} + L_{2,2} \quad (3)$$

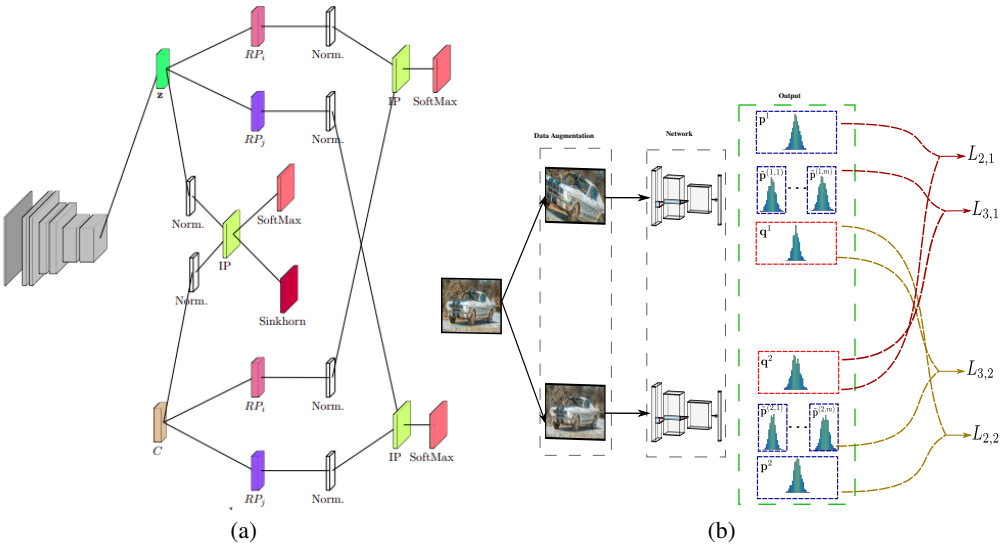

Figure 1: (a) Model architecture. ($\mathbf{z}$: Embeddings; C: Prototypes; Norm: Normalization; IP: Inner Product; RP_i: Random transformations on embeddings); The network diagram explains the computation of predicted cluster assignments $\mathbf{p}$ like in eqn. (1) and codes $\mathbf{q}$ like in eqn. (2). The RP_i blocks perform random transformations on the embeddings $\mathbf{z}$ and the prototypes C, which are used to compute $\tilde{\mathbf{p}}$ (b) The resulting ensemble of clusterings $\{\mathbf{p}^i, \mathbf{q}^i, \{\tilde{\mathbf{p}}^{(i,m)}\}_m\}_{i=\{1,2\}}$ is represented in the green block. $L_{2,1}, L_{2,2}$ represent soft clustering loss as in eqn. (3), $L_{3,1}, L_{3,2}$ represent consensus loss as in eqn. (4)

3.1.3 Consensus Clustering

We generate a cluster ensemble by first performing transformations on the embeddings Z^1, Z^2 and the prototypes C. At the beginning of the algorithm we randomly initialize M such transformations

and fix them throughout training. Suppose using a particular random transformation (a randomly generated matrix A), we get $\tilde{\mathbf{z}} = A\mathbf{z}$, $\tilde{\mathbf{c}} = A\mathbf{c}$. We then compute the softmax probabilities $\tilde{\mathbf{p}}_{ij}$ using the normalized vectors $\tilde{\mathbf{z}}/||\tilde{\mathbf{z}}||$ and $\tilde{\mathbf{c}}/||\tilde{\mathbf{c}}||$. Repeating this with the M transformations results in M predicted cluster assignment probabilities for each view. When the network is untrained, the embeddings $\mathbf{z}$ are random and applying the random transformations followed by computing the predicted cluster assignments leads to a diverse set of soft cluster assignments. The motivation is to use the diversity of this ensemble by enforcing a consensus on these cluster assignment and guide the training. This helps to learn a representation that gives a good clustering accuracy even when that representation is projected in some space using a random transformation. Creating such an ensemble leads to better representations and better clusters. To compute the consensus loss, once the probabilities $\tilde{\mathbf{p}}_{ij}$ are computed, we compare the codes generated using (2) of view 1 with the $\tilde{\mathbf{p}}$ of view 2 and vice versa, given as

$$L_{31} = -\frac{1}{2BM}\sum_{i=1}^{B}\sum_{m=1}^{M}\sum_{j=1}^{K}\mathbf{q}_{ij}^{2}\log\tilde{\mathbf{p}}_{ij}^{(1,m)}, \quad L_{32} = -\frac{1}{2BM}\sum_{i=1}^{B}\sum_{m=1}^{M}\sum_{j=1}^{K}\mathbf{q}_{ij}^{1}\log\tilde{\mathbf{p}}_{ij}^{(2,m)} \tag{4}$$

$$L_3 = L_{31} + L_{32} \tag{5}$$

The architecture for computing the ensemble and the computation of consensus loss is shown in Fig. (1a, 1b). The final loss that we sought to minimize is the combination of the losses L_1, L_2, L_3

$$L_{\text{total}} = \alpha L_1 + \beta L_2 + \gamma L_3. \tag{6}$$

During inference, to compute a clustering of the input images, we use the computed assignments $\{\mathbf{q}_i\}_{i=1}^{N}$ and assign the cluster index as $c_i = \arg\max_k \mathbf{q}_{ik}$ for the ith datapoint.

Algorithm 1: Consensus Clustering algorithm (ConCURL)

Input: Dataset $\mathcal{X} = \{x_i\}_{i=1}^{N}$, K, B, α, β, γ, M, d
Output: Cluster label c_i of $x_i \in \mathcal{X}$

Randomly initialize network parameters $\mathbf{w}$, K prototypes $\mathbf{c}_{1:K}$, M random projection matrices $R_{1:M}$to dimension d and $e = 0$;
while $e <$ *total epoch number* **do**
  for $b \in \{1, 2, \ldots, \lfloor \frac{N}{B} \rfloor\}$ **do**
    Select B samples as $\mathcal{X}_b$ from $\mathcal{X}$;
    Make a forward pass on two views of the input batch $(\mathcal{X}_b^1, \mathcal{X}_b^2)$, and obtain the features $\mathbf{z}_{1:B}^1$, $\mathbf{z}_{1:B}^2$;
    Compute loss L_1 which is the representation loss
    Compute probability of ith sample belonging to the jth cluster, $\mathbf{p}_{i,j}$ for both views separately, using normalized $\mathbf{z}$, $\mathbf{c}$ eq (1);
    Compute codes of the current batch $\mathbf{q}$ using eq (2);
    Compute loss L_2 using eq(3);
    for $m \in \{1, 2, \ldots, M\}$ **do**
      $\tilde{\mathbf{z}}, \tilde{\mathbf{c}} \longleftarrow$ Compute random projections of $\mathbf{z}$, $\mathbf{c}$;
      Compute probability of ith sample belonging to the jth cluster, $(\tilde{\mathbf{p}}_{i,j}^{(1,m)}, \tilde{\mathbf{p}}_{i,j}^{(2,m)})$ using normalized $\tilde{\mathbf{z}}$, $\tilde{\mathbf{c}}$ eq (1);
    end
    Compute loss L_3 using eq (5) ;
    Compute total loss using eq (6). Update the parameters, and prototypes using gradients
  end
  $e := e + 1$
end
Make forward pass on all the data and store the features;
foreach $x_i \in \mathcal{X}$ **do**
  Compute probability of ith sample belonging to the jth cluster, $\mathbf{p}_{i,j}$ using normalized $\mathbf{z}_i$, $\mathbf{c}_j$ eq (1);
  Compute codes $\mathbf{q}$ using eq (2);
  $c_i := \arg\max_k \mathbf{q}_{ik}$;
end

4 EXPERIMENTS

4.1 DATASETS

We evaluated our algorithm and compared against existing work on nine popular image datasets with a mix of high and medium resolution datasets: ImageNet-10 and ImageNet-Dogs (subsets of ImageNet (Deng et al., 2009)) using the same classes as Chang et al. (2017) , STL10 (Coates et al.,

2011), CIFAR-10, CIFAR100-20 (Krizhevsky et al., 2009), CUB (Wah et al., 2011), Caltech-101 (Fei-Fei et al., 2004), AwA2(Xian et al., 2018) and Intel Image classification. The dataset summary is given in Table 4. The resolution column shows the size to which we resized the images in our algorithm.

4.2 Implementation Details

We use a residual network (He et al., 2016) with 34 layers (current state of the art clustering results Huang et al. (2020) also use the same architecture). The MLP projection head consists of a hidden layer of size 2048, followed by batch normalization and ReLU layers, and an output layer of size 256. We use the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.0005. We implemented our algorithm using the Pytorch framework and trained our algorithm using a V100 GPU, taking 8 hours to train ImageNet-10 (13,000 training size) with a batch size of 128 for 500 epochs using 1 GPU.

During training, we use different augmentations to compute different views of the input images, following Chen et al. (2020); Grill et al. (2020). In particular, we first crop a random patch of the image with scale ranging from 0.08 to 1.0, and resize the cropped patch to 224×224 (128×128 and 96×96 in the case of smaller resolution datasets such as Intel and STL10 respectively). The resulting image was then flipped horizontally with a probability of 0.5. We then apply color transformations, starting by randomly changing the brightness, contrast, saturation and hue with a probability of 0.8. The image was then changed to gray-scale with a probability of 0.2. Then we applied a Gaussian Blur with kernel size (23×23) and a sigma chosen uniformly randomly between 0.1 and 2.0. The probability of applying the Gaussian Blur was 1.0 for view 1 and 0.5 for view 2. During evaluation, we resized the image such that the smaller edge of the image is of size 256 (not required for STL, Intel, CIFAR10, CIFAR100-20), and a center crop is performed with the resolution mentioned in Table 4. The color transformations were computed using Kornia (Riba et al., 2020) which is a differentiable computer vision library for Pytorch.

To compute the random transformations on the embeddings $\mathbf{z}$, we followed two techniques. We used random projections (Bingham & Mannila, 2001) with output dimension d, and transformed the embeddings $\mathbf{z}$ to the new space with dimension d. We also used diagonal transformation (Hsu et al., 2018) where we multiply $\mathbf{z}$ with a randomly generated diagonal matrix of the same dimension as $\mathbf{z}$.

We performed model selection on the hyperparameters of the random transformations on the embedding space such as the number of random transformations M (ranging from 10 to 100) and the dimensionality of the output space if using a random projection (we used [32, 64, 128, 256, 512]). We evaluated the models based on the metrics mentioned in Section A.2 on the data used for training the representations. Note that we fixed the number of prototypes to be equal to the number of ground truth classes. It was shown however that over-clustering leads to better representations (Caron et al., 2020; Ji et al., 2019; Asano et al., 2019) and we can extend our model to include an over-clustering block with a larger set of prototypes (Ji et al., 2019) and alternate the training procedure between the blocks.

4.3 Results

We show results on nine popular computer vision datasets described in Table 4 and evaluate it on three metrics - clustering accuracy, NMI and ARI (see A.2). Five of the datasets in Table 4 (ImageNet-10, ImageNet-Dogs, STL10, CIFAR10, and CIFAR100-20) have been widely used to evaluate these clustering metrics and to the best of our knowledge the other four datasets (Intel, Caltech101, CUB, and AwA2) are being used for systematically evaluating clustering results for the first time. In Table 1, we show the best accuracy we get for the proposed method ConCURL and compare it against the best accuracy achieved by state-of-the-art methods. Note that results highlighted in red color use both train and test data splits of those particular datasets for training. But in our case, we only use train split of the datasets as shown in Table 4. Even in that case we are able to outperform all state of the art methods in all but one dataset. In the ImageNet-10 dataset, ConCURL performs the best and it is better by 5.9% and 21% compared to PICA and GATCluster respectively in terms of clustering accuracy. ConCURL is better than PICA by more than 9% in terms of NMI and by more than 11% in terms of ARI in the ImageNet-10 dataset. We beat the next-best DCCM method by 18% in ImageNet-Dogs. In STL-10, PICA is the best followed by

ConCURL and in other datasets ConCURL is better than all other datases followed by PICA in terms of clustering accuracy.

Table 1: Clustering evaluation metrics

Datasets	ImageNet-10			ImageNet-Dogs			STL10			CIFAR10			CIFAR100-20		
Methods\Metrics	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
DCCM	0.710	0.608	0.555	0.383	0.321	0.182	0.482	0.376	0.262	0.623	0.496	0.408	0.327	0.285	0.173
GATCluster	0.762	0.609	0.572	0.333	0.322	0.200	0.583	0.446	0.363	0.610	0.475	0.402	0.281	0.215	0.116
PICA	0.870	0.802	0.761	0.352	0.352	0.201	0.713	0.611	0.531	0.696	0.591	0.512	0.337	0.310	0.171
ConCURL	0.922	0.877	0.852	0.452	0.447	0.288	0.623	0.514	0.428	0.705	0.545	0.507	0.366	0.351	0.195

In Table 2, we show results on "test data" that was not used during representation learning. We could not get trained models for all the datasets for GATCluster. Numbers in red violate this constraint because even test split was used for training. Even then, we outperform GATCluster in both ImageNet-10 and STL-10. Compared to PICA, we perform better both in ImageNet-10 and ImageNet-Dogs datasets. In Table 3, we show evaluation metrics on other four datasets for our method on both train and test split.

We also perform an ablation study on the affect of the losses L_1, L_2, L_3 (see A.4) and observe that the using consensus loss L_3 almost always improves accuracy. This shows the importance of consensus loss (L_3) and how ensemble learning through proposed consensus helps in achieving better clusters.

Table 2: Clustering evaluation metrics: On test split of the data

Datasets	ImageNet-10			ImageNet-Dogs			STL10			CIFAR10			CIFAR100-20		
Methods\Metrics	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
GATCluster [1]	0.762	0.609	0.571	-	-	-	0.588	0.462	0.379	-	-	-	-	-	-
PICA	0.758	0.724	0.602	0.375	0.399	0.208	0.692	0.588	0.496	0.694	0.588	0.507	0.331	0.311	0.166
ConCURL	0.864	0.840	0.770	0.455	0.477	0.274	0.611	0.498	0.410	0.693	0.527	0.488	0.363	0.354	0.193

Table 3: More datasets: Clustering evaluation metrics of ConCURL on train and test splits

Datasets	Intel			Caltech101			CUB			AwA2		
Split\Metrics	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
Train	0.910	0.801	0.800	0.339	0.651	0.223	0.127	0.452	0.033	0.539	0.681	0.448
Test	0.899	0.783	0.776	0.361	0.691	0.218	0.113	0.442	0.025	0.542	0.685	0.446

5 Conclusion

In this work, we leverage the ideas in unsupervised representation learning along with ensemble learning to perform clustering. We propose a novel ensemble learning algorithm which learns a representation by creating a consensus on multiple clustering outputs. Our proposed method outperforms all state of the art methods on various computer vision datasets. We also present the issue of overfitting in clustering and show that our method generalizes well on new data samples that were not available during training. This work is one of the first successful applications of ensemble learning in the deep clustering domain. This idea could easily be extended to ensembles of different clustering algorithms instead of variations in data representation (which was used in this paper). A possible extension can be to leverage the knowledge reuse framework of Strehl & Ghosh (2002) and use the clusterings output by the ensemble to compute a better quality partition of the input data. We believe that ensemble learning algorithms could also be effective in increasing robustness in clustering and we are planning to investigate this point further.

References

Adjusted Rand Score. `https://scikit-learn.org/stable/modules/clustering.html#adjusted-rand-score`. Accessed: 2020-09-23.

[1] GATCluster trained models were obtained from https://github.com/niuchuangnn/GATCluster, PICA trained models were obtained from https://github.com/Raymond-sci/PICA

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019.

Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250, 2001.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, 1998.

Leo Breiman. Stacked regressions. *Machine learning*, 24(1):49–64, 1996.

Sébastien Bubeck and Ulrike Von Luxburg. Overfitting of clustering and how to avoid it. 2007.

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)*, pp. 132–149, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

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

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

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*, pp. 215–223, 2011.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pp. 2292–2300, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Aniket Anand Deshmukh. *Kernel Methods for Learning with Limited Labeled Data*. PhD thesis, 2019.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pp. 226–231, 1996.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pp. 178–178. IEEE, 2004.

Xiaoli Z Fern and Carla E Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 186–193, 2003.

Ana LN Fred and Anil K Jain. Combining multiple clusterings using evidence accumulation. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):835–850, 2005.

Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. Citeseer, 1996.

Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

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*, pp. 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*, 2018.

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*, pp. 8849–8858, 2020.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

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*, pp. 9865–9874, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

Francesco Masulli and Andrea Schenone. A fuzzy clustering based segmentation system as support to diagnosis in medical imaging. *Artificial intelligence in medicine*, 16(2):129–147, 1999.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 1003–1011, 2009.

Chuang Niu, Jun Zhang, Ge Wang, and Jimin Liang. Gatcluster: Self-supervised gaussian-attention network for image clustering. *arXiv preprint arXiv:2002.11863*, 2020.

Rebecca Nugent and Marina Meila. An overview of clustering applied to molecular biology. In *Statistical methods in molecular biology*, pp. 369–404. Springer, 2010.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020. URL `https://arxiv.org/pdf/1910.02190.pdf`.

Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Alexander Topchy, Anil K Jain, and William Punch. Clustering ensembles: Models of consensus and weak partitions. *IEEE transactions on pattern analysis and machine intelligence*, 27(12): 1866–1881, 2005.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Jianlong Wu, Keyu Long, Fei Wang, Chen Qian, Cheng Li, Zhouchen Lin, and Hongbin Zha. Deep comprehensive correlation mining for image clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8150–8159, 2019.

Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2691–2699, 2015.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pp. 478–487, 2016.

Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pp. 649–666. Springer, 2016.

A Appendix

A.1 Dataset summary

The dataset summary is given in Table 4.

Table 4: Dataset Summary

Dataset	Classes	Train Data	Test Data	Resolution
ImageNet-10	10	13000	500	224× 224
Imagenet-Dogs	15	19500	750	224× 224
STL-10	10	5000	8000	96× 96
CIFAR10	10	50000	10000	32× 32
CIFAR100-20	20	50000	10000	32× 32
CUB	200	5994	5794	224× 224
Caltech-101	101	7020	1657	224× 224
Intel	6	14034	3000	128 × 128
AwA2	50	29865	7457	224 × 224

A.2 Evaluation metrics

We evaluate our algorithm by computing traditional clustering metrics (Cluster Accuracy, Normalized Mutual Information, and Adjusted Rand Index) (Zhang et al., 2016; Oord et al., 2018). Note that for measuring clustering metrics, the usual approach taken in literature is to evaluate the cluster metrics on the train data. Here, we report results on both the train data, as well as test data separately.

Cluster Accuracy The clustering accuracy is computed by first computing a cluster partition of the input data. Once the partitions are computed and cluster indices assigned to each input data point, the linear assignment map is computed using Kuhn-Munkres (Hungarian) algorithm that reassigns the cluster indices to the true labels of the data. Clustering accuracy is then given by

$$ACC = \frac{\sum_{i=1}^{N} \mathbb{1}\{y_{true}(x_i) = c(x_i)\}}{N},$$

where $y_{true}(x_i)$ is a true label of x_i and $c(x_i)$ is the cluster assignment produced by an algorithm (after Hungarian mapping).

Normalized Mutual Information For two clusterings U, V, with each containing $|U|, |V|$ clusters respectively, and let $|U_i|$ be the number of samples in cluster U_i of clustering U (similarly for V) , Mutual Information (MI) is given by

$$MI(U,V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_i|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|}$$

where N is the number of data points under consideration. Normalized Mutual Information is defined as

$$NMI(U,V) = \frac{MI(U,V)}{\sqrt{MI(U,U)MI(V,V)}}$$

Adjusted Rand Index (Hubert & Arabie, 1985; Skl) Suppose R is the groundtruth clustering and S is a partition, the RI of S is given as follows. Let a be the number of pairs of elements that are in the same set in R as well as in S; b be the number of pairs of elements that are in diferent sets in R, and different sets in S. Then

$$RI = \frac{a+b}{\binom{n}{2}}$$

$$ARI = \frac{RI - \mathbb{E}[RI]}{\max(RI) - \mathbb{E}[RI]}$$

A.3 T-SNE PLOTS

In figure 2, we show t-sne plot of the ImageNet-10 embeddings obtained from ConCURL trained model. One can clearly see the separation between various clusters with the exception of airline and airship clusters. Airline and airship clusters are mixed together on leftmost and righmost part of the t-sne plot.

A.4 ABLATION STUDY

In this subsection, we study the effect of weights α, β and γ on the final metrics.Results for the weight configuration corresponding to $\alpha = 1, \beta = 1, \gamma = 1$ is what is shown in the main paper. For the case of ($\alpha = 1, \beta = 0, \gamma = 0$), we computed the cluster accuracy, NMI, ARI by computing the embeddings of all the data output by the representation learning algorithm used for L_1 (here Grill et al. (2020)). Then we computed a K-means clustering on the embeddings (the target projection layer embeddings in this case) to obtain a partition of the data, and follow the same procedure mentioned in A.2

Table 5: Cluster metrics evaluation on data points that available during representation learning

Datasets	ImageNet-10			ImageNet-Dogs			STL10			Intel			Caltech101			CUB			AwA2		
Methods\Metrics	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
$\alpha = 1, \beta = 0, \gamma = 0$	0.818	0.843	0.757	0.492	0.464	0.289	-	-	-	0.889	0.764	0.758	0.348	0.641	0.212	0.134	0.460	0.041	0.528	0.713	0.416
$\alpha = 1, \beta = 1, \gamma = 0$	0.905	0.875	0.841	0.400	0.386	0.245	0.565	0.503	0.373	0.907	0.797	0.795	0.309	0.634	0.190	0.128	0.454	0.034	0.539	0.684	0.461
$\alpha = 1, \beta = 1, \gamma = 1$	0.922	0.877	0.852	0.452	0.447	0.288	0.623	0.514	0.428	0.910	0.801	0.800	0.339	0.651	0.223	0.127	0.452	0.033	0.539	0.681	0.448

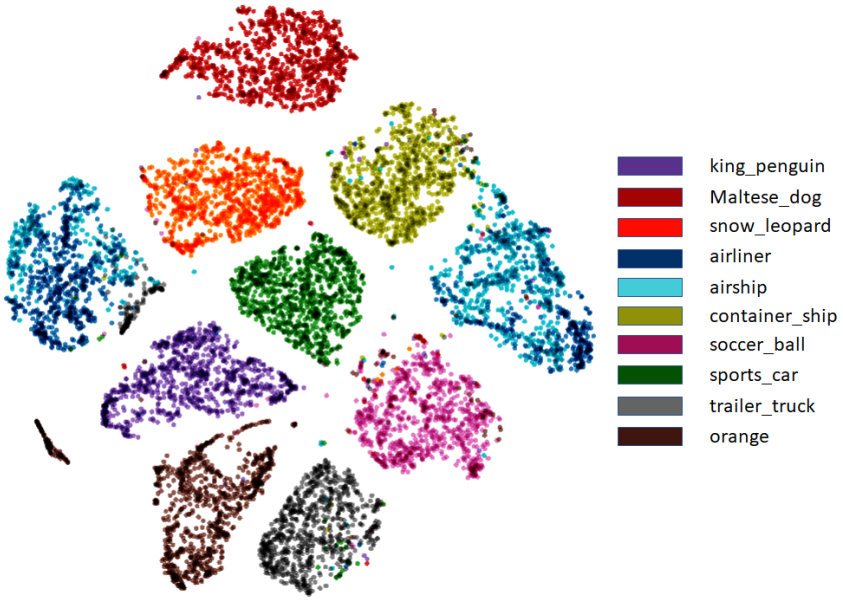

Figure 2: t-sne plot of ImageNet-10

Table 6: Cluster metrics evaluation on data points that were not available during representation learning

Datasets	ImageNet-10			ImageNet-Dogs			STL10			Intel			Caltech101			CUB			AwA2		
Methods\Metrics	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
$\alpha = 1, \beta = 0, \gamma = 0$	0.782	0.778	0.630	0.444	0.507	0.279	-	-	-	0.893	0.772	0.765	0.325	0.644	0.136	0.128	0.453	0.034	0.552	0.701	0.413
$\alpha = 1, \beta = 1, \gamma = 0$	0.884	0.867	0.811	0.408	0.441	0.238	0.560	0.484	0.362	0.903	0.787	0.785	0.325	0.667	0.181	0.117	0.445	0.027	0.541	0.681	0.456
$\alpha = 1, \beta = 1, \gamma = 1$	0.864	0.840	0.770	0.455	0.477	0.274	0.611	0.498	0.410	0.899	0.783	0.776	0.361	0.691	0.218	0.113	0.442	0.025	0.542	0.685	0.446