Efficient Learning Framework for Training Deep Learning Models with Limited Supervision

by

Kamran Ghasedi Dizaji

Master of Science, Amirkabir University of Technology, 2012

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

THE SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Kamran Ghasedi Dizaji

It was defended on

April 5, 2021

and approved by

Efficient Learning Framework for Training Deep Learning Models with Limited Supervision

Kamran Ghasedi Dizaji, PhD

University of Pittsburgh, 2021

In recent years, deep learning has shown tremendous success in different applications, however these modes mostly need a large labeled dataset for training their parameters. In this work, we aim to explore the potentials of efficient learning frameworks for training deep models on different problems in the case of limited supervision or noisy labels.

For the image clustering problem, we introduce a new deep convolutional autoencoder with an unsupervised learning framework. We employ a relative entropy minimization as the clustering objective regularized by the frequency of cluster assignments and a reconstruction loss.

In the case of noisy labels obtained by crowdsourcing platforms, we proposed a novel deep hybrid model for sentiment analysis of text data like tweets based on noisy crowd labels. The proposed model consists of a crowdsourcing aggregation model and a deep text autoencoder. We combine these sub-models based on a probabilistic framework rather than a heuristic way, and derive an efficient optimization algorithm to jointly solve the corresponding problem.

In order to improve the performance of unsupervised deep hash functions on image similarity search in big datasets, we adopt generative adversarial networks to propose a new deep image retrieval model, where the adversarial loss is employed as a data-dependent regularization in our objective function.

We also introduce a balanced self-paced learning algorithm for training a GAN-based model for image clustering, where the input samples are gradually included into training from easy to difficult, while the diversity of selected samples from all clusters are also considered.

In addition, we explore adopting discriminative approaches for unsupervised visual representation learning rather than the generative algorithms, such as maximizing the mutual information between an input image and its representation and a contrastive loss for decreasing the distance between the representations of original and augmented image data.

Table of Contents

List of Tables

List of Figures

Preface

First and foremost, I would like to sincerely thank my doctoral supervisor, Dr. Heng Huang, who has been very generous with his advice, encouragement, and support throughout my PhD program. He is an exceptional advisor and provided me an invaluable opportunity to dive deep into the machine learning field. I have always been inspired by his insightful vision to become a profound researcher, learned a lot from our in-depth discussions to see the differences and connections of various topics and acquire the big picture, been impressed by his dedication which motivated me to be a patient researcher, and was fortunate to have the freedom to explore new ideas and unconventional solutions because of his strong support and patience.

I would also like to thank my dissertation committee members, Dr. Zhi-Hong Mao, Dr. Wei Gao, Dr. Liang Zhan, and Dr. Wei Chen, for their insightful questions and constructive comments on my dissertation proposal and defense.

Many thanks to my dear friends in Data Science Lab, Xiaoqian, Hongchang, Zhouyuan, Alireza, Guodong, Shangqian, Yanfu, Feiping, Lei, Wenhao, Bin, Peng, De, An, Feihu, Runxue, Haoteng, Wenhan, Junyi, Zhengmian, for all their suggestions and comments and great times we had together.

Finally, my greatest gratitude is to my family, Mom, Dad, Keivan, and especially my lovely spouse Najmeh for their eternal and unconditional love. Thanks for being by my side through ups and downs, and being the best family that I could have ever asked for.

1.0 Introduction

In recent years, deep learning has shown impressive performance in wide range of applications, such as computer vision [78], natural language processing [26], social network embedding [153], speech recognition [59], and even biological science [31]. The competence of deep models is based on leaning hierarchical representations of data using scalable learning methods. However, learning wide and deep sets of features in multi-layer neural networks is a challenging task, since deep models are mostly prone to overfitting and getting stuck in undesirable local minima in the training process. Usually a large labeled dataset is utilized to train the parameters of deep models, but this is not a possible option in several practical problems. Several tricks and techniques are developed in the literature to alleviate this issue such as dropout [61], normalization layers like batch normalization [67] and weight normalization [134], non-saturating activation functions like rectified linear unit (ReLU) [103] and leaky rectified linear unit (LReLU) [98], clever architectures like Inception model [144], ResNet [56, 187] and DenseNet [65], and advanced optimization algorithms like Adam [74] and AdaGrad [34]. However, these techniques are not sufficient to address the problem of training deep models, where the training labeled data is scares. In this work, we propose efficient learning frameworks for training deep models with limited supervision. We mainly focus on image clustering, sentiment analysis and hashing functions problems as described in the following sections.

1.1 Unsupervised Deep Image Clustering Autoencoder

Clustering is one of the essential active research topics in machine learning and data mining, and has many different applications in various fields. The clustering problem has been extensively studied in the literature, and numerous algorithms were introduced to train clustering models without any supervisory signals. However, the current methods suffer either from inflexible shallow models or unstable deep embedding functions [41, 180]. The shallow methods cannot often capture the nonlinear nature of data due to their shallow and linear embedding function, have difficulties in scaling to large datasets because of their non-stochastic learning approach, and mostly degrades the results by using uncustomized hand-crafted features. Although the deep models are able to model the nonlinearity of data and efficiently deal with large-scale datasets, they are prone to getting stuck in bad local minima on training of their models with huge complexity, since there is no supervised information in the training process.

To address the mentioned challenging issues, we propose a new clustering algorithm, called deep embedded regularized clustering (*DEPICT*), which exploits the advantages of both discriminative clustering methods and deep embedding models. *DEPICT* generally consists of two main parts, a multinomial logistic regression (soft-max) layer stacked on top of a multi-layer convolutional autoencoder. The soft-max layer along with the encoder pathway can be considered as a discriminative clustering model, which is trained using the relative entropy (*KL* divergence) minimization. We further add a regularization term based on a prior distribution for the frequency of cluster assignments. The regularization term penalizes unbalanced cluster assignments and prevents allocating clusters to outlier samples. Moreover, we utilize the reconstruction loss function of autoencoder models as a data-dependent regularization term for avoiding the overfitting issue. In order to benefit from a joint learning framework for embedding and clustering, we simultaneously train all of the encoder and decoder layers together along with the soft-max layer by summing up the squared error reconstruction loss functions between the decoder and their corresponding (clean) encoder layers and add them to the clustering loss function.

1.2 Sentiment Analysis via Deep Hybrid Text-Crowd Model

Recently rapidly growing use of social media has provided a huge source of public opinions about different topics. Efficient mining of these opinions is very valuable for various industries and businesses. However, exploring the sentiment of public opinions is a very challenging task for automatic language models due to different variations in the texts, such as diverse contexts, genders of authors, writing styles and varied viewpoints. Crowdsourcing platforms provide an efficient tool to solve this type of the problems by using the knowledge of crowd workers in different tasks at low cost and time. Hence, the human skills in language understanding can be used to interpret the sentiments of texts with different variations. However, the collected labels via crowdsourcing are often noisy and inaccurate, because crowd workers are usually inexpert in the assigned task. In order to address this issue, it is common to collect multiple crowd labels for each sample to increase the credibility of the estimated true labels. Several studies have proposed different models to aggregate the crowd labels and estimate the potential true labels, which are also called truths [27, 22, 43]. But these crowdsourcing aggregation models become drastically incompetent, when the number of crowd labels per worker is not enough to train the reliability parameters of workers, or a document dataset is extremely large that collecting crowd labels for all samples is not practically feasible. In addition, crowdsourcing aggregation models do not utilize text data, and only use crowd labels as the source of information (i.e. input data).

We introduce a new hybrid model for sentiment analysis, which utilizes both crowd labels and text data. In particular, our proposed model, called *CrowdDeepAE*, consists of a generative aggregation model for crowd labels and a deep autoencoder for text data. These two sub-models are coupled in a probabilistic framework rather than a heuristic approach. Using this probabilistic framework, we introduce a unified objective function that incorporates the interests of both sub-models. We further derive an efficient optimization algorithm to solve the corresponding problem via an alternating approach, in which the parameters are updated while the truths are assumed to be known, and the truths are estimated when the parameters are fixed.

1.3 Unsupervised Deep Image retrieval Network

Image similarity search in big datasets has gained tremendous attentions in different applications such as information retrieval, data mining and pattern recognition [156]. With rapid growth of image data, it has become crucial to find compact and discriminative representations of images in huge datasets in order to have efficient storage and real-time matching for millions of images. Hashing functions provide an effective solution for this problem by attributing a binary code to each image, and consequently reducing the similarity search between high dimensional images to calculating the Hamming distance between their binary codes [47, 163]. Typically, hash functions are carefully designed to extract distinctive patterns from images relevant to their semantic categorizes, while being robust to various image transformations such as rotation, translation, scale, and lightning [93, 179, 66].

Generally, hash functions can be divided into supervised [94, 172, 51, 89] and unsupervised methods [55, 163, 157, 58]. The supervised hashing methods [82, 32, 199, 179] showed remarkable performance in representing input data with binary codes. Although, these deep hash functions take advantages of deep learning models in representing images with discriminative attributes, they require costly human-annotated labels to train their large set of parameters. Thus, their performance is dramatically degraded by getting stuck in bad local minima when there is not enough labeled data for training. The unsupervised hashing methods address this issue by providing learning frameworks without requiring any supervisory signals. The unsupervised hashing methods either use shallow models with hand-crafted features [17, 85, 3] as inputs, or employ deep architectures for obtaining both discriminative features and binary hash codes together. However, the unsupervised shallow functions may not capture the non-linear similarities between real-world images due to their low capacity. They also suffer from hand-crafted features and dimension reductions techniques, which are not robust to noise and image transformations.

We propose a new unsupervised deep hashing model, called *HashGAN*, which nor suffers from shallow hash functions and hand-crafted features, neither needs the supervised pre-training to have discriminative binary codes. Our framework jointly learns a hash function with a generative adversarial network (GAN). In particular, we tie the discriminator of the GAN with the hash function, employing the adversarial loss function as a data-dependent regularization term in training our deep hash function. Furthermore, we introduce a novel loss function for hashing real images, minimizing the entropy of hash bits for each image, maximizing the entropy of frequency of hash bits, improving the consistency of hash codes against different image transformations, and providing independent hash bits. Moreover, we provide a collaborative loss function, which enforces the encoder to have the same binary hash code for a synthesized image by the generator, as the binary input variable provided to the generator while synthesizing the image.

1.4 Generative Adversarial Clustering Network

Clustering is one of the essential active research topics in computer vision and machine learning communities with various applications. Clustering problem has been extensively studied in the literature by introducing numerous algorithms with unsupervised learning frameworks [177]. However, the existing methods that employ shallow or deep models suffer from different issues. The shallow clustering models may not capture the nonlinear nature of data due to their shallow and linear embedding functions, adversely affect their performance by using inflexible hand-crafted features, and have difficulties in scaling to large datasets. In contrast, the deep clustering methods have enough capacity to model the non-linear and complex data, and are able to deal with large-scale datasets. But they are prone to the overfitting issue leading to get stuck in bad local minima, since there is no reliable supervisory signal for training their large number of parameters.

To tackle these issues, we propose a generative adversarial clustering network, called *ClusterGAN*, as a novel deep clustering model to address the aforementioned issues. *ClusterGAN* adopts the adversarial game in *GAN* for the clustering task, and employs an efficient self-paced learning algorithm to boost its performance. Unlike the traditional *GAN*, *ClusterGAN* consists of three networks, a discriminator $\mathscr{D}$, a generator $\mathscr{G}$, and a clusterer $\mathscr{C}$ (*i.e.* a clustering network). The generator and clusterer are both conditional generative networks, where $\mathscr{G} : \mathbf{z} \rightarrow \hat{\mathbf{x}}$ generates the realistic data samples given the latent variables and $\mathscr{C} : \mathbf{x} \rightarrow \hat{\mathbf{z}}$ generates the discriminative latent variables given the real data. The discriminator $\mathscr{D}$ accepts a joint distribution of samples and features (*i.e.* latent variables) as the input, and tries to identify whether the paired samples belong to the generator $(\mathbf{z}, \hat{\mathbf{x}})$ or the clusterer $(\hat{\mathbf{z}}, \mathbf{x})$. Thus, training the generator and clusterer to fool the discriminator leads to generating synthesized samples similar to real data and learning discriminative embedding space in the clusterer similar to the generator latent variables.

Moreover, we introduce a novel clustering objective, which is directly applied on the output of the clusterer given the real samples. The basic idea is to impose a block diagonal constraint on the adjacency matrix of the real data. To do so, we first compute the similarity values between real samples using the cosine similarity function applied on the clusterer outputs. Then, a minimum entropy loss function is imposed to the similarity values to push them towards 0 (*i.e.* dissimilar) or 1 (*i.e.* similar). However, the main challenge is that the ground-truth similarities are unknown in unsupervised learning, which makes it difficult to train a deep clustering model from the scratch. In order to tackle this issue, we enhance the minimum entropy objective by utilizing a novel self-paced learning algorithm. The self-paced learning algorithm initiates the training process with easy samples, and then gradually takes more difficult samples into the training. In addition, we take the prior of selected samples into consideration using an exclusive lasso regularization. This helps us to select a more diverse set of samples in each training step, and prevents learning from easy samples belonging only to a few clusters.

1.5 Unsupervised Visual Representation Learning

The explosive growth of image data in the internet and social media has driven huge interest in efficient unsupervised models that are able to find similar patterns among the data. For instance, there are many studies on approximate nearest neighbor search (ANNS) algorithms, which aim to provide efficient image similarity search on large-scale image datasets. Hashing-based ANNS methods tackle this problem by representing image data with binary codes, providing an effective solution for the similarity search and storage of millions of images [47, 163, 94, 156, 89]. Categorizing similar/dissimilar images into the same/different sets is another essential task in machine learning and computer vision. This problem is extensively studied in the literature by introducing models that find discriminative boundaries between different image categories [177, 96]. These models are required to extract semantic features from image data related to their categories, and be robust to different image styles caused by spatial/geometric transformations and color distortions.

Supervised deep models have shown remarkable performance in image classification and retrieval by training their flexible mapping function using large sets of labeled data [55, 163, 157, 56, 65, 145]. However, unsupervised deep hashing and clustering models generally lag behind their supervised counterparts on image data, since the lack of reliable supervisory signals may lead to learning some arbitrary representations in deep models with large numbers of free parameters. In order to address this problem, some studies employ auxiliary reconstruction or generative loss functions as additional regularizations [174, 101, 18]. However, these regularizations usually enforce the models to contain some unnecessary generative information that is not directly relevant to the required ability of discriminative representations. Also the unsupervised deep models usually provide insignificant improvements compared to their shallow counterparts, and sometimes need a variant of supervised pretrainings to initialize their parameters [93, 66].

To address these issues, we propose a new unsupervised learning framework for deep models in image retrieval and clustering tasks based on three loss functions, an adversarial loss between three networks including a critic, a generator, and an encoder, a maximum mutual information loss and a contrastive loss for the encoder. The adversarial loss enforces the generator to learn the conditional image distribution given a set of random content and style latent variables, the encoder to map the input images to a set of content and style latent representations, and the critic to distinguish its joint input data belonging to the generator or encoder. The maximum mutual information loss aims to increase the correlation of between images and their latent representations based on a Jensen–Shannon (*i.e.* JS) divergence in a GAN-style sub-network. The contrastive loss tries to disentangle the content and style representations by decreasing/increasing the distance between content features of an image and its augmented variant/other images.

1.6 Contribution

We summarize our contribution as follows:

- Proposing a discriminative non-linear embedding subspace via the deep convolutional autoencoder, that is trained with an end-to-end joint learning approach unifying the clustering and embedding tasks and avoiding layer-wise pretraining;
- Introducing a hybrid crowd-text model for sentiment analysis, consisting of a generative crowd aggregation model and a deep sentimental autoencoder, which are combined based on a probabilistic framework;
- Proposing a novel framework for unsupervised hashing model by coupling a deep hash function and a generative adversarial network.
- introduce a deep clustering model by adopting the generative adversarial network for clustering and employing a novel balanced self-paced learning algorithm to gradually include samples into training steps from easy to difficult while considering the diversity of selected samples from all clusters.
- Proposing a novel unsupervised visual representation learning framework for training deep models to map image data into disentangled content and style representations using a generative adversarial loss and discriminative contrastive and mutual information loss functions.

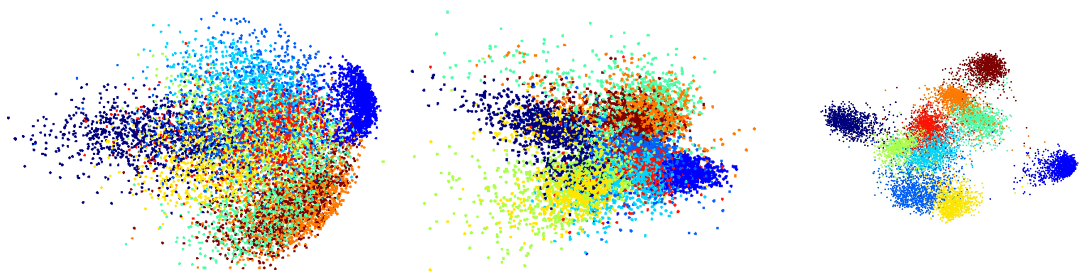
2.0 Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimizatione

2.1 Introduction

Clustering is one of the fundamental research topics in machine learning and computer vision research, and it has gained significant attention for discriminative representation of data points without any need for supervisory signals. The clustering problem has been extensively studied in various applications; however, the performance of standard clustering algorithms is adversely affected when dealing with high-dimensional data, and their time complexity dramatically increases when working with large-scale datasets. Tackling the curse of dimensionality, previous studies often initially project data into a low-dimensional manifold, and then cluster the embedded data in this new subspace [126, 146, 160]. Handling large-scale datasets, there are also several studies which select only a subset of data points to accelerate the clustering process [138, 97, 90].

However, dealing with real-world image data, existing clustering algorithms suffer from different issues: 1) Using inflexible hand-crafted features, which do not depend on the input data distribution; 2) Using shallow and linear embedding functions, which are not able to capture the non-linear nature of data; 3) Non-joint embedding and clustering processes, which do not result in an optimal embedding subspace for clustering; 4) Complicated clustering algorithms that require tuning the hyper-parameters using labeled data, which is not feasible in real-world clustering tasks.

To address the mentioned challenging issues, we propose a new clustering algorithm, called deep embedded regularized clustering (*DEPICT*), which exploits the advantages of both discriminative clustering methods and deep embedding models. *DEPICT* generally consists of two main parts, a multinomial logistic regression (soft-max) layer stacked on top of a multi-layer convolutional autoencoder. The soft-max layer along with the encoder pathway can be considered as a discriminative clustering model, which is trained using the relative entropy (*KL* divergence) minimization. We further add a regularization term based



(a) Raw Data (b) NonJoint *DEPICT* (c) Joint *DEPICT*

Figure 1: Visualization to show the discriminative capability of embedding subspaces using *MNIST-test* data. (a) The space of raw data. (b) The embedding subspace of non-joint *DEPICT* using standard stacked denoising autoencoder (SdA). (c) The embedding subspace of joint *DEPICT* using our joint learning approach (MdA).

on a prior distribution for the frequency of cluster assignments. The regularization term penalizes unbalanced cluster assignments and prevents allocating clusters to outlier samples.

Although this deep clustering model is flexible enough to discriminate the complex real-world input data, it can easily get stuck in non-optimal local minima during training and result in undesirable cluster assignments. In order to avoid overfitting the deep clustering model to spurious data correlations, we utilize the reconstruction loss function of autoencoder models as a data-dependent regularization term for training parameters.

In order to benefit from a joint learning framework for embedding and clustering, we introduce a unified objective function including our clustering and auxiliary reconstruction loss functions. We then employ an alternating approach to efficiently update the parameters and estimate the cluster assignments. It is worth mentioning that in the standard learning approach for training a multi-layer autoencoder, the encoder and decoder parameters are first pretrained layer-wise using the reconstruction loss, and the encoder parameters are then fine-tuned using the objective function of the main task [152]. However, it has been argued that the non-joint fine-tuning step may overwrite the encoder parameters entirely and consequently cancel out the benefit of the layer-wise pretraining step [194]. To avoid this problem and achieve optimal joint learning results, we simultaneously train all of the encoder and decoder layers together along with the soft-max layer. To do so, we sum up the squared error reconstruction loss functions between the decoder and their corresponding (clean) encoder layers and add them to the clustering loss function.

Figure 1 demonstrates the importance of our joint learning strategy by comparing different data representations of *MNIST-test* data points [83] using principle component analysis (PCA) visualization. The first figure indicates the raw data representation; The second one shows the data points in the embedding subspace of non-joint *DEPICT*, in which the model is trained using the standard layer-wise stacked denoising autoencoder (SdA); The third one visualizes the data points in the embedding subspace of joint *DEPICT*, in which the model is trained using our multi-layer denoising autoencoder learning approach (MdA). As shown, joint *DEPICT* using MdA learning approach provides a significantly more discriminative embedding subspace compared to non-joint *DEPICT* using standard SdA learning approach.

Moreover, experimental results show that *DEPICT* achieves superior or competitive results compared to the state-of-the-art algorithms on the image benchmark datasets while having faster running times. In addition, we compared different learning strategies for *DEPICT*, and confirm that our joint learning approach has the best results. It should also be noted that *DEPICT* does not require any hyper-parameter tuning using supervisory signals, and consequently is a better candidate for the real-world clustering tasks. Thus, we summarize the advantages of *DEPICT* as:

- Providing a discriminative non-linear embedding subspace via the deep convolutional autoencoder;
- Introducing an end-to-end joint learning approach, which unifies the clustering and embedding tasks, and avoids layer-wise pretraining;
- Achieving superior or competitive clustering results on high-dimensional and large-scale datasets with no need for hyper-parameter tuning using labeled data.

2.2 Related Works

There is a large number of clustering algorithms in literature, which can be grouped into different perspectives, such as hierarchical [57, 167, 191], centroid-based [95, 11, 108, 6], graph-based [137, 109, 159, 106], sequential (temporal) [72, 135, 131, 198, 128], regression model based [38, 155], and subspace clustering models [1, 70, 37, 107]. In another sense, they are generally divided into two subcategories, generative and discriminative clustering algorithms. The generative algorithms like K*-means* and Gaussian mixture model [12] explicitly represent the clusters using geometric properties of the feature space, and model the categories via the statistical distributions of input data. Unlike the generative clustering algorithms, the discriminative methods directly identify the categories using their separating hyperplanes regardless of data distribution. Information theoretic [86, 7, 76], max-margin [193, 176], and spectral graph [105] algorithms are examples of discriminative clustering models. Generally it has been argued that the discriminative models often have better results compared to their generative counterparts, since they have fewer assumptions about the data distribution and directly separate the clusters, but their training can suffer from overfitting or getting stuck in undesirable local minima [76, 105, 121]. Our *DEPICT* algorithm is also a discriminative clustering model, but it benefits from the auxiliary reconstruction task of autoencoder to alleviate this issues in training of our discriminative clustering algorithm.

There are also several studies regarding the combination of clustering with feature embedding learning. Ye *et al.* introduced a kernelized K*-means* algorithm, denoted by *DisKmeans*, where embedding to a lower dimensional subspace via linear discriminant analysis (*LDA*) is jointly learned with K*-means* cluster assignments [182]. [154] proposed to a new method to simultaneously conduct both clustering and feature embedding/selection tasks to achieve better performance. But these models suffer from having shallow and linear embedding functions, which cannot represent the non-linearity of real-world data.

A joint learning framework for updating code books and estimating image clusters was proposed in [175] while *SIFT* features are used as input data. A deep structure, named *TAGnet* was introduced in [160], where two layers of sparse coding followed by a clustering algorithm are trained with an alternating learning approach. Similar work is presented in

[161] that formulates a joint optimization framework for discriminative clustering and feature extraction using sparse coding. However, the inference complexity of sparse coding forces the model in [161] to reduce the dimension of input data with *PCA* and the model in [160] to use an approximate solution. Hand-crafted features and dimension reduction techniques degrade the clustering performance by neglecting the distribution of input data.

Tian *et al.* learned a non-linear embedding of the affinity graph using a stacked autoencoder, and then obtained the clusters in the embedding subspace via *K-means* [146]. Trigeorgis *et al.* extended semi non-negative matrix factorization (*semi-NMF*) to stacked multi-layer (deep) *semi-NMF* to capture the abstract information in the top layer. Afterwards, they run *K-means* over the embedding subspace for cluster assignments [148]. More recently, Xie *et al.* employed denoising stacked autoencoder learning approach, and first pretrained the model layer-wise and then fine-tuned the encoder pathway stacked by a clustering algorithm using Kullback-Leibler divergence minimization [174]. Unlike these models that require layer-wise pretraining as well as non-joint embedding and clustering learning, *DEPICT* utilizes an end-to-end optimization for training all network layers simultaneously using the unified clustering and reconstruction loss functions.

Yang *et al.* introduced a new clustering model, named *JULE*, based on a recurrent framework, where data is represented via a convolutional neural network and embedded data is iteratively clustered using an agglomerative clustering algorithm [180]. They derived a unified loss function consisting of the merging process for agglomerative clustering and updating the parameters of the deep representation. While *JULE* achieved good results using the joint learning approach, it requires tuning of a large number of hyper-parameters, which is not practical in real-world clustering tasks. In contrast, our model does not need any supervisory signals for hyper-parameter tuning.

2.3 Deep Embedded Regularized Clustering

In this section, we first introduce the clustering objective function and the corresponding optimization algorithm, which alternates between estimating the cluster assignments and updating model parameters. Afterwards, we show the architecture of *DEPICT* and provide the joint learning framework to simultaneously train all network layers using the unified clustering and reconstruction loss functions.

2.3.1 DEPICT Algorithm

Let's consider the clustering task of N samples, $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$, into K categories, where each sample $\mathbf{x}_i \in \mathbb{R}^{d_x}$. Using the embedding function, $\varphi_W : X \rightarrow Z$, we are able to map raw samples into the embedding subspace $\mathbf{Z} = [\mathbf{z}_1, ..., \mathbf{z}_n]$, where each $\mathbf{z}_i \in \mathbb{R}^{d_z}$ has a much lower dimension compared to the input data (i.e. $d_z \ll d_x$). Given the embedded features, we use a multinomial logistic regression (soft-max) function $f_\theta : Z \rightarrow Y$ to predict the probabilistic cluster assignments as follows.

$$p_{ik} = P(y_i = k|\mathbf{z}_i, \mathbf{\Theta}) = \frac{exp(\boldsymbol{\theta}_k^T \mathbf{z}_i)}{\sum\limits_{k'=1}^{K} exp(\boldsymbol{\theta}_{k'}^T \mathbf{z}_i)}\,, \tag{2.1}$$

where $\mathbf{\Theta} = [\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k] \in \mathbb{R}^{d_z \times K}$ are the soft-max function parameters, and p_{ik} indicates the probability of the i-th sample belonging to the k-th cluster.

In order to define our clustering objective function, we employ an auxiliary target variable $\mathbf{Q}$ to refine the model predictions iteratively. To do so, we first use Kullback-Leibler (KL) divergence to decrease the distance between the model prediction $\mathbf{P}$ and the target variable $\mathbf{Q}$.

$$\mathscr{L} = KL(\mathbf{Q}\|\mathbf{P}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik} \log \frac{q_{ik}}{p_{ik}}\,, \tag{2.2}$$

In order to avoid degenerate solutions, which allocate most of the samples to a few clusters or assign a cluster to outlier samples, we aim to impose a regularization term to the target variable. To this end, we first define the empirical label distribution of target variables as:

$$f_k = P(\mathbf{y} = k) = \frac{1}{N}\sum_{i} q_{ik}\,, \tag{2.3}$$

where f_k can be considered as the soft frequency of cluster assignments in the target distribution. Using this empirical distribution, we are able to enforce our preference for having balanced assignments by adding the following KL divergence to the loss function.

$$\mathscr{L} = KL(\mathbf{Q}\|\mathbf{P}) + KL(\mathbf{f}\|\mathbf{u}) \tag{2.4}$$
$$= \Big[\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log\frac{q_{ik}}{p_{ik}}\Big] + \Big[\frac{1}{N}\sum_{k=1}^{K} f_k\log\frac{f_k}{u_k}\Big] = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log\frac{q_{ik}}{p_{ik}} + q_{ik}\log\frac{f_k}{u_k}\,,$$

where $\mathbf{u}$ is the uniform prior for the empirical label distribution. While the first term in the objective minimizes the distance between the target and model prediction distributions, the second term balances the frequency of clusters in the target variables. Utilizing the balanced target variables, we can force the model to have more balanced predictions (cluster assignments) $\mathbf{P}$ indirectly. It is also simple to change the prior from the uniform distribution to any arbitrary distribution in the objective function if there is any extra knowledge about the frequency of clusters.

An alternating learning approach is utilized to optimize the objective function. Using this approach, we estimate the target variables $\mathbf{Q}$ via fixed parameters (expectation step), and update the parameters while the target variables $\mathbf{Q}$ are assumed to be known (maximization step). The problem to infer the target variable $\mathbf{Q}$ has the following objective:

$$\min_{\mathbf{Q}} \quad \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log\frac{q_{ik}}{p_{ik}} + q_{ik}\log\frac{f_k}{u_k}\,, \tag{2.5}$$

where the target variables are constrained to $\sum_k q_{ik} = 1$. This problem can be solved using first order methods, such as gradient descent, projected gradient descent, and Nesterov optimal method [104], which only require the objective function value and its (sub)gradient at each iteration. In the following equation, we show the partial derivative of the objective function with respect to the target variables.

$$\frac{\partial\mathscr{L}}{\partial q_{ik}} \propto \log\Big(\frac{q_{ik}f_k}{p_{ik}}\Big) + \frac{q_{ik}}{\sum\limits_{i'=1}^{N} q_{i'k}} + 1\,, \tag{2.6}$$

Investigating this problem more carefully, we approximate the gradient in Eq.(4.5) by removing the second term, since the number of samples N is often big enough to ignore the second term. Setting the gradient equal to zero, we are now able to compute the closed form solution for $\mathbf{Q}$ accordingly.

$$q_{ik} = \frac{p_{ik}/(\sum_{i'} p_{i'k})^{\frac{1}{2}}}{\sum\limits_{k'} p_{ik'}/(\sum_{i'} p_{i'k'})^{\frac{1}{2}}}\,, \tag{2.7}$$

For the maximization step, we update the network parameters $\boldsymbol{\psi} = \{\mathbf{\Theta}, \mathbf{W}\}$ using the estimated target variables with the following objective function.

$$\min_{\boldsymbol{\psi}} \quad -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log p_{ik}\,, \tag{2.8}$$

Interestingly, this problem can be considered as a standard cross entropy loss function for classification tasks, and the parameters of soft-max layer $\mathbf{\Theta}$ and embedding function $\mathbf{W}$ can be efficiently updated by backpropagating the error.

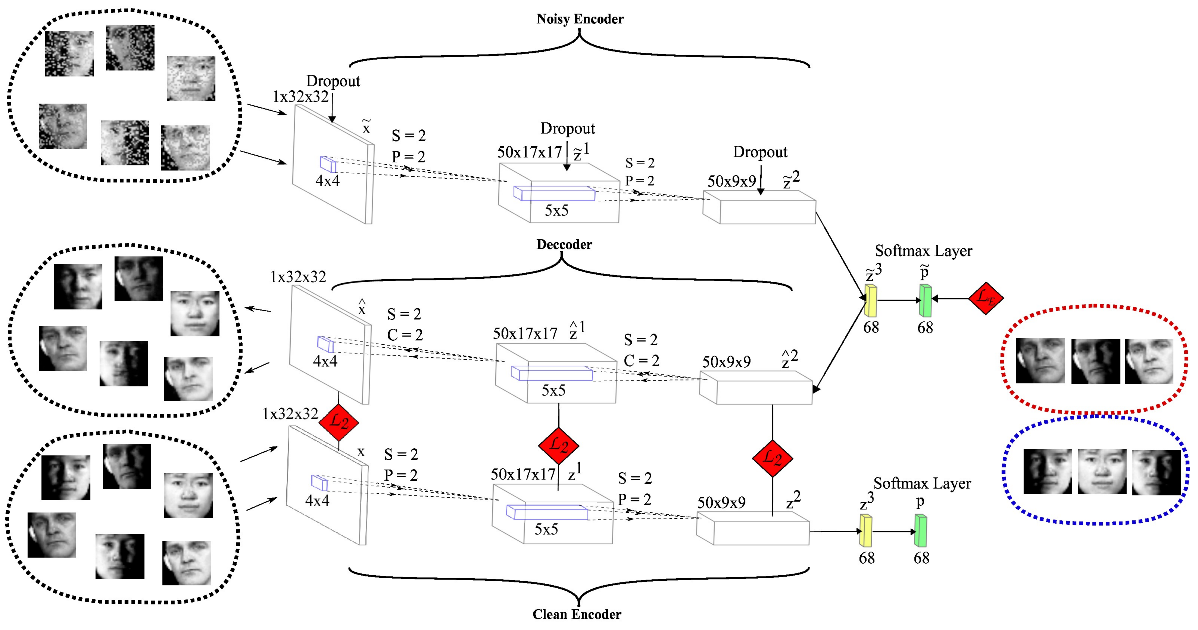


Figure 2: Architecture of *DEPICT* for *CMU-PIE* dataset. *DEPICT* consists of a soft-max layer stacked on top of a multi-layer convolutional autoencoder. In order to illustrate the joint learning framework, we consider the following four pathways for *DEPICT*: Noisy (corrupted) encoder, Decoder, Clean encoder and Soft-max layer. The clustering loss function, L_E, is applied on the noisy pathway, and the reconstruction loss functions, L_2, are between the decoder and clean encoder layers. The output size of convolutional layers, kernel sizes, strides (S), paddings (P) and crops (C) are also shown.

2.3.2 DEPICT Architecture

In this section, we extend our general clustering loss function using a denoising autoencoder. The deep embedding function is useful for capturing the non-linear nature of input data; However, it may overfit to spurious data correlations and get stuck in undesirable local minima during training. To avoid this overfitting, we employ autoencoder structures and use the reconstruction loss function as a data-dependent regularization for training the parameters. Therefore, we design *DEPICT* to consist of a soft-max layer stacked on top of a multi-layer convolutional autoencoder. Due to the promising performance of strided convolutional layers in [120, 184], we employ convolutional layers in our encoder and strided convolutional layers in the decoder pathways, and avoid deterministic spatial pooling layers (like max-pooling). Strided convolutional layers allow the network to learn its own spatial upsampling, providing a better generation capability.

Unlike the standard learning approach for denoising autoencoders, which contains layer-wise pretraining and then fine-tuning, we simultaneously learn all of the autoencoder and soft-max layers. As shown in Figure 2, *DEPICT* consists of the following components:

1) Corrupted feedforward (encoder) pathway maps the noisy input data into the embedding subspace using a few convolutional layers followed by a fully connected layer. The following equation indicates the output of each layer in the noisy encoder pathway.

$$\tilde{\mathbf{z}}^l = \mathit{Dropout}\big[g(\mathbf{W}_e^l \tilde{\mathbf{z}}^{l-1})\big] \,, \tag{2.9}$$

where $\tilde{\mathbf{z}}^l$ are the noisy features of the l-th layer, *Dropout* is a stochastic mask function that randomly sets a subset of its inputs to zero [143], g is the activation function of convolutional or fully connected layers, and $\mathbf{W}_e^l$ indicates the weights of the l-th layer in the encoder. Note that the first layer features, $\tilde{\mathbf{z}}^0$, are equal to the noisy input data, $\tilde{\mathbf{x}}$.

2) Followed by the corrupted encoder, the decoder pathway reconstructs the input data through a fully connected and multiple strided convolutional layers as follows,

$$\hat{\mathbf{z}}^{l-1} = g(\mathbf{W}_d^l \hat{\mathbf{z}}^l) \,, \tag{2.10}$$

where $\hat{\mathbf{z}}^l$ is the l-th reconstruction layer output, and $\mathbf{W}_d^l$ shows the weights for the l-th layer of the decoder. Note that input reconstruction, $\hat{\mathbf{x}}$, is equal to $\hat{\mathbf{z}}^0$.

3) Clean feedforward (encoder) pathway shares its weights with the corrupted encoder, and infers the clean embedded features. The following equation shows the outputs of the clean encoder, which are used in the reconstruction loss functions and obtaining the final cluster assignments.

$$\mathbf{z}^l = g(\mathbf{W}_e^l \mathbf{z}^{l-1})\,, \tag{2.11}$$

where $\mathbf{z}^l$ is the clean output of the l-th layer in the encoder. Consider the first layer features $\mathbf{z}^0$ equal to input data $\mathbf{x}$.

4) Given the top layer of the corrupted and clean encoder pathways as the embedding subspace, the soft-max layer obtains the cluster assignments using Eq.(C.2).

Note that we compute target variables $\mathbf{Q}$ using the clean pathway, and model prediction $\tilde{\mathbf{P}}$ via the corrupted pathway. Hence, the clustering loss function $KL(\mathbf{Q}\|\tilde{\mathbf{P}})$ forces the model to have invariant features with respect to noise. In other words, the model is assumed to have a dual role: a clean model, which is used to compute the more accurate target variables; and a noisy model, which is trained to achieve noise-invariant predictions.

As a crucial point, *DEPICT* algorithm provides a joint learning framework that optimizes the soft-max and autoencoder parameters together.

$$\min_{\psi} \; -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log \tilde{p}_{ik} + \frac{1}{N}\sum_{i=1}^{N}\sum_{l=0}^{L-1}\frac{1}{|\mathbf{z}_i^l|}\|\mathbf{z}_i^l - \hat{\mathbf{z}}_i^l\|_2^2\,, \tag{2.12}$$

where $|\mathbf{z}_i^l|$ is the output size of the l-th hidden layer (input for $l = 0$), and L is the depth of the autoencoder model.

The benefit of joint learning frameworks for training multi-layer autoencoders is also reported in semi-supervised classification tasks [122, 194]. However, *DEPICT* is different from previous studies, since it is designed for the unsupervised clustering task, it also does not require max-pooling switches used in stacked what-where autoencoder (SWWAE) [194], and lateral (skip) connections between encoder and decoder layers used in ladder network [122]. Algorithm 1 shows a brief description of *DEPICT* algorithm.

Algorithm 1: *DEPICT* Algorithm

```
1 Initialize $\mathbf{Q}$ using a clustering algorithm
2 while not converged do
3     $\min_{\boldsymbol{\psi}} \; -\frac{1}{N}\sum_{ik} q_{ik}\log\tilde{p}_{ik} + \frac{1}{N}\sum_{il}\frac{1}{|\mathbf{z}_i^l|}\|\mathbf{z}_i^l - \hat{\mathbf{z}}_i^l\|_2^2$
4     $p_{ik}^{(t)} \propto exp(\boldsymbol{\theta}_k^T \mathbf{z}_i^L)$
5     $q_{ik}^{(t)} \propto p_{ik}/(\sum_{i'} p_{i'k})^{\frac{1}{2}}$
6 end
```

2.4 Experiments

In this section, we first evaluate *DEPICT*[1] in comparison with state-of-the-art clustering methods on several benchmark image datasets. Then, the running speed of the best clustering models are compared. Moreover, we examine different learning approaches for training *DEPICT*. Finally, we analyze the performance of *DEPICT* model on semi-supervised classification tasks.

Datasets: In order to show that *DEPICT* works well with various kinds of datasets, we have chosen the following handwritten digit and face image datasets. Considering that clustering tasks are fully unsupervised, we concatenate the training and testing samples when applicable. *MNIST-full*: A dataset containing a total of 70,000 handwritten digits with 60,000 training and 10,000 testing samples, each being a 32 by 32 monochrome image [83]. *MNIST-test*: A dataset which only consists of the testing part of *MNIST-full* data. *USPS*: It is a handwritten digits dataset from the *USPS* postal service, containing 11,000 samples of 16 by 16 images. *CMU-PIE*: A dataset including 32 by 32 face images of 68 people with 4 different expressions [139]. *Youtube-Face (YTF)*: Following [180], we choose the first 41 subjects of YTF dataset. Faces inside images are first cropped and then resized to 55 by 55 sizes [168]. *FRGC*: Using the 20 random selected subjects in [180] from the original dataset, we collect 2,462 face images. Similarly, we first crop the face regions and resize them into 32 by 32 images. Table 10 provides a brief description of the datasets.

[1]Our code is available in `https://github.com/herandy/DEPICT`

Table 1: Dataset Descriptions

Dataset	# Samples	# Classes	# Dimensions
MNIST-full	70,000	10	1×28×28
MNIST-test	10,000	10	1×28×28
USPS	11,000	10	1×16×16
FRGC	2,462	20	3×32×32
YTF	10,000	41	3×55×55
CMU-PIE	2,856	68	1×32×32

Clustering Metrics: We have used 2 of the most popular evaluation criteria widely used for clustering algorithms, accuracy (ACC) and normalized mutual information (NMI). The best mapping between cluster assignments and true labels is computed using the Hungarian algorithm [79] to measure accuracy. NMI calculates the normalized measure of similarity between two labels of the same data [178]. Results of NMI do not change by permutations of clusters (classes), and they are normalized to have $[0, 1]$ range, with 0 meaning no correlation and 1 exhibiting perfect correlation.

Table 2: Clustering performance of different algorithms on image datasets based on accuracy (ACC) and normalized mutual information (NMI). The numbers of tuned hyper-parameters (# tuned HPs) using the supervisory signals are also shown for each algorithm. The results of alternative models are reported from original projects, except the ones marked by (∗) on top, which are obtained by us running the released code. We put dash marks (-) for the results that are not practical to obtain.

Dataset	*MNIST-full*		*MNIST-test*		*USPS*		*FRGC*		*YTF*		*CMU-PIE*		# tuned
	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	HPs
K-means	0.500*	0.534*	0.501*	0.547*	0.450*	0.460*	0.287*	0.243*	0.776*	0.601*	0.432*	0.223*	0
N-Cuts	0.411	0.327	0.753	0.304	0.675	0.314	0.285	0.235	0.742	0.536	0.411	0.155	0
SC-ST	0.416	0.311	0.756	0.454	0.726	0.308	0.431	0.358	0.620	0.290	0.581	0.293	0
SC-LS	0.706	0.714	0.756	0.740	0.681	0.659	0.550	0.407	0.759	0.544	0.788	0.549	0
AC-GDL	0.017	0.113	0.844	0.933	0.824	0.867	0.351	0.266	0.622	0.430	0.934	0.842	1
AC-PIC	0.017	0.115	0.853	0.920	0.840	0.855	0.415	0.320	0.697	0.472	0.902	0.797	0
SEC	0.779*	0.804*	0.790*	0.815*	0.511*	0.544*	-	-	-	-	-	-	1
LDMGI	0.802*	0.842*	0.811*	0.847*	0.563*	0.580*	-	-	-	-	-	-	1
NMF-D	0.152*	0.175*	0.241*	0.250*	0.287*	0.382*	0.259*	0.274*	0.562*	0.536*	0.920*	0.810*	0
TSC-D	0.651	0.692	-	-	-	-	-	-	-	-	-	-	2
DEC	0.816*	0.844*	0.827*	0.859*	0.586*	0.619*	0.505*	0.378*	0.446*	0.371*	0.924*	0.801*	1
JULE-SF	0.906	0.959	0.876	0.940	0.858	0.922	0.566	0.461	**0.848**	**0.684**	0.984	0.980	3
JULE-RC	0.913	0.964	**0.915**	0.961	0.913	0.950	0.574	0.461	**0.848**	**0.684**	**1.00**	**1.00**	3
DEPICT	**0.917**	**0.965**	**0.915**	**0.963**	**0.927**	**0.964**	**0.610**	**0.470**	0.802	0.621	0.974	0.883	0

2.4.1 Evaluation of Clustering Algorithm

Alternative Models: We compare our clustering model, *DEPICT*, with several baseline and state-of-the-art clustering algorithms, including *K-means*, normalized cuts (*N-Cuts*) [137], self-tuning spectral clustering (*SC-ST*) [188], large-scale spectral clustering (*SC-LS*) [23], graph degree linkage-based agglomerative clustering (*AC-GDL*) [191], agglomerative clustering via path integral (*AC-PIC*) [192], spectral embedded clustering (*SEC*) [110], local discriminant models and global integration (*LDMGI*) [181], *NMF* with deep model (*NMF-D*) [148], task-specific clustering with deep model (*TSC-D*) [160], deep embedded clustering (*DEC*) [174], and joint unsupervised learning (*JULE*) [180].

Implementation Details: We use a common architecture for *DEPICT* and avoid tuning any hyper-parameters using the labeled data in order to provide a practical algorithm for real-world clustering tasks. For all datasets, we consider two convolutional layers followed by a fully connected layer in encoder and decoder pathways. While for all convolutional layers, the feature map size is 50 and the kernel size is about 5×5, the dimension of the embedding subspace is set equal to the number of clusters in each dataset. We also pick the proper stride, padding and crop to have an output size of about 10×10 in the second convolutional layer. Inspired by [120], we consider leaky rectified (leaky RELU) non-linearity [98] as the activation function of convolutional and fully connected layers, except in the last layer of encoder and first layer of decoder, which have Tanh non-linearity functions. Consequently, we normalize the image intensities to be in the range of $[-1, 1]$. Moreover, we set the learning rate and dropout to 10^{-4} and 0.1 respectively, adopt adam as our optimization method with the default hyper-parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-08$ [74]. The weights of convolutional and fully connected layers are all initialized by Xavier approach [46]. Since the clustering assignments in the first iterations are random and not reliable for clustering loss, we first train *DEPICT* without clustering loss function for a while, then initialize the clustering assignment q_{ik} by clustering the embedding subspace features via simple algorithms like K-means or *AC-PIC*. More details about architecture of our networks are represented in Appendix A.1.

Quantitative Comparison: We run *DEPICT* and other clustering methods on each dataset. We followed the implementation details for *DEPICT* and report the average results from 5 runs. For the rest, we present the best reported results either from their original papers or from [180]. For unreported results on specific datasets, we run the released code with hyper-parameters mentioned in the original papers, these results are marked by ($*$) on top. But, when the code is not publicly available, or running the released code is not practical, we put dash marks (-) instead of the corresponding results. Moreover, we mention the number of hyper-parameters that are tuned using supervisory signals (labeled data) for each algorithm. Note that this number only shows the quantity of hyper-parameters, which are set differently for various datasets for better performance.

Table 2 reports the clustering metrics, normalized mutual information (NMI) and accuracy (ACC), of the algorithms on the aforementioned datasets. As shown, *DEPICT* outperforms other algorithms on four datasets and achieves competitive results on the remaining two. It should be noted that we think hyper-parameter tuning using supervisory signals is not feasible in real-world clustering tasks, and hence *DEPICT* is a significantly better clustering algorithm compared to the alternative models in practice. For example, *DEC*, *SEC*, and *LDMGI* report their best results by tuning one hyper-parameter over nine different options, and *JULE-SF* and *JULE-RC* achieve their good performance by tweaking several hyper-parameters over various datasets. However, we do not tune any hyper-parameters for *DEPICT* using the labeled data and only report the result with the same (default) hyper-parameters for all datasets.

2.4.2 Running Time Comparison

In order to evaluate the efficiency of our clustering algorithm in dealing with large-scale and high dimensional data, we compare the running speed of *DEPICT* with its competing algorithms, *JULE-SF* and *JULE-RC*. Moreover, the fast versions of *JULE-SF* and *JULE-RC* are also evaluated. Note that JULE-SF(fast) and *JULE-RC*(fast) both require tuning one extra hyper-parameter for each dataset to achieve results similar to the original *JULE* algorithms in Table 2 [180]. We run *DEPICT* and the released code for *JULE* algorithms[2]

[2]`https://github.com/jwyang/JULE-Torch`

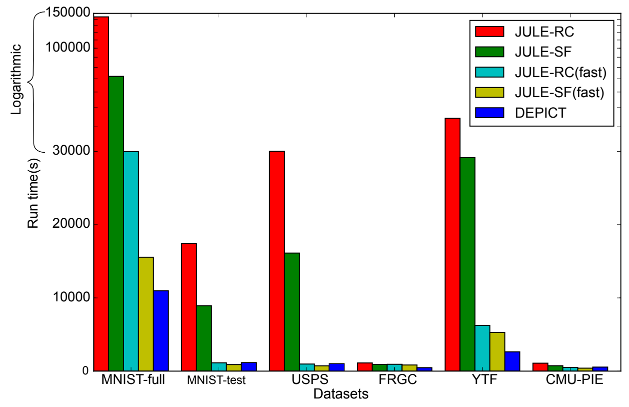


Figure 3: Running time comparison of *DEPICT* and *JULE* clustering algorithms on image datasets.

on a machine with one Titan X pascal GPU and a Xeon E5-2699 CPU.

Figure 3 illustrates the running time for *DEPICT* and *JULE* algorithms on all datasets. Note that running times of *JULE-SF* and *JULE-RC* are shown linearly from 0 to 30,000 and logarithmically for larger values for the sake of readability. In total, *JULE-RC*, *JULE-SF*, *JULE-RC*(fast), *JULE-SF*(fast) and *DEPICT* take 66.1, 35.5, 11.0, 6.6 and 4.7 hours respectively to run over all datasets. While all algorithms have approximately similar running times on small datasets (*FRGC* and *CMU-PIE*), when dealing with the large-scale and high-dimensional datasets (*MNIST-full* and *YTF*), *DEPICT* almost shows a linear increase in the running time, but the running times of original JULE algorithms dramatically grow with the size and number of input data. This outcome again emphasizes the practicality of *DEPICT* for real-world clustering tasks.

2.4.3 Evaluation of Learning Approach

In order to evaluate our joint learning approach, we compare several strategies for training *DEPICT*. For training a multi-layer convolutional autoencoder, we analyze the following three approaches : 1) Standard stacked denoising autoencoder (SdA), in which the model is first pretrained using the reconstruction loss function in a layer-wise manner, and the

Table 3: Clustering performance of different learning approaches, including SdA, RdA and MdA, for training *DEPICT* and *Deep-ConvAE+AC-PIC* models.

Dataset		*MNIST-full*		*MNIST-test*		*USPS*		*FRGC*		*YTF*		*CMU-PIE*	
		NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
Deep-ConvAE	SdA	0.255	0.348	0.313	0.345	0.223	0.290	0.120	0.230	0.414	0.302	0.354	0.266
+	RdA	0.615	0.455	0.859	0.900	0.886	0.866	0.443	0.363	0.597	0.425	0.912	0.817
AC-PIC	MdA	0.729	0.506	0.876	0.942	0.906	0.878	0.583	0.427	0.640	0.448	0.931	0.883
	SdA	0.365	0.427	0.353	0.390	0.328	0.412	0.211	0.300	0.414	0.302	0.354	0.266
DEPICT	RdA	0.808	0.677	0.899	0.950	0.901	0.923	0.551	0.444	0.652	0.450	0.951	0.926
	MdA	**0.917**	**0.965**	**0.915**	**0.963**	**0.927**	**0.964**	**0.610**	**0.470**	**0.802**	**0.621**	**0.974**	**0.883**

encoder pathway is then fine-tuned using the clustering objective function [152]. 2) Another approach (RdA) is suggested in [174] to improve the SdA learning approach, in which all of the autoencoder layers are retrained after the pretraining step, only using the reconstruction of input layer while data is not corrupted by noise. The fine-tuning step is also done after the retraining step. 3) Our learning approach (MdA), in which the whole model is trained simultaneously using the joint reconstruction loss functions from all layers along with the clustering objective function.

Furthermore, we also examine the effect of clustering loss (through error back-prop) in constructing the embedding subspace. To do so, we train a similar multi-layer convolutional autoencoder (*Deep-ConvAE*) only using the reconstruction loss function to generate the embedding subspace. Then, we run the best shallow clustering algorithm (*AC-PIC*) on the embedded data. Hence, this model (*Deep-ConvAE+AC-PIC*) differs from *DEPICT* in the sense that its embedding subspace is only constructed using the reconstruction loss and does not involve the clustering loss.

Table 8 indicates the results of *DEPICT* and *Deep-ConvAE+AC-PIC* when using the different learning approaches. As expected, *DEPICT* trained by our joint learning approach (MdA) consistently outperforms the other alternatives on all datasets. Interestingly, MdA learning approach shows promising results for *Deep-ConvAE+AC-PIC* model, where only reconstruction losses are used to train the embedding subspace. Thus, our learning approach is an efficient strategy for training autoencoder models due to its superior results and fast end-to-end training.

2.4.4 Semi-Supervised Classification Performance

Representation learning in an unsupervised manner or using a small number of labeled data has recently attracted great attention. Due to the potential of our model in learning a discriminative embedding subspace, we evaluate *DEPICT* in a semi-supervised classification task. Following the semi-supervised experiment settings [122, 194], we train our model using a small random subset of *MNIST-training* dataset as labeled data and the remaining as unlabeled data. The classification error of *DEPICT* is then computed using the *MNIST-test* dataset, which is not seen during training. Compared to our unsupervised learning approach, we only utilize the clusters corresponding to each labeled data in training process. In particular, only for labeled data, the cluster labels (assignments) are set using the best map technique from the original classification labels once, and then they will be fixed during the training step.

Table 4 shows the error results for several semi-supervised classification models using different numbers of labeled data. Surprisingly, *DEPICT* achieves comparable results with the state-of-the-art, despite the fact that the semi-supervised classification models use 10,000 validation data to tune their hyper-parameters, *DEPICT* only employs the labeled training data (e.g. 100) and does not tune any hyper-parameters. Although *DEPICT* is not mainly designed for classification tasks, it outperforms several models including *SWWAE* [194], *M1+M2* [75], and *AtlasRBF* [118], and has comparable results with the complicated *Ladder* network [122]. These results further confirm the discriminative quality of the embedding features of *DEPICT*.

2.5 Conclusion

In this paper, we proposed a new deep clustering model, *DEPICT*, consisting of a soft-max layer stacked on top of a multi-layer convolutional autoencoder. We employed a reg-

Table 4: Comparison of *DEPICT* and several semi-supervised classification models in *MNIST* dataset with different numbers of labeled data.

Model	100	1000	3000
T-SVM [149]	16.81	5.38	3.45
CAE [125]	13.47	4.77	3.22
MTC [124]	12.03	3.64	2.57
PL-DAE [84]	10.49	3.46	2.69
AtlasRBF [118]	8.10	3.68	-
M1+M2 [75]	3.33±0.14	2.40±0.05	2.18±0.04
SWWAE [194]	8.71±0.34	2.83±0.10	2.10±0.22
Ladder [122]	**1.06±0.37**	**0.84±0.08**	-
DEPICT	2.65±0.35	2.10±0.11	**1.91±0.06**

ularized relative entropy loss function for clustering, which leads to balanced cluster assignments. Adopting our autoencoder reconstruction loss function enhanced the embedding learning. Furthermore, a joint learning framework was introduced to train all network layers simultaneously and avoid layer-wise pretraining. Experimental results showed that *DEPICT* is a good candidate for real-world clustering tasks, since it achieved superior or competitive results compared to alternative methods while having faster running speed and not needing hyper-parameter tuning. Efficiency of our joint learning approach was also confirmed in clustering and semi-supervised classification tasks.

3.0 Sentiment Analysis via Deep Hybrid Textual-Crowd Learning Model

3.1 Introduction

Recently rapidly growing use of social media has provided a huge source of public opinions about different topics. Efficient mining of these opinions is very valuable for various industries and businesses. For instance, hotels, airlines, lenders, banks and even politicians utilize these data to find new costumers, target new products, analyze the personality of clients and make better decisions. However, exploring the sentiment of public opinions is a very challenging task for automatic language models due to different variations in the texts, such as diverse contexts, genders of authors, writing styles and varied viewpoints.

Crowdsourcing platforms like Amazon Mechanical Turk[1] provide an efficient tool to solve this type of the problems by using the knowledge of crowd workers in different tasks at low cost and time. Hence, the human skills in language understanding can be used to interpret the sentiments of texts with different variations. However, the collected labels via crowdsourcing are often noisy and inaccurate, because crowd workers are usually inexpert in the assigned task. In order to address this issue, it is common to collect multiple crowd labels for each sample to increase the credibility of the estimated true labels. Several studies have proposed different models to aggregate the crowd labels and estimate the potential true labels, which are also called truths [27, 22, 165, 196, 43]. *But these crowdsourcing aggregation models become drastically incompetent, when the number of crowd labels per worker is not enough to train the reliability parameters of workers, or a document dataset is extremely large that collecting crowd labels for all samples is not practically feasible. In addition, crowdsourcing aggregation models do not utilize text data, and only use crowd labels as the source of information (i.e. input data).*

In this project, we propose a new hybrid model for sentiment analysis, which utilizes both crowd labels and text data. In particular, our proposed model, called *CrowdDeepAE*, consists of a generative aggregation model for crowd labels and a deep autoencoder for text

[1]https://www.mturk.com/

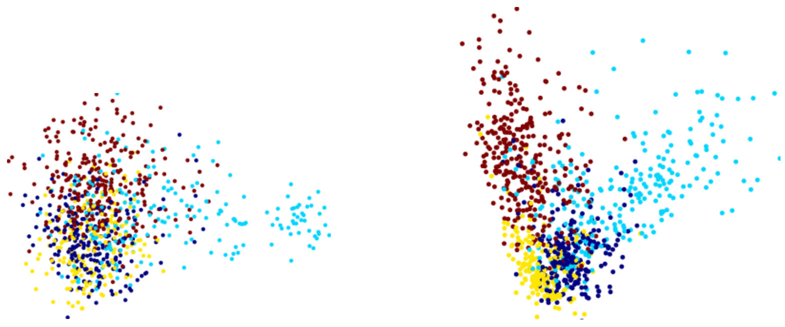

(a) *MV-DeepAE* (b) *CrowdDeepAE*

Figure 4: 2D visualization of *CrowdDeepAE* (ours) and *MV-DeepAE* features on *CrowdFlower* dataset using PCA, when only 20% of the crowd data is available.

data. These two sub-models are coupled in a probabilistic framework rather than a heuristic approach. Using this probabilistic framework, we introduce a unified objective function that incorporates the interests of both sub-models. We further derive an efficient optimization algorithm to solve the corresponding problem via an alternating approach, in which the parameters are updated while the truths are assumed to be known, and the truths are estimated when the parameters are fixed.

Therefore, CrowdDeepAE exploits the intelligence of crowd workers and the underlying informations of text data to categorize sentiments more accurately. To do so, it employs a non-linear generative aggregation model to flexibly aggregate noisy crowd labels, and leverages a deep denoising autoencoder to learn a discriminative embedding for text data. In particular, the deep autoencoder uses text data to find similar patterns between text samples and prevent the crowd aggregation model from overfitting, and the crowd aggregation model utilizes human language skills to assist the autoencoder in differentiating the samples with large (semantic) variations.

Experimental results indicate that our model achieves superior or competitive results compared to the state-of-the-art models on two large text-crowd datasets. Specifically, CrowdDeepAE outperforms the alternative models with significant margins when the crowd labels are scarce. Figure 4 visualizes the discriminative ability of our model (*CrowdDeepAE*) compared to an alternative hybrid model (*MV-DeepAE*), when only 20% of the crowd labels are available on *CrowdFlower* dataset. *MV-DeepAE* contains majority voting aggregation method (*MV*) and our autoencoder sub-model (*DeepAE*). The outcome demonstrates more discriminative features using our model, indicating the importance of our joint learning framework. The contribution of this project can be summarized as follows:

- Proposing a hybrid crowd-text model for sentiment analysis, consisting of a generative crowd aggregation model and a deep sentimental autoencoder, which are combined based on a probabilistic framework;
- Defining a unified objective function for the hybrid model, and deriving an efficient optimization algorithm to solve the problem;
- Achieving superior or competitive results compared to alternative models in our experiments, especially when the crowd labels are scarce.

3.2 Related Works

There are several datasets in different applications, which are labeled using crowdsourcing platforms like Amazon Mechanical Turk [4, 166, 130, 129]. However, crowd labels are often noisy and unreliable, since crowd workers mostly lack expertise in the assigned tasks. To tackle this issue, each sample is usually labeled by multiple crowd workers, then these redundant crowd labels are used to estimate the potential true labels (*i.e.* truths). There are several studies, which proposed discriminative and generative models to efficiently aggregate crowd labels [136, 195]. The discriminative aggregation models directly estimate the truths regardless of the crowd data distribution. Majority voting (*MV*) is the simplest discriminative aggregation model, which considers equal reliability for crowd workers and simply averages their votes. An intuitive and fast extension of majority voting, called iterative weighted majority voting (*IWMV*), is introduced in [88], which improves *MV* by considering a reliability parameter for each worker. Tian and Zhu also enhanced *MV* model by adopting the notion of max-margin from support vector machines, and introduced max-margin mjority voting (M^3V) as a new discriminative aggregation models [147].

In contrast to the discriminative aggregation models, the generative models employ a probabilistic model to represent the distribution of noisy observations (crowd labels) given the unknown variables (true labels) and model parameters. Dawid and Skene introduced a well known model (DS), which considers a confusion matrix as a reliability parameter for each worker in [27]. Furthermore, several studies extended DS by assuming a prior distribution for parameters, and used Bayesian approach to compute their posterior distributions [123, 22]. Another generative model, called $GLAD$, considers a scalar parameter for the reliability of each worker and the difficulty of each task, and calculates the probability of truths using the logistic function of the parameters [165]. Moreover, $GLAD$ is extended in [164] such that a vector instead of a scalar is considered as the parameter of each worker and sample. In addition to a confusion matrix as the reliability parameter of each worker, Zhou *et al.* assigned a confusion matrix as a difficulty parameter for each sample, and proposed an aggregation model based on minimax conditional entropy of crowd labels [196, 197]. Later, Tian and Zhu regularized a variant of DS with the discriminative M^3V model, and jointly learned the parameters of both sub-models [147]. In order to tackle the aggregation problem when crowd labels per worker are scarce, Venanzi *et al.* proposed CommunityBCC, which groups crowd workers into a few types (communities) and learns similar reliability parameters for each community [150].

The aforementioned aggregation models only use crowd labels to estimate the truths, but do not benefit from text data. There are a few studies on sentiment analysis using both crowd and text data [14, 102, 141]. In a recent work [141], a Bayesian model is employed to combine the two modalities by considering a confusion matrix for each word and worker. Our proposed model also utilizes both crowd labels and text data; however, it leverages the power of deep models to provide a more discriminative language model, despite the shallow *BCCwords* model in [141]. Our model is also unique in the way that it combines a generative crowd aggregation model with a deep sentimental autoencoder using a probabilistic framework. Moreover, experimental results show the superiority of our model compared to *BCCwords*, especially when crowd labels are scarce.

3.3 Hybrid Sentiment Analysis Model

In this section, we first introduce our hybrid model by showing its architecture and explaining the intuition behind it. We then formulate its unified objective function based on a probabilistic framework, and derive an optimization algorithm for updating parameters and estimating truths.

3.3.1 CrowdDeepAE Architecture

The proposed hybrid model, denoted by *CrowdDeepAE*, consists of two main parts, a deep denoising autoencoder for text data and an aggregation model for crowd labels. Figure 5 demonstrates the architecture of *CrowdDeepAE*, in which the deep denoising autoencoder has two tasks, reconstructing the corrupted text data by noise and estimating the truths from text data. The crowdsourcing aggregation model is also supposed to estimate the truths from the noisy crowd labels. Hence, the truths are obtained by contributions of both crowd and text data.

Coupling the deep autoencoder and crowd aggregation model in *CrowdDeepAE* has several advantages:

- *CrowdDeepAE* exploits two sources of information to estimate the truths more accurately, text data via the encoder pathway of the autoencoder and crowd data through the aggregation model;
- The multi-layer autoencoder provides powerful discriminative features for the text samples, which have more capabilities than shallow models in learning the non-linear embedding space of real-world text data;
- The reconstruction loss function in the denoising autoencoder plays a role of data-dependent regularization term, indirectly preventing the crowdsourcing aggregation model from overfitting;
- *CrowdDeepAE* is able to annotate the entire dataset, even the samples without any crowd labels, since the autoencoder can be efficiently trained using the supervision of limited number of crowd labels and the unsupervised reconstruction task;

- The aggregation model assists training the autoencoder using the semantic knowledge of crowd workers, which is very beneficial due to the large variations on text data;
- The joint learning framework used for *CrowdDeepAE* leads to more optimal results compared to a naive non-joint learning approach, where the textual and crowd sub-models are trained separately.

3.3.2 CrowdDeepAE Objective Function

Lets consider the crowdsourcing task includes N questions, each with K possible options. The crowd and text data are represented by $\mathbf{X} = \{\mathbf{X}^{Cr}, \mathbf{X}^{Te}\}$, respectively, and $\mathbf{Y}$ indicates the unknown true labels. We provide a probabilistic framework to combine our autoencoder and aggregation sub-models, and consequently define a unified objective function for our hybrid model. The general likelihood function of *CrowdDeepAE* parameters ($\boldsymbol{\psi}$) given the observations ($\mathbf{X}^{Cr}, \mathbf{X}^{Te}$) is:

$$\begin{aligned} P(\mathbf{X}^{Cr},&\mathbf{X}^{Te}|\boldsymbol{\psi}) = \prod_{i=1}^{N} P(\mathbf{X}_i^{Cr}, \mathbf{X}_i^{Te}|\boldsymbol{\psi}) \qquad (3.1) \\ &= \prod_{i=1}^{N}\sum_{c=1}^{K} P(\mathbf{X}_i^{Cr}, \mathbf{X}_i^{Te}, Y_i = c|\boldsymbol{\psi}) \\ &= \prod_{i=1}^{N}\sum_{c=1}^{K} P(\mathbf{X}_i^{Cr}, \mathbf{X}_i^{Te}|Y_i = c, \boldsymbol{\psi})P(Y_i = c|\boldsymbol{\psi}) \\ &= \prod_{i=1}^{N}\sum_{c=1}^{K} \underbrace{P(\mathbf{X}_i^{Cr}|Y_i = c, \boldsymbol{\theta})}_{\text{Crowd Aggregation Model}} \underbrace{P(Y_i = c|\mathbf{X}_i^{Te}, \mathbf{W})P(\mathbf{X}_i^{Te}|\mathbf{W})}_{\text{Deep Autoencoder}}, \end{aligned}$$

where i and c are the indices of questions and options, and $\mathbf{W}$ and $\boldsymbol{\theta}$ represent the parameters of autoencoder and crowd aggregation sub-models, respectively. Note that the samples are assumed independent and identically distributed (i.i.d), and $\mathbf{X}_i^{Cr}$ and $\mathbf{X}_i^{Te}$ are supposed to be conditionally independent given the true labels.

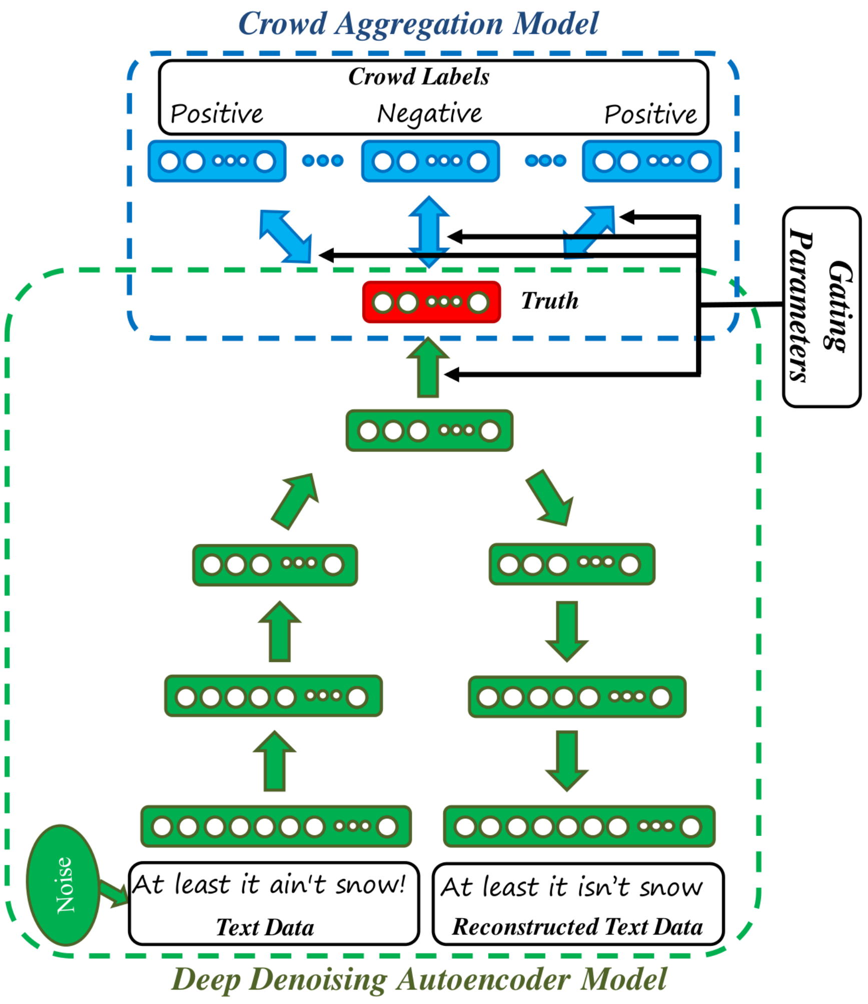


Figure 5: *CrowdDeepAE* architecture, consisting of a deep denoising autoencoder and a crowd aggregation model.

We are now able to decompose the likelihood function in Eq. (C.2) into the crowd aggregation and deep autoencoder objectives. Considering that M crowd workers are hired in the crowdsourcing task, our generative crowd aggregation model has the following form.

$$P(\mathbf{X}_i^{Cr}|Y_i = c, \boldsymbol{\theta}) = \prod_{j=1}^{M}\prod_{k=1}^{K}\Big[\frac{exp(\theta_{jck})}{\sum\limits_{k'} exp(\theta_{jck'})}\Big]^{\mathbf{1}(x_{ij}^{Cr}=k)} = \prod_{j=1}^{M}\prod_{k=1}^{K}[p_{ijck}]^{\mathbf{1}(x_{ij}^{Cr}=k)}, \qquad (3.2)$$

where $\mathbf{X}_i^{Cr} = \{\mathbf{x}_{i1}^{Cr}, ..., \mathbf{x}_{iM}^{Cr}\}$ is the set of crowd labels for the i-th question. Also p_{ijck} shows the probability of a crowd label such that the j-th worker selects the k-th option for the i-th question, when c is the true label. Therefore, the joint probability of crowd data for each question is based on the probability of each conditionally independent crowd label. The aggregation model considers a confusion matrix $\boldsymbol{\theta}_j$ as the reliability parameter of each worker, in which higher diagonal elements θ_{jkk} indicate more reliability for the worker. Moreover, the exponential non-linearity increases the flexibility of our crowdsourcing aggregation model in dealing with the noisy crowd labels.

The direct optimization of log-likelihood function $\mathscr{L}(\boldsymbol{\psi}|\mathbf{X}) = \log P(\mathbf{X}^{Cr}, \mathbf{X}^{Te}|\boldsymbol{\psi})$ is difficult, hence we use Expectation-Maximization (EM) learning approach to solve this problem. Following, we present Proposition 1 to alleviate the optimization problem, and provide its proof in Appendix B.1. For the sake of simpler notations, hereafter we denote $P(Y_i = c|\mathbf{X}_i^{Te}, \mathbf{W}) = e_{ic}$ and $P(\mathbf{X}_i^{Te}|\mathbf{W}) = d_i$ as the probability of $\underline{e}$ncoder and $\underline{d}$ecoder pathways, respectively, and $\mathbf{1}(x_{ij}^{Cr} = k) = \mathbf{1}_{ijk}$.

Proposition 1: Iteratively improving the following auxiliary function $\mathbb{Q}$ is sufficient to maximize the log-likelihood function $\mathscr{L}(\boldsymbol{\psi}|\mathbf{X})$.

$$\mathbb{Q}(\boldsymbol{\psi}|\boldsymbol{\psi}^{(t)}) = \sum_{ijck} q_{ic}^{(t)} \log\Big(d_i e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}\Big) where\ q_{ic}^{(t)} = \frac{\prod\limits_{jk} e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}}{\sum\limits_{c'}\prod\limits_{jk} e_{ic'}[p_{ijc'k}]^{\mathbf{1}_{ijk}}} \qquad (3.3)$$

where $q_{ic}^{(t)}$ shows the probability distribution of a truth. Technically, it is the expectation of an unknown true label with respect to the current parameters. Thus, we can iteratively improve the auxiliary function $\mathbb{Q}$ instead of the log-likelihood function $\mathscr{L}$.

In the $\mathbb{Q}$ function, the autoencoder and crowd workers have similar effects in the objective function and calculating the truths. However, it is expected that the deep autoencoder has more accurate predictions than the inexpert crowd workers due to the learned knowledge from all of questions. Hence, we control the influence of each factor in our objective function using adjustable weights. Following, we present the updated objective function, which can be derived similar to proposition 1.

$$\max_{\boldsymbol{\theta}, \mathbf{W}, \mathbf{1}^T\boldsymbol{\alpha}=M+1, \boldsymbol{\alpha}\geq 0} \sum_{ijck} q_{ic}^{(t)} \log \left([d_i]^{\lambda_d} [e_{ic}]^{\alpha_0} [p_{ijck}]^{\alpha_j \mathbf{1}_{ijk}} \right)$$
$$where \ q_{ic}^{(t)} \propto \prod_{jk} (e_{ic})^{\alpha_0} (p_{ijck})^{\alpha_j \mathbf{1}_{ijk}} \tag{3.4}$$

where $\boldsymbol{\alpha}$ and λ_d are the adjustable weights and the hyperparameter for the reconstruction loss of autoencoder, respectively. Note that $\boldsymbol{\alpha}$ can be seen as the gating parameters (see Figure 5), which adjust the contribution of each worker and also the autoencoder in estimating the truths. In other words, $\boldsymbol{\alpha}$ gives one more degree of freedom to our hybrid model about the credibility of crowd workers and autoencoder. For example, when there are several (non-expert) crowd workers labeling a question with (very noisy) crowd labels, a high weight for (discriminative) autoencoder can help estimating the truth accurately. Note that we define the weight for probability of decoder pathway by λ_d, since d_i does not affect the truths, and only regulates the autoencoder objective function. Furthermore, we add two more regularization penalty terms for the parameters to avoid overfitting.

$$\min_{\boldsymbol{\theta}, \mathbf{W}, \mathbf{1}^T\boldsymbol{\alpha}=M+1, \boldsymbol{\alpha}\geq 0} -\sum_{ijck} q_{ic}^{(t)} \log \left([d_i]^{\lambda_d} [e_{ic}]^{\alpha_0} [p_{ijck}]^{\alpha_j \mathbf{1}_{ijk}} \right) + \lambda_\theta \sum_j \|\boldsymbol{\theta}_j\|_F + \lambda_\alpha \|\boldsymbol{\alpha}\|_2 \,, \tag{3.5}$$

where λ_θ and λ_α are the hyperparameters of regularization terms. Also adding two constraints for $\boldsymbol{\alpha}$ (under min operation) is beneficial in our objective function for having competitive learning and avoiding the trivial solution $\boldsymbol{\alpha} = \mathbf{0}$.

3.3.3 CrowdDeepAE Optimization Algorithm

In order to efficiently solve problem (3.5), we employ an alternating learning strategy to update the parameters and estimate the truths. In particular, each one of the parameters $\boldsymbol{\psi} = \{\boldsymbol{\theta}, \boldsymbol{\alpha}, \mathbf{W}\}$ is updated while the other parameters and truths are fixed, and the probability of truths $\mathbf{Q} = \{\mathbf{q}_1, ..., \mathbf{q}_N\}$ are estimated when the parameters are assumed to be known.

Update $\boldsymbol{\theta}$: The problem for updating the parameters of crowd aggregation model is reduced to:

$$\min_{\boldsymbol{\theta}} \; -\sum_{ijck} q_{ic}^{(t)} \log \left([p_{ijck}]^{\alpha_j \mathbf{1}_{ijk}} \right) + \lambda_\theta \sum_j \|\boldsymbol{\theta}_j\|_F \tag{3.6}$$

There are several first-order optimization algorithms that can be used to solve this problem. Using the following gradient of the objective function *wrt* the parameter $\boldsymbol{\theta}$, we employ L-BFGS algorithm to iteratively update the parameters.

$$\frac{\partial \mathbb{Q}}{\partial \theta_{jck}} = \sum_i q_{ic}^{(t)} \alpha_j [\mathbf{1}_{ijk} - p_{ijck}] \tag{3.7}$$

Update $\boldsymbol{\alpha}$: The problem to update the gating parameters boils down to:

$$\min_{\mathbf{1}^T\boldsymbol{\alpha} = M+1, \boldsymbol{\alpha} \geq 0} \; \lambda_\alpha \boldsymbol{\alpha}^T \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \boldsymbol{\beta} \tag{3.8}$$

where $\beta_0 = \sum_{ic} q_{ic}^{(t)} \log e_{ic}$, $\beta_j = \sum_{ick} q_{ic}^{(t)} \mathbf{1}_{ijk} \log p_{ijck}$. We efficiently solve this problem using the Lagrangian multiplier method as shown in Appendix B.2.

Update $\mathbf{W}$: The problem to update the parameters of deep denoising autoencoder has the following form.

$$\min_W \; -\sum_{ic} q_{ic}^{(t)} \log P_W(Y_i = c|\mathbf{X}_i^{Te}) - \frac{\lambda_d}{\alpha_0} \log P_W(\mathbf{X}_i^{Te}) \,,$$

where the first term is the standard cross entropy loss function for classification problems. But for the second probability term, we use a theorem in [9] in order to change the term to reconstruction loss function in the standard denoising autoencoder.

The general idea is that if the observation variable X is corrupted into $\tilde{X}$ by a noise with conditional distribution $\mathscr{C}(\tilde{X}|X)$, training a denoising autoencoder actually estimates

Algorithm 2: *CrowdDeepAE* Algorithm

```
1 Initialize $\mathbf{q}_i$ by majority voting  $\forall i \in \{1, ..., N\}$
2 while not converged do
3     Solve problem (3.6) to update $\boldsymbol{\theta}$
4     Solve problem ( B.8) to update $\boldsymbol{\alpha}$
5     Solve problem (3.9) to update $\mathbf{W}$
6     $q_{ic} \propto \prod_{jk} (e_{ic})^{\alpha_0} (p_{ijck})^{\alpha_j \mathbf{1}_{ijk}}$
7 end
```

the reverse conditional distribution $P(X|\tilde{X})$. It has been shown that a consistent estimator of $P(X)$ can be estimated using a Markov chain that alternates between sampling from $P(X|\tilde{X})$ and sampling from $\mathscr{C}(\tilde{X}|X)$ as follows.

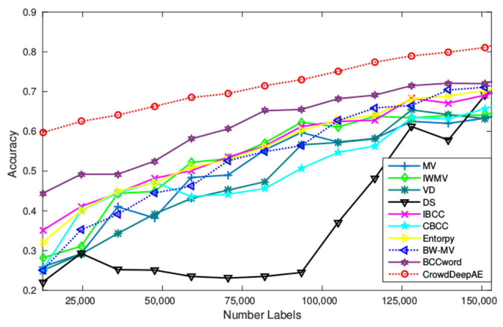
$$X_t \sim P_W(X|\tilde{X}_{t-1}) \qquad \tilde{X}_t \sim \mathscr{C}(\tilde{X}|X_t)$$

The theorem proves that $P_W(X|\tilde{X})$ of conventional denoising autoencoder [151, 9, 41] is a consistent estimator of the true conditional distribution. Also as the number of samples $N \to \infty$, the asymptotic distribution of the generated samples by the denoising autoencoder converges to original data-generating distribution. Hence, we reformulate the objective function of the text model as follows:

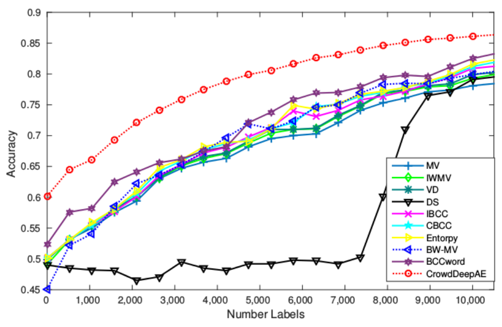
$$\min_{W} \; -\sum_{ic} q_{ic}^{(t)} \log P_W(Y_i = c|\mathbf{X}_i^{Te}) - \frac{\lambda_d}{\alpha_0} \log P_W(\mathbf{X}_i^{Te}|\tilde{\mathbf{X}}_i^{D})\,, \tag{3.9}$$

where $\tilde{\mathbf{X}}_i^D$ is a sample corrupted by a random noise. Now it is clear how we can use the denoising autoencoder as our text-based sub-model.

Interestingly, our learning approach does not have memory exhaustion problems when handling very large datasets. In order to learn the reliability parameters $\boldsymbol{\theta}$ for large number of crowd workers, we can split the crowd data into several mini-batches, each one only including the crowd labels of a few workers. Dealing with a large set of text samples, we are able to distribute the text samples into a set of mini-batches and train the autoencoder parameters with stochastic optimization algorithms. Therefore, the computation and space complexities can be managed using stochastic and parallel learning approaches.



(a) *CrowdFlower* (*CF*)



(b) *SentimentPolarity* (*CF*)

Figure 6: Accuracy of crowdsourcing aggregation models on *CrowdFlower* (*CF*) and *SentimentPolarity* (SP) datasets, when increasing the number of crowd labels.

Algorithm 2 shows the *CrowdDeepAE* algorithm, in which the truths are first initialized by majority voting. It then alternates between updating the model parameters and estimating the truths until convergence. It is worth mentioning that we compute the truths using the clean text samples in E-step. But the classification loss function with noisy text inputs in Eq. (3.9) has the regularization effect in training the parameters $\mathbf{W}$, and results in to the more robust and generalized autoencoder model.

3.4 Experiments and Discussions

In this section, we first evaluate the performance of our hybrid model in the crowd aggregation task, and then examine the quality of the learned language models. In order to compare the proposed model with the state-of-the-art aggregation models, we use two large-scale crowdsourcing datasets, which have text data along with crowd labels for sentiment analysis.

Table 5: Comparison of crowdsourcing aggregation models on *CrowdFlower* (*CF*) and *SentimentPolarity* (*SP*) datasets with 20% of crowd labels. The comparison metrics are accuracy, ave. recall, AUC (the higher the better), and NLPD (the lower the better).

		CF (20% labels)				SP (20% labels)			
	Model	Accuracy	Ave. recall	NLPD	AUC	Accuracy	Ave. recall	NLPD	AUC
Crowd	*MV*	0.625	0.550	1.392	0.725	0.710	0.710	1.192	0.704
	IWMV	0.630	0.562	1.368	0.735	0.710	0.710	1.167	0.715
	VD	0.650	0.585	1.252	0.745	0.710	0.710	1.112	0.728
	DS	0.610	0.488	1.285	0.681	0.500	0.500	0.695	0.500
	IBCC	0.688	0.545	0.972	0.822	0.740	0.740	0.516	0.835
	CBCC	0.635	0.532	1.052	0.800	0.726	0.726	0.540	0.818
	Entropy	0.688	0.545	1.014	0.818	0.745	0.745	0.508	0.842
Crowd-Text	*MV-BW*	0.665	0.602	2.133	0.749	0.722	0.722	0.648	0.784
	MV-DeepAE	0.682	0.611	1.372	0.792	0.738	0.738	0.615	0.800
	BCCwords	0.715	0.578	0.918	0.830	0.750	0.750	0.516	0.840
	CrowdDeepAE	**0.790**	**0.642**	**0.889**	**0.876**	**0.816**	**0.816**	**0.500**	**0.875**

Datasets: *CrowdFlower* (*CF*) dataset was a part of the 2013 Crowdsourcing at Scale shared task challenge, collected by CrowdFlower[2] as a rich source for the sentiment analysis of tweets about the weather. The dataset includes 569,375 crowd labels for 98,980 tweets. But the gold-standard (true) labels are only provided for 300 tweets, which correspond to 1720 crowd labels collected from 461 workers. In the crowd task, workers are requested to label the sentiment of tweets related to weather using the following options, negative (0), neutral (1), positive (2) and not related to weather (4). The crowd workers are also able to skip the questions by the can not tell (5) option.

Sentiment Polarity (*SP*) dataset includes the sentiment analysis of crowd workers about the movie reviews across two categories, "fresh" (positive) and "rotten" (negative). The dataset consists of 5,000 sentences from the movie reviews in RottenTomatoes website[3], which is extracted by [115]. A task requester hired 203 crowd workers to label the dataset, resulting in 27,747 crowd labels totally. The gold-standard labels for all the questions are available in *SP* dataset.

Implementation details: For both *CF* and *SP* datasets, we first use the stemming approach to parse the texts [119], then remove the common English stop words and finally extract the top 1000 words according to the term frequency-inverse document frequency (tf-idf) score [5].

For the deep autoencoder, we consider three fully connected layers for both encoder and decoder pathways with 512, 256, and 128 neurons as the feature maps, and then add a soft-max layer on top of the encoder pathway. The leaky rectified activation (leaky RELU) is used as the activation function for the autoencoder layers, except the reconstruction layer at the end of decoder pathway, which has rectified activation (RELU) to reconstruct text samples. Moreover, we set the learning rate to 10^{-4} and adopt Adam [73] as our optimization method. The weights of all layers are also initialized by the Xavier or GlorotUniform initialization approach [46].

Since the crowdsourcing task is an unsupervised problem, we did not use any true labels for setting the hyper-parameters $\{\lambda_\theta, \lambda_\alpha, \lambda_d\}$ and dropout noise value. We use a trick in [147],

[2]www.crowdflower.com
[3]www.rottentomatoes.com

that employs the non-related likelihood for selecting the hyper-parameters. In particular, we utilize the likelihood function $p(\mathbf{X}^{Cr}|\mathbf{Y}, \boldsymbol{\theta})$ to choose λ_α, λ_d and dropout from $\lambda_\alpha^{set} = \{0.01, 0.1, 1\}$, $\lambda_d^{set} = \{0.01, 0.1, 1\}$ and dropout$^{set} = \{0.1, 0.2, 0.3\}$, and adopt $p(\mathbf{Y}|\mathbf{X}^{Te}, \mathbf{W})$ as a criterion to choose λ_θ from $\lambda_\theta^{set} = \{0.01, 0.1, 1\}$. Thus using this approach, we make sure to select the hyper-parameters without any knowledge from the true labels.

3.4.1 Evaluation of Aggregation Models

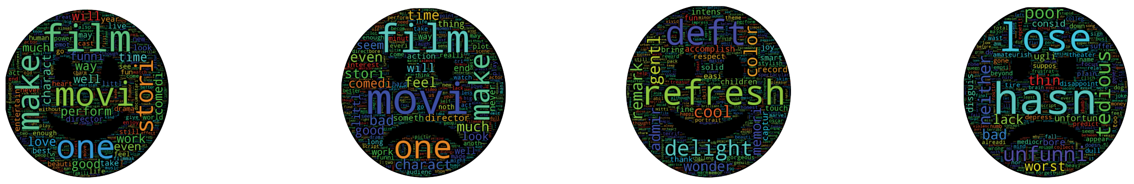
To evaluate the performance of our model, we run several experiments using *CF* and *SP* datasets to estimate the truths using crowd labels and text data. For the sake of comparison, we use the following alternative models and comparison metrics.

Alternative models: We compare our model, *CrowdDeepAE*, with several baseline methods, including majority voting (*MV*), iterative weighted majority voting (*IWMV*) [88], vote distribution (*VD*), Dawid and Skene model (*DS*) [27], Independent Bayesian Classifier Combination model (*IBCC*) [140], Community-Based Bayesian Classifier Combination model (*CBCC*) [150], multi-class minimax entropy model (*Entropy*) [196], combination of majority voting aggregation model and bag-of-words text classifier (*MV-BW*), combination of majority voting aggregation model and a deep sentimental autoencoder similar to our autoencoder (*MV-DeepAE*), and Bayesian classifier combination with words model (*BCC-words*) [141].

It should be noted that *VD* can be considered as a probabilistic version of *MV*, since it computes the probability of each option, while assuming equal reliability for all the workers. Moreover, the *MV-BW* model trains a classical bag-of-words classifier for text data using the target label induced by majority voting aggregation model. Similarly, *MV-DeepAE* uses the predicted labels of majority voting aggregation model to train the deep autoencoder model for text data. The results of alternative models are reported from reference papers, except *MV-DeepAE* that is implemented by us with the similar autoencoder network to *CrowdDeepAE*.

Table 6: Comparison of crowdsourcing aggregation models on *CrowdFlower* (*CF*) and *SentimentPolarity* (*SP*) datasets, when all crowd labels are available. The comparison metrics are accuracy, ave. recall, AUC (the higher the better), and NLPD (the lower the better).

		CF (all labels)				*SP* (all labels)			
	Model	Accuracy	Ave. recall	NLPD	AUC	Accuracy	Ave. recall	NLPD	AUC
Crowd	*MV*	0.840	0.764	0.921	0.852	0.852	0.852	0.797	0.885
	IWMV	0.860	0.764	0.912	0.041	0.885	0.885	0.752	0.891
	VD	0.883	0.779	0.458	0.942	0.887	0.887	**0.338**	0.947
	DS	0.830	0.745	0.459	0.897	0.914	0.914	0.340	**0.957**
	IBCC	0.860	0.763	**0.437**	0.935	**0.915**	**0.915**	0.374	**0.957**
	CBCC	0.886	0.746	0.526	0.942	**0.915**	**0.915**	0.383	**0.957**
	Entropy	0.886	0.746	0.551	0.938	0.914	0.914	0.391	**0.957**
Crowd-Text	*MV-BW*	0.867	0.764	0.921	0.859	0.885	0.885	0.797	0.891
	MV-DeepAE	0.880	0.768	0.571	0.922	0.885	0.885	0.752	0.891
	BCCwords	0.890	0.807	0.591	0.877	**0.915**	**0.915**	0.389	**0.957**
	CrowdDeepAE	**0.912**	**0.825**	0.479	**0.948**	**0.915**	**0.915**	0.389	**0.957**



(a) Pos-docStatistic (b) Neg-docStatistic (c) Pos-CrowdDeepAE (d) Neg-CrowdDeepAE

Figure 7: Word clouds of the positive (Pos) and negative (Neg) sentiments in SP dataset. The extracted word clouds using the statistics of documents (docStatistic) and our language model (*CrowdDeepAE*) are shown in the left and right, respectively. The colors are only for legibility.

Comparison metrics: Following [141], we measure the performance of models using *accuracy*, *average recall*, *negative log-probability density (NLPD)* [150], and *area under curve (AUC)* [140]. For *CF* dataset, we use mean AUC over pair of classes as shown in [53].

Performance comparison: In order to examine the effectiveness of the aforementioned aggregation models, we run several experiments with different subsets (number of crowd labels) of CF and *SP* datasets. Following [140], we estimate the truths using the aggregation models when only 2% randomly-chosen crowd labels are available. Then, we increase the number of crowd labels by adding an extra 2% randomly-chosen crowd labels, and rerun all the models. This process is repeated until all of the crowd labels are used for training.

Figure 6 shows the accuracy of aggregation models on both *CF* and *SP* datasets. As it is shown, *CrowdDeepAE* consistently outperforms the other models with significant margins, especially when a small number of crowd labels are available. Interestingly in *CF* dataset, our model only requires 16% of the crowd labels to have a better accuracy than the all of other models, which are using 30% of the crowd labels. *CrowdDeepAE* also achieves a higher accuracy with 8% of the crowd labels in *CF* dataset versus *MV* model with 30% of the crowd labels. Furthermore, *CrowdDeepAE* consistently improves the performance of *MV-DeepAE*, and consequently confirms the importance of our joint learning framework and our crowd aggregation sub-model. Note that we only show a limited portion of the results (approximately 150,000 and 10,000 crowd labels in *CF* and *SP* datasets) in Figure 6 for the sake of a clear visualization.

Furthermore, Table 5 and 6 report the mentioned comparison metrics for the aggregation models on *CF* and *SP* datasets, when 20% and 100% of crowd labels are available, respectively. We divide the models in the tables into two groups of single and hybrid models, where the first ones only employ crowd labels to estimate the truths, and the second ones utilize both crowd labels and text data for the prediction task. Using only 20% of crowd labels, approximately 70% of the text samples have at least one crowd label. In this case, using text data is more crucial, since enough crowd labels are not available for training the crowd parameters. The hybrid crowd-text models have relatively better performances than the crowd models, because the hybrid models are able to employ language model to classify the samples with no crowd labels. But the crowd models suffer from insufficient crowd labels for training, and assign a default category for the unlabeled samples based on their prior distribution. Our proposed model, *CrowdDeepAE*, benefits from the deep autoencoder trained by a small subset of crowd data, and is able to efficiently label the samples with no crowd labels. When only 20% of crowd labels are available, our model outperforms the alternative models on both *SP* and *CF* datasets according to all metrics. In addition, *CrowdDeepAE* still achieves superior or competitive results in comparison with the state-of-the-art models on both datasets using all crowd labels. It indicates that *CrowdDeepAE* leverages the powerful deep language model along with the efficient crowd aggregation model to provide accurate predictions using crowd and text data.

3.4.2 Evaluation of Language Models

In order to visualize the learned language model in *CrowdDeepAE*, we show the word clouds for both *CF* and *SP* datasets. In particular, the word cloud represents the importance (probability) of each word in a document with its font size. Using this visual representation, a viewer can quickly identify the dominant words in a document using their relative sizes. For each word in the datasets, we generate an auxiliary variable by setting the corresponding element in $\mathbf{X}_i^{Te}$ equal to 1 and the remaining ones to zero, and then compute the probability of the word for every class. Figure 7 demonstrates the word clouds of *CrowdDeepAE* in *CF* dataset for the positive and negative classes. We also show the word cloud of *CF* dataset using the probability (frequency) of each word in every sentiment class. The world clouds extracted from the documents statistic (docStatistic) mostly assign greater importance to the highly repeated words like "movie", "film", and "stori", which do not differentiate the two classes. However, the word clouds of *CrowdDeepAE* discriminantly represent the positive sentiments using the words with roots like "refresh", "deft", "delight" and "gentl"; and the negative class with the words like "lose", "hasn", "tedious", and "unfunni". The word clouds for *CF* dataset are shown in Appendix B.3.

3.5 Conclusion

In this project, we proposed a new crowdsourcing aggregation model that is augmented by a deep sentimental autoencoder. The crowd aggregation and autoencoder sub-models are combined in a probabilistic framework rather than a heuristic way. We introduced a unified objective function, and then derived an efficient optimization algorithm to alternatingly solve the corresponding problem. Experimental results showed that our model outperforms the alternative models, especially when the crowd labels are scarce. Although the proposed model was applied only in sentiment analysis, it can be used as the general hybrid model for different applications in future works.

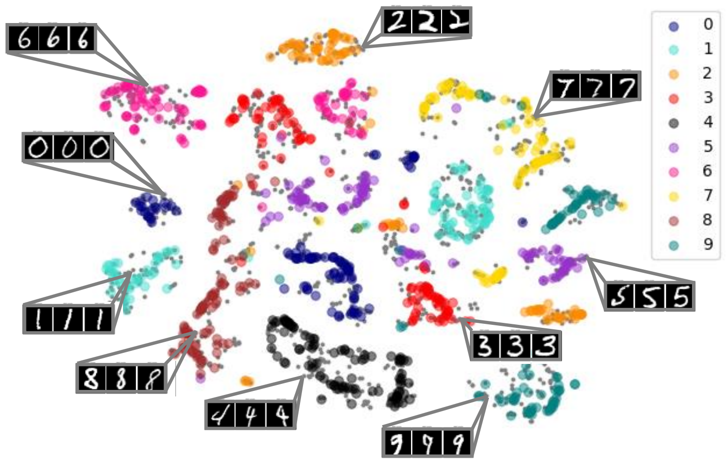


Figure 8: Visualization of *HashGAN* representations for a query set on *MNIST* using *TSNE* projection. The real and synthesized data are indicated by colored and gray circles respectively. Some of the synthesized images are randomly shown from different parts of space.

4.0 Unsupervised Deep Generative Adversarial Hashing Network

4.1 Introduction

Image similarity search in big datasets has gained tremendous attentions in different applications such as information retrieval, data mining and pattern recognition [156]. With rapid growth of image data, it has become crucial to find compact and discriminative representations of images in huge datasets in order to have efficient storage and real-time matching for millions of images. Hashing functions provide an effective solution for this problem by attributing a binary code to each image, and consequently reducing the similarity search between high dimensional images to calculating the Hamming distance between their binary codes [47, 163, 94, 89]. Typically, hash functions are carefully designed to extract distinctive patterns from images relevant to their semantic categorizes, while being robust to various image transformations such as rotation, translation, scale, and lightning [93, 179, 66].

Generally, hash functions can be divided into supervised [94, 172, 51, 89] and unsupervised methods [55, 163, 157, 58]. The supervised hashing methods, especially deep hash functions [82, 32, 199, 179], showed remarkable performance in representing input data with binary codes. Although, these deep hash functions take advantages of deep learning models in representing images with discriminative attributes, they require costly human-annotated labels to train their large set of parameters. Thus, their performance is dramatically degraded by getting stuck in bad local minima when there is not enough labeled data for training.

The unsupervised hashing methods address this issue by providing learning frameworks without requiring any supervisory signals. The unsupervised hashing methods either use shallow models with hand-crafted features [17, 127, 85, 3] as inputs, or employ deep architectures for obtaining both discriminative features and binary hash codes together. However, the unsupervised shallow functions may not capture the non-linear similarities between real-world images due to their low capacity. They also suffer from hand-crafted features and dimension reductions techniques (e.g. principle component analysis (PCA)), which are not robust to noise and image transformations. On the other hand, the unsupervised deep hash functions usually have insignificant improvements against the shallow models, since they can not exploit the power of deep models due to overfitting and lack of supervision. Some of the unsupervised deep hash functions tackle this issue by initializing their parameters using supervised pretraining with large datasets (*e.g.* ImageNet dataset [28]) [93, 66].

We propose a new unsupervised deep hashing model, called *HashGAN*, which nor suffers from shallow hash functions and hand-crafted features, neither needs the supervised pretraining to have discriminative binary codes. Our framework jointly learns a hash function with a generative adversarial network (GAN). In particular, we tie the discriminator of the GAN with the hash function, employing the adversarial loss function as a data-dependent regularization term in training our deep hash function. Furthermore, we introduce a novel loss function for hashing real images, minimizing the entropy of hash bits for each image, maximizing the entropy of frequency of hash bits, improving the consistency of hash codes against different image transformations, and providing independent hash bits. Moreover, we provide a collaborative loss function, which enforces the encoder to have the same binary hash code for a synthesized image by the generator, as the binary input variable provided to the generator while synthesizing the image. We show that this collaborative loss function is a helpful auxiliary task for obtaining discriminative hash codes.

Figure 8 illustrates a 2D visualization of *HashGAN* hash codes for a query set of real and fake images on *MNIST* dataset [83]. As shown, *HashGAN* not only achieves discriminative representations for real data, but also generates synthesized images conditioned on their binary inputs, representing the semantic categories. Experimental results indicate that our proposed model outperforms unsupervised hash functions with significant margin in information retrieval tasks. Moreover, *HashGAN* achieves superior or competitive results compared to the state-of-the-art models in image clustering tasks. We also explore the effect of each term in our loss function using an ablation study. Therefore, our experiments confirm the effectiveness of *HashGAN* in unsupervised attribute learning across different tasks. Our contributions are summarized as follows:

- Proposing a novel framework for unsupervised hashing model by coupling a deep hash function and a generative adversarial network.
- Introducing a new hashing objective for real images, regularized by the adversarial and collaborative loss functions on synthesized images, resulting in minimum-entropy, uniform frequency, consistent, and independent hash bits.
- Achieving state-of-the-art results compared to alternatives on information retrieval and clustering tasks.

4.2 Related Works

4.2.1 Hash Functions

Generally, hash functions can be grouped into supervised [94, 39, 172, 92, 51] and unsupervised methods [55, 163, 157, 58]. The supervised methods require class labels or pairwise similarity ground truths in their learning process, whereas the unsupervised approaches need only input samples. With the growing success of deep learning in different applications, several studies have been published about supervised deep hash functions [82, 32, 199, 89, 179]. They mostly use pairwise relationships in different variants of ranking loss functions (*e.g.* triplet [162], contrastive [52] objectives) to simultaneously learn discriminative features and encode hash bits. However, the performance of these supervised hashing models crucially depends on availability of labeled data in the training process.

Among the shallow models, locality sensitivity hashing (*LSH*) [45] maps original data into a low dimensional feature space using random linear projections, and then obtains binary hash codes. Later in [80, 171], *LSH* was extended to kernel-based variants of hash functions. Gong et al. introduced another well-known model, called iterative quantization (*ITQ*) [47], which uses an alternating optimization approach for learning efficient projections and performing binarization. Spectral hashing (*SpeH*) [163] computes binary hash codes by implementing spectral graph partitioning using the similarity information in a feature space. However, these models suffer from shallow hash functions and inflexible hand-crafted features, which limit their capabilities in dealing with complex and high dimensional real world data.

In unsupervised deep hashing models, semantic hashing [132] is one of the early studies, which adopts Restricted Boltzmann Machine (*RBM*) [60] model as a deep hash function, and trains its parameters using an unsupervised learning approach. Deep Hashing (*DH*) [36] applies an unsupervised loss function to a hierarchical neural networks to have quantized, balanced and independent hash code bits. Lin *et al.* introduced *DeepBit* [93] as an unsupervised deep hashing algorithm by defining an objective function based on quantization loss and balanced and rotation invariant hash bits. In addition to quantization and balanced hash bits loss functions, unsupervised triplet hashing (*UTH*) [66] employs an unsupervised triplet loss, which minimizes the distance of an anchor image and its rotated version (*i.e.* positive pair) while maximizing the distance of the anchor image with a random image (*i.e.* negative pair). Another method with two steps is introduced in [64] to learn discriminative binary representations in an unsupervised manner. A convolutional neural network (*CNN*) is trained using a clustering algorithm in the first step, and then the learned cluster assignments are used as soft pseudo labels in a triplet ranking loss for training a deep hash function in the second step.

Our proposed model falls in the category of unsupervised deep hash functions. But unlike the unsupervised deep hash functions, which have insignificant improvements over the shallow alternatives, and/or require supervised pretraining using a large labeled dataset, *HashGAN* outperforms unsupervised alternatives with significant margins without any supervised pretraining.

4.2.2 Applications of GAN

Goodfellow *et al.* proposed a powerful generative model, called generative adversarial networks (*GAN*) [48], which is able to synthesize realistic images with great details. Particularly, *GAN* objective includes a two-player minimax game between two networks, a generator and a discriminator. The discriminator aims to distinguish between the real and synthesized (*i.e.* fake) images, and the generator maps samples from arbitrary distribution (*i.e.* random noise) to the distribution of real images, trying to synthesize fake images that fool the discriminator. Several studies [30, 120] further addressed problems such as the unstable training process of *GAN* and noisy and blurry synthesized images, resulting in higher quality images. Moreover, some works [99, 113] tried to improve the quality and diversity of generated images by conditioning on the supervisory signals like class labels and text descriptions, and incorporating these supervised information into the generative and discriminative pathways. In addition, *GAN* has been adopted in supervised and semi-supervised tasks to use the input data distribution as a generalization force, and enhance the classification results [142, 133, 120]. Unlike these supervised/semi-supervised studies, our model employs *GAN* in the unsupervised hashing task, and does not require any supervisory signals like class labels and image captions.

In recent years, deep learning has shown remarkable results in wide range of applications, such as computer vision [78], natural language processing [26], speech recognition [59], and even biological science [31]. The impressive capability of deep models is due to efficient and scalable leaning of discriminative features from raw data via multi-layer networks. Among different models, Goodfellow et. al. proposed a powerful generative model, called generative adversarial networks (GAN) [48], particularly for image generation task. GAN consists of two sub-networks, a generator and a discriminator, and aims to play a minimax game between these sub-networks. While the generator's goal is to fool the discriminator by synthesizing realistic images from arbitrary distribution (*i.e.* random noise), the discriminator tries to distinguish between the real and synthesized (*i.e.* fake) images. GAN model is also applied to different tasks, including image generation [30, 71], image translation [200], semi-supervised image classification [133], image inpainting [117, 185], speech enhancement [116] and drug discovery [10].

We also adopts adversarial loss on GAN objective to regularize our graph CNN model, which is different with previous studies. Besides, our task is regression on graph-structured data, which is differing from supervised classification on image data in standard GAN model.

4.3 Unsupervised Deep Generative Adversarial Hashing Network

In this section, we first introduce *HashGAN* by showing its architecture and explaining the intuition behind the model. Then, we define its loss function and describe the effect of each term in our learning framework.

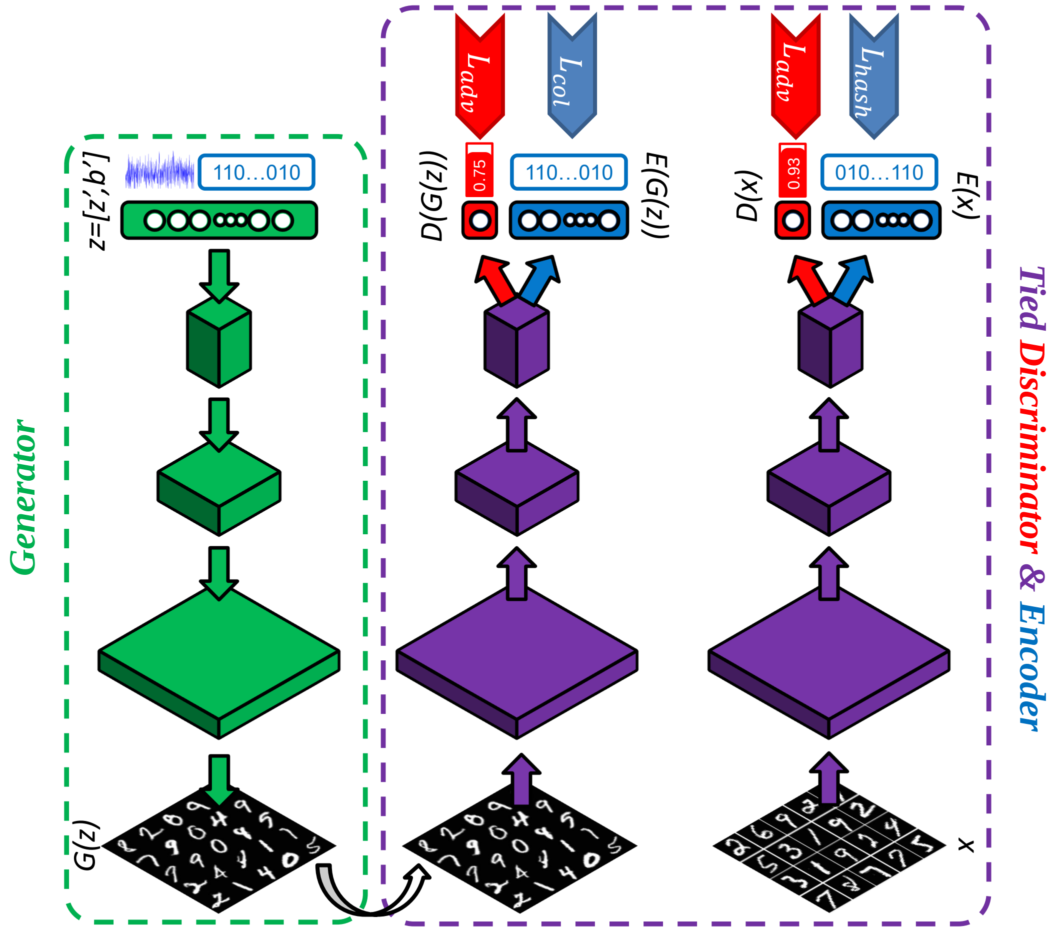


Figure 9: *HashGAN* architecture, including a generator (green), a discriminator (red) and an encoder (blue), where the last two share their parameters in several layers (red⊕blue=purple). The arrows on top represent the loss functions.

4.3.1 HashGAN Architecture

Our proposed *HashGAN* model consists of three components, a generator, a discriminator and an encoder. The generator is supposed to synthesize images that fool the discriminator by mapping samples from a random distribution to the real data distribution. The discriminator is expected to distinguish the synthesized images from real ones. The encoder is designed to map the images to discriminative binary hash codes. As shown in Figure 9, the discriminator and encoder share all of their parameters except for the weights of their last layer. The inputs of generator are also the concatenation of samples from two random distributions, including binary and uniform random variables.

In order to train the discriminator parameters, we use the standard adversarial loss function in *GAN* models. The parameters of encoder are trained via a hashing loss on real data and an ℓ_2-norm loss on fake data. The hashing loss ensures having quantized, balanced, consistent and independent hash codes for real images, and the ℓ_2-norm loss is determined to have similar hash codes as the generator binary inputs for synthesized images. To train the parameters of generator, we utilize the feature matching loss, introduced in [133], to match the statistics of the real and fake images. To do so, the expected value of the features in the last hidden layer of discriminator (encoder) network is selected in the feature matching loss function.

HashGAN architecture has several advantages in our unsupervised deep learning framework. First, tying the discriminator and encoder is very useful in unsupervised training of our deep hash function, because the adversarial loss can be considered as a data-dependent regularization term in training *HashGAN*, which avoids overfitting and getting stuck in bad local minima. From another point of view, the encoder pathway utilizes the information in the data distribution, which is discovered in the latent variables of discriminator.

It has been shown that interpolations in the input space of the generator produce semantic variations along data distribution [120, 33]. Hence, training the encoder to utilize these information hidden in the input variables of generator is helpful in learning discriminative binary codes. The feature matching loss and the ℓ_2-norm loss for training the generator and encoder networks can be considered as collaborative loss functions, which aim to use the generator binary inputs as the pseudo-hash-labels for the synthesized images, while they have similar statistics with the real images. This novel approach fits our unsupervised hashing problem, and it is different with the conventional conditional *GAN* models [99, 113], which need supervisory signals.

4.3.2 HashGAN Loss Function

Consider there are N images in the gallery set, denoted by $\mathbf{X} = \{\mathbf{x}_i | i = 1, \cdots, N\}$, which are used in training our deep hash function. We utilize a multi-layer hash encoder to map the input images into the K-bit hash codes. To do so, there are K independent sigmoid functions in the last layer of our encoder network. Thus, the output of encoder for each image is represented by $\mathbf{t}_i = \mathscr{E}(\mathbf{x}_i)$, which shows the composition of K independent probabilities as $t_{ik} = P(b_{ik} = 1 | \mathbf{x}_i; \mathbf{W}_{\mathscr{E}})$, where t_{ik} and b_{ik} are the k-th output of encoder and binary hash code for the i-th image, and $\mathbf{W}_{\mathscr{E}}$ indicates the encoder parameters. Note that the binary hash codes are simply computed using $b_{ik} = \mathbf{1}(t_{ik} > 0.5)$, where $\mathbf{1}(.)$ is the indicator function.

Our *HashGAN* model employs a generator network, which maps the samples from a random distribution to the data distribution. As mentioned earlier, the random input of generator is concatenation of binary and uniform random variable as $\mathbf{z}_i = [\mathbf{z}'_i, \mathbf{b}'_i]$, where $\mathbf{z}'_i \sim \mathscr{U}(0, 1)$ shows the uniform random noise and $\mathbf{b}'_i \sim \mathscr{B}$ indicates the binary random noise. While the real images are shown by $\mathbf{x}_i$, the synthesized images by the generator are represented by $\hat{\mathbf{x}}_i = \mathscr{G}(\mathbf{z}_i)$. We also obtain the encoder outputs for the synthesized images as $\hat{\mathbf{t}}_i = \mathscr{E}(\hat{\mathbf{x}}_i) = \mathscr{E}(\mathscr{G}(\mathbf{z}_i))$.

The discriminator of *HashGAN* is supposed to determine whether its input image is a real or a synthesized sample. A sigmoid function is considered as the last layer of discriminator, computing the probabilities $p_i = \mathscr{D}(\mathbf{x}_i) = P(y_i = 1 | \mathbf{x}_i; \mathbf{W}_{\mathscr{D}})$ and $\hat{p_i} = \mathscr{D}(\mathscr{G}(\mathbf{z}_i)) = P(y_i = 1 | \hat{\mathbf{x}}_i; \mathbf{W}_{\mathscr{D}})$, where p_i and $\hat{p_i}$ are the probabilities of being real ($y_i = 1$) for the i-th real and synthesized images respectively.

Now, we are able to define the loss function in our learning framework. The total loss function is summation of the adversarial loss, hashing loss, and collaborative loss for the real and synthesized images:

$$\mathscr{L}_{total} = \mathscr{L}_{adv} + \mathscr{L}_{hash} + \mathscr{L}_{col}\,. \tag{4.1}$$

Following, we describe each term of the loss function in more details, and explain the role of each one in achieving discriminative binary hash codes. As proposed in [48], the adversarial loss in *GAN* models is designed as a minimax play between the discriminator and the generator models, in which the discriminator is trained to correctly distinguish the real and synthesized images, and the generator is trained to synthesize fake images that fool the discriminator. The adversarial loss function for training our discriminator has the following form:

$$\max_{\mathscr{D}} \quad \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\big[\log(\mathscr{D}(\mathbf{x}))\big] + \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\big[\log(1-\mathscr{D}(\mathscr{G}(\mathbf{z}))\big] \tag{4.2}$$

where the goal is to train the discriminator $\mathscr{D}$ to distinguish the real image $\mathbf{x}$ from the synthesized sample $\mathscr{G}(\mathbf{z})$. The adversarial loss is maximized *w.r.t.* the discriminator to increase the log-likelihood of correct predictions on real images and decrease the log-likelihood of mis-prediction on fake samples.

The hashing objective for real data contains four losses, including minimum-entropy, uniform frequency, consistent, and independent bits loss functions. The following equation shows these loss functions:

$$\begin{aligned}\min_{\mathscr{E}} \quad & \underbrace{-\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik}\log t_{ik} + (1-t_{ik})\log(1-t_{ik})}_{\text{minimum entropy bits}} \\ & + \underbrace{\sum_{k=1}^{K} f_k \log f_k + (1-f_k)\log(1-f_k)}_{\text{uniform frequency bits}} \\ & + \underbrace{\sum_{i=1}^{N}\sum_{k=1}^{K} \|t_{ik}-\tilde{t}_{ik}\|_2^2}_{\text{consistent bits}} + \underbrace{\|\mathbf{W}_{\mathscr{E}}^{L^\top}\mathbf{W}_{\mathscr{E}}^{L} - \mathbf{I}\|_2^2}_{\text{independent bits}}\,, \end{aligned} \tag{4.3}$$

where $\tilde{t}_{ik} = P(b_{ik}|\tilde{\mathbf{x}}_i; \mathbf{W}_{\mathscr{E}})$ is the k-th encoder output for the i-th real image, transformed by translation, rotation, flipping, or noise, $f_k = 1/N \sum_{i=1}^{N} t_{ik}$ is the frequency of the k-th hash bit code over sampled images, and $\mathbf{W}_{\mathscr{E}}^{L}$ is the weights of the last layer on the encoder network.

The first term in the hashing loss function is equivalent to entropy of each hash bit, and minimizing this term pushes hash bits for each image toward 0 or 1. Thus, the minimum-entropy bits loss function reduces the quantization loss without using the sign function. Considering f_k as the empirical frequency of each hash bits, the second term in this loss function is a negative of entropy for the bits frequency. By maximizing (i.e. minimizing negative of) the entropy of bits frequency, the encoder tends to have balanced hash codes. The third term in the loss function constrains the encoder to extract similar hash codes for an image and its transformed variants, making the encoder robust to the transformations. Finally, the last term in this loss function pushes the encoder to have independent hash bits.

We also take advantages of the synthesized images in training the encoder network by a ℓ_2-norm loss function, which minimizes the distance of encoder outputs and generator binary inputs. Following equation shows the ℓ_2-norm loss on the synthesized data:

$$\min_{\mathscr{E}} \quad \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\left[\|\mathscr{E}(\mathscr{G}(\mathbf{z})) - \mathbf{b}'\|_2^2\right], \tag{4.4}$$

where $\mathbf{b}'$ is the binary random variable in the generator input $\mathbf{z} = [\mathbf{z}', \mathbf{b}']$. Using this ℓ_2-norm loss function, the encoder network is able to provide similar hash codes for the synthesized images, which share the same binary attributes $\mathbf{b}'$, but vary due to different uniform random variables $\mathbf{z}'$.

In order to train the generator network, we used the feature matching loss instead of directly optimizing the output of the discriminator via the traditional adversarial loss function. The feature matching loss requires the generator to synthesize images that have similar statistic to the real images. We consider the last hidden layer of discriminator, denoted by $\mathscr{F}$, as the source of statistic, and define the following loss function:

$$\min_{\mathscr{G}} \quad \|\mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\mathscr{F}(\mathbf{x}) - \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\mathscr{F}(\mathscr{G}(\mathbf{z}))\|_2^2, \tag{4.5}$$

where $\mathscr{F}$ is also the last hidden layer of encoder network, affecting the hash codes and the adversarial probability. The feature matching loss provides more stability in training our model, and leads the synthesized images to share statistic with real data. This is very helpful in collaborating with ℓ_2-norm loss, making the pseudo-hash-labels for fake data effective on obtaining discriminative binary representations for real images.

In order to train our *HashGAN* model, we are able to use stochastic learning techniques. Thus, we alternatively train the generator and tie the discriminator and the encoder networks. In particular, we optimize the parameters of discriminator and encoder jointly using the adversarial, hashing and ℓ_2-norm loss functions in one step, and train the parameters of generator using the feature matching loss in the next step.

4.4 Experiments

We perform several experiments to evaluate the performance of our proposed model on multiple datasets. The quality of hash codes extracted by *HashGAN* is explored in image retrieval and clustering tasks. We also investigate the effect of each component in our loss function using an ablation study.

Implementation details: We use almost similar architectures for *HashGAN* to the *Improved-GAN* networks in [133]. We avoid pooling layers and use strided convolutional layers, utilize weight normalization [134] to stabilize the training process, consider ReLU and leaky-ReLU non-linearities [98] as the activation function of convolutional layers in our discriminator and encoder. For image preprocessing, we only normalize the image intensities to be in the range of $[0, 1]$ or $[-1, 1]$, and consequently use sigmoid and TanH functions in the last layer of our generator. A zero mean Gaussian noise with standard deviation of 0.15 is also added to the input images of our discriminator/encoder. Moreover, we set the learning rate to 9×10^{-4} and linearly decrease it to 3×10^{-4}, and adopt Adam [74] as our optimization method with the hyper-parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = 1e-08$. Since our hashing task is unsupervised, we did not tune any hyper-parameters for adjusting the effect of our losses in different datasets, and use the default setting. In particular, we set

Table 7: Image retrieval results (mAP and mAP@1000) of unsupervised hash functions on CIFAR-10 and MNIST datasets, when the number of hash bits are 16, 32 and 64. The usage of supervised pretraining is shown for each model using the tick sign. The results of alternative models are reported from the reference papers, except for the ones marked by ($*$) on top, which are obtained by us running the released code.

	Dataset	*CIFAR-10*						*MNIST*						Super. Pretrain
		mAP (%)			mAP@1000 (%)			mAP (%)			mAP@1000 (%)			
	Model	16	32	64	16	32	64	16	32	64	16	32	64	
Shallow	*KMH* [55]	13.59	13.93	14.46	24.08*	23.56*	25.19*	32.12	33.29	35.78	59.12*	70.32*	67.62*	✗
	SphH [58]	13.98	14.58	15.38	24.52*	24.16*	26.09*	25.81	30.77	34.75	52.97*	65.45*	65.45*	✗
	SpeH [163]	12.55	12.42	12.56	22.10*	21.79*	21.97*	26.64	25.72	24.10	59.72*	64.37*	67.60*	✗
	PCAH [157]	12.91	12.60	12.10	21.52*	21.62*	20.54*	27.33	24.85	21.47	60.98*	64.47*	63.31*	✗
	LSH [45]	12.55	13.76	15.07	12.63*	16.31*	18.00*	20.88	25.83	31.71	42.10*	50.45*	66.23*	✗
	ITQ [47]	15.67	16.20	16.64	26.71*	27.41*	28.93*	41.18	43.82	45.37	70.06*	76.86*	80.23*	✗
Deep	*DH* [36]	16.17	16.62	16.96	-	-	-	43.14	44.97	46.74	-	-	-	✗
	DAR [64]	16.82	17.01	17.21	-	-	-	-	-	-	-	-	-	✗
	DeepBit [93]	-	-	-	19.43	24.86	27.73	-	-	-	28.18	32.02	44.53	✓
	UTH [66]	-	-	-	28.66	30.66	32.41	-	-	-	43.15	46.58	49.88	✓
	HashGAN [ours]	**29.94**	**31.47**	**32.53**	**44.65**	**46.34**	**48.12**	**91.13**	**92.70**	**93.93**	**94.31**	**95.48**	**96.37**	✗

the weights for the adversarial ($\mathcal{L}_{adv}$), feature matching ($\mathcal{L}_{feat}$), independent bits ($\mathcal{L}_{indBit}$), uniform frequency bits ($\mathcal{L}_{uniFrqBit}$), consistent bits ($\mathcal{L}_{consBit}$) loss functions equal to 1, and the weight of ℓ_2-norm loss ($\mathcal{L}_2$) equal to 0.1. For $\mathcal{L}_{minEntrpBit}$ in the hash loss function, the weight is selected from $\lambda_{minEntrpBit} = \{10^{-3}, 10^{-2}\}$ based on the final epoch loss value in the training process. Besides, we first train *HashGAN* without the hash and ℓ_2-norm loss functions by setting its weight equal to zero for one tenth of the maximum epoch, since the obtained hash codes in the first iterations may not be reliable for training the encoder parameters. We use Theano toolbox [2] for writing our code, and run the algorithm in a machine with one Titan X Pascal GPU.

Datasets: We compare our model with unsupervised hash functions in the image retrieval task on *CIFAR-10* [77] and *MNIST* [83]. Furthermore, we analyze the discriminative capability of *HashGAN* binary codes in the image clustering task on *MNIST*, *USPS*, *FRGC* [180] and *STL-10* [25] datasets. Following, we describe each dataset briefly.

CIFAR-10 dataset [77] contains 60K 32×32 colored images balanced across 10 classes (*i.e.* airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck).

MNIST dataset [83] includes 70K 28×28 gray scale images of hand written digits (0-9) across 10 classes.

USPS is a dataset of 11K 16×16 gray scale handwritten digits from USPS postal service, with unbalanced distribution across the ten digits.

FRGC contains $2,462$ facial images from randomly selected 20 subjects on this dataset [180]. Similar to [180], we crop the images to 32×32 colored facial images.

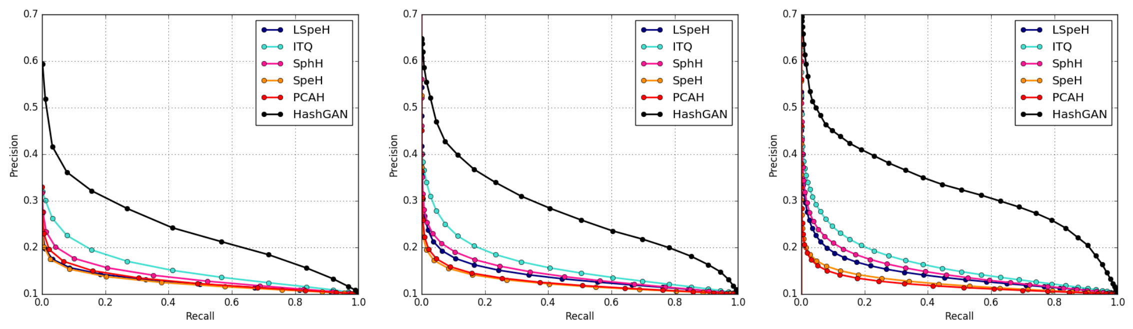
STL-10 database [25] includes 13K colored images across 10 classes (*i.e.* airplane, bird, car, cat, deer, dog, horse, monkey, ship and truck). The images are resized to 32×32.

4.4.1 Image Retrieval

Alternative models: For image retrieval, we compare our method with the previous unsupervised hash functions including K-means hashing (*KMH*) [55], spherical hashing (*SphH*) [58], spectral hashing (*SpeH*) [163], PCA-based hashing (*PCAH*) [157], locality sensitivity hashing (*LSH*) [45], iterative quantization (*ITQ*) [47], deep hashing (*DH*) [36], discriminative attributes representations (*DAR*) [64], *DeepBit* [93] and unsupervised triplet hashing (*UTH*) [66].

Evaluation metrics: We evaluate the performance of *HashGAN* compared to the aforementioned unsupervised hashing functions using precision and mean average precision (mAP). We follow the standard protocol for both *MNIST* and *CIFAR-10* datasets, and randomly sample 1000 images (100 per class) as the query set and use the remaining data as the gallery set. In particular, we report the results of the image retrieval in terms of precision@1000, mAP, and mAP@1000[1], where precision@1000 is the fraction of correctly retrieved samples from the top 1000 retrieved samples in gallery, mAP is the mean of the average precision of query images over all the relevant images, mAP@1000 is mAP calculated over the top 1000 ranked images from the gallery set. The reported results are the average of 5 experimental results.

[1]Note that comparisons in some of the previous studies are confusing, as they comapre mAP results of baseline models with mAP@1000 results of other models. To avoid such confusion, we provide evaluations in terms of both of these metrics, separately.



(a) 16 bits (b) 32 bits (c) 64 bits

Figure 10: Precision-Recall curves on *CIFAR-10* database for *HashGAN* and five baselines with 16, 32, and 64 hash bits.

Performance comparison: Table 15 shows the mAP and mAP@1000 results of *HashGAN* and other alternative models across different hash bit sizes. To better compare the models, we divide the hash functions into two groups of shallow and deep models, and indicate whether they use supervised pretraining or not. The results demonstrate that our model consistently outperforms other models with significant margins across different number of bits, datasets and metrics. Although, *HashGAN* gives better performance with more number of hash bits, its performance has small drops with less hash bits. Interestingly, the unsupervised deep hash functions, which use supervised pretraining via ImageNet dataset, show better results on *CIFAR-10* dataset compared to the shallow models, but have relatively lower performance on *MNIST* dataset. This shows that pretraining on ImageNet dataset is more helpful for *CIFAR-10* than for *MNIST*, which is not that surprising, given that ImageNet data distribution looks more similar to the *CIFAR-10* image distribution than *MNIST*. However, our model does not require any supervised pretraining, and consequently is not affected by pretraining biases, and achieves superior results on both datasets.

Table 8 indicates the results of precision@1000 for *HashGAN* and some of the unsupervised hash functions. Similar to Table 15, *HashGAN* achieves superior results in comparison with the alternative shallow and deep models. The improvements of our model are consis-

Table 8: Image retrieval results (precision@1000) of unsupervised hash functions on CIFAR-10 and MNIST datasets, when the number of hash bits are 16, 32 and 64. The results of alternative models are reported from the reference papers, except for the ones marked by $(*)$ on top, which are obtained by us running the released code.

	Dataset	CIFAR-10			MNIST		
		precision@1000 (%)			precision@1000 (%)		
	Model	16	32	64	16	32	64
Shallow	*KMH* [55]	18.83	19.72	20.16	51.08*	53.82*	54.13*
	SphH [58]	18.90*	20.91*	23.25*	46.31*	54.74*	62.50*
	SpeH [163]	18.83	19.72	20.16	51.08*	53.75*	54.13*
	PCAH [157]	18.89	19.35	18.73	51.79*	51.90*	48.36*
	LSH [45]	16.21	19.10	22.25	31.95*	45.05*	55.31*
	ITQ [47]	22.46	25.30	27.09	61.94*	68.80*	71.00*
Deep	*DH* [36]	16.17	16.62	16.96	-	-	-
	DAR [64]	24.54	26.62	28.06	-	-	-
	HashGAN [ours]	**41.76**	**43.62**	**45.51**	**93.52**	**94.83**	**95.60**

tent across both *MNIST* and *CIFAR-10* datasets and different hash code sizes, showing the effectiveness of our learning framework in dealing with different conditions. We also compare *HashGAN* with the baselines using precision-recall curves on *CIFAR-10* dataset. Figure 18 clearly demonstrates better performance for *HashGAN* consistently across different number of bits.

Moreover, we visualize the *HashGAN*'s top 10 retrieved images for some query data on *CIFAR-10* dataset, when the hash bit size is 32. Figure 11 illustrates these results, qualitatively showing that our hash function is able to extract semantic binary attributes.

4.4.2 Ablation Study

We perform an ablation study to examine the contribution of each loss component in the achieved results. We evaluate this experiment across L_{indBit}, L_2, $L_{consBit}$, $L_{uniFrqBit}$ and $L_{adv} + L_{feat} + L_2$. Note that in the absence of adversarial loss, the feature matching and ℓ_2-norm losses are also excluded due to their co-dependencies with the adversarial loss.

Table 9: Clustering performance of *HashGAN* and several other algorithms on four image datasets based on accuracy (ACC) and normalized mutual information (NMI). The results of alternative models are reported from the reference papers, except for the ones marked by (∗) on top, which are obtained by us running the released code.

	Dataset	*MNIST*		*USPS*		*FRGC*		*STL-10*	
	Model	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
Shallow	*K-means*	0.500	0.534	0.450	0.460	0.287	0.243	0.209*	0.284
	N-Cuts [137]	0.411	0.327	0.675	0.314	0.285	0.235	-	-
	SC-LS [23]	0.706	0.714	0.681	0.659	0.550	0.407	-	-
	AC-PIC [192]	0.017	0.115	0.840	0.855	0.415	0.320	-	-
	SEC [110]	0.779	0.804	0.511	0.544	-	-	0.245*	0.307
	LDMGI [181]	0.802	0.842	0.563	0.580	-	-	0.260*	0.331
Deep	*NMF-D* [148]	0.152	0.175	0.287	0.382	0.259	0.274	-	-
	DEC [174]	0.816	0.844	0.586	0.619	0.505	0.378	0.284*	0.359
	JULE-RC [180]	0.913	0.964	0.913	0.950	0.574	0.461	-	-
	DEPICT [41]	**0.917**	**0.965**	**0.927**	0.964	**0.610**	**0.470**	0.303*	0.371*
	HashGAN [ours]	0.913	**0.965**	0.920	0.958	0.602	0.465	**0.316**	**0.394**

We exclude loss components one at a time, measuring the difference in precision@1000 on *MNIST* and *CIFAR-10* datasets (See Fig. 17). The first observation is that all of the loss components contribute in improving the results. Furthermore, the figure shows the strong effect of $L_{adv} + L_{feat} + L_2$ as the key components in avoiding overfitting. In other words, employing GAN in our model has the highest practical contribution, and removing the discriminator and generator degrades the performance substantially. It also demonstrates that the presence of uniform frequency loss is very important. Examining the results achieved in the absence of this loss demonstrates that some of the binary codes collapse to either zero or one, reducing the capacity of the assigned hash bit size. The relative analysis of the results in each dataset demonstrates that consistency loss is more effective in *CIFAR-10* than in *MNIST*. This is expected as we only use noise for image transformation on MNIST since the images are centered and scaled, but rely on extra transformations including translations and horizontal flipping for *CIFAR-10*. The figure also demonstrates considerable contribution from the ℓ_2-norm loss, showing the effectiveness of our framework in using the synthesized

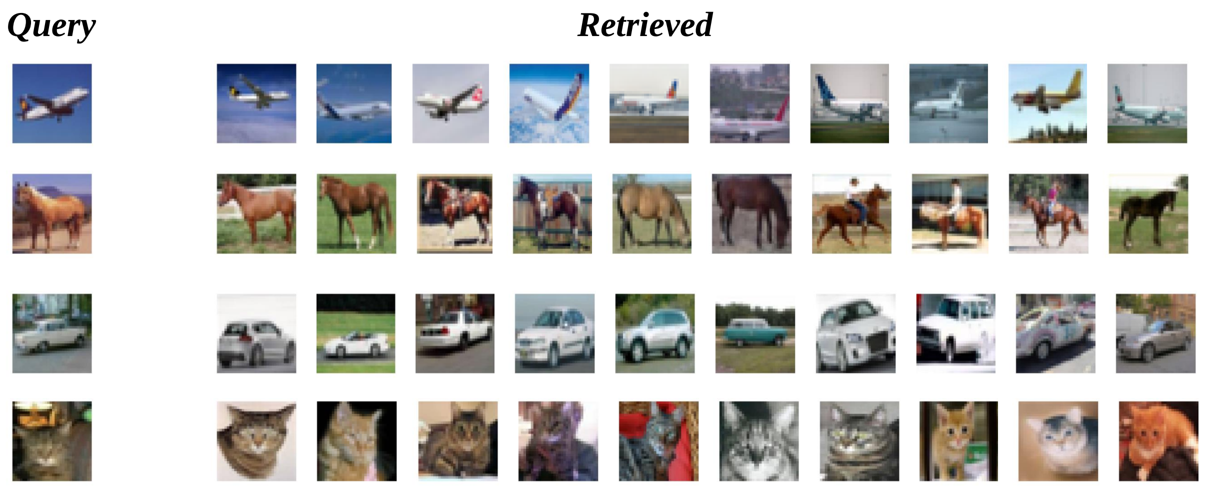


Figure 11: Top 10 retrieved images for query data by *HashGAN* on *CIFAR-10* dataset with 32 bits hash code.

images for training the encoder network. The lowest effect is provided by the independent bit loss.

4.4.3 Image Clustering

One way to measure whether the hash function is effective in extracting distinctive codes is to evaluate their performance in clustering tasks. Hence, we assess *HashGAN*'s ability in clustering, by using the extracted hash codes as low dimensional input features for K-means and compare the results with alternative clustering models.

Alternative Models: We compare our clustering method with several baselines and state-of-the-art clustering algorithms, including *K-means*, normalized cuts (*N-Cuts*) [137], large-scale spectral clustering (*SC-LS*) [23], agglomerative clustering via path integral (*AC-PIC*) [192], spectral embedded clustering (*SEC*) [110], local discriminant models and global integration (*LDMGI*) [181], *NMF* with deep model (*NMF-D*) [148], task-specific clustering with deep model (*TSC-D*) [160], deep embedded clustering (*DEC*) [174], joint unsupervised learning (*JULE-RC*) [180] and *DEPICT* [41].

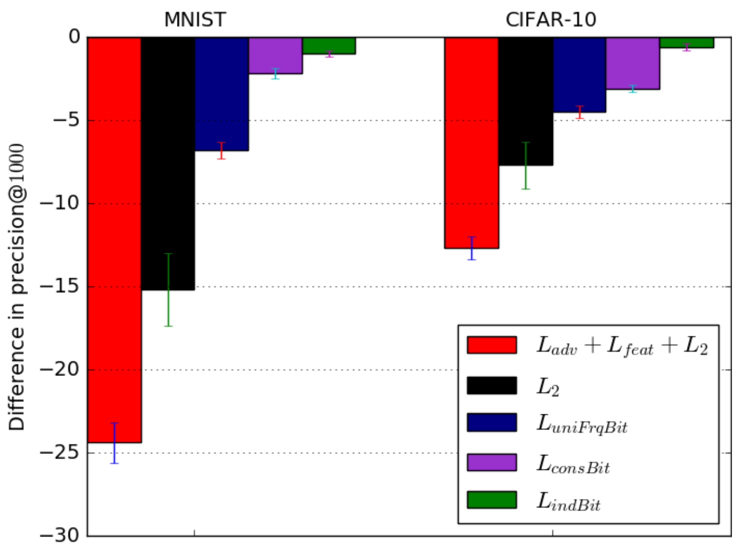


Figure 12: The difference in the precision@1000, when each of the loss components are excluded from the *HashGAN*'s objective function on *MNIST* and *CIFAR-10* datasets.

Evaluation metrics: To compare the clustering results of our model with previous studies, we rely on the two popular metrics used to evaluate clustering: *normalized mutual information* (NMI), and *accuracy* (ACC). NMI provides a measure of similarity between two data with the same label, which is normalized between 0 (lowest similarity) to 1 (highest similarity) [178]. To calculate ACC we find the best mapping between the predicted clusters and the true labels, following the approach proposed by [79].

Performance comparison: Table 13 gives the evaluation results for our clustering method and the mentioned algorithms in terms of NMI and ACC across *MNIST*, *USPS*, *FRGC*, and *STL-10* datasets. The results demonstrate that our method (*HashGAN* + K-means) achieves superior or competitive results compared to the state-of-the-art clustering algorithms. Note that our method is not specially designed for clustering, since we only run K-means algorithm on the *HashGAN* representations without backpropagating clustering error through the network. The table also indicates clear advantage of deep models compared with shallow models, emphasizing the importance of deep representations in image clustering.

Overall, this experiment demonstrates the effectiveness of *HashGAN* model in extracting discriminative representations on different datasets in completely unsupervised manner.

4.5 Conclusion

This project introduced *HashGAN*, an unsupervised deep hashing model, composed of a generator, a discriminator and an encoder. We defined a novel objective function to efficiently train our deep hash function without any supervision. Using the tied discriminator and encoder, we employed the adversarial loss as a data-dependent regularization for unsupervised learning of our hash function. Our novel hashing loss also led to quantized, balanced, consistent and independent hash bits for real images. Furthermore, we introduced a collaborative loss to use the synthesized images in training our hash function. *HashGAN* outperformed unsupervised hashing models in information retrieval with significant margin, and achieved state-of-the-art results in image clustering.

5.0 Balanced Self-Paced Learning for Generative Adversarial Clustering Network

5.1 Introduction

Clustering is one of the essential active research topics in computer vision and machine learning communities with various applications. Clustering problem has been extensively studied in the literature by introducing numerous algorithms with unsupervised learning frameworks [177]. However, the existing methods that employ shallow or deep models suffer from different issues. The shallow clustering models may not capture the nonlinear nature of data due to their shallow and linear embedding functions, adversely affect their performance by using inflexible hand-crafted features, and have difficulties in scaling to large datasets. In contrast, the deep clustering methods have enough capacity to model the non-linear and complex data, and are able to deal with large-scale datasets. But they are prone to the overfitting issue leading to get stuck in bad local minima, since there is no reliable supervisory signal for training their large number of parameters.

In this project, we propose a generative adversarial clustering network, called *ClusterGAN*, as a novel deep clustering model to address the aforementioned issues. *ClusterGAN* adopts the adversarial game in *GAN* for the clustering task, and employs an efficient self-paced learning algorithm to boost its performance. The standard *GAN* is formulated as an adversarial game between two networks, a discriminator and a generator [48]. In particular, the generator $\mathscr{G}$ is supposed to synthesize realistic images to fool the discriminator $\mathscr{D}$ by mapping the random input $\mathbf{z}$ into the data space, and the discriminator aims to distinguish the real data from the generated samples. The objective function in this two-player adversarial game between $\mathscr{D}$ and $\mathscr{G}$ is:

$$\min_{\mathscr{G}} \max_{\mathscr{D}} \ \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\big[\log \mathscr{D}(\mathbf{x})\big] + \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\big[\log\big(1-\mathscr{D}(\mathscr{G}(\mathbf{z}))\big)\big]\,, \tag{5.1}$$

where $P(\mathbf{x})$ is the real data distribution, and $P(\mathbf{z})$ is the generator random input distribution. In this adversarial loss, $\mathscr{G}$ is trained to learn the conditional distribution of real data given the random variables, and $\mathscr{D}$ is trained to find the boundaries between samples drawn from the real and generated data distributions.

Unlike the traditional *GAN*, *ClusterGAN* consists of three networks, a discriminator $\mathscr{D}$, a generator $\mathscr{G}$, and a clusterer $\mathscr{C}$ (*i.e.* a clustering network). The generator and clusterer are both conditional generative networks, where $\mathscr{G} : \mathbf{z} \rightarrow \hat{\mathbf{x}}$ generates the realistic data samples given the latent variables and $\mathscr{C} : \mathbf{x} \rightarrow \hat{\mathbf{z}}$ generates the discriminative latent variables given the real data. The discriminator $\mathscr{D}$ accepts a joint distribution of samples and features (*i.e.* latent variables) as the input, and tries to identify whether the paired samples belong to the generator $(\mathbf{z}, \hat{\mathbf{x}})$ or the clusterer $(\hat{\mathbf{z}}, \mathbf{x})$. Thus, training the generator and clusterer to fool the discriminator leads to generating synthesized samples similar to real data and estimating features similar to the generator latent variables. By considering a discriminative distribution for the generator inputs, we employ the adversarial game between $\mathscr{D}$, $\mathscr{G}$ and $\mathscr{C}$, and learn a discriminative embedding space in the output of the clusterer. Figure 13 illustrates the architecture of *ClusterGAN*.

Moreover, we introduce a novel clustering objective, which is directly applied on the output of the clusterer given the real samples. The basic idea is to impose a block diagonal constraint on the adjacency matrix of the real data. To do so, we first compute the similarity values between real samples using the cosine similarity function applied on the clusterer outputs. Then, a minimum entropy loss function is imposed to the similarity values to push them towards 0 (*i.e.* dissimilar) or 1 (*i.e.* similar). However, the main challenge is that the ground-truth similarities are unknown in unsupervised learning, which makes it difficult to train a deep clustering model from the scratch. In order to tackle this issue, we enhance the minimum entropy objective by utilizing a novel self-paced learning algorithm. Generally, the standard self-paced learning algorithm initiates the training process with easy samples, and then gradually takes more difficult samples into the training. Considering the difficulty level of samples based on their loss values, the self-paced learning is reported to alleviate the problem of getting stuck in bad local minima, and provides better generalization for the models [81]. In addition to this gradually learning approach, we take the prior of selected samples into consideration using an exclusive lasso regularization. This helps us to select a more diverse set of samples in each training step, and prevents learning from easy samples belonging only to a few clusters. We also provide a theoretical proof for our balanced self-paced learning algorithm in regard to achieving the global optimum closed form solution.

In our experiments, *ClusterGAN* achieves state-of-the-art results compared to the alternative clustering methods on several datasets. We also examine the effects of each component in our learning objective function using an ablation study. Moreover, we evaluate the performance of *ClusterGAN* representations in comparison with unsupervised hash functions on information retrieval tasks. The experimental results confirm the effectiveness of our learning framework in training unsupervised models with large depth. Therefore, the contribution of this project can be summarized as the following points.

- We introduce a deep clustering model by adopting the generative adversarial network for clustering.
- We propose a novel balanced self-paced learning algorithm for clustering by gradually incorporating easy to more difficult samples into training steps, while keeping the prior of selected samples balanced in each step.
- Our proposed model achieves comparable results to the state-of-the-art methods on clustering and information retrieval tasks.

5.2 Related Works

5.2.1 Clustering Algorithms

Countless number of clustering methods have been proposed in the literature, which can be divided into shallow and deep models. In shallow clustering algorithms, K*-means* and Gaussian mixture model (*GMM*) [12] are two classical examples of distance-based clustering methods, which represent the clusters using geometric properties of the data points. The kernel-based algorithms, like max-margin methods [193, 176], attempt to model the non-linearity of data via the proper kernel functions. The connectivity-based algorithms, including spectral methods [105, 189], aim to partition the data points that are highly connected. However, these algorithms are not able to model the complex real-world data because of their shallow and linear models.

Recently, deep clustering models attract more attentions due to their capabilities in dealing with complex, high-dimension and large-scale datasets. A mutli-layer sparse coding network followed by a clustering algorithm is introduced in [160], where an alternative learning approach is used to update the code books and estimate the clustering assignments. Trigeorgis *et al.* stacked multiple semi non-negative matrix factorization layers to achieve discriminative representations at the top layer, and used K*-means* to get cluster assignments [148].

Autoencoder network is also adopted in multiple deep clustering models to build discriminative embedding space using the reconstruction task. Tian *et al.* trained a stacked autoencoder on the affinity matrix of a graph, and then obtained the clusters by running K*-means* at the top layer features [146]. Xie *et al.* introduced deep embedded clustering (*DEC*), which is first pre-trained using the reconstruction loss, and then fine-tuned via Kullback-Leibler divergence minimization [174]. Dizaji *et al.* proposed *DEPICT* as a deep clustering autoencoder network, that is trained using a joint reconstruction loss and relative entropy minimization. *DEPICT* also benefits from a regularization term for balancing the prior probability of cluster assignments [41].

Moreover, *JULE* employs a convolutional neural network to represent the features, which are iteratively clustered using an agglomerative clustering algorithm [180]. Yu *et. al.* extended *GMM* to GAN mixture model by allocating a *GAN* model for each cluster [186]. Hu *et. al.* introduced a clustering algorithm, called *IMSAT*, by encouraging the predictions for augmented samples to be close to the original ones, and maximizing the mutual information of the predicted representations. *IMSAT* employs the virtual adversarial training [100] and geometric transformations as data augmentation approaches [63]. *ClusterGAN* differs from the previous models, because it adopts the adversarial game in *GAN* for unsupervised learning of discriminative representations, and employs a novel self-paced learning algorithm for clustering. Consequently, it is able to efficiently train deeper clusterers compared to alternative algorithms.

5.2.2 Self-Paced Learning Algorithms

Inspired by the human learning principle, curriculum learning starts learning with easier examples, and then gradually takes more complex examples into consideration [8]. But in order to avoid heuristic "easiness" measures, Kumar *et. al.* proposed self-paced learning algorithm that incorporates curriculum learning into the model optimization. It adds a regularization term to the objective function, and consequently defines "easiness" measures by the loss value regarding each sample [81]. Jiang *et. al.* extended self-paced learning to also consider the diversity of samples selected in each training step [69]. Many studies further adopted self-paced learning in their tasks to avoid getting stuck in bad local minima and improve the generalization of their models [190, 91, 87]. Our balanced self-paced learning approach differs with the existing methods, since it is applied to an unsupervised loss based on adjacency matrix. It is also specially different with the algorithm in [69], which uses the $\ell_{2,1}$-norm regularization and supervised class labels, but our approach utilizes the exclusive lasso regularization with no need to supervisory signals.

5.2.3 Generative Adversarial Networks

GAN [48] is a powerful class of deep generative models, and is able to generate realistic images with great details. Particularly, its effective approach is relied on a minimax game between a generator and a discriminator, which compete each other to synthesize more realistic samples and detect the real samples. Several studies further attempted to improve the quality of generated images, for instance by using Laplacian pyramid framework [30], strided convolution layers and batch normalization [120], and a generator conditioning on the class labels or text descriptions [99, 113]. In addition, *GAN* has been adopted in supervised, semi-supervised and unsupervised tasks, which have an inference model (*e.g.* classifier) [21, 29, 133, 44, 42, 158]. Among them, ALI [35] and Triple-GAN [24] are more close to our proposed model, where they are specifically designed for semi-supervised classification, but *ClusterGAN* is developed for clustering. In particular, our learning framework is unique by utilizing a novel self-paced learning algorithm and customized generative adversarial network for clustering.

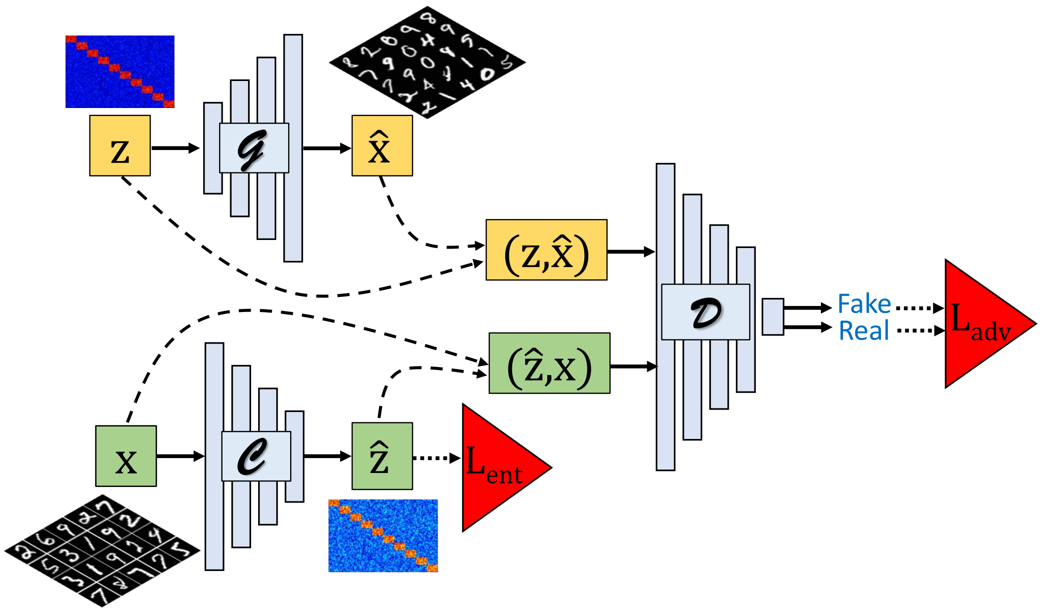


Figure 13: Architecture of *ClusterGAN* along with the applied loss functions. *ClusterGAN* consists of three networks, a generator $\mathscr{G}$, a clusterer $\mathscr{C}$ and a discriminator $\mathscr{D}$. The generator synthesizes the realistic samples given the discriminative random inputs. The clusterer maps the real images into the discriminative latent variables. The discriminator distinguishes whether its input pair belongs to the generator or the clusterer. The adversarial L_{adv} and minimum entropy L_{ent} loss functions are applied to the discriminator and clusterer outputs respectively.

5.3 Method

In this section, we first define the adversarial game regarding the minimax objective in *ClusterGAN*, and then explain our conditional entropy minimization loss, which is enhanced by the proposed balanced self-paced learning algorithm. Given n unlabeled samples $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$ as the inputs, we aim to cluster them into c categories, where the ground-truth labels are represented by $\mathbf{y} = [y_1, ..., y_n]$. While *ClusterGAN* contains three networks, a discriminator, a generator, and a clusterer, our final goal is to construct a block diagonal adjacency matrix $\mathbf{A}$ based on the outputs of the clusterer, where $a_{ij} = 1$ if $y_i = y_j$ and $a_{ij} = 0$ otherwise. Achieving the proper block diagonal adjacency matrix leads to easy clustering assignments with no need to a complicated clustering algorithm. Since the output layer of the clusterer is sigmoid function, we simply use the cosine similarity function to compute the adjacency matrix as $a_{ij} = \hat{\mathbf{z}}_i^\intercal \hat{\mathbf{z}}_j / \|\hat{\mathbf{z}}_i\| \|\hat{\mathbf{z}}_j\|$, where $\hat{\mathbf{z}}_i$ is the clusterer output for the i-th sample, and $\|.\|$ represents the ℓ_2-norm function.

5.3.1 Cluster-GAN Adversarial Loss

As shown in Figure 13, *ClusterGAN* consists of a discriminator $\mathscr{D}$, a generator $\mathscr{G}$ and a clusterer $\mathscr{C}$, in which the generator and clusterer aims to fool the discriminator by synthesizing realistic samples by $\mathscr{G} : \mathbf{z} \rightarrow \hat{\mathbf{x}}$ and similar latent variable to the generator inputs by $\mathscr{C} : \mathbf{x} \rightarrow \hat{\mathbf{z}}$, and the discriminator tries to distinguish the joint distribution of samples $(\hat{\mathbf{z}}, \mathbf{x})$ and $(\mathbf{z}, \hat{\mathbf{x}})$ coming from the clusterer and generator respectively.

In order to assist constructing the block diagonal adjacency matrix $\mathbf{A}$, we set the random input vectors of generator $\mathbf{z}$ to be orthogonal or parallel. To do so, we consider a binary random variable with m/c elements equal to 1 and the remaining equal to 0, where m is the length of $\mathbf{z}$ vector. In this case, if the distribution of clusterer output $\hat{\mathbf{z}}$ becomes similar to the generator input variables $\mathbf{z}$, we achieve the goal of an adjacency matrix with block diagonal structure. But in order to represent the intra-cluster variations, we add small uniform random noise to the inputs of the generator. This trick empirically helps to generate realistic samples with more diversity, and has insignificant effect on the block diagonal adjacency matrix.

As mentioned, the discriminator in *ClusterGAN* tries to discriminate the two joint distributions $P(\mathbf{z}, \hat{\mathbf{x}}) = P(\mathbf{z})P_{\mathscr{G}}(\mathbf{x}|\mathbf{z})$ and $P(\hat{\mathbf{z}}, \mathbf{x}) = P(\mathbf{x})P_{\mathscr{C}}(\mathbf{z}|\mathbf{x})$, which are coming form the generator and clusterer respectively. Since the generator random variable distribution $P(\mathbf{z})$ and the empirical distribution of real data $P(\mathbf{x})$ are known, our objective is to learn the conditional distribution of $P_{\mathscr{G}}(\mathbf{x}|\mathbf{z})$ and $P_{\mathscr{C}}(\mathbf{z}|\mathbf{x})$ to match the distributions $P(\mathbf{z}, \hat{\mathbf{x}})$ and $P(\hat{\mathbf{z}}, \mathbf{x})$. In order to acquire this condition, we employ the adversarial game between $\mathscr{D}$, $\mathscr{G}$ and $\mathscr{C}$ such that the discriminator is trained to identify whether joint pairs are sampled from $P(\mathbf{z}, \hat{\mathbf{x}})$ or $P(\hat{\mathbf{z}}, \mathbf{x})$, whereas the generator and clusterer are learned to fool the discriminator. Therefore, the objective function of this adversarial game for *ClusterGAN* is:

$$\min_{\mathscr{G},\mathscr{C}} \max_{\mathscr{D}} \; \mathbf{U}(\mathscr{D}, \mathscr{G}, \mathscr{C}) = \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\big[\log \mathscr{D}\big(\mathscr{C}(\mathbf{x}), \mathbf{x}\big)\big] + \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\big[\log\big(1 - \mathscr{D}\big(\mathbf{z}, \mathscr{G}(\mathbf{z})\big)\big)\big] . \quad (5.2)$$

Using this minimax objective function, we are able to alleviate the overfitting issue in training of a deep network with large complexity. This becomes more important in unsupervised clustering task, since there is no reliable supervised information to learn the deep clustering model. It can be shown that the optimal discriminator defined by this objective is balanced between the joint distribution of pairs belonging to the clusterer $P(\hat{\mathbf{z}}, \mathbf{x})$ and generator $P(\mathbf{z}, \hat{\mathbf{x}})$.

Lemma 1. *For any fixed $\mathscr{G}$ and $\mathscr{C}$, the optimal $\mathscr{D}$ defined by the utility function $\mathbf{U}(\mathscr{D}, \mathscr{G}, \mathscr{C})$ is:*

$$\mathscr{D}^*(\mathbf{z}, \mathbf{x}) = \frac{P(\mathbf{x})P_{\mathscr{C}}(\mathbf{z}|\mathbf{x})}{P(\mathbf{x})P_{\mathscr{C}}(\mathbf{z}|\mathbf{x}) + P(\mathbf{z})P_{\mathscr{G}}(\mathbf{x}|\mathbf{z})}$$

Given $\mathscr{D}^*(\mathbf{x}, \mathbf{z})$, we can further replace $\mathscr{D}$ in the utility function $\mathbf{U}(\mathscr{D}, \mathscr{G}, \mathscr{C})$ and reformulate the objective as $\mathbf{V}(\mathscr{G}, \mathscr{C}) = \max_{\mathscr{D}} \mathbf{U}(\mathscr{D}, \mathscr{G}, \mathscr{C})$, whose optimal value is shown in the following Lemma.

Lemma 2. *The global optimum point of $\mathbf{V}(\mathscr{G}, \mathscr{C})$ is achieved if and only if $P(\mathbf{z}, \hat{\mathbf{x}}) = P(\hat{\mathbf{z}}, \mathbf{x})$.*

Employing this adversarial game in *ClusterGAN*, we can attain the desired clusterer and generator for our problem. In fact, the generator is trained to synthesize the images similar to the real data distribution. The clusterer is trained to learn the inverse mapping function of the generator, estimating discriminative features for the real data. Thus, we can construct an almost block diagonal adjacency matrix from the clusterer outputs. In another point of view, this adversarial loss can be considered as a data-dependent regularization in training our deep clustering model, helping to avoid getting stuck in bad local minima. The proof for Lemma 1 and Lemma 2 are presented in the Appendix C.1 and C.2.

5.3.2 Cluster-GAN Entropy Minimization Loss

In addition to the adversarial loss, we introduce a clustering objective based on conditional entropy minimization, which is directly applied to the adjacency matrix constructed from the real data. Maximizing the mutual information or minimizing the conditional entropy has been reported to have successful results in clustering [15, 76]. The conditional entropy minimization loss in our problem has the following form:

$$\min_{\mathscr{C}} \quad -\sum_{i,j=1}^{n} \left[a_{ij}\log a_{ij} + (1-a_{ij})\log(1-a_{ij})\right], \tag{5.3}$$

in which the adjacency elements a_{ij} are pushed towards 0 or 1. Therefore, minimizing the conditional entropy is in favor of the block diagonal adjacency matrix. However, the similarity values computed from the clusterer features are not reliable especially at the first iterations of training. To tackle this issue, we can use the standard self-paced learning approach, which embeds gradual learning from easy to more difficult samples into model optimization as follows.

$$\min_{\mathscr{C},\boldsymbol{\nu}} \quad \sum_{i=1}^{n}\nu_i l_i - \lambda_\nu \sum_{i=1}^{n}\nu_i\,, \quad s.t.\ \boldsymbol{\nu}\in[0,1]^n \tag{5.4}$$

where $l_i = -\sum_{j=1}^{n} a_{ij}\log a_{ij} - (1-a_{ij})\log(1-a_{ij})$ is a loss related to the i-th sample, ν_i is the self-paced learning parameter, and λ_ν is a hyper-parameter for controlling the learning pace. The parameters of self-paced learning algorithm and clusterer are generally trained using an alternative learning strategy. Keeping the model parameters fixed, the globally optimum solution for the self-paced learning parameters is $\nu_i^* = 1$ if $l_i < \lambda_\nu$, and $\nu_i^* = 0$ otherwise. It is obvious that by increasing λ_ν throughout training, the self-paced learning algorithm allows more difficult samples into the training process. However, the standard self-paced learning does not consider selecting a balanced set of samples from all clusters, and may choose easy samples of only a few clusters. In order to address this issue, we

Algorithm 3: *ClusterGAN* Algorithm

```
1 for number of training iterations do
2     Sample a batch of pairs $\big(\mathbf{x}, \mathscr{C}(\mathbf{x})\big)$ and $\big(\mathscr{G}(\mathbf{z}), \mathbf{z}\big)$ using the clusterer and generator
3     Update the discriminator parameters by $\max_{\mathscr{D}} \sum_{i=1}^{n} \log \mathscr{D}\big(\mathscr{C}(\mathbf{x}_i), \mathbf{x}_i\big) + \sum_{j=1}^{n} \log\big(1 - \mathscr{D}\big(\mathbf{z}_j, \mathscr{G}(\mathbf{z}_j)\big)\big)$
4     Update the generator parameters by $\min_{\mathscr{G}} \sum_{j=1}^{n} \log\big(1 - \mathscr{D}\big(\mathbf{z}_j, \mathscr{G}(\mathbf{z}_j)\big)\big)$
5     Update the clusterer parameters by $\min_{\mathscr{C}} \sum_{j=1}^{n} \log \mathscr{D}\big(\mathscr{C}(\mathbf{x}_i), \mathbf{x}_i\big) + \nu_i l_i + \|\mathscr{C}(\mathbf{x}_i) - \mathscr{C}(\tilde{\mathbf{x}}_i)\|^2$
6     Update the self-paced learning parameters by $\min_{\boldsymbol{\nu}} \sum_{i=1}^{n} \nu_i l_i - \lambda_\nu \|\boldsymbol{\nu}\|_1 + \gamma \|\boldsymbol{\nu}\|_e$  $s.t.\ \boldsymbol{\nu} \in [0,1]^n$
7 end
```

propose balanced self-paced learning algorithm, which penalizes the lack of diversity using the following objective function:

$$\min_{\mathscr{C},\boldsymbol{\nu}} \quad \sum_{k=1}^{c} \Bigg[\sum_{i=1}^{n_k} \nu_{ki}(l_{ki} - \lambda_\nu) + \gamma \Big(\sum_{i=1}^{n_k} |\nu_{ki}| \Big)^2 \Bigg]$$
$$s.t. \quad \boldsymbol{\nu} \in [0,1]^n \,, \tag{5.5}$$

where γ is the regularization hyper-parameter, and ν_{ki} represents the self-paced learning parameter for the i-th sample of the k-th cluster, where the data are assumed to belong to c clusters as $\sum_{k=1}^{c} n_k = n$. The second term in the loss is also the exclusive lasso regularization $\|\boldsymbol{\nu}\|_e = \sum_{k=1}^{c} \big(\sum_{i=1}^{n_k} |\nu_{ki}| \big)^2$. Note that the balanced self-paced learning objective has two regularization terms, $-\|\boldsymbol{\nu}\|_1 = -\lambda_\nu \sum_{k=1}^{c} \sum_{i=1}^{n_k} \nu_{ki}$ that is in favor of selecting easier samples, and $\|\boldsymbol{\nu}\|_e$ that penalizes groups with more selected samples. Thus, the proposed balanced self-paced learning algorithm consider both the easiness and diversity of selected samples to ensure robust and unbiased training steps. In order to solve this objective function, we use an alternative learning approach, where the clusterer parameters are fixed while obtaining the self-paced learning parameters, and the self-paced parameters are assumed to be known while updating the clusterer parameters. Given the fixed $\mathscr{C}$, the objective function for estimating $\boldsymbol{\nu}$ is:

$$\min_{\boldsymbol{\nu}} \sum_{i=1}^{n} \nu_i l_i - \lambda_\nu \|\boldsymbol{\nu}\|_1 + \gamma \|\boldsymbol{\nu}\|_e \quad s.t.\ \boldsymbol{\nu} \in [0,1]^n. \tag{5.6}$$

Table 10: Dataset Descriptions

Dataset	# Samples	# Classes	# Dimensions
MNIST	70,000	10	1×28×28
USPS	11,000	10	1×16×16
FRGC	2,462	20	3×32×32
CIFAR-10	60,000	10	3×32×32
STL-10	13,000	10	3×96×96

We derive the global optimum solution for this optimization problem as shown in the following theorem.

Theorem 1. *For any fixed* $\mathscr{C}$*, the optimal* $\boldsymbol{\nu}^*$ *defined by the objective function in Eq. (5.6) is:*

$$\begin{cases} \nu^*_{kq} = 1, & if \;\; l_{kq} < \lambda_\nu - 2\gamma q \\ \nu^*_{kq} = \frac{\lambda_\nu - l_{kq}}{2\gamma} - q, & if \;\; \lambda_\nu - 2\gamma q \leq l_{kq} < \lambda_\nu - 2\gamma(q-1) \\ \nu^*_{kq} = 0, & if \;\; l_{kq} \geq \lambda_\nu - 2\gamma(q-1) \end{cases}$$

where $q \in \{1, ..., n_k\}$ *is the sorted index based on the loss values* $\{l_{k1}, ..., l_{kn_k}\}$ *in the k-th group.*

This solution intuitively makes sense, since the samples with loss greater/less than the threshold $\lambda_\nu - 2\gamma(q-1)$ are considered as the difficult/easy samples, and are not-involved/involved in the current training step. Interestingly, the threshold is also a function of the ordered loss in each group, and consequently is increased as the number of samples in a cluster increases. Hence, the balanced self-paced learning algorithm considers both the easiness and diversity of selected samples in our learning framework. The proof for Theorem 1 is presented in Appendix C.3.

In addition to the adversarial loss and the minimum entropy loss, we utilize a consistency loss to train the clusterer parameters. The consistency loss encourages the clusterer to have similar outputs for each samples $\mathbf{x}$ and its variations $\tilde{\mathbf{x}}$ augmented by image transformations or noise as follows:

$$\min_{\mathscr{C}} \sum_{i=1}^{n} \|\mathscr{C}(\mathbf{x}_i) - \mathscr{C}(\tilde{\mathbf{x}}_i)\|^2 \,. \tag{5.7}$$

The minimum entropy loss function in Eq. (5.3) is defined on the full-batch, and has quadratic complexity *w.r.t.* the number of samples. However, we practically alleviate this scalability issue by applying the loss only to the samples of each mini-batch. Algorithm 3 shows the training steps for *ClusterGAN*, where all of the networks are trained using our alternative leaning framework.

5.4 Experiments

We perform several experiments to evaluate the performance of *ClusterGAN* in clustering and information retrieval tasks on several datasets. We also examine the effect of each component in our learning framework using an ablation study.

Datasets: We examine *ClusterGAN* clustering performance in comparison with alternative algorithms on *MNIST* [83], *USPS*, *FRGC* [180], *CIFAR-10* [77] and *STL-10* [25] datasets. Moreover, we compare *ClusterGAN* with unsupervised hash functions in the image retrieval task on *MNIST* and *CIFAR-10* datasets. Table 10 provides the summary of datasets statistics.

Implementation details: We mainly use the architectures of *Triple-GAN* in [24] for *ClusterGAN* except the last layer of clusterer, which is set as same as the size of generator input with the sigmoid non-linearity. For image preprocessing, we only normalize the image intensities to be in the range of $[-1, 1]$ on *CIFAR-10* and *STL-10* and $[0, 1]$ for the others, and consequently use the tangent-hyperbolic and sigmoid functions in the last layer of the generator. The added noise to the generator inputs has uniform distribution with range

$[0, 0.5]$ which is linearly shrinking to $[0, 0.1]$ throughout training. Moreover, we set the learning rate to 10^{-4} and linearly decrease it to 10^{-5}, and adopt Adam [74] as our optimization method with the hyper-parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = 1e - 08$. In order to avoid manually setting λ_ν and γ for different datasets, we choose them based on the loss values of samples such that we start training with only 1% of samples at the first iteration, and then linearly increase λ_ν to include all samples in 3/4 of the maximum epoch. We run K-*means* on the clusterer outputs for clustering, and use the indicator function $\mathbf{1}(. > 0.5)$ to binarize the clusterer outputs for hashing. The reported results are all the average of 5 experimental outcomes. We use Theano toolbox [2] for writing our code, and run the algorithm on a machine with one Titan X Pascal GPU.

Table 11: Clustering performance of *ClusterGAN* and several alternative models on several datasets based on ACC and NMI. The results of other models are reported from the reference papers, except for the ones marked by (∗) on top, which are obtained by us running the released code. The result with † sign are for the models with supervised pre-training.

	Dataset	*MNIST*		*USPS*		*FRGC*		*CIFAR-10*		*STL-10*	
	Model	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
Shallow	*K-means*	0.500	0.534	0.450	0.460	0.287	0.243	0.102*	0.239*	0.209*	0.284*
	N-Cuts [137]	0.411	0.327	0.675	0.314	0.285	0.235	-	-	-	-
	SC-LS [23]	0.706	0.714	0.681	0.659	0.550	0.407	0.114*	0.258*	0.105*	0.168*
	AC-PIC [192]	0.017	0.115	0.840	0.855	0.415	0.320	0.118*	0.264*	0.235*	0.329*
	SEC [110]	0.779	0.804	0.511	0.544	-	-	0.107*	0.249*	0.245*	0.307*
	LDMGI [181]	0.802	0.842	0.563	0.580	-	-	0.109*	0.253*	0.260*	0.331*
Deep	*NMF-D* [148]	0.152	0.175	0.287	0.382	0.259	0.274	-	-	-	-
	DEC [174]	0.816	0.844	0.586	0.619	0.505	0.378	0.267*	0.312*	0.284*	0.359*
	JULE-RC [180]	0.913	0.964	0.913	0.950	0.574	0.461	0.194*	0.275*	0.204*	0.288*
	DEPICT [41]	0.917	0.965	0.927	0.964	0.610	0.470	0.274*	0.326*	0.303*	0.371*
	IMSAT [63]	-	**0.984**	-	-	-	-	-	0.456†	-	0.941†
	ClusterGAN	**0.921**	0.964	**0.931**	**0.970**	**0.615**	**0.476**	**0.323**	**0.412**	**0.335**	**0.423**

Table 12: Image retrieval results (%) of *ClusterGAN* and unsupervised hash functions on CIFAR-10 and MNIST datasets, when the number of hash bits are 32 and 64. The results of other models are reported from the reference papers, except for the ones marked by (∗) on top, which are obtained by us running the released code. The result with † sign are for the models with supervised pre-training.

	Dataset	*CIFAR-10*						*MNIST*					
		precision@1000		mAP		mAP@1000		precision@1000		mAP		mAP@1000	
	Model	32	64	32	64	32	64	32	64	32	64	32	64
Shallow	*KMH* [55]	19.72	20.16	13.93	14.46	23.56*	25.19*	53.82*	54.13*	33.29	35.78	70.32*	67.62*
	SphH [58]	20.91*	23.25*	14.58	15.38	24.16*	26.09*	54.74*	62.50*	30.77	34.75	65.45*	65.45*
	SpeH [163]	18.83	19.72	12.42	12.56	21.79*	21.97*	53.75*	54.13*	25.72	24.10	64.37*	67.60*
	PCAH [157]	19.35	18.73	12.60	12.10	21.62*	20.54*	51.90*	48.36*	24.85	21.47	64.47*	63.31*
	LSH [45]	19.10	22.25	13.76	15.07	16.31*	18.00*	45.05*	55.31*	25.83	31.71	50.45*	66.23*
	ITQ [47]	25.30	27.09	16.20	16.64	27.41*	28.93*	68.80*	71.00*	43.82	45.37	76.86*	80.23*
Deep	*DH* [36]	16.62	16.96	16.62	16.96	-	-	-	-	44.97	46.74	-	-
	DAR [64]	26.62	28.06	17.01	17.21	-	-	-	-	-	-	-	-
	DeepBit [93]	-	-	-	-	24.86†	27.73†	-	-	-	-	32.02†	44.53†
	UTH [66]	-	-	-	-	30.66†	32.41†	-	-	-	-	46.58†	49.88†
	ClusterGAN	**40.62**	**42.51**	**29.47**	**30.53**	**43.34**	**45.12**	**90.83**	**91.60**	**88.70**	**89.93**	**91.48**	**92.37**

5.4.1 Image Clustering

Alternative Models: We compare our clustering model with several baselines and state-of-the-art clustering algorithms, including *K-means*, normalized cuts (*N-Cuts*) [137], large-scale spectral clustering (*SC-LS*) [23], agglomerative clustering via path integral (*AC-PIC*) [192], spectral embedded clustering (*SEC*) [110], local discriminant models and global integration (*LDMGI*) [181], *NMF* with deep model (*NMF-D*) [148], deep embedded clustering (*DEC*) [174], joint unsupervised learning (*JULE-RC*) [180], *DEPICT* [41] and *IMSAT* [63].

Evaluation metrics: To compare the clustering performance of our model with previous studies, we rely on the two popular metrics used to evaluate clustering: *normalized mutual information* (NMI), and *accuracy* (ACC). NMI provides a measure of similarity between two data with the same label, which is normalized between 0 (lowest similarity) to 1 (highest similarity) [178]. To calculate ACC, we find the best map between the predicted clusters and the true labels [79].

Performance comparison: Table 13 shows the clustering results of *ClusterGAN* and the alternative models on five datasets. As it is expected, the deep clustering models mostly have better results than their shallow alternatives. Among the deep models, *ClusterGAN* outperforms the other methods almost on all datasets. Note that the *IMSAT* results on *CIFAR-10* and *STL-10* are obtained using the 50-layer pre-trained deep residual networks on ImageNet dataset [28], and cannot be compared to the results of other models trained with no supervisory signals. It is worth mentioning that *ClusterGAN* is able to train deeper clustering networks (*e.g.* 9 hidden layers on *CIFAR-10*) compared to the other deep models (*e.g.* 3 or 4 hidden layers on *CIFAR-10*). This effective learning framework could be the reason for *ClusterGAN*'s better performances on the more complex datasets like *CIFAR-10* and *STL-10*. This experiment confirms the efficiency *ClusterGAN* discriminative representations in clustering of different datasets with various sizes, dimensions and complexities.

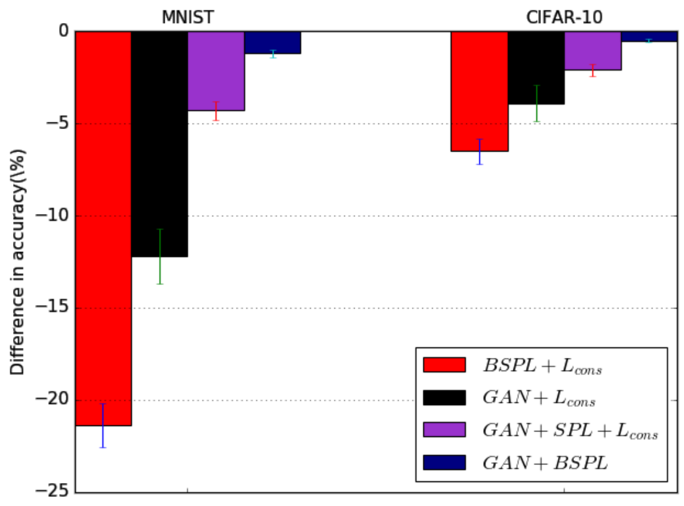


Figure 14: The difference in clustering accuracy, when *ClusterGAN* is trained using some components of the original objective function.

5.4.2 Ablation Study

We perform an ablation study to examine the contribution of the adversarial loss (GAN), the balanced self-paced learning algorithm ($BSPL$), and the consistency loss (L_{cons}). To do so, we train *ClusterGAN* without GAN architecture and adversarial loss ($BSPL$+L_{cons}), without the balanced self-paced learning algorithm (GAN+L_{cons}), and without the consistency loss (GAN+$BSPL$). Moreover, we explore the effect of exclusive lasso regularization in $BSPL$ by training *ClusterGAN* using the standard self-paced learning algorithm (GAN+SPL+L_{cons}). Figure 17 illustrates the difference in accuracy between each scenario and the original *ClusterGAN* on *MNIST* and *CIFAR-10* datasets.

The first observation is that all of the terms contribute in improving the results. Moreover, the figure shows the strong effect for GAN as a key components to avoid getting stuck in bad local minima. It also demonstrates that the balanced self-paced learning is important in stable training, and also has better results compared to standard self-paced learning approach. Furthermore, the relative analysis of the results in both dataset demonstrates that consistency loss is more effective on *CIFAR-10* than on *MNIST*. This is expected as we only use noise for image transformation on MNIST since the images are centered and scaled, but employ extra transformations including translations and horizontal flipping on *CIFAR-10*.

Moreover, we visualize the embedding subspace of a few clustering models on *USPS dataset* in Figure 15. The figure shows the 2D visualization of clusterer outputs for $BSPL{+}L_{cons}$ and *ClusterGAN* using principle component analysis (PCA). In addition, we also illustrate the raw data in the input space. As shown in the figure, *ClusterGAN* provides a significantly more discriminative embedding subspace compared to the other model and raw data.

5.4.3 Image Retrieval

Alternative models: For image retrieval, we compare our method with the previous unsupervised hash functions including K-means hashing (*KMH*) [55], spherical hashing (*SphH*) [58], spectral hashing (*SpeH*) [163], PCA-based hashing (*PCAH*) [157], locality sensitivity hashing (*LSH*) [45], iterative quantization (*ITQ*) [47], deep hashing (*DH*) [36], discriminative attributes representations (*DAR*) [64], *DeepBit* [93] and unsupervised triplet hashing (*UTH*) [66].

Evaluation metrics: We evaluate the performance of *ClusterGAN* compared to the other unsupervised hashing functions using precision and mean average precision (mAP) on *MNIST* and *CIFAR-10* datasets. We follow the standard protocol, and randomly sample 1000 images as the query set and use the remaining data as the gallery set. In particular, we report the results of the image retrieval in terms of precision@1000, mAP, and mAP@1000.

Performance comparison: Another way to measure the effectiveness of *ClusterGAN* discriminative representation is to evaluate its performance in hashing tasks. As shown in Table 15, *ClusterGAN* consistently outperforms alternative models with significant margins across different number of bits, datasets and metrics. Interestingly, the unsupervised deep hash functions, which use supervised pre-training via ImageNet dataset, show better results on *CIFAR-10* dataset compared to the shallow models, but have relatively lower performance on *MNIST* dataset due to the difference in transfer data distribution. With no need to supervised pre-training, our model is not affected by pre-training bias and achieves superior results on both datasets.

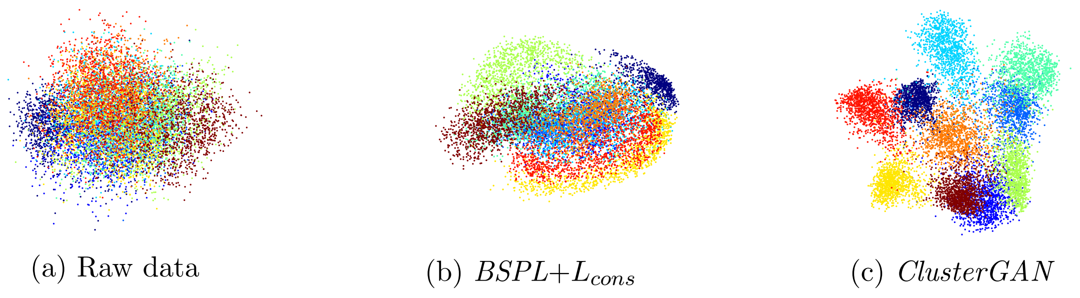

(a) Raw data (b) *BSPL*+L_{cons} (c) *ClusterGAN*

Figure 15: Visualization of different data representations on *USPS* dataset using principle component analysis (PCA) . (a) The space of raw data. (b) The embedding subspace of *ClusterGAN* without *GAN* architecture and adversarial loss denoted by *BSPL*+L_{cons}. (c) The embedding subspace of *ClusterGAN*.

5.5 Conclusion

In this project, we proposed a generative adversarial clustering network, denoted by *ClusterGAN*, as a new deep clustering model. *ClusterGAN* consists of three networks, a generator, a discriminator, and a clusterer. In order to efficiently train the deep clusterer without any supervised information, we introduced an adversarial game between the three networks, such that the generator synthesizes the realistic images given the discriminative random inputs, the clusterer inversely maps the real samples into the discriminative features. We further proposed a minimum entropy loss on the real data along with a balanced self-paced learning algorithm to enhance the training of the clusterer. The balanced self-paced learning algorithm improves the generalization of our clusterer by gradually decreasing the easiness of included samples in the training process, while considering the diversity of selected samples. Experimental results demonstrated that *ClusterGAN* achieves state-of-the-art results in the clustering and information retrieval tasks, and confirmed the effectiveness of each component in our learning framework using an ablation study.

6.0 Contrastive Generative Adversarial Network for Unsupervised Image Retrieval and Clustering

6.1 Introduction

The explosive growth of image data in the internet and social media has driven huge interest in efficient unsupervised models that are able to find similar patterns among the data. For instance, there are many studies on approximate nearest neighbor search (ANNS) algorithms, which aim to provide efficient image similarity search on large-scale image datasets. Hashing-based ANNS methods tackle this problem by representing image data with binary codes, providing an effective solution for the similarity search and storage of millions of images [47, 163, 94, 156, 89]. Categorizing similar/dissimilar images into the same/different sets is another essential task in machine learning and computer vision. This problem is extensively studied in the literature by introducing models that find discriminative boundaries between different image categories [177, 96]. These models are required to extract semantic features from image data related to their categories, and be robust to different image styles caused by spatial/geometric transformations and color distortions.

Supervised deep models have shown remarkable performance in image classification and retrieval by training their flexible mapping function using large sets of labeled data [55, 163, 157, 56, 65, 145]. However, unsupervised deep hashing and clustering models generally lag behind their supervised counterparts on image data, since the lack of reliable supervisory signals may lead to learning some arbitrary representations in deep models with large numbers of free parameters. In order to address this problem, some studies employ auxiliary reconstruction or generative loss functions as additional regularizations [174, 101, 18]. However, these regularizations usually enforce the models to contain some unnecessary generative information that is not directly relevant to the required ability of discriminative representations. Also the unsupervised deep models usually provide insignificant improvements compared to their shallow counterparts, and sometimes need a variant of supervised pretrainings to initialize their parameters [93, 66].

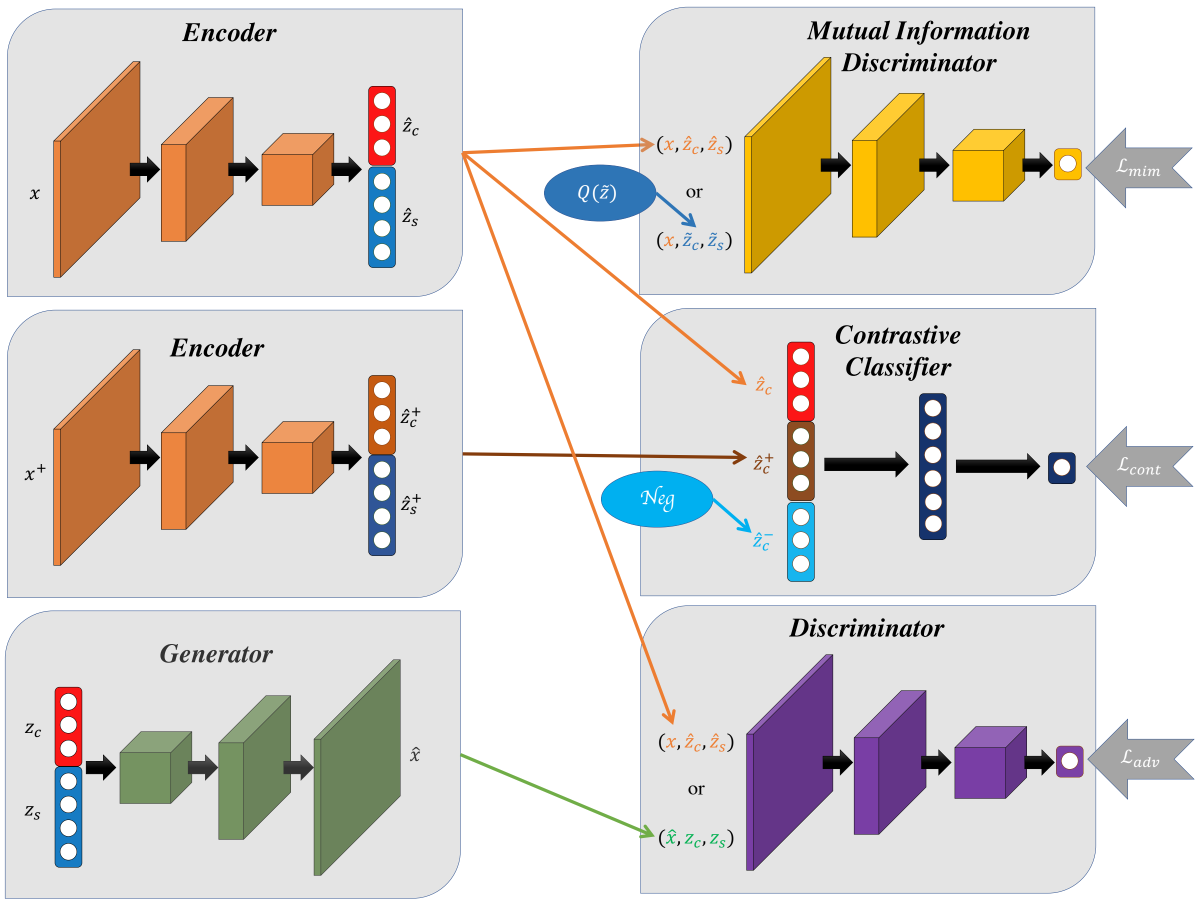


Figure 16: Architecture of *Contra-Info GAN*, consisting of an encoder to map input images into latent representations, a generator to synthesize images given latent variables, a discriminator to distinguish the encoder data from the generator data, a contrastive classifier to increase/decrease the distance of negative/positive data pairs, and a mutual information discriminator to preserve the relevant information of image data in the latent representations. Three loss functions, $\mathscr{L}_{mim}$, $\mathscr{L}_{cont}$ and $\mathscr{L}_{adv}$, are applied to train the networks.

In this project, we propose a new unsupervised learning framework for deep models in image retrieval and clustering tasks based on three loss functions: 1) We utilize an adversarial loss between three networks, including a critic, a generator, and an encoder (*i.e.* our unsupervised discriminative network), enforcing the generator to learn the conditional image distribution given a set of random content and style latent variables, the encoder to map the input images to a set of content and style latent representations, and the critic to distinguish its joint input data (*i.e.* a pair of image and latent features) belonging to the generator or encoder. This adversarial loss is not only beneficial for training the encoder due to the learned knowledge in the generator network, but also provides this opportunity to impose the desired constraints and prior to the content and style latent representations. 2) We also exploit the maximum mutual information loss to increase the correlation of between images and their latent representations. Instead of directly maximizing the Kullback-Leibler (*i.e.* KL) divergence between a latent representations of an image and other representations, we use a Jensen–Shannon (*i.e.* JS) divergence in a GAN-style sub-network to better approximate the mutual information. 3) We introduce a novel contrastive loss to disentangle the content and style representations by decreasing/increasing the distance between content features of an image (*i.e.* anchor point) and its augmented variant/other images (*i.e.* positive point/negative points). Our contrastive loss does not require a large batch size in training to cover enough negative pairs, and has small overhead to the computation time and memory size.

In summary, we present Contrastive Information-based Generative Adversarial Network, denoted by *Contra-Info GAN*, as an effcient solution for unsupervised image retrieval and clustering tasks. Our experimental results indicate that *Contra-Info GAN* achieves state-of-the-art results compared to alternative models on image retrieval and clustering. Moreover, we examine the effect of each component in our learning framework using an ablation study. Therefore, our main contributions of this project can be summarized as follows:

- Proposing a novel unsupervised learning framework for training deep models to map image data into disentangled content-style representations.
- Introducing a threefold loss function based on a contrastive learning of visual representations with small time and space complexities, approximation of maximum mutual

information via a JS-divergence, and an adversarial game for learning discriminative and generative pathways between images and latent representations.

- Outperforming state-of-the-art models on image retrieval and clustering with significant margins.

6.2 Methodology

Given N samples $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$ as the images in the training (*i.e.* gallery) set, *Contra-Info GAN* aims to find a discriminative encoder that maps the input images into a latent representation $\mathbf{z} = [\mathbf{z}_c, \mathbf{z}_s]$, consisting of disentangled content and style representations. We use K dimensional softmax layer or K independent sigmoid layers to obtain the content representation $\mathbf{z}_c$ in clustering or retrieval tasks, and a linear layer to obtain the style representation $\mathbf{z}_s$. While the content representation shows the probability of the input image belonging to a cluster or a binary hash code (*i.e.* inter-class information), the style representation indicates the other factors of variations in the image (*i.e.* intra-class information). As shown in Figure 1, we employ three loss functions in our learning framework to accomplish this goal, and describe them with more details in the following sections.

6.2.1 Contra-Info GAN Adversarial Loss

The vanilla *GAN* is formulated as an adversarial game between two networks, a discriminator and a generator [48]. In particular, the generator $\mathcal{G}$ is supposed to synthesize realistic images to fool the discriminator $\mathcal{D}$, and the discriminator aims to distinguish the real data from the generated samples. However, the adversarial game in our learning framework includes three networks, a discriminator $\mathcal{D}$, a generator $\mathcal{G}$ and an encoder $\mathcal{E}$. The generator aims to synthesize realistic images as $\mathcal{G} : \mathbf{z} \rightarrow \hat{\mathbf{x}}$, and the encoder tries to map the input images into latent representations similar to the generator inputs as $\mathcal{E} : \mathbf{x} \rightarrow \hat{\mathbf{z}}$. Given a pair set of images and latent features as the input, the discriminator is supposed to distinguish whether the paired samples belonging to the generator $(\mathbf{z}, \hat{\mathbf{x}})$ or the encoder $(\hat{\mathbf{z}}, \mathbf{x})$. We utilize the adversarial loss in Wasserstein GAN with gradient penalty (*i.e.* WGAN-GP) [50] to learn the joint distribution of images and latent representations through the generator and encoder using the following objective,

$$\begin{aligned}\min_{\mathscr{G},\mathscr{E}} \max_{\mathscr{D}} \quad & \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\left[\mathscr{D}\big(\mathscr{E}(\mathbf{x}),\mathbf{x}\big)\right] - \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\left[\mathscr{D}\big(\mathbf{z},\mathscr{G}(\mathbf{z})\big)\right] \\ & + \lambda_W \mathbb{E}_{(\mathbf{z},\mathbf{x})\sim P(\bar{\mathbf{z}},\bar{\mathbf{x}})}\left[\big(\|\nabla_{\mathbf{z},\mathbf{x}}\mathscr{D}(\mathbf{z},\mathbf{x})\|_2 - 1\big)^2\right], \end{aligned} \tag{6.1}$$

where $P(\mathbf{x})$ is the real images distribution, and $P(\mathbf{z})$ is the generator random input distribution consisting of the content and style latent variables as $\mathbf{z} = [\mathbf{z}_c, \mathbf{z}_s]$. The content latent variables sampled from a categorical distribution $\mathbf{z}_c \sim Cat(K, 1/K)$ in clustering and K independent random binary distributions as $\mathbf{z}_c^j \sim Bernoulli(\mathscr{U}(0,1))$ in image retrieval. The style latent variables sampled from a normal distribution $\mathbf{z}_s \sim \mathscr{N}(0,1)$. $P(\bar{\mathbf{z}}, \bar{\mathbf{x}})$ is used for sampling uniformly along straight lines between pairs of images and latent features as $\bar{\mathbf{z}} = \epsilon\mathbf{z} + (1-\epsilon)\hat{\mathbf{z}}$ and $\bar{\mathbf{x}} = \epsilon\mathbf{x} + (1-\epsilon)\hat{\mathbf{x}}$, where $\hat{\mathbf{z}} = \mathscr{E}(\mathbf{x})$ and $\hat{\mathbf{x}} = \mathscr{G}(\mathbf{z})$. λ_W is the gradient penalty hyper-parameter, adjusting the 1-Lipschitz constraint effect on the discriminator (also known as critic in WGAN-GP).

This loss function assists the encoder to exploit the learned knowledge in the generation pathway by learning the inverse function of synthesizing realistic images from the latent variables in the generator. Moreover, we are able to easily impose the desired constraints (*i.e.* prior) on the encoder latent representations $\hat{\mathbf{z}}$ via the chosen distributions for the generator latent variables $\mathbf{z}$. This point helps us to avoid degenerate solutions on the encoder such as assigning all images to a few clusters, allocating a cluster to a few outlier samples, or having imbalanced frequency in hash bits. It is also beneficial to directly use $\hat{\mathbf{z}}$ as the cluster assignments or hash codes, since $\hat{\mathbf{z}}$ is enforced to become like one-hot or binary vectors of $\mathbf{z}$. We also explored the helpfulness of pretraining the generator network using the vanilla adversarial loss in GAN or WGAN-GP, since the generator task in synthesizing realistic images is way more difficult than the encoder task in mapping input images to the latent representations. Our experimental results show that this pretraining is useful in stabilizing our adversarial loss and achieving better results. In addition, we chose the Wasserstein distance instead of other GAN divergence measures, because it is continuous everywhere,

Algorithm 4: *Contra-Info GAN* algorithm for image clustering

```
Input: Unlabeled Dataset $\mathbf{X} = \{\mathbf{x}^i\}_{i=1}^N$ and number of clusters $K$; Initial parameters of the encoder $\mathscr{E}$, generator $\mathscr{G}$, discriminator $\mathscr{D}$, mutual information discriminator $\mathscr{M}$, and contrastive classifier $\mathscr{C}$; Hyper-parameters $M$, $B$, $\lambda_W$, $n_{critic}$, $\lambda_{adv}$, $\lambda_{cont}$, $\lambda_{mim}$; Augmentation function $T$
while $\mathscr{L}_{adv}$ not converged do
    for $t = 1, ..., n_{critic}$ do
        $\{\mathbf{x}^i\}_{i=1}^B \sim P(\mathbf{x})$ ;   // Sample image data
        $\{\mathbf{z}^i = [\mathbf{z}_c^i, \mathbf{z}_s^i]\}_{i=1}^B \sim [Cat(K, 1/K), \mathcal{N}(0,1)]$ ;   // Sample latent variables
        $\hat{\mathbf{z}}^i \leftarrow \mathscr{E}(\mathbf{x}^i) \quad \forall i \in [1, ..., B]$ ;   // Obtain latent representations
        $\hat{\mathbf{x}}^i \leftarrow \mathscr{G}(\mathbf{z}^i) \quad \forall i \in [1, ..., B]$ ;   // Generate images
        $\bar{\mathbf{z}}^i \leftarrow \epsilon \mathbf{z}^i + (1-\epsilon)\hat{\mathbf{z}}^i$ ;   // Mix representations
        $\bar{\mathbf{x}}^i \leftarrow \epsilon \mathbf{x}^i + (1-\epsilon)\hat{\mathbf{x}}^i \quad \forall i \in [1, ..., B]$ ;   // Mix images
        $\theta_{\mathscr{D}} \leftarrow \max_{\mathscr{D}} \sum_{i=1}^B \mathscr{D}\big(\hat{\mathbf{z}}^i, \mathbf{x}^i\big) - \mathscr{D}\big(\mathbf{z}^i, \hat{\mathbf{x}}^i\big) + \lambda_W \big(\|\nabla_{\bar{\mathbf{z}}^i, \bar{\mathbf{x}}^i} \mathscr{D}(\bar{\mathbf{z}}^i, \bar{\mathbf{x}}^i)\|_2 - 1\big)^2$ ;   // Train $\mathscr{D}$
    end
    Sample latent variables $[\mathbf{z}_c]_{i=1}^B \sim Cat(K, 1/K)$, $[\mathbf{z}_s]_{i=1}^B \sim \mathcal{N}(0,1)$ ;   // Sample latent variables
    Sample image data $[\mathbf{x}^i]_{i=1}^B \sim P(\mathbf{x})$ ;   // Sample image data
    $\hat{\mathbf{z}}^i = [\hat{\mathbf{z}}_c^i, \hat{\mathbf{z}}_s^i] \leftarrow \mathscr{E}(\mathbf{x}^i) \quad \forall i \in [1, ..., B]$ ;   // Obtain latent representations
    $\hat{\mathbf{z}}^{i+} = [\hat{\mathbf{z}}_c^{i+}, \hat{\mathbf{z}}_s^{i+}] \leftarrow \mathscr{E}(T(\mathbf{x}^i)) \quad \forall i \in [1, ..., B]$ ;   // Obtain latent representations of positive samples
    $\hat{\mathbf{z}}^{ij-} = [\hat{\mathbf{z}}_c^{ij-}, \hat{\mathbf{z}}_s^{ij-}] \leftarrow Neg^i \quad \forall i, j \in [1, ..., B], [1, ..., M]$ ;   // Obtain latent representations of negative samples
    $\theta_{\mathscr{G}} \leftarrow \min_{\mathscr{G}} - \sum_{i=1}^B \mathscr{D}\big(\mathbf{z}^i, \mathscr{G}([\mathbf{z}_c^i])\big)$ ;   // Train $\mathscr{G}$
    $\theta_{\mathscr{C}} \leftarrow \min_{\mathscr{C}} - \sum_{i=1}^B \log \frac{exp\left[\mathscr{C}\left(\hat{\mathbf{z}}_c^i, \hat{\mathbf{z}}_c^{i+}\right)\right]}{exp\left[\mathscr{C}\left(\hat{\mathbf{z}}_c, \hat{\mathbf{z}}_c^+\right)\right] + \sum_{j=1}^M exp\left[sim\left(\hat{\mathbf{z}}_c, \hat{\mathbf{z}}_c^-\right)\right]}$ ;   // Train $\mathscr{C}$
    $\theta_{\mathscr{M}} \leftarrow \max_{\mathscr{M}} \sum_{i=1}^B \log \mathscr{M}\big(\hat{\mathbf{z}}^i, \mathbf{x}^i\big) + \sum_{i=1, j\neq i}^B \log\big(1 - \mathscr{M}\big(\hat{\mathbf{z}}^j, \mathbf{x}^i\big)\big)$ ;   // Train $\mathscr{M}$
    $\theta_{\mathscr{E}} \leftarrow \min_{\mathscr{E}} \mathscr{L}_{tot}^{\mathscr{E}}$ ;   // Train $\mathscr{E}$ using $\mathscr{L}_{tot}^{\mathscr{E}}$ in Eq. 6.6
end
```

does not suffer from vanishing or exploding gradient issues, and can be used to estimate the training convergence of the encoder.

6.2.2 Contrastive Loss of Contra-Info GAN

In order to attain the disentangled latent representations on the encoder, we introduce a contrastive loss based on the fact that image transformations should not change the content representations $\hat{\mathbf{z}}_c$ and may only affect the style representations $\hat{\mathbf{z}}_s$ representing the other factors of variations in images. In particular, we augment each image with transformations such as rotation, scaling, cropping, and color jittering, and enforce the encoder to have similar content representations for an image $\mathbf{x}$ and its augmented variant $\mathbf{x}^+ = T(\mathbf{x})$. To do so, we use the following contrastive loss function:

$$\min_{\mathscr{E}, sim} -\mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})} \log \frac{e^{sim(\hat{\mathbf{z}}_c, \hat{\mathbf{z}}_c^+)}}{e^{sim(\hat{\mathbf{z}}_c, \hat{\mathbf{z}}_c^+)} + \sum_{\mathbf{x}^- \in Neg} e^{sim(\hat{\mathbf{z}}_c, \hat{\mathbf{z}}_c^-)}}, \tag{6.2}$$

where $\hat{\mathbf{z}}_c$ is the content representation for image $\mathbf{x}$ as $[\hat{\mathbf{z}}_c, \hat{\mathbf{z}}_s] = \mathscr{E}(\mathbf{x})$, $\hat{\mathbf{z}}_c^+$ is the content representation for the augmented image $\mathbf{x}^+ = T(\mathbf{x})$ as $[\hat{\mathbf{z}}_c^+, \hat{\mathbf{z}}_s^+] = \mathscr{E}(\mathbf{x}^+)$, $\hat{\mathbf{z}}_c^-$ is the content representation for another random image $\mathbf{x}^-$ as $[\hat{\mathbf{z}}_c^-, \hat{\mathbf{z}}_s^-] = \mathscr{E}(\mathbf{x}^-)$, and sim is a similarity function like dot product. Intuitively, the loss value is low when $\hat{\mathbf{z}}_c$ is similar to its augmented variant $\hat{\mathbf{z}}_c^+$ (*i.e.* positive example) and dissimilar to latent representations of other images $\hat{\mathbf{z}}_c^-$ (*i.e.* negative examples).

In order to define the set of negative examples for each image (Neg in Eq. 6.2), the previous contrastive learning studies either use the samples in the current mini-batch [20, 114] or a memory bank containing the representations of all samples in the dataset [170]. While the large number of negative examples is often required to have efficient unsupervised contrastive loss, the former studies have challenges with the GPU memory size due to the large mini-batch size, and the latter studies suffer from less consistent representations because of the slow update rate of sample representations in the memory bank (*i.e.* only one update per epoch when the sample is seen during training).

We introduce an efficient contrastive loss in our learning framework that does not limit the number of negative examples by the mini-batch size and benefits from consistent representations of the samples throughout training. To provide an approach with small computation time and memory size overhead, we maintain a queue of representations obtained from the content latent variables of the generator as $Neg = [\mathbf{z}_c^1, ..., \mathbf{z}_c^M]$, where M is the predefined number of negative examples. The samples of the queue are progressively replaced, such that the content representations of the new generated images that fool the discriminator $\mathscr{D}$ are added to the queue while the same number of older samples are removed. Note that we exclude the positive examples of each anchor point $\hat{\mathbf{z}}_c$ from its Neg set by removing the examples similar to the binarized version of $\hat{\mathbf{z}}_c$. Using this approach the size of the queue can be much larger than the mini-batch size. We consider a fully connected networks with one hidden layer as sim function in Eq. 6.2 that can be seen as a $M + 1$-way deep classifier.

6.2.3 Maximum Mutual Information Estimation in Contra-Info GAN

The encoder network may suffer from learning arbitrary representations for the input images due to its flexible non-linear mapping function and lack of reliable supervisory signals. To avoid this problem, we leverage the mutual information maximization to preserve the relevant information in the latent representations regarding the image retrieval and clustering tasks. The mutual information between an image $\mathbf{x}$ and its encoded latent representation $\hat{\mathbf{z}}$ is defined as follows,

$$\begin{aligned} I(\mathbf{x}, \hat{\mathbf{z}}) &= \iint P(\hat{\mathbf{z}}|\mathbf{x})P(\mathbf{x}) \log \frac{P(\hat{\mathbf{z}}|\mathbf{x})P(\mathbf{x})}{Q(\hat{\mathbf{z}})P(\mathbf{x})} d\mathbf{x} d\hat{\mathbf{z}} \\ &= KL\big(P(\hat{\mathbf{z}}|\mathbf{x})P(\mathbf{x}) \| Q(\hat{\mathbf{z}})P(\mathbf{x})\big) , \end{aligned} \tag{6.3}$$

where $P(\hat{\mathbf{z}}|\mathbf{x})$ is the encoder output distribution, $P(\mathbf{x})$ is the input image distribution, and $Q(\hat{\mathbf{z}}) = \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\big[P(\hat{\mathbf{z}}|\mathbf{x})\big]$ is the empirical posterior distribution of the latent representations. In order to alleviate the issues of maximizing the unbounded KL-divergence in Eq. 6.3, we follow [62] and replace the KL-divergence with more stable JS-divergence to approximate the mutual information as

$$I(\mathbf{x}, \hat{\mathbf{z}}) \approx JS\big(P(\hat{\mathbf{z}}|\mathbf{x})P(\mathbf{x}), Q(\hat{\mathbf{z}})P(\mathbf{x})\big) . \tag{6.4}$$

Note that our goal is maximizing the mutual information and not obtaining its precise value, thus we can use the JS-divergence as the optimization criterion instead of the KL-divergence. One effective way to estimate the JS-divergence between two distributions is to employ the adversarial game in the GAN framework [48, 112]. To do so, we employ a discriminator to approximate the mutual information using the following objective,

$$\begin{aligned} \max_{\mathscr{M},\mathscr{E}} \;\; & \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\big[\log \mathscr{M}\big(\mathscr{E}(\mathbf{x}), \mathbf{x}\big)\big] \\ & + \mathbb{E}_{\mathbf{x},\mathbf{z}\sim P(\mathbf{x}),Q(\hat{\mathbf{z}})}\big[\log\big(1 - \mathscr{M}\big(\mathbf{z}, \mathbf{x}\big)\big)\big] , \end{aligned} \tag{6.5}$$

where $\mathscr{M}$ is the discriminator used for estimating the mutual information. Optimizing this objective function encourages the encoder network to increase the relevance of an image $\mathbf{x}$ and its latent representation $\hat{\mathbf{z}} = \mathscr{E}(\mathbf{x})$ rather than the representation of another image using the discriminator $\mathscr{M}$ by considering $\big(\mathscr{E}(\mathbf{x}^i), \mathbf{x}^i\big)$ as a positive sample and $\big(\mathscr{E}(\mathbf{x}^j), \mathbf{x}^i\big)$ as a negative sample.

Algorithm 4 shows the training steps of our learning framework in the clustering task, in which the encoder $\mathscr{E}$ maps image data into the disentangled latent representations, the generator $\mathscr{G}$ synthesizes realistic images given the latent variables, discriminator $\mathscr{D}$ distinguishes whether its input belonging to the generator or encoder, the discriminator $\mathscr{M}$ helps in approximating the mutual information, and the contrastive classifier $\mathscr{C}$ enforce the encoder to have transformation-invariant representations. The loss function for updating the parameters of the encoder is

$$\mathscr{L}^{\mathscr{E}}_{tot} = \lambda_{adv}\mathscr{L}_{adv} + \lambda_{cont}\mathscr{L}_{cont} + \lambda_{mim}\mathscr{L}_{mim}\,, \tag{6.6}$$

where, $\mathscr{L}_{adv}$ is the adversarial loss shown in Eq. 6.1, $\mathscr{L}_{cont}$ is the contrastive loss presented in Eq. 6.2, $\mathscr{L}_{mim}$ is the maximum mutual information objective indicated in Eq. 6.5, and λ_{adv}, λ_{cont} and λ_{mim} are the hyper-parameters balancing the effect of components.

6.3 Experiments

Datasets: We compare our model with clustering models on five datasets including *MNIST* containing $70,000$ gray images with (28×28) size [83], *Fashion-MNIST* containing $70,000$ gray images with (28×28) size [173], *ImageNet-10* [19] containing color $13,000$ images with (96×96) size, *CIFAR-10* containing color $60,000$ images with (32×32) size [77] and *STL-10* [25] containing color $13,000$ images with (96×96) size. All of the datasets have 10 clusters. Following [68], we convert the color images to gray images to discourage clustering based on trivial colour cues. In addition, we use the *CIFAR-10* and *MNIST* datasets for comparing our model on the image retrieval task.

Implementation Details: For the encoder $\mathscr{E}$, generator $\mathscr{G}$, discriminator $\mathscr{D}$ and mutual information discriminator $\mathscr{M}$, we mainly use similar architectures to [50, 101] with different number of layers and units for different sizes of input images. The contrastive classifier $\mathscr{C}$ is a multi-layer fully connected network. The dimension of $\mathbf{z}_s$ and $\hat{\mathbf{z}}_s$ is set to 100, and the dimension of $\hat{\mathbf{z}}_c$ is set to the priori known number of clusters or size of hash bits. We also set the learning rate to 10^{-4} and linearly decrease it to 10^{-5}, and adopt Adam [74] as our optimization method with the hyper-parameters $\beta_1 = 0.5$, $\beta_2 = 0.9$. We add a small uniform random noise to the content latent representations $\mathbf{z}_c$ of the generator (*i.e.* $1 \rightarrow 1-u$ and $0 \rightarrow 0+u$ where $u \sim \mathcal{U}[0, 0.1]$) to make the discriminator job more difficult. Since the image retrieval and clustering tasks are naturally unsupervised, we did not tune any hyper-parameters using the supervisory signals. We set the WGAN hyper-parameters $\lambda_W = 10$ and $n_{critic} = 5$, the batch size $B = 64$, the Neg set size $M = 2048$, and the loss weights $\lambda_{adv} = 1$ and $lambda_{mim} = 1$. Since the contrastive loss has different effects on easy and difficult datasets (*i.e. MNIST* vs. *CIFAR-10*), we set the $\lambda_{cont} = 2$ for *MNIST* and *Fashion-MNIST* and $\lambda_{cont} = 4$ for the other datasets. The data augmentation function includes cropping, horizontal flipping, color jittering and channel shuffling. In particular, we crop images with the randomly sampled aspect ratio and area from the range of $[3/4, 4/3]$ and $[40\%, 100\%]$ respectively, and resize them back to the original image size. The input images are also flipped horizontally with 50% probability except for the MNIST dataset. We scale brightness of images with random weights sampled from $[0.6, 1.4]$, and hue with random coefficients sampled from $[0.875, 1.125]$. The RGB channels of color images are also randomly shuffled before graying the images.

Table 13: Performance of clustering models on five datasets based on ACC and NMI. The results of alternative models are reported from the reference papers, except for the ones marked by † on top, which are obtained by us running the released code. The results with ‡ sign are for the models with supervised pre-training.

Dataset	*MNIST*		*Fashion-MNIST*		*ImageNet-10*		*CIFAR-10*		*STL-10*	
Model	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
K-means	0.500†	0.568†	0.510†	0.472†	0.118†	0.239†	0.090†	0.235†	0.136†	0.228†
N-Cuts [137]	0.411	0.327	0.575	0.508	0.151	0.274	0.103	0.247	0.098	0.159
AC [49]	0.609	0.609	0.570†	0.502†	0.138	0.242	0.105	0.228	0.239	0.332
SC-LS [23]	0.706	0.714	0.662†	0.657†	0.118†	0.246†	0.114	0.258	0.105	0.168
NMF [16]	0.608	0.545	0.425	0.434	0.138	0.242	0.105	0.228	0.239	0.239
NMF-D [148]	0.152	0.175	0.297†	0.386†	0.103†	0.184†	0.078†	0.200†	0.208†	0.222†
DEC [174]	0.772	0.843	0.546	0.518	0.282	0.381	0.301	0.257	0.276	0.359
JULE-RC [180]	0.913	0.964	0.608	0.563	0.175	0.300	0.192	0.272	0.182	0.277
DEPICT [41]	0.917	0.965	0.392	0.392	0.170†	0.252†	0.274	0.326	0.303	0.371
IMSAT [63]	-	0.984	-	-	-	-	-	0.456‡	-	0.941‡
ClusterGAN-1 [101]	0.890	0.950	0.640	0.630	-	-	-	-	-	-
ClusterGAN-2 [40]	0.921	0.964	-	-	-	-	0.323	0.412	0.335	0.423
DAC [19]	0.935	0.978	0.588†	0.615†	0.396	0.522	0.366	0.470	0.394	0.527
RTM [111]	0.933	0.968	0.685	0.710	-	-	0.197	0.309	-	-
IIC [68]	-	**0.992**	0.610†	0.657†	-	-	-	0.617	-	0.499
DCCM [169]	-	-	-	-	0.608	0.710	0.496	0.408	0.376	0.482
Contra-Info GAN	**0.970**	0.989	**0.710**	**0.758**	**0.645**	**0.744**	**0.572**	**0.659**	**0.488**	**0.530**

Table 14: Performance of Cluster assignments with/without K-means on $\hat{\mathbf{z}}_c$, $\hat{\mathbf{z}}_s$ and $\hat{\mathbf{z}}$.

Dataset	*MNIST*		*Fashion-MNIST*		*ImageNet-10*		*CIFAR-10*		*STL-10*	
Method	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
K-means on $\hat{\mathbf{z}}$	0.491	0.538	0.496	0.502	0.122	0.297	0.123	0.300	0.423	0.479
K-means on $\hat{\mathbf{z}}_s$	**0.970**	**0.989**	**0.710**	**0.758**	0.638	0.740	0.568	0.656	0.486	0.527
K-means on $\hat{\mathbf{z}}_c$	**0.970**	0.987	**0.710**	**0.758**	0.642	0.741	0.570	0.657	**0.488**	**0.532**
Argmax on $\hat{\mathbf{z}}_c$	**0.970**	**0.989**	**0.710**	**0.758**	**0.645**	**0.744**	**0.572**	**0.659**	**0.488**	0.530

Evaluation Metrics: To compare the performance of clustering models, we rely on the two widely used metrics, *normalized mutual information* (NMI) and *accuracy* (ACC). NMI measures the similarity between two data with the same label, and is normalized between 0 (lowest similarity) to 1 (highest similarity) [178]. Following [79], we find the best map between the predicted clusters and the true labels to calculate ACC. We also evaluate the performance of hashing functions using precision and mean average precision (mAP). We follow the standard protocol for *MNIST* and *CIFAR-10*, and randomly sample 1000 images (100 per class) as the query set and use the remaining data as the gallery set. In particular, we report the results of the image retrieval in terms of precision@1000, mAP, and mAP@1000, where precision@1000 is the fraction of correctly retrieved samples from the top 1000 retrieved samples in gallery, mAP is the mean of the average precision of query images over all the relevant images, mAP@1000 is mAP calculated over the top 1000 ranked images from the gallery set. The reported results in image retrieval and clustering tasks are the average of 5 runs.

Performance Comparison on Image Clustering: Table 13 shows the clustering results of *Contra-Info GAN* and several alternative models on the five datasets. *Contra-Info GAN* outperforms the other methods on all distastes with large margins based on the both evaluation metrics except *MNIST*. While *IIC* has a slightly better accuracy on *MNIST* than *Contra-Info GAN*, our model has significantly better performance than *IIC* on *Fashion-MNIST*, *CIFAR-10* and *STL-10* datasets. Note that the *IMSAT* results on *CIFAR-10* and *STL-10* are obtained using the 50-layer pre-trained deep residual networks on ImageNet dataset [28], and cannot be compared to the results of other models trained

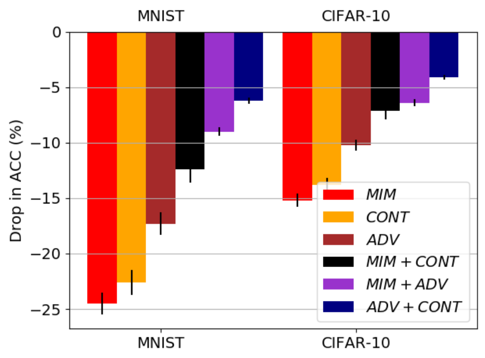
Table 15: Image retrieval results (mAP and mAP@1000) of unsupervised hash functions on CIFAR-10 and MNIST datasets, when the number of hash bits are 16, 32 and 64. The results of alternative models are reported from the reference papers.

Dataset	*CIFAR-10*									*MNIST*								
	mAP (%)			mAP@1000 (%)			precision@1000 (%)			mAP (%)			mAP@1000 (%)			precision@1000 (%)		
Model	16	32	64	16	32	64	16	32	64	16	32	64	16	32	64	16	32	64
KMH [55]	13.59	13.93	14.46	24.08	23.56	25.19	18.83	19.72	20.16	32.12	33.29	35.78	59.12	70.32	67.62	51.08	53.82	54.13
SphH [58]	13.98	14.58	15.38	24.52	24.16	26.09	18.90	20.91	23.25	25.81	30.77	34.75	52.97	65.45	65.45	46.31	54.74	62.50
SpeH [163]	12.55	12.42	12.56	22.10	21.79	21.97	18.83	19.72	20.16	26.64	25.72	24.10	59.72	64.37	67.60.	51.08	53.75	54.13
PCAH [157]	12.91	12.60	12.10	21.52	21.62	20.54	18.89	19.35	18.73	27.33	24.85	21.47	60.98	64.47	63.31	51.79	51.90	48.36
LSH [45]	12.55	13.76	15.07	12.63	16.31	18.00	16.21	19.10	22.25	20.88	25.83	31.71	42.10	50.45	66.23	31.95	45.05	55.31
ITQ [47]	15.67	16.20	16.64	26.71	27.41	28.93	22.46	25.30	27.09	41.18	43.82	45.37	70.06	76.86	80.23	61.94	68.80	71.00
DH [36]	16.17	16.62	16.96	-	-	-	16.17	16.62	16.96	43.14	44.97	46.74	-	-	-	-	-	-
DAR [64]	16.82	17.01	17.21	-	-	-	24.54	26.62	28.06	-	-	-	-	-	-	-	-	-
DeepBit [93]	-	-	-	19.43	24.86	27.73	-	-	-	-	-	-	28.18	32.02	44.53	-	-	-
UTH [66]	-	-	-	28.66	30.66	32.41	-	-	-	-	-	-	43.15	46.58	49.88	-	-	-
DistillHash	28.44	28.53	28.67	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HashGAN [44]	29.94	31.47	32.53	44.65	46.34	48.12	41.76	43.62	45.51	91.13	92.70	93.93	94.31	95.48	96.37	93.52	94.83	95.60
Contra-Info GAN	**34.63**	**36.12**	**38.78**	**48.15**	**51.32**	**54.98**	**45.34**	**47.72**	**50.08**	**93.71**	**95.04**	**95.89**	**96.66**	**97.35**	**98.12**	**95.08**	**96.12**	**97.11**

with no supervisory signals. The improvements over the existing models especially on more complex datasets, like *ImageNet-10*, *CIFAR-10* and *STL-10*, confirm the effectiveness of our learning framework in unsupervised training of deep clustering models.

Evaluation of Latent Representations: In order to assess the capability of *Contra-Info GAN* in learning disentangled latent representations, we explore the clustering performance of $\mathbf{z}$, $\mathbf{z}_c$ and $\mathbf{z}_s$ using Argmax and K-means. Table 14 demonstrates the results of four different ways of obtaining the cluster assignments. Similar performance on $\mathbf{z}_c$ with or without K-means indicates the effectiveness of $\mathscr{L}_{adv}$ on imposing desired constraints and prior to the content representations. In addition, the results K-means on $\mathbf{z}$, $\mathbf{z}_c$ are similar, but the outcomes of K-means on $\mathbf{z}_s$ are considerably worse than the other two. This results supports the claim that the encoder network is able to learn disentangled representations on image data.

Performance Comparison on Image Retrieval: Table 15 indicates the results of *Contra-Info GAN* and several hashing models on *CIFAR-10* and *MNIST* across different hash bit sizes. The results demonstrate that *Contra-Info GAN* consistently outperforms other models with significant margins with different numbers of bits and evaluation metrics on the both datasets. *Contra-Info GAN* has higher improvements on large hash bit size



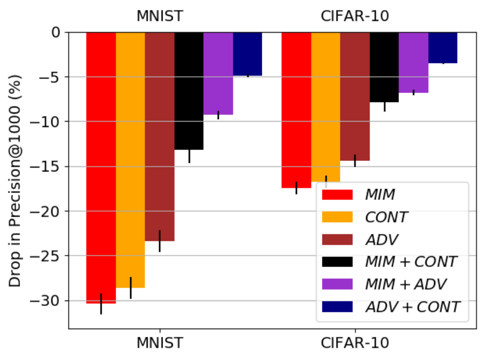


Figure 17: Ablation study of the adversarial loss (ADV), the contrastive loss (CONT) and the mutual information loss (MIM) in image clustering (left figure) and image retrieval (right figure).

relatively, utilizing the hash codes more efficiently. The unsupervised deep hash functions with supervised pretraining on ImageNet dataset, like *DeepBit* and *UTH*, have better results on *CIFAR-10* dataset compared to the shallow models, but have relatively lower performance on *MNIST* dataset. This shows that pretraining on the ImageNet dataset is helpful on *CIFAR-10*, which has a similar distribution to ImageNet compared to *MNIST*. However, *Contra-Info GAN* achieves superior results on the both datasets, since it does not require any supervised pretraining, and consequently is not affected by pretraining biases. Figure 18 shows the precision-recall curve for *LSH*, *ITQ*, *SphH*, *Speh*, *PCAH* and *Contra-Info GAN* with 16, 32 and 64 bits on *CIFAR-10* dataset. Overall, these experiments validate the effectiveness of our learning framework in dealing with different datasets and hash code sizes. Based on the *Contra-Info GAN* hash function, we also visualize the the top 10 retrieved images for some query data on CIFAR-10 dataset. Figure **??** illustrates these retrieved images using 32 bits hash codes, indicating that our model is able to extract semantic binary attributes. We also demonstrates samples assigned to four clusters on *MNIST* dataset using the *Contra-Info GAN* clustering model. Figure **??** shows these images with various assignment probabilities.

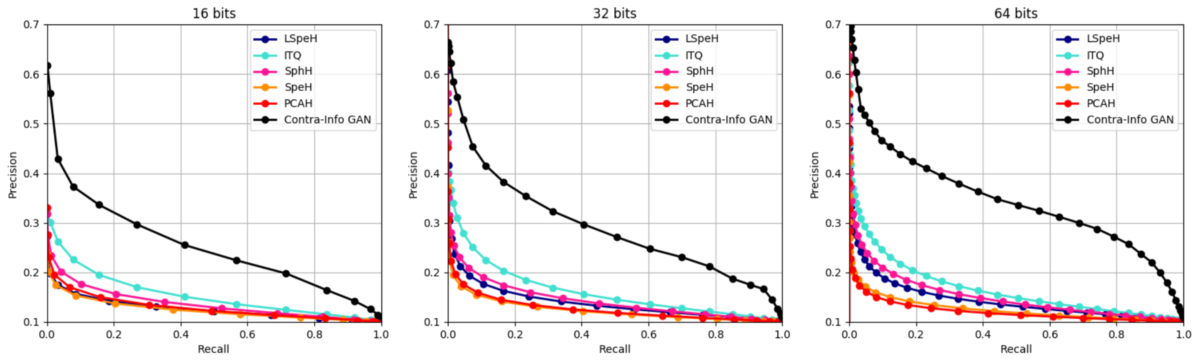
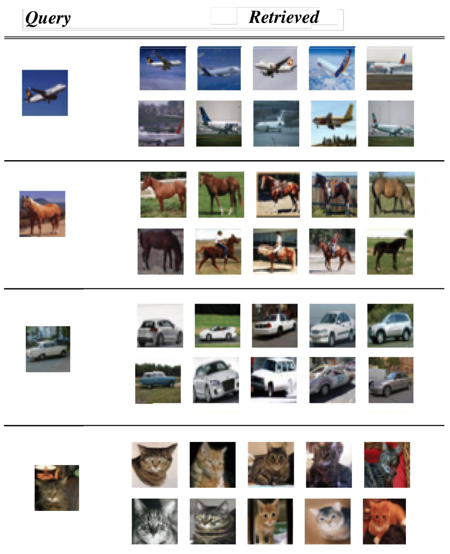
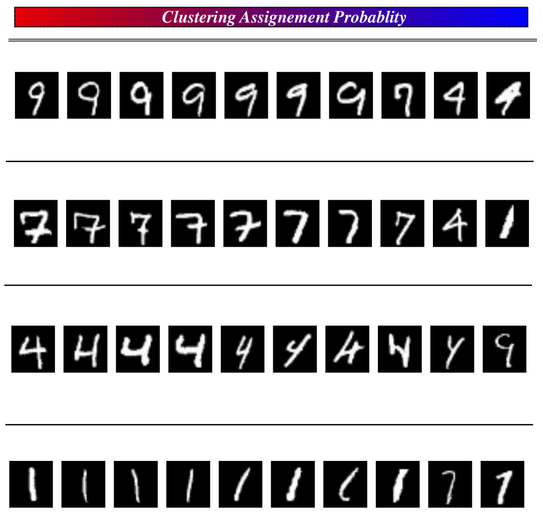


Figure 18: Precision-Recall curves on *CIFAR-10* dataset for *Contra-Info GAN* and five baselines with 16, 32, and 64 hash bits.





a)   **b)**

Figure 19: a) Top 10 retrieved images for query data by the *Contra-Info GAN* hash function on *CIFAR-10* dataset with 32 bits hash code. b) Image samples assigned to four clusters by the *Contra-Info GAN* clustering model on *MNIST* dataset.

Ablation Study: We perform an ablation study to examine the contribution of the adversarial loss (ADV), the contrastive loss (CONT) and the mutual information loss (MIM). Figure 17 illustrates the performance drop of different scenarios on *MNIST* and *CIFAR-10* datasets based on ACC in clustering and precision@1000 in image retrieval. The first observation is that all of the components in our loss function contribute in improving the results. While removing two of the losses degrades the results substantially, the adversarial loss (ADV) has the strongest effect on the results consistently on both tasks due to importance imposing prior and avoiding degenerate solutions. Moreover, the figure shows that the contrastive loss is more important than the mutual information loss in our learning framework. Furthermore, the contrastive loss has relatively more effect on *CIFAR-10* compared to *MNIST*, since *CIFAR-10* has more intra-class variations.

6.4 Related Work

Clustering Algorithms: Many clustering methods have been proposed in the literature, which can be generally divided into shallow and deep models. Among shallow clustering algorithms, there are distance-based clustering methods, such as *K-means* and Gaussian mixture model (*GMM*) [12], representing clusters using geometric properties of the data points, the kernel-based algorithms, like max-margin methods [193, 176], modeling the non-linearity of data using kernel functions, the connectivity-based algorithms, including spectral methods [105, 189], partitioning data points that are highly connected. However, these algorithms are not usually able to model the complex real-world data because of their shallow and linear models. Among deep clustering models, several studies employ autoencoder networks to build discriminative embedding space using the reconstruction loss [146, 174, 41]. There are also some methods using generative adversarial networks for clusterings [101, 40]. The latent representations in our model has some similarity to the ones in [101], however ours can be directly used for cluster assignments due to our effective contrastive loss. There are a few works on direct cluster assignments like [68, 169], where the mutual information between the latent representations of an image and its augmented variant is maximized in [68], and the correlations of image data are increased/decreased based on a pseudo-graph estimating the similarity of images in [169]. Unlike our learning framework, the former method requires large batch size to achieve good results (similar to many pair-wised consistency and contrastive losses [20, 183]), and the latter method suffers from imprecise pseudo-graph in its training procedure. The momentum contrastive learning benefits from similar approach to our model in containing the negative examples in a queue, but requires keeping a copy of a model that is not the case in our contrastive loss [54].

Hash Functions: The unsupervised hash function can be also grouped into shallow and deep models. As an instance of shallow models, locality sensitivity hashing (*LSH*) uses random linear projections as the mapping function [45]. Iterative quantization (*ITQ*) method uses an alternative approach for learning its projection and estimating the binary codes [47]. Spectral hashing (*SpeH*) obtains binary hash codes using spectral graph partitioning on the similarity information obtained from features [163]. However, these shallow models may not capture the non-linear nature of real-world data due to their shallow model, non-flexible mapping function, and hand-crafted features. Among unsupervised deep hashing models, semantic hashing adopts the restricted Boltzmann machine [60] model as a deep hash function [132]. Deep Hashing (*DH*) obtains quantized, balanced and independent hash bits using an unsupervised loss function [36]. *DeepBit* employs a quantization-based loss and a rotation-invariant consistency loss on deep models [93]. Unsupervised triplet hashing (*UTH*) uses an unsupervised triplet loss to decrease/increase the distance of positive/negative pairs [66]. Another study utilized a clustering algorithm to obtain pseudo labels for training a deep hash function based on a triplet loss on a CNN model [64]. Tying a discriminator and encoder parameters, HashGAN regularizes the encoder parameters using an adversarial loss of a discriminator and a collaborative loss obtained via a generator. Our proposed model differs with the previous works, as we employ effective contrastive, mutual information and adversarial losses to obtain disentangled representations.

6.5 Conclusion

In this project, we proposed a novel learning framework for unsupervised training of deep models for image retrieval and clustering tasks. Our learning framework benefits from three losses, the adversarial loss to exploit the generative knowledge and impose the desired prior to our deep model, the contrastive loss to achieve disentangled content and style representations, and mutual information loss to preserve the relevant information in the representations. Experimental results showed the superiority of our model compared to alternative clustering algorithms and hash functions.

Appendix A

A.1 Architecture of Convolutional Autoencoder Networks

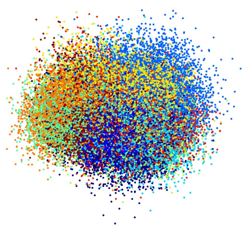
In this project, we have two convolutional layers plus one fully connected layer in both encoder and decoder pathways for all datasets. In order to have same size outputs for corresponding convolutional layers in the decoder and encoder, which is necessary for calculating the reconstruction loss functions, the kernel size, stride and padding (crop in decoder) are varied in different datasets. Moreover, the number of fully connected features (outputs) is chosen equal to the number of clusters for each dataset. Table 16 represents the detailed architecture of convolutional autoencoder networks for each dataset.

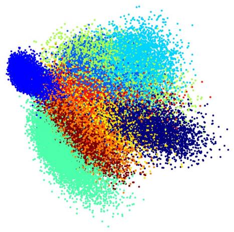
A.2 Visualization of learned embedding subspace

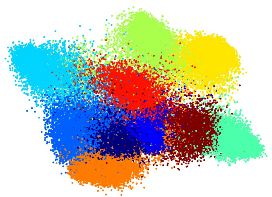
In this section, we visualize the learned embedding subspace (top encoder layer) in different stages using the first two principle components. The embedding representations are shown in three stages: 1) initial stage, where the parameters are randomly initialized with GlorotUniform; 2) intermediate stage before adding L_E, where the parameters are trained only using reconstruction loss functions; 3) final stage, where the parameters are fully trained using both clustering and reconstruction loss functions. Figure 20 illustrates the three stages of embedding features for *MNIST-full*, *MNIST-test*, and *USPS* datasets, and Figure 21 shows the three stages for *FRGC*, *YTF*, and *CMU-PIE* datasets.

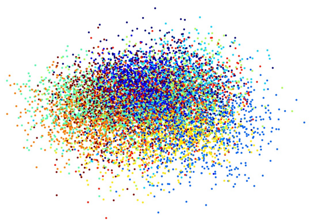
Table 16: Architecture of deep convolutional autoencoder networks. Conv1, Conv2 and Fully represent the specifications of the first and second convolutional layers in encoder and decoder pathways and the stacked fully connected layer.

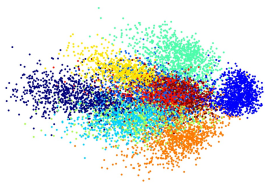
Dataset	Conv1				Conv2				Fully
	#feature	kernel	stride	padding	#feature	kernel	stride	padding	#features
MNIST-full	50	4×4	2	0	50	5×5	2	2	10
MNIST-test	50	4×4	2	0	50	5×5	2	2	10
USPS	50	4×4	2	0	50	5×5	2	2	10
FRGC	50	4×4	2	2	50	5×5	2	2	20
YTF	50	5×5	2	2	50	4×4	2	0	41
CMU-PIE	50	4×4	2	2	50	5×5	2	2	68

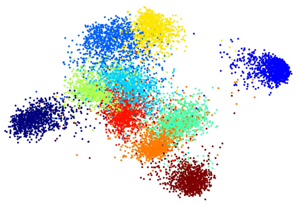

(a) Initial stage on *MNIST-full*

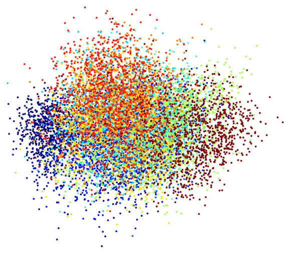

(b) Intermediate stage on *MNIST-full*

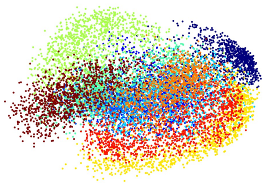

(c) Final stage on *MNIST-full*


(d) Initial stage on *MNIST-test*


(e) Intermediate stage on *MNIST-test*


(f) Final stage on *MNIST-test*


(g) Initial stage on *USPS*


(h) Intermediate stage on *USPS*

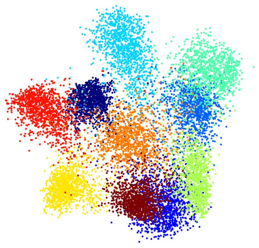
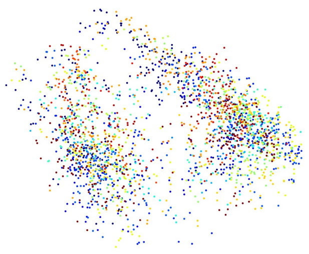
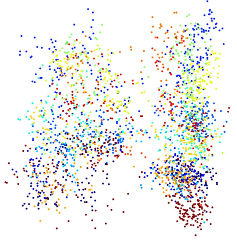
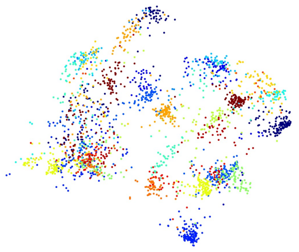
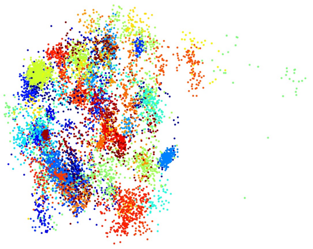

(i) Final stage on *USPS*

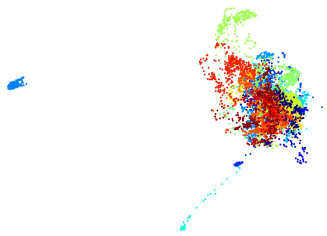
Figure 20: Embedding features in different learning stages on *MNIST-full*, *MNIST-test*, and *USPS* datasets. Three stages including Initial stage, Intermediate stage before adding clustering loss, and Final stage are shown for all datasets.

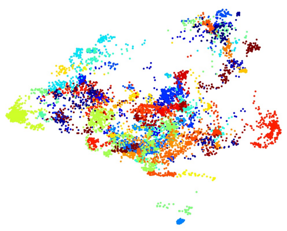

(a) Initial stage on *FRGC*

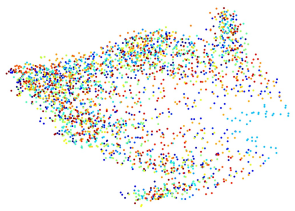

(b) Intermediate stage on *FRGC*

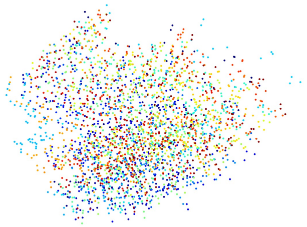

(c) Final stage on *FRGC*


(d) Initial stage on *YTF*


(e) Intermediate stage on *YTF*


(f) Final stage on *YTF*


(g) Initial stage on *CMU-PIE*


(h) Intermediate stage on *CMU-PIE*

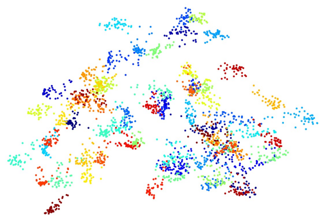

(i) Final stage on *CMU-PIE*

Figure 21: Embedding features in different learning stages on *FRGC*, *YTF* and *CMU-PIE* datasets. Three stages including Initial stage, Intermediate stage before adding clustering loss, and Final stage are shown for all datasets.

Appendix B

B.1 Proof for Proposition 1

Proposition 1: Iteratively improving the following auxiliary function $\mathbb{Q}$ is enough to maximize the log-likelihood function $\mathscr{L}(\boldsymbol{\psi}|\mathbf{X})$.

$$\mathbb{Q}(\boldsymbol{\psi}|\boldsymbol{\psi}^{(t)}) = \sum_{ijck} q_{ic}^{(t)} \log\Big(d_i e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}\Big) \tag{B.1}$$

$$where\ \ q_{ic}^{(t)} = \frac{\prod\limits_{jk} e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}}{\sum\limits_{c'}\prod\limits_{jk} e_{ic'}[p_{ijc'k}]^{\mathbf{1}_{ijk}}}$$

Proof: We can decompose the log-likelihood function into two auxiliary functions as follows.

$$\begin{aligned}\mathscr{L}(\boldsymbol{\psi}|\mathbf{X}) &= \log\Big(\prod_{i=1}^{N}\sum_{c=1}^{K}\prod_{j=1}^{M}\prod_{k=1}^{K} d_i e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}\Big) \\ &= \sum_{ijck} z_{ic}\log\Big(d_i e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}\Big) \\ &- \sum_{ic} z_{ic}\log\Big(\frac{d_i\prod\limits_{jk} e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}}{d_i\sum\limits_{c'}\prod\limits_{jk} e_{ic'}[p_{ijc'k}]^{\mathbf{1}_{ijk}}}\Big)\end{aligned} \tag{B.2}$$

where $z_{ic} \in \{0,1\}$ is the unknown true label. Since, we do not have access to the actual true labels (z_{ic}) in crowdsourcing task, they can be replaced by their expectation with respect to the current parameter.

$$q_{ic}^{(t)} = \mathbf{E}_{\psi^{(t)}}[z_{ic}=1|\mathbf{X_i}] = \frac{\prod\limits_{jk} e_{ic}^{(t)}[p_{ijck}^{(t)}]^{\mathbf{1}_{ijk}}}{\sum\limits_{c'}\prod\limits_{jk} e_{ic'}^{(t)}[p_{ijc'k}^{(t)}]^{\mathbf{1}_{ijk}}} \tag{B.3}$$

Thus, we approximate the log-likelihood function using,

$$\begin{aligned}\mathscr{L}(\boldsymbol{\psi}|\mathbf{X}) &= \sum_{ijck} q_{ic}^{(t)} \log\Big(d_i e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}\Big) \\ &- \sum_{ic} q_{ic}^{(t)} \log\Big(\frac{\prod_{jk} e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}}{\sum_{c'}\prod_{jk} e_{ic'}[p_{ijc'k}]^{\mathbf{1}_{ijk}}}\Big) \\ &= \mathbb{Q}(\boldsymbol{\psi}|\boldsymbol{\psi}^{(t)}) - \mathscr{H}(\boldsymbol{\psi}|\boldsymbol{\psi}^{(t)}) \qquad \text{(B.4)}\end{aligned}$$

Note that d_i is simply canceled out in the last line of Eq. (C.4).

In order to maximize the log-likelihood function, we can iteratively increase the right hand side of the above equation. The following equation shows the consecutive difference of the log-likelihood function.

$$\begin{aligned}&\mathscr{L}(\boldsymbol{\psi}^{(t+1)}|\mathbf{X}) - \mathscr{L}(\boldsymbol{\psi}^{(t)}|\mathbf{X}) \qquad \text{(B.5)} \\ &\quad = \mathbb{Q}(\boldsymbol{\psi}^{(t+1)}|\boldsymbol{\psi}^{(t)}) - \mathbb{Q}(\boldsymbol{\psi}^{(t)}|\boldsymbol{\psi}^{(t)}) \\ &\quad + \mathscr{H}(\boldsymbol{\psi}^{(t)}|\boldsymbol{\psi}^{(t)}) - \mathscr{H}(\boldsymbol{\psi}^{(t+1)}|\boldsymbol{\psi}^{(t)})\end{aligned}$$

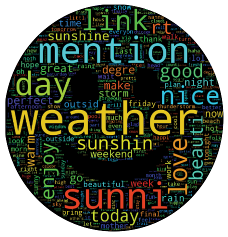
It can be shown that the consecutive difference of $\mathscr{H}$ is always non-negative based on Jansen inequality.

$$\mathscr{H}(\boldsymbol{\psi}^{(t)}|\boldsymbol{\psi}^{(t)}) - \mathscr{H}(\boldsymbol{\psi}^{(t+1)}|\boldsymbol{\psi}^{(t)}) = \sum_{ic} q_{ic}^{(t)} log\Big(\frac{q_{ic}^{(t)}}{q_{ic}^{(t+1)}}\Big) \geq 0 \qquad \text{(B.6)}$$

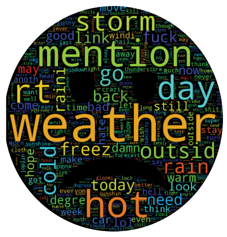
Therefore, iteratively improving $\mathbb{Q}$ function is sufficient to maximize the log-likelihood function.

$$\begin{aligned}\hat{\boldsymbol{\psi}} &= argmax\, \mathbb{Q}(\boldsymbol{\psi}|\boldsymbol{\psi}^{(t)}) \\ &= argmax \sum_{ijck} q_{ic}^{(t)} \log\Big(d_i e_{ic}[p_{ijck}]^{\mathbf{1}_{ijk}}\Big) \qquad \text{(B.7)}\end{aligned}$$

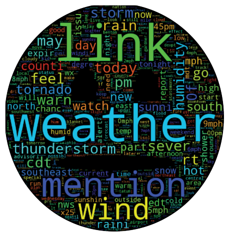
□



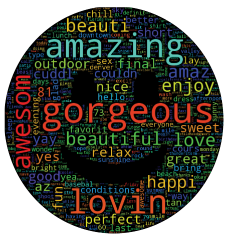
(a) Pos-docStatistic



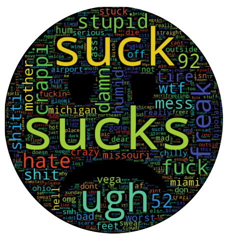
(b) Neg-docStatistic



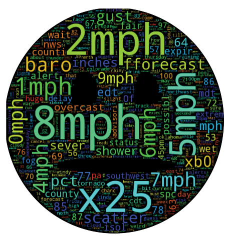
(c) Neut-docStatistic



(d) Pos-CrowdDeepAE



(e) Neg-CrowdDeepAE



(f) Neut-CrowdDeepAE

Figure 22: Word clouds of the positive (Pos), negative (Neg) and neutral (Neut) sentiments in SP dataset. The extracted word clouds using the statistics of documents (docStatistic) and our language model (CrowdDeepAE) are shown in the top and bottom rows respectively. The colors are only for legibility.

B.2 Solution to update gating parameters

Update $\boldsymbol{\alpha}$: The problem to update the gating parameters is:

$$\min_{\mathbf{1}^T\boldsymbol{\alpha}=M+1,\boldsymbol{\alpha}\geq 0} \lambda_\alpha \boldsymbol{\alpha}^T\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{\beta} \tag{B.8}$$

where $\beta_0 = \sum_{ic} q_{ic}^{(t)} \log\left(e_{ic}\right)$, $\beta_j = \sum_{ick} q_{ic}^{(t)} \mathbf{1}_{ijk} \log p_{ijck}$. We efficiently solve this problem as follows. The Lagrangian function for problem (B.8) is:

$$\ell(\boldsymbol{\alpha}, \eta, \boldsymbol{\mu}) = \lambda_\alpha \boldsymbol{\alpha}^T\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{\beta} - \eta(\mathbf{1}^T\boldsymbol{\alpha} - M - 1) - \boldsymbol{\mu}^T\boldsymbol{\alpha}\,,$$

where η and $\boldsymbol{\mu}$ are Lagrangian multipliers associated with the equality and inequality constraints respectively. Suppose $\boldsymbol{\alpha}^*$, η^* and $\boldsymbol{\mu}^*$ as the optimal variables, Karush-Kuhn-Tucker (KKT) conditions [13] yield the following equations,

$$\begin{cases} \lambda_\alpha \boldsymbol{\alpha}^* - \boldsymbol{\beta} - \eta^*\mathbf{1} - \boldsymbol{\mu}^* = 0 \\ \eta^* \geq 0 \\ \boldsymbol{\mu}^* \geq 0 \\ \boldsymbol{\mu}^{*T}\boldsymbol{\alpha}^* = 0. \end{cases}$$

The first equation in the above KKT conditions can be reformulated as $\alpha_j^* = (\beta_j + \eta^* + \mu_j^*)/\lambda_\alpha$ for all $j \in \{0, ..., M\}$. Using the inequality $\boldsymbol{\mu}^{*T}\boldsymbol{\alpha}^* = 0$, we further have:

$$\alpha_j^* = \frac{1}{\lambda_\alpha}(\beta_j + \eta^*)_+ \tag{B.9}$$

where $(.)_+ = \max(0, .)$. So the optimal variable α_j^* can be computed by knowing η^*. Using equality constraint $\mathbf{1}^T\boldsymbol{\alpha} = M + 1$, we define the following function,

$$f(\eta^*) = \sum_j \frac{1}{\lambda_\alpha}(\beta_j + \eta^*)_+ - M - 1\,. \tag{B.10}$$

Because $f(\eta^*) = 0$, we simply obtain η^* as the root of this function. Note that $f(\eta^*)$ is piece-wise linear and monotonically increasing, hence the root can be easily computed using the Newton method.

B.3 Word clouds for CrowdFlower dataset

In order to visualize the learned language model in CrowdDeepAE, we show the word clouds for both CF and SP datasets. In particular, the word cloud represents the importance (probability) of each word in a document with its font size. Using this visual representation, a viewer can quickly identify the dominant words in a document using the relative sizes. For each word in the datasets, we generate an auxiliary variable $\mathbf{X}_i^{Te}$ by setting the corresponding element equal to 1 and the other elements to zero, and then compute the probability of the word for every class. Figure 22 demonstrates the word cloud of CrowdDeepAE in CF dataset for the three sentiment options. We also show the word cloud of CF dataset using the probability (frequency) of each word in every sentiment class. The world clouds extracted from the documents statistic (docStatistic) mostly assign more importance to the highly repeated words in CF dataset like "weather", "mention" and "link", which do not represent the sentiment classes. But interestingly, the word clouds extracted from the language model of CrowdDeepAE have higher probabilities for the discriminative words, such as "gorgeous", "amazing" and "awesome" for the positive class; "sucks", "ugh" and "freak" for the negative option; and "mph", "scatter" and "forecast" for neutral class.

Appendix C

C.1 Proofs of Lemma 1

The objective function of the adversarial game for *ClusterGAN* is:

$$\min_{\mathscr{G},\mathscr{C}} \max_{\mathscr{D}} \; \mathbf{U}(\mathscr{D},\mathscr{G},\mathscr{C}) = \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\big[\log \mathscr{D}\big(\mathscr{C}(\mathbf{x}),\mathbf{x}\big)\big] + \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}\big[\log\big(1-\mathscr{D}\big(\mathbf{z},\mathscr{G}(\mathbf{z})\big)\big)\big]. \tag{C.1}$$

Lemma 1. *For any fixed* $\mathscr{G}$ *and* $\mathscr{C}$*, the optimal* $\mathscr{D}$ *defined by the utility function* $\mathbf{U}(\mathscr{D},\mathscr{G},\mathscr{C})$ *is:*

$$\mathscr{D}^*(\mathbf{z},\mathbf{x}) = \frac{P(\mathbf{x})P_{\mathscr{C}}(\mathbf{z}|\mathbf{x})}{P(\mathbf{x})P_{\mathscr{C}}(\mathbf{z}|\mathbf{x}) + P(\mathbf{z})P_{\mathscr{G}}(\mathbf{x}|\mathbf{z})}$$

$$= \frac{P_{\mathscr{C}}(\mathbf{z},\mathbf{x})}{P_{\mathscr{C}}(\mathbf{z},\mathbf{x}) + P_{\mathscr{G}}(\mathbf{z},\mathbf{x})}$$

Proof. Given the clusterer and generator, the utility function $\mathbf{U}(\mathscr{D},\mathscr{G},\mathscr{C})$ can be rewritten as

$$\begin{aligned}\mathbf{U}(\mathscr{D},\mathscr{G},\mathscr{C}) =& \iint P(\mathbf{x})P_{\mathscr{C}}(\mathbf{z}|\mathbf{x})\log(\mathscr{D}(\mathbf{z},\mathbf{x}))d\mathbf{x}d\mathbf{z} \\ &+ \iint P(\mathbf{z})P_{\mathscr{G}}(\mathbf{x}|\mathbf{z})\log(1-\mathscr{D}(\mathbf{z},\mathbf{x}))d\mathbf{x}d\mathbf{z} \\ =& \iint P_{\mathscr{C}}(\mathbf{z},\mathbf{x})\log(\mathscr{D}(\mathbf{z},\mathbf{x}))d\mathbf{x}d\mathbf{z} \\ &+ \iint P_{\mathscr{G}}(\mathbf{z},\mathbf{x})\log(1-\mathscr{D}(\mathbf{z},\mathbf{x}))d\mathbf{x}d\mathbf{z} \\ =& f(\mathscr{D}(\mathbf{z},\mathbf{x}))\end{aligned} \tag{C.2}$$

For any $\big(P_{\mathscr{C}}(\mathbf{z},\mathbf{x}), P_{\mathscr{G}}(\mathbf{z},\mathbf{x})\big) \in \mathbb{R}^2\ \{0,0\}$, the function $f(\mathscr{D}(\mathbf{z},\mathbf{x}))$ achieves its maximum at $\frac{P_{\mathscr{C}}(\mathbf{z},\mathbf{x})}{P_{\mathscr{C}}(\mathbf{z},\mathbf{x})+P_{\mathscr{G}}(\mathbf{z},\mathbf{x})}$. □

C.2 Proofs of Lemma 2

Given $\mathscr{D}^*(\mathbf{x},\mathbf{z})$, we can further replace $\mathscr{D}$ in the utility function $\mathbf{U}(\mathscr{D},\mathscr{G},\mathscr{C})$ and reformulate the objective as $\mathbf{V}(\mathscr{G},\mathscr{C}) = \max_{\mathscr{D}} \mathbf{U}(\mathscr{D},\mathscr{G},\mathscr{C})$.

Lemma 2. *The global optimum point of* $\mathbf{V}(\mathscr{G},\mathscr{C})$ *is achieved if and only if* $P(\mathbf{z},\hat{\mathbf{x}}) = P(\hat{\mathbf{z}},\mathbf{x})$.

Proof. Given $\mathscr{D}^*(\mathbf{x},\mathbf{z})$, the utility function $\mathbf{V}(\mathscr{G},\mathscr{C})$ can be reformulated as:

$$\begin{aligned}\mathbf{V}(\mathscr{G},\mathscr{C}) = &\iint P_{\mathscr{C}}(\mathbf{z},\mathbf{x}) \log\Big(\frac{P_{\mathscr{C}}(\mathbf{z},\mathbf{x})}{P_{\mathscr{C}}(\mathbf{z},\mathbf{x}) + P_{\mathscr{G}}(\mathbf{z},\mathbf{x})}\Big) d\mathbf{x}d\mathbf{z} \\ &+ \iint P_{\mathscr{G}}(\mathbf{z},\mathbf{x}) \log\Big(\frac{P_{\mathscr{G}}(\mathbf{z},\mathbf{x})}{P_{\mathscr{C}}(\mathbf{z},\mathbf{x}) + P_{\mathscr{G}}(\mathbf{z},\mathbf{x})}\Big) d\mathbf{x}d\mathbf{z}\end{aligned} \tag{C.3}$$

Sketching the proof in original *GAN* paper [48], $\mathbf{V}(\mathscr{G},\mathscr{C})$ cab be rewritten as:

$$\mathbf{V}(\mathscr{G},\mathscr{C}) = -\log 4 + 2JSD(P_{\mathscr{C}}(\mathbf{z},\mathbf{x}) \| P_{\mathscr{G}}(\mathbf{z},\mathbf{x}))\,, \tag{C.4}$$

where JSD represents the Jensen-Shannon divergence, which is always non-negative. Therefore, the unique optimum of $\mathbf{V}(\mathscr{G},\mathscr{C})$ is achieved if and only if $P_{\mathscr{C}}(\mathbf{z},\mathbf{x}) = P_{\mathscr{G}}(\mathbf{z},\mathbf{x})$, or in other words

$$P(\mathbf{z},\hat{\mathbf{x}}) = P(\hat{\mathbf{z}},\mathbf{x})$$

□

C.3 Proofs of Theorem 1

The optimization problem for estimating our balanced self-paced learning algorithm is:

$$\min_{\boldsymbol{\nu}} \quad \mathbf{L}(\boldsymbol{\nu}) = \sum_{i=1}^{n} \nu_i l_i - \lambda_\nu \|\boldsymbol{\nu}\|_1 + \gamma \|\boldsymbol{\nu}\|_e \quad s.t.\ \boldsymbol{\nu} \in [0,1]^n. \tag{C.5}$$

Theorem 1. *For any fixed $\mathscr{C}$, the optimal $\boldsymbol{\nu}$ defined by the objective function $\mathbf{L}(\boldsymbol{\nu})$ is:*

$$\begin{cases} \nu^*_{kq} = 1, & if \quad l_{kq} < \lambda_\nu - 2\gamma q \\ \nu^*_{kq} = \frac{\lambda_\nu - l_{kq}}{2\gamma} - q, & if \quad \lambda_\nu - 2\gamma q \leq l_{kq} < \lambda_\nu - 2\gamma(q-1) \\ \nu^*_{kq} = 0, & if \quad l_{kq} \geq \lambda_\nu - 2\gamma(q-1) \end{cases}$$

where $q \in \{1, ..., n_k\}$ is the sorted index of loss values $\{l_{k1}, ..., l_{kn_k}\}$ in the k-th group.

Proof.

$$\min_{\nu} \quad \mathbf{L}(\boldsymbol{\nu}) = \sum_{k=1}^{c} \mathbf{L}(\boldsymbol{\nu}_k) \tag{C.6}$$
$$= \sum_{k=1}^{c} \Big[\sum_{i=1}^{n_k} \nu_{ki}(l_{ki} - \lambda_\nu) + \gamma\Big(\sum_{i=1}^{n_k} |\nu_{ki}|\Big)^2 \Big], \quad s.t. \ \boldsymbol{\nu} \in [0,1]^n,$$

We can handle the c groups in Problem (C.6) separately. Given k, lets define:

$$\mathbf{b} = [\frac{(l_{k1} - \lambda_\nu)}{\gamma}, \frac{(l_{k2} - \lambda_\nu)}{\gamma}, \ldots, \frac{(l_{kn_k} - \lambda_\nu)}{\gamma}], \tag{C.7}$$

as the optimization problem *w.r.t.* the k-th group can be formulated as follows:

$$\min_{\mathbf{u}} \mathbf{b}^T\mathbf{u} + \mathbf{u}^T\mathbf{1}\mathbf{1}^T\mathbf{u}, \quad s.t. \ \ \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}, \tag{C.8}$$

where $\mathbf{u} = [v_{k1}, v_{k2}, \ldots v_{kn_k}]$. The Lagrangian function of Problem (C.8) is

$$\min_{\mathbf{u}} \mathbf{b}^T\mathbf{u} + \mathbf{u}^T\mathbf{1}\mathbf{1}^T\mathbf{u} - \eta^T\mathbf{u} - \lambda^T(\mathbf{1} - \mathbf{u}). \tag{C.9}$$

where $\eta \geq \mathbf{0}$ and $\lambda \geq \mathbf{0}$ are Lagrangian multipliers. Take derivate of Problem (C.9) *w.r.t.* $\mathbf{u}$ and set it to zero, we get

$$\eta + \lambda - \mathbf{b} = 2m\mathbf{1}. \tag{C.10}$$

where $m = \mathbf{1}^T\mathbf{u}$. From the KKT condition we can derive $\eta^T\mathbf{u} = 0$ and $\lambda^T(\mathbf{1} - \mathbf{u}) = 0$. Consequently, we can derive

$$\begin{cases} u_q = 0 & \Longrightarrow \eta_q > 0, \lambda_q = 0 \ \Longrightarrow \frac{b_q}{2} + m > 0, \\ 0 < u_q < 1 & \Longrightarrow \eta_q = 0, \lambda_q = 0 \ \Longrightarrow \frac{b_q}{2} + m = 0, \\ u_q = 1 & \Longrightarrow \eta_q = 0, \lambda_q > 0 \ \Longrightarrow \frac{b_q}{2} + m < 0, \end{cases} \tag{C.11}$$

where $q \in \{1, ..., n_k\}$. Without loss of generality, suppose $\mathbf{b}$ is a sorted vector such that $b_1 < b_2 < \cdots < b_{n_k}$, then according to Eq. (C.11) we have $1 \geq u_1 \geq u_2 \geq \cdots \geq u_{n_k} \geq 0$, from which we can derive

$$\begin{cases} u_q = 0 & \Longrightarrow u_r = 0, \forall r \geq q & \Longrightarrow m \leq q-1\,, \\ 0 < u_q < 1 & \Longrightarrow u_r = 0, \forall r > q, \text{and } u_r = 1, \forall r < q & \Longrightarrow q-1 < m < q\,, \\ u_q = 1 & \Longrightarrow u_r = 1, \forall r \leq q & \Longrightarrow m \geq q\,. \end{cases} \tag{C.12}$$

Combining Eq. (C.11) and Eq. (C.12) we can derive the solution to Problem (C.8) as follows:

$$\begin{cases} -\frac{b_q}{2} \leq q-1 & \Longrightarrow u_q = 0\,, \\ q-1 < -\frac{b_q}{2} < q & \Longrightarrow u_q = -\frac{b_q}{2} - q + 1\,, \\ -\frac{b_q}{2} \geq q & \Longrightarrow u_q = 1\,, \end{cases}$$

which can be rewritten based on $\boldsymbol{\nu}$ as:

$$\begin{cases} \nu_{kq}^* = 1, & if \quad l_{kq} < \lambda_\nu - 2\gamma q \\ \nu_{kq}^* = \frac{\lambda_\nu - l_{kq}}{2\gamma} - q, & if \quad \lambda_\nu - 2\gamma q \leq l_{kq} < \lambda_\nu - 2\gamma(q-1) \\ \nu_{kq}^* = 0, & if \quad l_{kq} \geq \lambda_\nu - 2\gamma(q-1) \end{cases}$$

□

Bibliography

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[2] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint*, 2016.

[3] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *IEEE conference on Computer vision and pattern recognition (CVPR)*, pages 510–517. Ieee, 2012.

[4] Yoram Bachrach, Thore Graepel, Tom Minka, and John Guiver. How to grade a test without knowing the answers, a bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *arXiv preprint arXiv:1206.6386*, 2012.

[5] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[6] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

[7] David Barber and Felix V Agakov. Kernelized infomax clustering. In *Advances in neural information processing systems (NIPS)*, pages 17–24, 2005.

[8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[9] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 899–907, 2013.

[10] Mostapha Benhenda. Chemgan challenge for drug discovery: can ai reproduce natural chemical diversity? *arXiv preprint arXiv:1708.08227*, 2017.

[11] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[12] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on pattern analysis and machine intelligence*, 22(7):719–725, 2000.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] Anthony Brew, Derek Greene, and Pádraig Cunningham. Using crowdsourcing and active learning to track sentiment in online media. In *ECAI*, pages 145–150, 2010.

[15] John S Bridle, Anthony JR Heading, and David JC MacKay. Unsupervised classifiers, mutual information and'phantom targets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1096–1101, 1992.

[16] Deng Cai, Xiaofei He, Xuanhui Wang, Hujun Bao, and Jiawei Han. Locality preserving nonnegative matrix factorization. In *Twenty-first international joint conference on artificial intelligence*, 2009.

[17] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. *European conference on Computer Vision (ECCV)*, pages 778–792, 2010.

[18] Miguel A Carreira-Perpinán and Ramin Raziperchikolaei. Hashing with binary autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 557–566, 2015.

[19] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 5879–5887, 2017.

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

[21] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NISP)*, pages 2172–2180, 2016.

[22] Xi Chen, Qihang Lin, and Dengyong Zhou. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 64–72, 2013.

[23] Xinlei Chen and Deng Cai. Large scale spectral clustering with landmark-based representation. In *AAAI*, 2011.

[24] LI Chongxuan, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 4091–4101, 2017.

[25] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[26] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning (ICML)*, pages 160–167. ACM, 2008.

[27] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[29] Zhijie Deng, Hao Zhang, Xiaodan Liang, Luona Yang, Shizhen Xu, Jun Zhu, and Eric P Xing. Structured generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3902–3912, 2017.

[30] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems (NIPS)*, pages 1486–1494, 2015.

[31] Pietro Di Lena, Ken Nagata, and Pierre Baldi. Deep architectures for protein contact map prediction. *Bioinformatics*, 28(19):2449–2457, 2012.

[32] Thanh-Toan Do, Anh-Dzung Doan, and Ngai-Man Cheung. Learning to hash with binary deep neural network. In *European Conference on Computer Vision (ECCV)*, pages 219–234. Springer, 2016.

[33] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[34] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[35] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[36] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2475–2483, 2015.

[37] Hongchang Gao, Feiping Nie, Xuelong Li, and Heng Huang. Multi-view subspace clustering. *International Conference on Computer Vision (ICCV 2015)*, pages 4238–4246, 2015.

[38] Hongchang Gao, Xiaoqian Wang, and Heng Huang. New robust clustering model for identifying cancer genome landscapes. *IEEE International Conference on Data Mining (ICDM 2016)*, pages 151–160, 2016.

[39] Tiezheng Ge, Kaiming He, and Jian Sun. Graph cuts for supervised binary coding. In *European Conference on Computer Vision*, pages 250–264. Springer, 2014.

[40] Kamran Ghasedi, Xiaoqian Wang, Cheng Deng, and Heng Huang. Balanced self-paced learning for generative adversarial clustering network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4391–4400, 2019.

[41] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (ICCV)*, pages 5736–5745, 2017.

[42] Kamran Ghasedi Dizaji, Xiaoqian Wang, and Heng Huang. Semi-supervised generative adversarial network for gene expression inference. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1435–1444. ACM, 2018.

[43] Kamran Ghasedi Dizaji, Yanhua Yang, and Heng Huang. Joint generative-discriminative aggregation model for multi-option crowd labels. In *WSDM*, 2017.

[44] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi, Yanhua Yang, Cheng Deng, and Heng Huang. Unsupervised deep generative adversarial hashing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3664–3673, 2018.

[45] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[46] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[47] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *In Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[48] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NIPS)*, pages 2672–2680, 2014.

[49] K Chidananda Gowda and G Krishna. Agglomerative clustering using the concept of mutual nearest neighbourhood. *Pattern recognition*, 10(2):105–112, 1978.

[50] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[51] Jinma Guo, Shifeng Zhang, and Jianmin Li. Hash learning with convolutional neural networks for semantic based image retrieval. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 227–238. Springer, 2016.

[52] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE computer society conference on Computer vision and pattern recognition*, volume 2, pages 1735–1742. IEEE, 2006.

[53] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

[54] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

[55] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2938–2945, 2013.

[56] 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 (CVPR)*, pages 770–778, 2016.

[57] Katherine A Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning (ICML)*. ACM, 2005.

[58] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2957–2964. IEEE, 2012.

[59] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[60] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[61] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[62] 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.

[63] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *International Conference on Machine Learning (ICML)*, pages 1558–1567, 2017.

[64] Chen Huang, Chen Change Loy, and Xiaoou Tang. Unsupervised learning of discriminative attributes and visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5175–5184, 2016.

[65] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

[66] Shanshan Huang, Yichao Xiong, Ya Zhang, and Jia Wang. Unsupervised triplet hashing for fast image retrieval. *arXiv preprint arXiv:1702.08798*, 2017.

[67] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[68] 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.

[69] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2078–2086, 2014.

[70] Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 246–256. SIAM, 2004.

[71] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[72] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[73] D Kinga and J Ba Adam. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

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

[75] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3581–3589, 2014.

[76] Andreas Krause, Pietro Perona, and Ryan G Gomes. Discriminative clustering by regularized information maximization. In *Advances in neural information processing systems (NIPS)*, pages 775–783, 2010.

[77] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 1(4):7, 2009.

[78] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

[79] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[80] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1092–1104, 2012.

[81] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1189–1197, 2010.

[82] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3278, 2015.

[83] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[84] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013.

[85] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555. IEEE, 2011.

[86] Haifeng Li, Keshu Zhang, and Tao Jiang. Minimum entropy clustering and applications to gene expression analysis. In *Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE*, pages 142–151. IEEE, 2004.

[87] Hao Li, Maoguo Gong, Deyu Meng, and Qiguang Miao. Multi-objective self-paced learning. In *AAAI*, pages 1802–1808, 2016.

[88] Hongwei Li and Bin Yu. Error rate bounds and iterative weighted majority voting for crowdsourcing. *arXiv preprint arXiv:1411.4086*, 2014.

[89] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1711–1717. AAAI Press, 2016.

[90] Yeqing Li, Feiping Nie, Heng Huang, and Junzhou Huang. Large-scale multi-view spectral clustering via bipartite graph. *Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 2015.

[91] Jian Liang, Zhihang Li, Dong Cao, Ran He, and Jingdong Wang. Self-paced cross-modal subspace matching. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 569–578. ACM, 2016.

[92] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton Van den Hengel, and David Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1963–1970, 2014.

[93] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1183–1192, 2016.

[94] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2074–2081. IEEE, 2012.

[95] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[96] Dengsheng Lu and Qihao Weng. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823–870, 2007.

[97] Dijun Luo, Chris Ding, Heng Huang, and Feiping Nie. Consensus spectral clustering in near-linear time. In *2011 IEEE 27th International Conference on Data Engineering*, pages 1079–1090. IEEE, 2011.

[98] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[99] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[100] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

[101] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan: Latent space clustering in generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4610–4617, 2019.

[102] Claudiu-Cristian Musat Thisone, Alireza Ghasemi, and Boi Faltings. Sentiment analysis using a novel human computation game. In *Proceedings of the 3rd Workshop on the People's Web Meets NLP: Collaboratively Constructed Semantic Resources and their Applications to NLP*, pages 1–9. Association for Computational Linguistics, 2012.

[103] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[104] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[105] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems (NIPS)*, 2:849–856, 2002.

[106] Feiping Nie, Cheng Deng, Hua Wang, Xinbo Gao, and Heng Huang. New l1-norm relaxations and optimizations for graph clustering. *Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, 2016.

[107] Feiping Nie and Heng Huang. Subspace clustering via new discrete group structure constrained low-rank model. *25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1874–1880, 2016.

[108] Feiping Nie, Xiaoqian Wang, and Heng Huang. Clustering and projected clustering via adaptive neighbor assignment. *The 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2014)*, pages 977–986, 2014.

[109] Feiping Nie, Xiaoqian Wang, Michael I Jordan, and Heng Huang. The constrained laplacian rank algorithm for graph-based clustering. In *AAAI*, pages 1969–1976. Citeseer, 2016.

[110] Feiping Nie, Zinan Zeng, Ivor W Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11):1796–1808, 2011.

[111] Oliver Nina, Jamison Moody, and Clarissa Milligan. A decoder-free approach for unsupervised clustering and manifold learning with random triplet mining. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[112] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

[113] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

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

[115] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[116] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.

[117] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[118] Nikolaos Pitelis, Chris Russell, and Lourdes Agapito. Semi-supervised learning using an unsupervised atlas. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 565–580. Springer, 2014.

[119] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

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

[121] Rajat Raina, Yirong Shen, Andrew Y Ng, and Andrew McCallum. Classification with hybrid generative/discriminative models. In *In Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2003.

[122] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3546–3554, 2015.

[123] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *The Journal of Machine Learning Research*, 11:1297–1322, 2010.

[124] Salah Rifai, Yann N Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2294–2302, 2011.

[125] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.

[126] Volker Roth and Tilman Lange. Feature selection in clustering problems. In *In Advances in Neural Information Processing Systems (NIPS)*, pages 473–480, 2003.

[127] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *IEEE international conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011.

[128] Najmeh Sadoughi and Carlos Busso. Retrieving target gestures toward speech driven animation with meaningful behaviors. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 115–122. ACM, 2015.

[129] Najmeh Sadoughi and Carlos Busso. Joint learning of speech-driven facial motion with bidirectional long-short term memory. In *International Conference on Intelligent Virtual Agents*, pages 389–402. Springer, 2017.

[130] Najmeh Sadoughi, Yang Liu, and Carlos Busso. Speech-driven animation constrained by appropriate discourse functions. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 148–155. ACM, 2014.

[131] Najmeh Sadoughi, Yang Liu, and Carlos Busso. Msp-avatar corpus: Motion capture recordings to study the role of discourse functions in the design of intelligent virtual agents. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, 2015.

[132] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[133] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2234–2242, 2016.

[134] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 901–909, 2016.

[135] Mehmet E Sargin, Yucel Yemez, Engin Erzin, and Ahmet M Tekalp. Analysis of head gesture and prosody patterns for prosody-driven head-gesture animation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1330–1345, 2008.

[136] Aashish Sheshadri and Matthew Lease. Square: A benchmark for research on computing crowd consensus. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[137] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[138] Hiroyuki Shinnou and Minoru Sasaki. Spectral clustering for a large data set by reducing the similarity matrix size. In *LREC*, 2008.

[139] Terence Sim, Simon Baker, and Maan Bsat. The cmu pose, illumination, and expression (pie) database. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 46–51. IEEE, 2002.

[140] Edwin Simpson, Stephen J Roberts, Ioannis Psorakis, and Arfon Smith. Dynamic bayesian combination of multiple imperfect classifiers. *Decision making and imperfection*, 474:1–35, 2013.

[141] Edwin D Simpson, Matteo Venanzi, Steven Reece, Pushmeet Kohli, John Guiver, Stephen J Roberts, and Nicholas R Jennings. Language understanding in the wild: Combining crowdsourcing and machine learning. In *Proceedings of the 24th international conference on world wide web (WWW)*, pages 992–1002. International World Wide Web Conferences Steering Committee, 2015.

[142] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[143] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[144] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–9, 2015.

[145] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[146] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, pages 1293–1299, 2014.

[147] Tian Tian and Jun Zhu. Max-margin majority voting for learning from crowds. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1612–1620, 2015.

[148] George Trigeorgis, Konstantinos Bousmalis, Stefanos Zafeiriou, and Bjoern Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, pages 1692–1700, 2014.

[149] Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[150] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *Proceedings of the 23rd international conference on World wide web (WWW)*, pages 155–164, 2014.

[151] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning (ICML)*, pages 1096–1103. ACM, 2008.

[152] 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(Dec):3371–3408, 2010.

[153] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. ACM, 2016.

[154] De Wang, Feiping Nie, and Heng Huang. Unsupervised feature selection via unified trace ratio formulation and k-means clustering (track). *European Conference on*

Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2014), pages 306–321, 2014.

[155] Hua Wang, Feiping Nie, and Heng Huang. Multi-view clustering and feature learning via structured sparsity. *The 30th International Conference on Machine Learning (ICML 2013)*, pages 352–360, 2013.

[156] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[157] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3424–3431. IEEE, 2010.

[158] Xiaoqian Wang, Kamran Ghasedi Dizaji, and Heng Huang. Conditional generative adversarial network for gene expression inference. *Bioinformatics*, 34(17):i603–i611, 2018.

[159] Xiaoqian Wang, Feiping Nie, and Heng Huang. Structured doubly stochastic matrix for graph based clustering: Structured doubly stochastic matrix. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1245–1254, 2016.

[160] Zhangyang Wang, Shiyu Chang, Jiayu Zhou, Meng Wang, and Thomas S Huang. Learning a task-specific deep architecture for clustering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 369–377. SIAM, 2016.

[161] Zhangyang Wang, Yingzhen Yang, Shiyu Chang, Jinyan Li, Simon Fong, and Thomas S Huang. A joint optimization framework of sparse coding and discriminative clustering. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, page 4, 2015.

[162] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[163] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems (NIPS)*, pages 1753–1760, 2009.

[164] Peter Welinder, Steve Branson, Serge J Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems (NIPS)*, volume 23, pages 2424–2432, 2010.

[165] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems (NIPS)*, pages 2035–2043, 2009.

[166] Kyle W Willett, Chris J Lintott, Steven P Bamford, Karen L Masters, Brooke D Simmons, Kevin RV Casteels, Edward M Edmondson, Lucy F Fortson, Sugata Kaviraj, William C Keel, et al. Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, 2013.

[167] Christopher KI Williams. A mcmc approach to hierarchical mixture modelling. In *Advances in neural information processing systems (NIPS)*, pages 680–686, 1999.

[168] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE, 2011.

[169] 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*, pages 8150–8159, 2019.

[170] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.

[171] Hao Xia, Pengcheng Wu, Steven CH Hoi, and Rong Jin. Boosting multi-kernel locality-sensitive hashing for scalable image retrieval. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 55–64. ACM, 2012.

[172] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, volume 1, pages 2156–2162, 2014.

[173] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[174] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning (ICML)*, 2016.

[175] Pengtao Xie and Eric P Xing. Integrating image clustering and codebook learning. In *AAAI*, pages 1903–1909, 2015.

[176] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Advances in neural information processing systems (NIPS)*, pages 1537–1544, 2004.

[177] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, pages 645–678, 2005.

[178] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003.

[179] Huei-Fang Yang, Kevin Lin, and Chu-Song Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

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

[181] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10):2761–2773, 2010.

[182] Jieping Ye, Zheng Zhao, and Mingrui Wu. Discriminative k-means for clustering. In *Advances in neural information processing systems (NIPS)*, pages 1649–1656, 2008.

[183] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6210–6219, 2019.

[184] Raymond Yeh, Chen Chen, Teck Yian Lim, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016.

[185] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5485–5493, 2017.

[186] Yang Yu and Wen-Ji Zhou. Mixture of gans for clustering. In *IJCAI*, 2018.

[187] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[188] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *In Advances in Neural Information Processing Systems (NIPS)*, volume 17, page 16, 2004.

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

[190] Dingwen Zhang, Deyu Meng, Chao Li, Lu Jiang, Qian Zhao, and Junwei Han. A self-paced multiple-instance learning framework for co-saliency detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 594–602, 2015.

[191] Wei Zhang, Xiaogang Wang, Deli Zhao, and Xiaoou Tang. Graph degree linkage: Agglomerative clustering on a directed graph. In *European Conference on Computer Vision*, pages 428–441. Springer, 2012.

[192] Wei Zhang, Deli Zhao, and Xiaogang Wang. Agglomerative clustering via maximum incremental path integral. *Pattern Recognition*, 46(11):3056–3065, 2013.

[193] Bin Zhao, Fei Wang, and Changshui Zhang. Efficient multiclass maximum margin clustering. In *ICML*, pages 1248–1255, 2008.

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

[195] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.

[196] Dengyong Zhou, Qiang Liu, John Platt, and Christopher Meek. Aggregating ordinal labels from crowds by minimax conditional entropy. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 262–270, 2014.

[197] Dengyong Zhou, Qiang Liu, John C Platt, Christopher Meek, and Nihar B Shah. Regularized minimax conditional entropy for crowdsourcing. *arXiv preprint arXiv:1503.07240*, 2015.

[198] Feng Zhou, Fernando De la Torre, and Jessica K Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):582–596, 2013.

[199] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, pages 2415–2421, 2016.

[200] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.