This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

# Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization

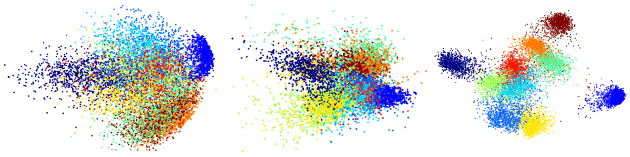
Kamran Ghasedi Dizaji[†], Amirhossein Herandi[‡], Cheng Deng[♯], Weidong Cai[♮], Heng Huang[†*]
[†]Electrical and Computer Engineering, University of Pittsburgh, USA
[‡]Computer Science and Engineering, University of Texas at Arlington, USA
[♯]School of Electronic Engineering, Xidian University, China
[♮]School of Information Technologies, University of Sydney, Australia
kamran.ghasedi@gmail.com, amirhossein.herandi@uta.edu, chdeng@mail.xidian.edu.cn
tom.cai@sydney.edu.au, heng.huang@pitt.edu

## Abstract

*In this paper, we propose a new clustering model, called **DEeP** Embedded Regular**I**zed **C**lus**T**ering (DEPICT), which efficiently maps data into a discriminative embedding subspace and precisely predicts cluster assignments. DEPICT generally consists of a multinomial logistic regression function stacked on top of a multi-layer convolutional autoencoder. We define a clustering objective function using relative entropy (KL divergence) minimization, regularized by a prior for the frequency of cluster assignments. An alternating strategy is then derived to optimize the objective by updating parameters and estimating cluster assignments. Furthermore, we employ the reconstruction loss functions in our autoencoder, as a data-dependent regularization term, to prevent the deep embedding function from overfitting. In order to benefit from end-to-end optimization and eliminate the necessity for layer-wise pre-training, we introduce a joint learning framework to minimize the unified clustering and reconstruction loss functions together and train all network layers simultaneously. Experimental results indicate the superiority and faster running time of DEPICT in real-world clustering tasks, where no labeled data is available for hyper-parameter tuning.*

## 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 [36, 42, 49]. Handling large-scale datasets, there are also several studies which select only a subset of data points to accelerate the clustering process [39, 21, 19].



(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).

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

*Corresponding Author. This work was partially supported by U.S. NIH R01 AG049371, NSF IIS 1302675, IIS 1344152, DBI 1356628, IIS 1619308, IIS 1633753.

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.

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 [45]. 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 [65]. 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 [16] 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. Related Works

There is a large number of clustering algorithms in literature, which can be grouped into different perspectives, such as hierarchical [9, 51], centroid-based [20, 3, 27], graph-based [38, 28, 48, 25], sequential (temporal) [11, 37, 66], regression model based [7, 47], and subspace clustering models [1, 10, 6, 26]. 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 [4] 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 [18, 2, 14], max-margin [64, 55], and spectral graph [24] 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 [14, 24, 32]. 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 [59]. [46] 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.

A joint learning framework for updating code books and estimating image clusters was proposed in [54] while *SIFT* features are used as input data. A deep structure, named *TAGnet* was introduced in [49], where two layers of sparse coding followed by a clustering algorithm are trained with an alternating learning approach. Similar work is presented in [50] 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 [50] to reduce the dimension of input data with *PCA* and the model in [49] 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* [42]. 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 [43]. 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 [53]. 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 [57]. 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.

# 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.

## 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, $\varphi_W : X \to Z$, 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 $f_\theta : Z \to Y$ to predict the probabilistic cluster assignments as follows.

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

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 $\mathbf{Q}$.

$$\mathscr{L} = KL(\mathbf{Q}\|\mathbf{P}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik} \log \frac{q_{ik}}{p_{ik}}\,, \tag{2}$$

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:

$$f_k = P(\mathbf{y} = k) = \frac{1}{N}\sum_i q_{ik}\,, \tag{3}$$

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.

$$\begin{aligned}\mathscr{L} &= KL(\mathbf{Q}\|\mathbf{P}) + KL(\mathbf{f}\|\mathbf{u}) \qquad (4)\\ &= \Big[\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log\frac{q_{ik}}{p_{ik}}\Big] + \Big[\frac{1}{N}\sum_{k=1}^{K} f_k \log\frac{f_k}{u_k}\Big]\\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log\frac{q_{ik}}{p_{ik}} + q_{ik}\log\frac{f_k}{u_k}\,,\end{aligned}$$

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:

$$\min_{\mathbf{Q}} \ \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log\frac{q_{ik}}{p_{ik}} + q_{ik}\log\frac{f_k}{u_k}\,, \tag{5}$$

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 [23], 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.

$$\frac{\partial \mathscr{L}}{\partial q_{ik}} \propto \log\Big(\frac{q_{ik} f_k}{p_{ik}}\Big) + \frac{q_{ik}}{\sum\limits_{i'=1}^{N} q_{i'k}} + 1\,, \tag{6}$$

Investigating this problem more carefully, we approximate the gradient in Eq.(6) 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.

$$q_{ik} = \frac{p_{ik}/(\sum_{i'} p_{i'k})^{\frac{1}{2}}}{\sum\limits_{k'} p_{ik'}/(\sum_{i'} p_{i'k'})^{\frac{1}{2}}}\,, \tag{7}$$

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

$$\min_{\boldsymbol{\psi}} \ -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log p_{ik}\,, \tag{8}$$

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

### 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 [31, 60], 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.

$$\tilde{\mathbf{z}}^l = Dropout\big[g(\mathbf{W}_e^l \tilde{\mathbf{z}}^{l-1})\big]\,, \tag{9}$$

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 [41], $g$ 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,

$$\hat{\mathbf{z}}^{l-1} = g(\mathbf{W}_d^l \hat{\mathbf{z}}^l)\,, \tag{10}$$

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.

$$\mathbf{z}^l = g(\mathbf{W}_e^l \mathbf{z}^{l-1})\,, \tag{11}$$

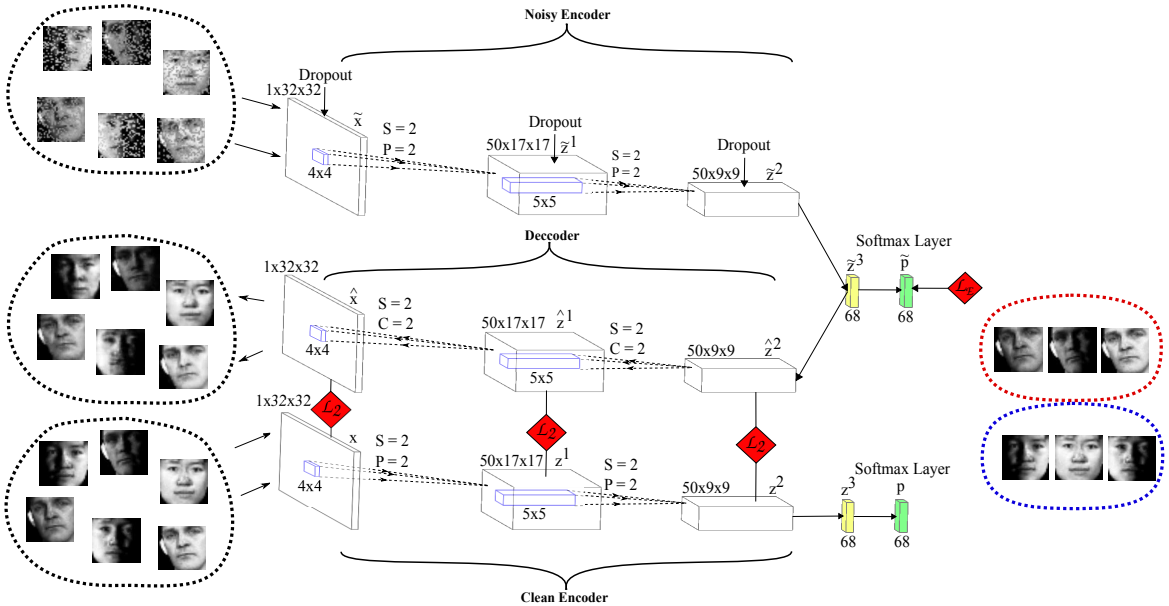



Figure 2: Architecture of *DEPICT* for *CMU-PIE* dataset. *DEPICT* consists of a soft-max layer stacked on top of a multi-layer convolutional autoencoder. In order to illustrate the joint learning framework, we consider the following four pathways for *DEPICT*: Noisy (corrupted) encoder, Decoder, Clean 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.

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.(1).

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.

$$\min_{\boldsymbol{\psi}} \; -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik}\log\tilde{p}_{ik} + \frac{1}{N}\sum_{i=1}^{N}\sum_{l=0}^{L-1}\frac{1}{|\mathbf{z}_i^l|}\|\mathbf{z}_i^l-\hat{\mathbf{z}}_i^l\|_2^2\,, \tag{12}$$

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 [33, 65]. 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) [65], and lateral (skip) connections between encoder and decoder layers used in ladder network [33]. Algorithm 1 shows a brief description of *DEPICT* algorithm.

**Algorithm 1:** *DEPICT* Algorithm

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

## 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.

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

**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 [16]. *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 [40]. *Youtube-Face (YTF)*: Following [57], we choose the first 41 subjects of YTF dataset. Faces inside images are first cropped and then resized to 55 by 55 sizes [52]. *FRGC*: Using the 20 random selected subjects in [57] 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 1 provides a brief description of each dataset.

| 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 |

Table 1: Dataset Descriptions

**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 [15] to measure accuracy. NMI calculates the normalized measure of similarity between two labels of the same data [56]. 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.

### 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*) [38], self-tuning spectral clustering (*SC-ST*) [61], large-scale spectral clustering (*SC-LS*) [5], graph degree linkage-based agglomerative clustering (*AC-GDL*) [62], agglomerative clustering via path integral (*AC-PIC*) [63], spectral embedded clustering (*SEC*) [29], local discriminant models and global integration (*LDMGI*) [58], *NMF* with deep model (*NMF-D*) [43], task-specific clustering with deep model (*TSC-D*) [49], deep embedded clustering (*DEC*) [53], and joint unsupervised learning (*JULE*) [57].

**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 $5 \times 5$, 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 $10 \times 10$ in the second convolutional layer. Inspired by [31], we consider leaky rectified (leaky RELU) non-linearity [22] 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 $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-08$ [12]. The weights of convolutional and fully connected layers are all initialized by Xavier approach [8]. 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*.

**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 [57]. 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,

| 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 |

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 papers, 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.

*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.

### 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 [57]. We run *DEPICT* and the released code for *JULE* algorithms[2] 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] https://github.com/jwyang/JULE-Torch

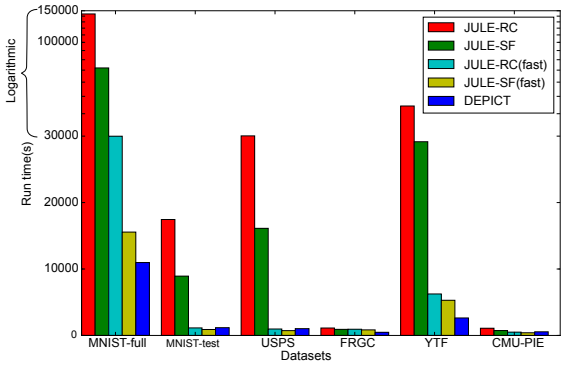



Figure 3: Running time comparison of *DEPICT* and *JULE* clustering algorithms on image datasets.

### 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 encoder pathway is then fine-tuned using the clustering objective function [45]. 2) Another approach (RdA) is suggested in [53] 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

| Dataset | | *MNIST-full* | | *MNIST-test* | | *USPS* | | *FRGC* | | *YTF* | | *CMU-PIE* | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NMI | ACC | NMI | ACC | NMI | ACC | NMI | ACC | NMI | ACC | NMI | ACC |
| *Deep-ConvAE + AC-PIC* | 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 |
| | 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 |
| *DEPICT* | 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 |
| | 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** |

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

(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 3 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.

### 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 [33, 65], 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* [65], *M1+M2* [13], and *AtlasRBF* [30], and has comparable results with the complicated *Ladder* network [33]. These results further confirm the discriminative quality of the embedding features of *DEPICT*.

| Model | 100 | 1000 | 3000 |
|---|---|---|---|
| *T-SVM* [44] | 16.81 | 5.38 | 3.45 |
| *CAE* [35] | 13.47 | 4.77 | 3.22 |
| *MTC* [34] | 12.03 | 3.64 | 2.57 |
| *PL-DAE* [17] | 10.49 | 3.46 | 2.69 |
| *AtlasRBF* [30] | 8.10 | 3.68 | - |
| *M1+M2* [13] | 3.33±0.14 | 2.40±0.05 | 2.18±0.04 |
| *SWWAE* [65] | 8.71±0.34 | 2.83±0.10 | 2.10±0.22 |
| *Ladder* [33] | **1.06±0.37** | **0.84±0.08** | - |
| *DEPICT* | 2.65±0.35 | 2.10±0.11 | **1.91±0.06** |

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

## 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 regularized 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.

## References


[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

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

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

[4] C. Biernacki, G. Celeux, and G. 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.

[5] X. Chen and D. Cai. Large scale spectral clustering with landmark-based representation. In *AAAI*, 2011.

[6] H. Gao, F. Nie, X. Li, and H. Huang. Multi-view subspace clustering. *International Conference on Computer Vision (ICCV 2015)*, pages 4238–4246, 2015.

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

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

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

[10] K. Kailing, H.-P. Kriegel, and P. 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.

[11] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[12] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3581–3589, 2014.

[14] A. Krause, P. Perona, and R. G. Gomes. Discriminative clustering by regularized information maximization. In *Advances in neural information processing systems (NIPS)*, pages 775–783, 2010.

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

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] D.-H. 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.

[18] H. Li, K. Zhang, and T. 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.

[19] Y. Li, F. Nie, H. Huang, and J. Huang. Large-scale multi-view spectral clustering via bipartite graph. *Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 2015.

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

[21] D. Luo, C. Ding, and H. Huang. Consensus spectral clustering. *ICDE*, 2010.

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

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

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

[25] F. Nie, C. Deng, H. Wang, X. Gao, and H. Huang. New l1-norm relaxations and optimizations for graph clustering. *Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, 2016.

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

[27] F. Nie, X. Wang, and H. 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.

[28] F. Nie, X. Wang, M. I. Jordan, and H. Huang. The constrained laplacian rank algorithm for graph-based clustering. In *AAAI*, pages 1969–1976. Citeseer, 2016.

[29] F. Nie, Z. Zeng, I. W. Tsang, D. Xu, and C. 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.

[30] N. Pitelis, C. Russell, and L. 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.

[31] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

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

[33] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3546–3554, 2015.

[34] S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio, and X. Muller. The manifold tangent classifier. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2294–2302, 2011.

[35] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. 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.

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

[37] M. E. Sargin, Y. Yemez, E. Erzin, and A. 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.

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

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

[40] T. Sim, S. Baker, and M. 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.

[41] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[42] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. Learning deep representations for graph clustering. In *AAAI*, pages 1293–1299, 2014.

[43] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, pages 1692–1700, 2014.

[44] V. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[45] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. 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.

[46] D. Wang, F. Nie, and H. 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.

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

[48] X. Wang, F. Nie, and H. 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.

[49] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. 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.

[50] Z. Wang, Y. Yang, S. Chang, J. Li, S. Fong, and T. 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.

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

[52] L. Wolf, T. Hassner, and I. 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.

[53] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning (ICML)*, 2016.

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

[55] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *Advances in neural information processing systems (NIPS)*, pages 1537–1544, 2004.

[56] W. Xu, X. Liu, and Y. 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.

[57] J. Yang, D. Parikh, and D. 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.

[58] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10):2761–2773, 2010.

[59] J. Ye, Z. Zhao, and M. Wu. Discriminative k-means for clustering. In *Advances in neural information processing systems (NIPS)*, pages 1649–1656, 2008.

[60] R. Yeh, C. Chen, T. Y. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016.

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

[62] W. Zhang, X. Wang, D. Zhao, and X. Tang. Graph degree linkage: Agglomerative clustering on a directed graph. In *European Conference on Computer Vision*, pages 428–441. Springer, 2012.

[63] W. Zhang, D. Zhao, and X. Wang. Agglomerative clustering via maximum incremental path integral. *Pattern Recognition*, 46(11):3056–3065, 2013.

[64] B. Zhao, F. Wang, and C. Zhang. Efficient multiclass maximum margin clustering. In *ICML*, pages 1248–1255, 2008.

[65] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015.

[66] F. Zhou, F. De la Torre, and J. 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.