arXiv:1611.05148v3 [cs.CV] 28 Jun 2017

Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering*

Zhuxi Jiang[1], Yin Zheng[2], Huachun Tan[1], Bangsheng Tang[3], Hanning Zhou[3]
[1]Beijing Institute of Technology, Beijing, China
[2]Tencent AI Lab, Shenzhen, China
[3]Hulu LLC., Beijing, China
{zjiang, tanhc}@bit.edu.cn, yinzheng@tencent.com,
bangsheng.tang@gmail.com, eric.zhou@hulu.com

June 29, 2017

Abstract

Clustering is among the most fundamental tasks in machine learning and artificial intelligence. In this paper, we propose Variational Deep Embedding (VaDE), a novel unsupervised generative clustering approach within the framework of Variational Auto-Encoder (VAE). Specifically, VaDE models the data generative procedure with a Gaussian Mixture Model (GMM) and a deep neural network (DNN): 1) the GMM picks a cluster; 2) from which a latent embedding is generated; 3) then the DNN decodes the latent embedding into an observable. Inference in VaDE is done in a variational way: a different DNN is used to encode observables to latent embeddings, so that the evidence lower bound (ELBO) can be optimized using the Stochastic Gradient Variational Bayes (SGVB) estimator and the *reparameterization* trick. Quantitative comparisons with strong baselines are included in this paper, and experimental results show that VaDE significantly outperforms the state-of-the-art clustering methods on 5 benchmarks from various modalities. Moreover, by VaDE's generative nature, we show its capability of generating highly realistic samples for any specified cluster, without using supervised information during training.

1 Introduction

Clustering is the process of grouping similar objects together, which is one of the most fundamental tasks in machine learning and artificial intelligence. Over the past decades, a large family of clustering algorithms have been developed and successfully

*This paper is accepted by IJCAI 2017, http://ijcai-17.org/accepted-papers.html

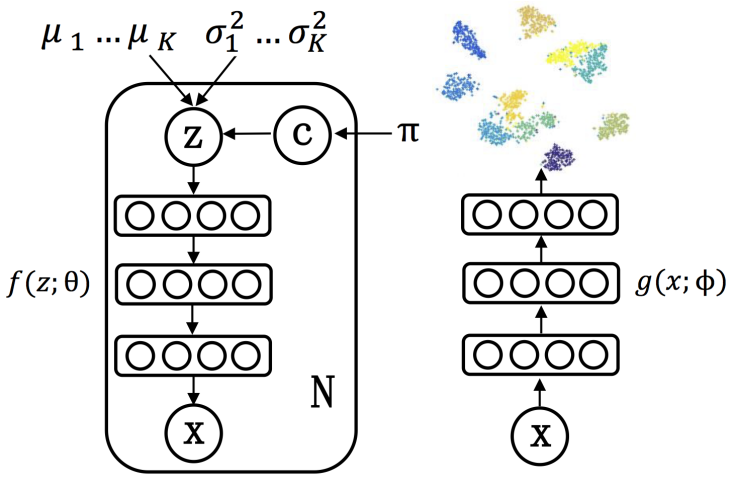


Figure 1: The diagram of VaDE. The data generative process of VaDE is done as follows: 1) a cluster is picked from a GMM model; 2) a latent embedding is generated based on the picked cluster; 3) DNN $f(\mathbf{z};\boldsymbol{\theta})$ decodes the latent embedding into an observable $\mathbf{x}$. A encoder network $g(\mathbf{x};\boldsymbol{\phi})$ is used to maximize the ELBO of VaDE.

applied in enormous real world tasks Ng *et al.* [2002]; Xie *et al.* [2016]; Yang *et al.* [2010]; Ye *et al.* [2008]. Generally speaking, there is a dichotomy of clustering methods: Similarity-based clustering and Feature-based clustering. Similarity-based clustering builds models upon a distance matrix, which is a $N \times N$ matrix that measures the distance between each pair of the N samples. One of the most famous similarity-based clustering methods is Spectral Clustering (SC) Von Luxburg [2007], which leverages the Laplacian spectra of the distance matrix to reduce dimensionality before clustering. Similarity-based clustering methods have the advantage that domain-specific similarity or kernel functions can be easily incorporated into the models. But these methods suffer scalability issue due to super-quadratic running time for computing spectra.

Different from similarity-based methods, a feature-based method takes a $N \times D$ matrix as input, where N is the number of samples and D is the feature dimension. One popular feature-based clustering method is K-means, which aims to partition the samples into K clusters so as to minimize the within-cluster sum of squared errors. Another representative feature-based clustering model is Gaussian Mixture Model (GMM), which assumes that the data points are generated from a Mixture-of-Gaussians (MoG), and the parameters of GMM are optimized by the Expectation Maximization (EM) algorithm. One advantage of GMM over K-means is that a GMM can generate samples by estimation of data density. Although K-means, GMM and their variants Liu *et al.* [2010]; Ye *et al.* [2008] have been extensively used, learning good representations most suitable for clustering tasks is left largely unexplored.

Recently, deep learning has achieved widespread success in numerous machine learning tasks He *et al.* [2016]; Krizhevsky *et al.* [2012]; Szegedy *et al.* [2015]; Zheng

et al. [2014a,b, 2015, 2016], where learning good representations by deep neural networks (DNN) lies in the core. Taking a similar approach, it is conceivable to conduct clustering analysis on good representations, instead of raw data points. In a recent work, Deep Embedded Clustering (DEC) Xie *et al.* [2016] was proposed to simultaneously learn feature representations and cluster assignments by deep neural networks. Although DEC performs well in clustering, similar to K-means, DEC cannot model the generative process of data, hence is not able to generate samples. Some recent works, e.g. VAE Kingma and Welling [2014], GAN Goodfellow *et al.* [2014] , PixelRNN Oord *et al.* [2016], InfoGAN Chen *et al.* [2016] and PPGN Nguyen *et al.* [2016], have shown that neural networks can be trained to generate meaningful samples. The motivation of this work is to develop a *clustering* model based on neural networks that 1) learns good representations that capture the statistical structure of the data, and 2) is capable of generating samples.

In this paper, we propose a clustering framework, Variational Deep Embedding (VaDE), that combines VAE Kingma and Welling [2014] and a Gaussian Mixture Model for clustering tasks. VaDE models the data generative process by a GMM and a DNN f: 1) a cluster is picked up by the GMM; 2) from which a latent representation $\mathbf{z}$ is sampled; 3) DNN f decodes $\mathbf{z}$ to an observation $\mathbf{x}$. Moreover, VaDE is optimized by using another DNN g to encode observed data $\mathbf{x}$ into latent embedding $\mathbf{z}$, so that the Stochastic Gradient Variational Bayes (SGVB) estimator and the *reparameterization* trick Kingma and Welling [2014] can be used to maximize the evidence lower bound (ELBO). VaDE generalizes VAE in that a Mixture-of-Gaussians prior replaces the single Gaussian prior. Hence, VaDE is by design more suitable for clustering tasks[1]. Specifically, the main contributions of the paper are:

- We propose an unsupervised generative clustering framework, VaDE, that combines VAE and GMM together.
- We show how to optimize VaDE by maximizing the ELBO using the SGVB estimator and the *reparameterization* trick;
- Experimental results show that VaDE outperforms the state-of-the-art clustering models on 5 datasets from various modalities by a large margin;
- We show that VaDE can generate highly realistic samples for any specified cluster, without using supervised information during training.

The diagram of VaDE is illustrated in Figure 1.

2 Related Work

Recently, people find that learning good representations plays an important role in clustering tasks. For example, DEC Xie *et al.* [2016] was proposed to learn feature representations and cluster assignments simultaneously by deep neural networks. In

[1] Although people can use VaDE to do unsupervised feature learning or semi-supervised learning tasks, we only focus on clustering tasks in this work.

fact, DEC learns a mapping from the observed space to a lower-dimensional latent space, where it iteratively optimizes the KL divergence to minimize the within-cluster distance of each cluster. DEC achieved impressive performances on clustering tasks. However, the feature embedding in DEC is designed specifically for clustering and fails to uncover the real underlying structure of the data, which makes the model lack of the ability to extend itself to other tasks beyond clustering, such as generating samples.

The deep generative models have recently attracted much attention in that they can capture the data distribution by neural networks, from which unseen samples can be generated. GAN and VAE are among the most successful deep generative models in recent years. Both of them are appealing unsupervised generative models, and their variants have been extensively studied and applied in various tasks such as semi-supervised classification Abbasnejad *et al.* [2016]; Kingma *et al.* [2014]; Maaløe *et al.* [2016]; Makhzani *et al.* [2016]; Salimans *et al.* [2016], clustering Makhzani *et al.* [2016] and image generation Dosovitskiy and Brox [2016]; Radford *et al.* [2016].

For example, Abbasnejad *et al.* [2016] proposed to use a mixture of VAEs for semi-supervised classification tasks, where the mixing coefficients of these VAEs are modeled by a Dirichlet process to adapt its capacity to the input data. SB-VAE Nalisnick and Smyth [2016] also applied Bayesian nonparametric techniques on VAE, which derived a stochastic latent dimensionality by a stick-breaking prior and achieved good performance on semi-supervised classification tasks. VaDE differs with SB-VAE in that the cluster assignment and the latent representation are jointly considered in the Gaussian mixture prior, whereas SB-VAE separately models the latent representation and the class variable, which fails to capture the dependence between them. Additionally, VaDE does not need the class label during training, while the labels of data are required by SB-VAE due to its semi-supervised setting. Among the variants of VAE, Adversarial Auto-Encoder(AAE) Makhzani *et al.* [2016] can also do unsupervised clustering tasks. Different from VaDE, AAE uses GAN to match the aggregated posterior with the prior of VAE, which is much more complex than VaDE on the training procedure. We will compare AAE with VaDE in the experiments part.

Similar to VaDE, Nalisnick *et al.* [2016] proposed DLGMM to combine VAE and GMM together. The crucial difference, however, is that VaDE uses a mixture of Gaussian prior to replace the single Gaussian prior of VAE, which is suitable for clustering tasks by nature, while DLGMM uses a mixture of Gaussian distribution as the approximate posterior of VAE and does not model the class variable. Hence, VaDE generalizes VAE to clustering tasks, whereas DLGMM is used to improve the capacity of the original VAE and is not suitable for clustering tasks by design. The recently proposed GM-CVAE Shu *et al.* [2016] also combines VAE with GMM together. However, the GMM in GM-CVAE is used to model the transitions between video frames, which is the main difference with VaDE.

3 Variational Deep Embedding

In this section, we describe Variational Deep Embedding (VaDE), a model for probabilistic clustering problem within the framework of Variational Auto-Encoder (VAE).

3.1 The Generative Process

Since VaDE is a kind of unsupervised generative approach to clustering, we herein first describe the generative process of VaDE. Specifically, suppose there are K clusters, an observed sample $\mathbf{x} \in \mathbb{R}^D$ is generated by the following process:

1. Choose a cluster $c \sim \text{Cat}(\boldsymbol{\pi})$
2. Choose a latent vector $\mathbf{z} \sim \mathcal{N}\left(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2 \mathbf{I}\right)$
3. Choose a sample $\mathbf{x}$:
   (a) If $\mathbf{x}$ is binary
      i. Compute the expectation vector $\boldsymbol{\mu}_x$

$$\boldsymbol{\mu}_x = f(\mathbf{z}; \boldsymbol{\theta}) \tag{1}$$

      ii. Choose a sample $\mathbf{x} \sim \text{Ber}(\boldsymbol{\mu}_x)$
   (b) If $\mathbf{x}$ is real-valued
      i. Compute $\boldsymbol{\mu}_x$ and $\boldsymbol{\sigma}_x^2$

$$[\boldsymbol{\mu}_x; \log \boldsymbol{\sigma}_x^2] = f(\mathbf{z}; \boldsymbol{\theta}) \tag{2}$$

      ii. Choose a sample $\mathbf{x} \sim \mathcal{N}\left(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x^2 \mathbf{I}\right)$

where K is a predefined parameter, π_k is the prior probability for cluster k, $\boldsymbol{\pi} \in \mathbb{R}_+^K$, $1 = \sum_{k=1}^K \pi_k$, $\text{Cat}(\boldsymbol{\pi})$ is the categorical distribution parametrized by $\boldsymbol{\pi}$, $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c^2$ are the mean and the variance of the Gaussian distribution corresponding to cluster c, $\mathbf{I}$ is an identity matrix, $f(\mathbf{z}; \boldsymbol{\theta})$ is a neural network whose input is $\mathbf{z}$ and is parametrized by $\boldsymbol{\theta}$, $\text{Ber}(\boldsymbol{\mu}_x)$ and $\mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x^2)$ are multivariate Bernoulli distribution and Gaussian distribution parametrized by $\boldsymbol{\mu}_x$ and $\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x$, respectively. The generative process is depicted in Figure 1.

According to the generative process above, the joint probability $p(\mathbf{x}, \mathbf{z}, c)$ can be factorized as:

$$p(\mathbf{x}, \mathbf{z}, c) = p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}|c) p(c), \tag{3}$$

since $\mathbf{x}$ and c are independent conditioned on $\mathbf{z}$. And the probabilities are defined as:

$$p(c) = \text{Cat}(c|\boldsymbol{\pi}) \tag{4}$$

$$p(\mathbf{z}|c) = \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2 \mathbf{I}\right) \tag{5}$$

$$p(\mathbf{x}|\mathbf{z}) = \text{Ber}(\mathbf{x}|\boldsymbol{\mu}_x) \quad or \quad \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x^2 \mathbf{I}) \tag{6}$$

3.2 Variational Lower Bound

A VaDE instance is tuned to maximize the likelihood of the given data points. Given the generative process in Section 3.1, by using Jensen's inequality, the log-likelihood of VaDE can be written as:

$$\log p(\mathbf{x}) = \log \int_{\mathbf{z}} \sum_c p(\mathbf{x}, \mathbf{z}, c) d\mathbf{z}$$

$$\geq E_{q(\mathbf{z},c|\mathbf{x})}[\log \frac{p(\mathbf{x},\mathbf{z},c)}{q(\mathbf{z},c|\mathbf{x})}] = \mathcal{L}_{\text{ELBO}}(\mathbf{x}) \tag{7}$$

where $\mathcal{L}_{\text{ELBO}}$ is the evidence lower bound (ELBO), $q(\mathbf{z},c|\mathbf{x})$ is the variational posterior to approximate the true posterior $p(\mathbf{z},c|\mathbf{x})$. In VaDE, we assume $q(\mathbf{z},c|\mathbf{x})$ to be a mean-field distribution and can be factorized as:

$$q(\mathbf{z},c|\mathbf{x}) = q(\mathbf{z}|\mathbf{x})q(c|\mathbf{x}). \tag{8}$$

Then, according to Equation 3 and 8, the $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$ in Equation 7 can be rewritten as:

$$\begin{aligned}
\mathcal{L}_{\text{ELBO}}(\mathbf{x}) &= E_{q(\mathbf{z},c|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z},c)}{q(\mathbf{z},c|\mathbf{x})}\right] \\
&= E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z},c) - \log q(\mathbf{z},c|\mathbf{x})\right] \\
&= E_{q(\mathbf{z},c|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}|c) \\
&\quad + \log p(c) - \log q(\mathbf{z}|\mathbf{x}) - \log q(c|\mathbf{x})]
\end{aligned} \tag{9}$$

In VaDE, similar to VAE, we use a neural network g to model $q(\mathbf{z}|\mathbf{x})$:

$$[\tilde{\boldsymbol{\mu}}; \log \tilde{\boldsymbol{\sigma}}^2] = g(\mathbf{x};\boldsymbol{\phi}) \tag{10}$$

$$q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z};\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\sigma}}^2\mathbf{I}) \tag{11}$$

where $\boldsymbol{\phi}$ is the parameter of network g.

By substituting the terms in Equation 9 with Equations 4, 5, 6 and 11, and using the SGVB estimator and the *reparameterization* trick, the $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$ can be rewritten as: [2]

$$\begin{aligned}
\mathcal{L}_{\text{ELBO}}(\mathbf{x}) =& \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{D} x_i \log \boldsymbol{\mu}_x^{(l)}|_i + (1-x_i)\log(1-\boldsymbol{\mu}_x^{(l)}|_i) \\
&- \frac{1}{2}\sum_{c=1}^{K}\gamma_c \sum_{j=1}^{J}(\log \boldsymbol{\sigma}_c^2|_j + \frac{\tilde{\boldsymbol{\sigma}}^2|_j}{\boldsymbol{\sigma}_c^2|_j} + \frac{(\tilde{\boldsymbol{\mu}}|_j - \boldsymbol{\mu}_c|_j)^2}{\boldsymbol{\sigma}_c^2|_j}) \\
&+ \sum_{c=1}^{K}\gamma_c \log \frac{\pi_c}{\gamma_c} + \frac{1}{2}\sum_{j=1}^{J}(1+\log \tilde{\boldsymbol{\sigma}}^2|_j)
\end{aligned} \tag{12}$$

where L is the number of Monte Carlo samples in the SGVB estimator, D is the dimensionality of $\mathbf{x}$ and $\boldsymbol{\mu}_x^{(l)}$, x_i is the i^{th} element of $\mathbf{x}$, J is the dimensionality of $\boldsymbol{\mu}_c$, $\boldsymbol{\sigma}_c^2$, $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}^2$, and $*|_j$ denotes the j^{th} element of $*$, K is the number of clusters, π_c is the prior probability of cluster c, and γ_c denotes $q(c|\mathbf{x})$ for simplicity.

In Equation 12, we compute $\boldsymbol{\mu}_x^{(l)}$ as

$$\boldsymbol{\mu}_x^{(l)} = f(\mathbf{z}^{(l)};\theta), \tag{13}$$

[2] This is the case when the observation $\mathbf{x}$ is binary. For the real-valued situation, the ELBO can be obtained in a similar way.

where $\mathbf{z}^{(l)}$ is the l^{th} sample from $q(\mathbf{z}|\mathbf{x})$ by Equation 11 to produce the Monte Carlo samples. According to the *reparameterization* trick, $\mathbf{z}^{(l)}$ is obtained by

$$\mathbf{z}^{(l)} = \tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\sigma}} \circ \boldsymbol{\epsilon}^{(l)}, \tag{14}$$

where $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$, $\circ$ is element-wise multiplication, and $\tilde{\boldsymbol{\mu}}$, $\tilde{\boldsymbol{\sigma}}$ are derived by Equation 10.

We now describe how to formulate $\gamma_c \triangleq q(c|\mathbf{x})$ in Equation 12 to maximize the ELBO. Specifically, $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$ can be rewritten as:

$$\begin{aligned}
&\mathcal{L}_{\text{ELBO}}(\mathbf{x}) = E_{q(\mathbf{z},c|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z},c)}{q(\mathbf{z},c|\mathbf{x})}\right] \\
&= \int_{\mathbf{z}} \sum_{c} q(c|\mathbf{x}) q(\mathbf{z}|\mathbf{x}) \left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} + \log \frac{p(c|\mathbf{z})}{q(c|\mathbf{x})}\right] d\mathbf{z} \\
&= \int_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} - \int_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) D_{KL}(q(c|\mathbf{x})||p(c|\mathbf{z})) d\mathbf{z}
\end{aligned} \tag{15}$$

In Equation 15, the first term has no relationship with c and the second term is non-negative. Hence, to maximize $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$, $D_{KL}(q(c|\mathbf{x})||p(c|\mathbf{z})) \equiv 0$ should be satisfied. As a result, we use the following equation to compute $q(c|\mathbf{x})$ in VaDE:

$$q(c|\mathbf{x}) = p(c|\mathbf{z}) \equiv \frac{p(c)p(\mathbf{z}|c)}{\sum_{c'=1}^{K} p(c')p(\mathbf{z}|c')} \tag{16}$$

By using Equation 16, the information loss induced by the mean-field approximation can be mitigated, since $p(c|\mathbf{z})$ captures the relationship between c and $\mathbf{z}$. It is worth noting that $p(c|\mathbf{z})$ is only an approximation to $q(c|\mathbf{x})$, and we find it works well in practice[3].

Once the training is done by maximizing the ELBO w.r.t the parameters of $\{\boldsymbol{\pi}, \boldsymbol{\mu}_c, \boldsymbol{\sigma}_c, \boldsymbol{\theta}, \boldsymbol{\phi}\}$, $c \in \{1, \cdots, K\}$, a latent representation $\mathbf{z}$ can be extracted for each observed sample $\mathbf{x}$ by Equation 10 and Equation 11, and the clustering assignments can be obtained by Equation 16.

3.3 Understanding the ELBO of VaDE

This section, we provide some intuitions of the ELBO of VaDE. More specifically, the ELBO in Equation 7 can be further rewritten as:

$$\mathcal{L}_{\text{ELBO}}(\mathbf{x}) = E_{q(\mathbf{z},c|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - D_{KL}(q(\mathbf{z},c|\mathbf{x})||p(\mathbf{z},c)) \tag{17}$$

The first term in Equation 17 is the *reconstruction* term, which encourages VaDE to explain the dataset well. And the second term is the Kullback-Leibler divergence from the Mixture-of-Gaussians (MoG) prior $p(\mathbf{z}, c)$ to the variational posterior $q(\mathbf{z}, c|\mathbf{x})$, which regularizes the latent embedding $\mathbf{z}$ to lie on a MoG manifold.

[3]We approximate $q(c|\mathbf{x})$ by: 1) sampling a $\mathbf{z}^{(i)} \sim q(\mathbf{z}|\mathbf{x})$; 2) computing $q(c|\mathbf{x}) = p(c|\mathbf{z}^{(i)})$ according to Equation 16

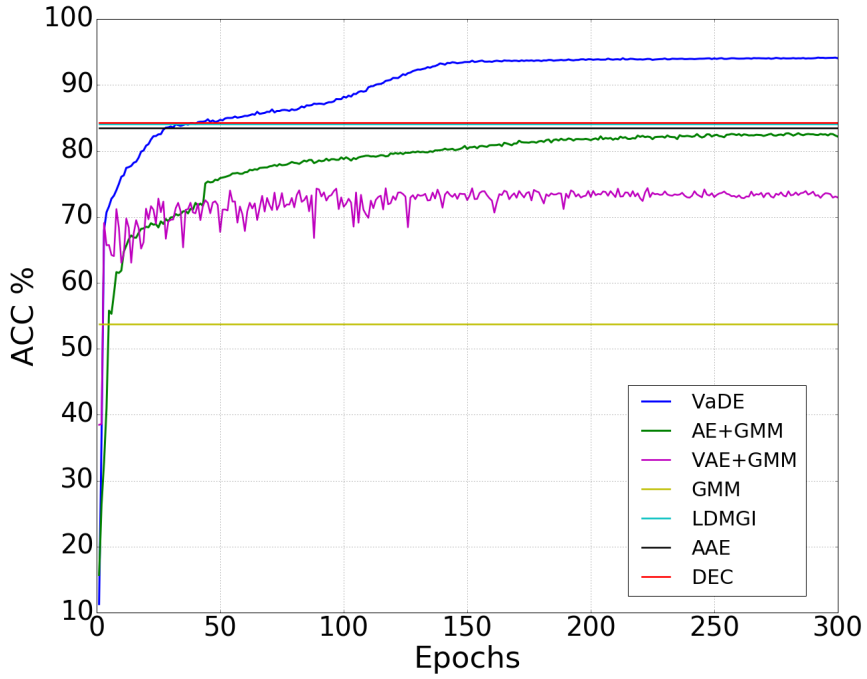
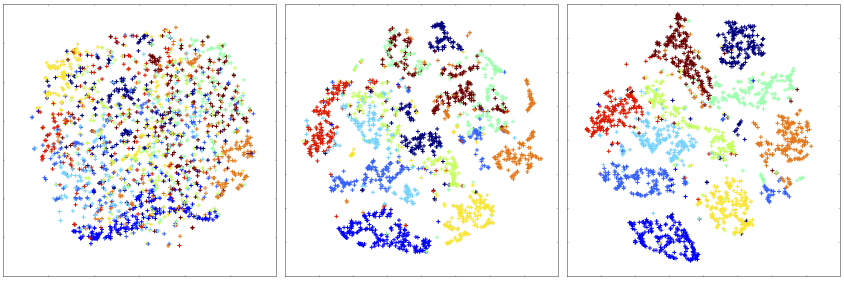


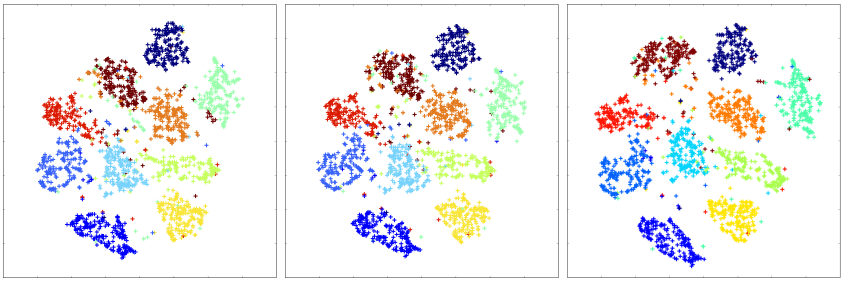
Figure 2: Clustering accuracy over number of epochs during training on MNIST. We also illustrate the best performances of DEC, AAE, LDMGI and GMM. It is better to view the figure in color.

To demonstrate the importance of the KL term in Equation 17, we train an Auto-Encoder (AE) with the same network architecture as VaDE first, and then apply GMM on the latent representations from the learned AE, since a VaDE model without the KL term is almost equivalent to an AE. We refer to this model as AE+GMM. We also show the performance of using GMM directly on the observed space (GMM), using VAE on the observed space and then using GMM on the latent space from VAE (VAE+GMM)[4], as well as the performances of LDMGI Yang *et al.* [2010], AAE Makhzani *et al.* [2016] and DEC Xie *et al.* [2016], in Figure 2. The fact that VaDE outperforms AE+GMM (without KL term) and VAE+GMM significantly confirms the importance of the regularization term and the advantage of jointly optimizing VAE and GMM by VaDE. We also present the illustrations of clusters and the way they are changed w.r.t. training epochs on MNIST dataset in Figure 3, where we map the latent representations $\mathbf{z}$ into 2D space by t-SNE Maaten and Hinton [2008].

[4]By doing this, VAE and GMM are optimized separately.



(a) Epoch 0 (11.35%) (b) Epoch 1 (55.63%) (c) Epoch 5 (72.40%)



(d) Epoch 50 (84.59%) (e) Epoch 120 (90.76%) (f) Epoch End (94.46%)

Figure 3: The illustration about how data is clustered in the latent space learned by VaDE during training on MNIST. Different colors indicate different ground-truth classes and the clustering accuracy at the corresponding epoch is reported in the bracket. It is clear to see that the latent representations become more and more suitable for clustering during training, which can also be proved by the increasing clustering accuracy.

4 Experiments

In this section, we evaluate the performance of VaDE on 5 benchmarks from different modalities: MNIST LeCun *et al.* [1998], HHAR Stisen *et al.* [2015], Reuters-10K Lewis *et al.* [2004], Reuters Lewis *et al.* [2004] and STL-10 Coates *et al.* [2011]. We provide quantitative comparisons of VaDE with other clustering methods including GMM, AE+GMM, VAE+GMM, LDGMI Yang *et al.* [2010], AAE Makhzani *et al.* [2016] and the strong baseline DEC Xie *et al.* [2016]. We use the same network architecture as DEC for a fair comparison. The experimental results show that VaDE achieves the state-of-the-art performance on all these benchmarks. Additionally, we also provide quantitatively comparisons with other variants of VAE on the discriminative quality of the latent representations. The code of VaDE is available at https://github.com/slim1017/VaDE.

4.1 Datasets Description

The following datasets are used in our empirical experiments.

Dataset	# Samples	Input Dim	# Clusters
MNIST	70000	784	10
HHAR	10299	561	6
REUTERS-10K	10000	2000	4
REUTERS	685071	2000	4
STL-10	13000	2048	10

Table 1: Datasets statistics

- **MNIST**: The MNIST dataset consists of 70000 handwritten digits. The images are centered and of size 28 by 28 pixels. We reshaped each image to a 784-dimensional vector.
- **HHAR**: The Heterogeneity Human Activity Recognition (HHAR) dataset contains 10299 sensor records from smart phones and smart watches. All samples are partitioned into 6 categories of human activities and each sample is of 561 dimensions.
- **REUTERS**: There are around 810000 English news stories labeled with a category tree in original Reuters dataset. Following DEC, we used 4 root categories: corporate/industrial, government/social, markets, and economics as labels and discarded all documents with multiple labels, which results in a 685071-article dataset. We computed tf-idf features on the 2000 most frequent words to represent all articles. Similar to DEC, a random subset of 10000 documents is sampled, which is referred to as Reuters-10K, since some spectral clustering methods (e.g. LDMGI) cannot scale to full Reuters dataset.
- **STL-10**: The STL-10 dataset consists of color images of 96-by-96 pixel size. There are 10 classes with 1300 examples each. Since clustering directly from raw pixels of high resolution images is rather difficult, we extracted features of images of STL-10 by ResNet-50 He *et al.* [2016], which were then used to test the performance of VaDE and all baselines. More specifically, we applied a 3×3 average pooling over the last feature map of ResNet-50 and the dimensionality of the features is 2048.

4.2 Experimental Setup

As mentioned before, the same network architecture as DEC is adopted by VaDE for a fair comparison. Specifically, the architectures of f and g in Equation 1 and Equation 10 are 10-2000-500-500-D and D-500-500-2000-10, respectively, where D is the input dimensionality. All layers are fully connected. Adam optimizer Kingma and Ba [2015] is used to maximize the ELBO of Equation 9, and the mini-batch size is 100. The learning rate for MNIST, HHAR, Reuters-10K and STL-10 is 0.002 and decreases every 10 epochs with a decay rate of 0.9, and the learning rate for Reuters is 0.0005 with a decay rate of 0.5 for every epoch. As for the generative process in Section 3.1,

Method	MNIST	HHAR	REUTERS-10K	REUTERS	STL-10
GMM	53.73	60.34	54.72	55.81	72.44
AE+GMM	82.18	77.67	70.13	70.98	79.83
VAE+GMM	72.94	68.02	69.56	60.89	78.86
LDMGI	84.09†	63.43	65.62	N/A	79.22
AAE	83.48	83.77	69.82	75.12	80.01
DEC	84.30†	79.86	74.32	75.63†	80.62
VaDE	**94.46**	**84.46**	**79.83**	**79.38**	**84.45**

†: Taken from Xie *et al.* [2016].

Table 2: Clustering accuracy (%) performance comparison on all datasets.

Method	k=3	k=5	k=10
VAE	18.43	15.69	14.19
DLGMM	9.14	8.38	8.42
SB-VAE	7.64	7.25	7.31
VaDE	**2.20**	**2.14**	**2.22**

Table 3: MNIST test error-rate (%) for kNN on latent space.

the multivariate Bernoulli distribution is used for MNIST dataset, and the multivariate Gaussian distribution is used for the others. The number of clusters is fixed to the number of classes for each dataset, similar to DEC. We will vary the number of clusters in Section 4.6.

Similar to other VAE-based models Kingma and Salimans [2016]; Sønderby *et al.* [2016], VaDE suffers from the problem that the reconstruction term in Equation 17 would be so weak in the beginning of training that the model might get stuck in an undesirable local minima or saddle point, from which it is hard to escape. In this work, pretraining is used to avoid this problem. Specifically, we use a Stacked Auto-Encoder to pretrain the networks f and g. Then all data points are projected into the latent space $\mathbf{z}$ by the pretrained network g, where a GMM is applied to initialize the parameters of $\{\boldsymbol{\pi}, \boldsymbol{\mu}_c, \boldsymbol{\sigma}_c\}$, $c \in \{1, \cdots, K\}$. In practice, few epochs of pretraining are enough to provide a good initialization of VaDE. We find that VaDE is not sensitive to hyperparameters after pretraining. Hence, we did not spend a lot of effort to tune them.

4.3 Quantitative Comparison

Following DEC, the performance of VaDE is measured by *unsupervised clustering accuracy (ACC)*, which is defined as:

$$\text{ACC} = \max_{m \in \mathcal{M}} \frac{\sum_{i=1}^{N} \mathbb{1}\{l_i = m(c_i)\}}{N}$$

where N is the total number of samples, l_i is the ground-truth label, c_i is the cluster assignment obtained by the model, and $\mathcal{M}$ is the set of all possible one-to-one mappings

between cluster assignments and labels. The best mapping can be obtained by using the KuhnMunkres algorithm Munkres [1957]. Similar to DEC, we perform 10 random restarts when initializing all clustering models and pick the result with the best objective value. As for LDMGI, AAE and DEC, we use the same configurations as their original papers. Table 2 compares the performance of VaDE with other baselines over all datasets. It can be seen that VaDE outperforms all these baselines by a large margin on all datasets. Specifically, on MNIST, HHAR, Reuters-10K, Reuters and STL-10 dataset, VaDE achieves ACC of 94.46%, 84.46%, 79.83%, 79.38% and 84.45%, which outperforms DEC with a relative increase ratio of 12.05%, 5.76%, 7.41%, 4.96% and 4.75%, respectively.

We also compare VaDE with SB-VAE Nalisnick and Smyth [2016] and DLGMM Nalisnick *et al.* [2016] on the discriminative power of the latent representations, since these two baselines cannot do clustering tasks. Following SB-VAE, the discriminative powers of the models' latent representations are assessed by running a k-Nearest Neighbors classifier (kNN) on the latent representations of MNIST. Table 3 shows the error rate of the kNN classifier on the latent representations. It can be seen that VaDE outperforms SB-VAE and DLGMM significantly[5].

Note that although VaDE can learn discriminative representations of samples, the training of VaDE is in a totally *unsupervised* way. Hence, we did not compare VaDE with other supervised models.

4.4 Generating Samples by VaDE

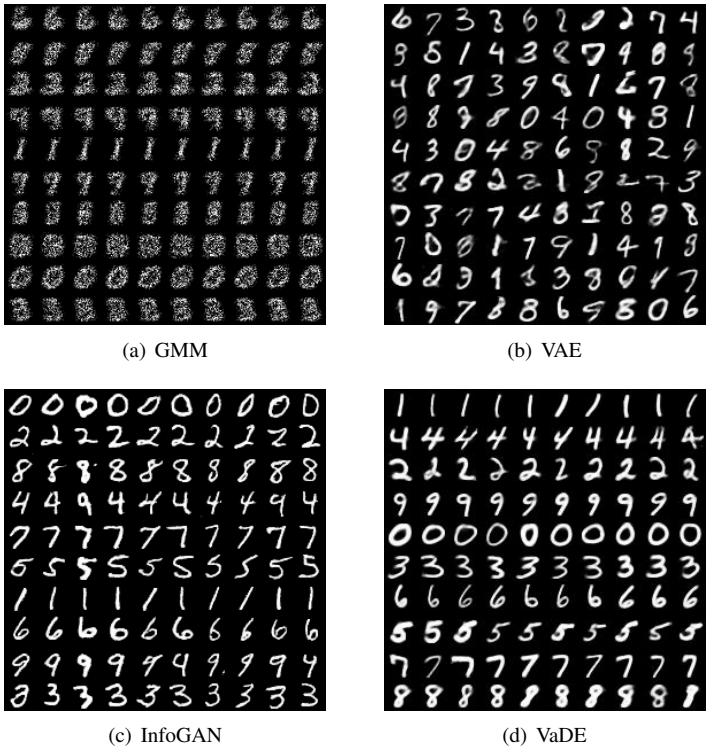
One major advantage of VaDE over DEC Xie *et al.* [2016] is that it is by nature a *generative* clustering model and can generate highly realistic samples for any specified cluster (class). In this section, we provide some qualitative comparisons on generating samples among VaDE, GMM, VAE and the state-of-art generative method InfoGAN Chen *et al.* [2016].

Figure 4 illustrates the generated samples for class 0 to 9 of MNIST by GMM, VAE, InfoGAN and VaDE, respectively. It can be seen that the digits generated by VaDE are smooth and diverse. Note that the classes of the samples from VAE cannot be specified. We can also see that the performance of VaDE is comparable with InfoGAN.

4.5 Visualization of Learned Embeddings

In this section, we visualize the learned representations of VAE, DEC and VaDE on MNIST dataset. To this end, we use t-SNE Maaten and Hinton [2008] to reduce the dimensionality of the latent representation $\mathbf{z}$ from 10 to 2, and plot 2000 randomly sampled digits in Figure 5. The first row of Figure 5 illustrates the ground-truth labels for each digit, where different colors indicate different labels. The second row of Figure 5 demonstrates the clustering results, where correctly clustered samples are colored with green and incorrect ones with red.

[5]We use the same network architecture for VaDE, SB-VAE in Table 3 for fair comparisons. Since there is no code available for DLGMM, we take the number of DLGMM directly from Nalisnick *et al.* [2016]. Note that Nalisnick and Smyth [2016] has already shown that the performance of SB-VAE is comparable to DLGMM.



(a) GMM (b) VAE

(c) InfoGAN (d) VaDE

Figure 4: The digits generated by GMM, VAE, InfoGAN and VaDE. Except (b), digits in the same row come from the same cluster.

From Figure 5 we can see that the original VAE which used a single Gaussian prior does not perform well in clustering tasks. It can also be observed that the embeddings learned by VaDE are better than those by VAE and DEC, since the number of incorrectly clustered samples is smaller. Furthermore, incorrectly clustered samples by VaDE are mostly located at the border of each cluster, where confusing samples usually appear. In contrast, a lot of the incorrectly clustered samples of DEC appear in the interior of the clusters, which indicates that DEC fails to preserve the inherent structure of the data. Some mistakes made by DEC and VaDE are also marked in Figure 5.

4.6 The Impact of the Number of Clusters

So far, the number of clusters for VaDE is set to the number of classes for each dataset, which is a prior knowledge. To demonstrate VaDE's representation power as an unsupervised clustering model, we deliberately choose different numbers of clusters K. Each row in Figure 6 illustrates the samples from a cluster grouped by VaDE on MNIST dataset, where K is set to 7 and 14 in Figure 6(a) and Figure 6(b), respectively. We

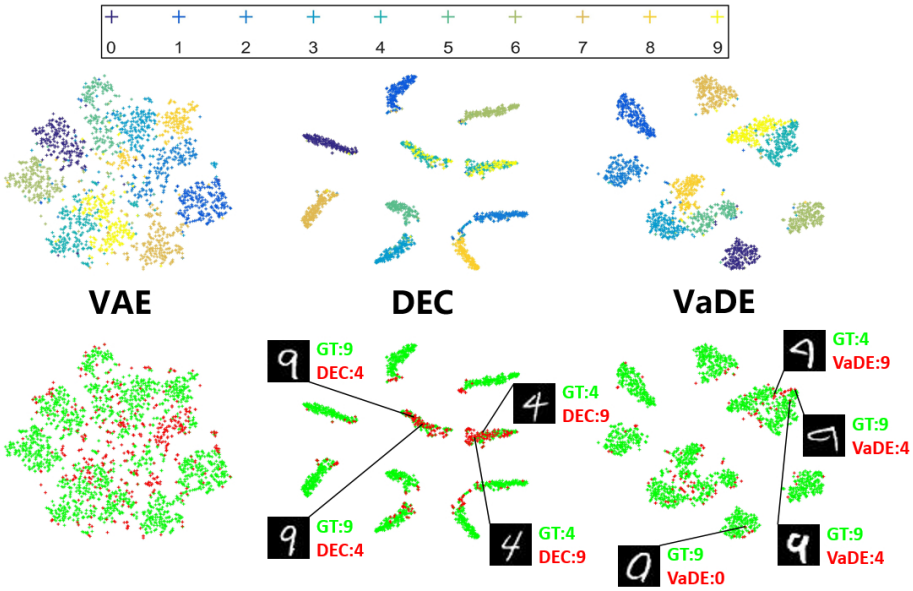
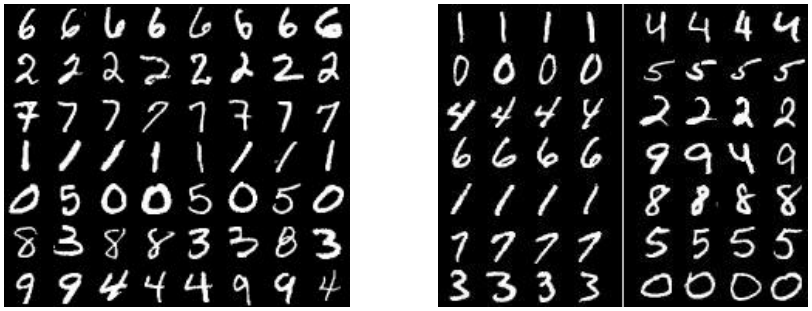


Figure 5: Visualization of the embeddings learned by VAE, DEC and VaDE on MNIST, respectively. The first row illustrates the ground-truth labels for each digit, where different colors indicate different labels. The second row demonstrates the clustering results, where correctly clustered samples are colored with green and, incorrect ones with red. GT:4 means the ground-truth label of the digit is 4, DEC:4 means DEC assigns the digit to the cluster of 4, and VaDE:4 denotes the assignment by VaDE is 4, and so on. It is better to view the figure in color.

can see that, if K is smaller than the number of classes, digits with similar appearances will be clustered together, such as 9 and 4, 3 and 8 in Figure 6(a). On the other hand, if K is larger than the number of classes, some digits will fall into sub-classes by VaDE, such as the fatter 0 and thinner 0, and the upright 1 and oblique 1 in Figure 6(b).

5 Conclusion

In this paper, we proposed Variational Deep Embedding (VaDE) which embeds the probabilistic clustering problems into a Variational Auto-Encoder (VAE) framework. VaDE models the data generative procedure by a GMM model and a neural network, and is optimized by maximizing the evidence lower bound (ELBO) of the log-likelihood of data by the SGVB estimator and the *reparameterization* trick. We compared the clustering performance of VaDE with strong baselines on 5 benchmarks from different modalities, and the experimental results showed that VaDE outperforms the state-of-the-art methods by a large margin. We also showed that VaDE could generate highly realistic samples conditioned on cluster information without using any supervised information during training. Note that although we use a MoG prior for VaDE in this



(a) 7 clusters (b) 14 clusters

Figure 6: Clustering MNIST with different numbers of clusters. We illustrate samples belonging to each cluster by rows.

paper, other mixture models can also be adopted in this framework flexibly, which will be our future work.

Acknowledgments

We thank the School of Mechanical Engineering of BIT (Beijing Institute of Technology) and Collaborative Innovation Center of Electric Vehicles in Beijing for their support. This work was supported by the National Natural Science Foundation of China (61620106002, 61271376). We also thank the anonymous reviewers.

References

Ehsan Abbasnejad, Anthony Dick, and Anton van den Hengel. Infinite variational autoencoder for semi-supervised learning. *arXiv preprint arXiv:1611.07800*, 2016.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016.

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*, 2011.

Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma and Tim Salimans. Improving variational autoencoders with inverse autoregressive flow. In *NIPS*, 2016.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 2004.

Jialu Liu, Deng Cai, and Xiaofei He. Gaussian mixture model with local consistency. In *AAAI*, 2010.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *ICML*, 2016.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. In *NIPS*, 2016.

James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 1957.

Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. *arXiv preprint arXiv:1605.06197*, 2016.

Eric Nalisnick, Lars Hertel, and Padhraic Smyth. Approximate inference for deep latent gaussian mixtures. 2016.

Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002.

Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

Rui Shu, James Brofos, Frank Zhang, Hung Hai Bui, Mohammad Ghavamzadeh, and Mykel Kochenderfer. Stochastic video prediction with conditional density estimation. In *ECCV Workshop on Action and Anticipation for Visual Learning*, 2016.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *NIPS*, 2016.

Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015.

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*, pages 1–9, 2015.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 2007.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

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*, 2010.

Jieping Ye, Zheng Zhao, and Mingrui Wu. Discriminative k-means for clustering. In *NIPS*, 2008.

Yin Zheng, Richard S Zemel, Yu-Jin Zhang, and Hugo Larochelle. A neural autoregressive approach to attention-based recognition. *International Journal of Computer Vision*, 113(1):67–79, 2014.

Yin Zheng, Yu-Jin Zhang, and H. Larochelle. Topic modeling of multimodal data: An autoregressive approach. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1370–1377, June 2014.

Y. Zheng, Yu-Jin Zhang, and H. Larochelle. A deep and autoregressive approach for topic modeling of multimodal data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PP(99):1–1, 2015.

Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. A neural autoregressive approach to collaborative filtering. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 764–773, 2016.

Appendix A

In this section, we provide the derivation of $q(c|\mathbf{x}) = E_{q(\mathbf{z}|\mathbf{x})}[p(c|\mathbf{z})]$.

The evidence lower bound $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$ can be rewritten as:

$$\begin{aligned}
\mathcal{L}_{\text{ELBO}}(\mathbf{x}) &= E_{q(\mathbf{z},c|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z},c)}{q(\mathbf{z},c|\mathbf{x})}\right] \\
&= \int_{\mathbf{z}} \sum_{c} q(\mathbf{z},c|\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z}|c)p(c)}{q(\mathbf{z},c|\mathbf{x})} d\mathbf{z} \\
&= \int_{\mathbf{z}} \sum_{c} q(c|\mathbf{x})q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{z})p(c|\mathbf{z})p(\mathbf{z})}{q(c|\mathbf{x})q(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int_{\mathbf{z}} \sum_{c} q(c|\mathbf{x})q(\mathbf{z}|\mathbf{x}) \left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} + \log \frac{p(c|\mathbf{z})}{q(c|\mathbf{x})}\right] d\mathbf{z} \\
&= \int_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} - \int_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \sum_{c} q(c|\mathbf{x}) \log \frac{q(c|\mathbf{x})}{p(c|\mathbf{z})} d\mathbf{z} \\
&= \int_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} - \int_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) D_{KL}(q(c|\mathbf{x})||p(c|\mathbf{z})) d\mathbf{z} \quad (18)
\end{aligned}$$

In Equation 18, the first term does not depend on c and the second term is non-negative. Thus, maximizing the lower bound $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$ with respect to $q(c|\mathbf{x})$ requires that $D_{KL}(q(c|\mathbf{x})||p(c|\mathbf{z})) = 0$. Thus, we have

$$\frac{q(c|\mathbf{x})}{p(c|\mathbf{z})} = \nu$$

where ν is a constant.

Since $\sum_c q(c|\mathbf{x}) = 1$ and $\sum_c p(c|\mathbf{z}) = 1$, we have:

$$\frac{q(c|\mathbf{x})}{p(c|\mathbf{z})} = 1$$

Taking the expectation on both sides, we can obtain:

$$q(c|\mathbf{x}) = E_{q(\mathbf{z}|\mathbf{x})}[p(c|\mathbf{z})]$$

Appendix B

Lemma 1 Given two multivariate Gaussian distributions $q(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}^2\mathbf{I})$ and $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$, we have:

$$\int q(\mathbf{z}) \log p(\mathbf{z})\, d\mathbf{z} = \sum_{j=1}^{J} -\frac{1}{2}\log\left(2\pi\sigma_j^2\right) - \frac{\tilde{\sigma}_j^2}{2\sigma_j^2} - \frac{(\tilde{\mu}_j - \mu_j)^2}{2\sigma_j^2} \tag{19}$$

where μ_j, σ_j, $\tilde{\mu}_j$ and $\tilde{\sigma}_j$ simply denote the j^{th} element of $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}$, respectively, and J is the dimensionality of $\mathbf{z}$.

Proof (of Lemma 1).

$$\begin{aligned}
&\int q(\mathbf{z}) \log p(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}^2\mathbf{I}) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})\, d\mathbf{z} \\
=&\int \prod_{j=1}^{J} \frac{1}{\sqrt{2\pi\tilde{\sigma}_j^2}} \exp(-\frac{(z_j-\tilde{\mu}_j)^2}{2\tilde{\sigma}_j^2}) \log \left[\prod_{j=1}^{J} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp(-\frac{(z_j-\mu_j)^2}{2\sigma_j^2})\right] d\mathbf{z} \\
=&\sum_{j=1}^{J} \int \frac{1}{\sqrt{2\pi\tilde{\sigma}_j^2}} \exp(-\frac{(z_j-\tilde{\mu}_j)^2}{2\tilde{\sigma}_j^2}) \log \left[\frac{1}{\sqrt{2\pi\sigma_j^2}} \exp(-\frac{(z_j-\mu_j)^2}{2\sigma_j^2})\right] dz_j \\
=&\sum_{j=1}^{J} \int \frac{1}{\sqrt{2\pi\tilde{\sigma}_j^2}} \exp(-\frac{(z_j-\tilde{\mu}_j)^2}{2\tilde{\sigma}_j^2}) \left[-\frac{1}{2}\log(2\pi\sigma_j^2)\right] dz_j - \int \frac{1}{\sqrt{2\pi\tilde{\sigma}_j^2}} \exp(-\frac{(z_j-\tilde{\mu}_j)^2}{2\tilde{\sigma}_j^2}) \frac{(z_j-\mu_j)^2}{2\sigma_j^2}\, dz_j \\
=&\sum_{j=1}^{J} -\frac{1}{2}\log(2\pi\sigma_j^2) - \int \frac{1}{\sqrt{2\pi\tilde{\sigma}_j^2}} \exp(-\frac{(z_j-\tilde{\mu}_j)^2}{2\tilde{\sigma}_j^2}) \frac{(z_j-\tilde{\mu}_j)^2 + 2(z_j-\tilde{\mu}_j)(\tilde{\mu}_j-\mu_j) + (\tilde{\mu}_j-\mu_j)^2}{2\tilde{\sigma}_j^2} \frac{\tilde{\sigma}_j^2}{\sigma_j^2}\, dz_j \\
=&C - \frac{\tilde{\sigma}_j^2}{\sigma_j^2} \int \frac{1}{\sqrt{2\pi\tilde{\sigma}_j^2}} \exp(-\frac{(z_j-\tilde{\mu}_j)^2}{2\tilde{\sigma}_j^2}) \frac{(z_j-\tilde{\mu}_j)^2}{2\tilde{\sigma}_j^2}\, dz_j - \int \frac{1}{\sqrt{2\pi\tilde{\sigma}_j^2}} \exp(-\frac{(z_j-\tilde{\mu}_j)^2}{2\tilde{\sigma}_j^2}) \frac{(\tilde{\mu}_j-\mu_j)^2}{2\sigma_j^2}\, dz_j \\
=&C - \frac{\tilde{\sigma}_j^2}{\sigma_j^2} \int \frac{1}{\sqrt{2\pi}} \exp(-\frac{x_j^2}{2}) \frac{x_j^2}{2}\, dx_j - \frac{(\tilde{\mu}_j-\mu_j)^2}{2\sigma_j^2} \\
=&C - \frac{\tilde{\sigma}_j^2}{\sigma_j^2} \int \frac{1}{\sqrt{2\pi}} (-\frac{x_j}{2})\, d(\exp(-\frac{x_j^2}{2})) - \frac{(\tilde{\mu}_j-\mu_j)^2}{2\sigma_j^2} \\
=&C - \frac{\tilde{\sigma}_j^2}{\sigma_j^2} \left[\frac{1}{\sqrt{2\pi}} (-\frac{x_j}{2}) \exp(-\frac{x_j^2}{2})\Big|_{-\infty}^{\infty} - \int \frac{1}{\sqrt{2\pi}} \exp(-\frac{x_j^2}{2})\, d(-\frac{x_j}{2})\right] - \frac{(\tilde{\mu}_j-\mu_j)^2}{2\sigma_j^2} \\
=&\sum_{j=1}^{J} -\frac{1}{2}\log\left(2\pi\sigma_j^2\right) - \frac{\tilde{\sigma}_j^2}{2\sigma_j^2} - \frac{(\tilde{\mu}_j-\mu_j)^2}{2\sigma_j^2}
\end{aligned}$$

where C denotes $\sum_{j=1}^{J} -\frac{1}{2}\log(2\pi\sigma_j^2)$ for simplicity.

Appendix C

In this section, we describe how to compute the evidence lower bound of VaDE. Specifically, the evidence lower bound can be rewritten as:

$$\begin{aligned}\mathcal{L}_{\text{ELBO}}(\mathbf{x}) =& E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] \\ &+ E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{z}|c)\right] \\ &+ E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(c)\right] \\ &- E_{q(\mathbf{z},c|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x})\right] \\ &- E_{q(\mathbf{z},c|\mathbf{x})}\left[\log q(c|\mathbf{x})\right] \end{aligned} \tag{20}$$

The Equation 20 can be computed by substituting Equation 4, 5, 6, 11 and 16 into Equation 20 and using **Lemma 1** in Appendix B. Specifically, each item of Equation 20 can be obtained as follows:

- $E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right]$:

  Recall that the observation $\mathbf{x}$ can be modeled as either a multivariate Bernoulli distribution or a multivariate Gaussian distribution. We provide the derivation of $E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right]$ for the multivariate Bernoulli distribution, and the derivation for the multivariate Gaussian case can be obtained in a similar way.

  Using the SGVB estimator, we can approximate the $E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right]$ as:

$$\begin{aligned} E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] &= \frac{1}{L}\sum_{l=1}^{L}\log p(\mathbf{x}|\mathbf{z}^{(l)}) \\ &= \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{D} x_i \log {\mu_x^{(l)}}_i + (1-x_i)\log(1-{\mu_x^{(l)}}_i) \end{aligned}$$

  where ${\boldsymbol{\mu}_x}^{(l)} = f(\mathbf{z}^{(l)};\boldsymbol{\theta})$, $\mathbf{z}^{(l)} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}^2\mathbf{I})$ and $\left[\tilde{\boldsymbol{\mu}}; \log\tilde{\boldsymbol{\sigma}}^2\right] = g(\mathbf{x};\boldsymbol{\phi})$. L is the number of Monte Carlo samples in the SGVB estimator and can be set to 1. D is the dimensionality of $\mathbf{x}$.

  Since the Monte Carlo estimate of the expectation above is non-differentiable w.r.t $\boldsymbol{\phi}$ when $z^{(l)}$ is directly sampled from $\mathbf{z} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}^2\mathbf{I})$, we use the *reparameterization* trick to obtain a differentiable estimation:

$$\mathbf{z}^{(l)} = \tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\sigma}} \circ \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$$

  where $\circ$ denotes the element-wise product.

- $E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{z}|c)\right]$:

$$E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{z}|c)\right] = \int_{\mathbf{z}}\sum_{c=1}^{K} q(c|\mathbf{x})q(\mathbf{z}|\mathbf{x})\log p(\mathbf{z}|c)\, d\mathbf{z}$$

$$= \sum_{c=1}^{K} q(c|\mathbf{x}) \int_{\mathbf{z}} \mathcal{N}(\mathbf{z}|\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}^2\mathbf{I}) \log \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2\mathbf{I})\, d\mathbf{z}$$

According to Lemma 1 in Appendix B, we have:

$$E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(\mathbf{z}|c)\right] = -\sum_{c=1}^{K} q(c|\mathbf{x}) \left[\frac{J}{2}\log(2\pi) + \frac{1}{2}(\sum_{j=1}^{J}\log\sigma_{cj}^2 + \sum_{j=1}^{J}\frac{\tilde{\sigma}_j^2}{\sigma_{cj}^2} + \sum_{j=1}^{J}\frac{(\tilde{\mu}_j - \mu_{cj})^2}{\sigma_{cj}^2})\right]$$

- $E_{q(\mathbf{z},c|\mathbf{x})}\left[\log p(c)\right]$:

$$\begin{aligned} E_{q(\mathbf{z},c|\mathbf{x})}[\log p(c)] &= \int_{\mathbf{z}} \sum_{c=1}^{K} q(\mathbf{z}|\mathbf{x}) q(c|\mathbf{x}) \log p(c)\, d\mathbf{z} \\ &= \int_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \sum_{c=1}^{K} q(c|\mathbf{x}) \log \pi_c\, d\mathbf{z} \\ &= \sum_{c=1}^{K} q(c|\mathbf{x}) \log \pi_c \end{aligned}$$

- $E_{q(\mathbf{z},c|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x})\right]$:

$$\begin{aligned} E_{q(\mathbf{z},c|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x})\right] &= \int_{\mathbf{z}} \sum_{c=1}^{K} q(c|\mathbf{x}) q(\mathbf{z}|\mathbf{x}) \log q(\mathbf{z}|\mathbf{x})\, d\mathbf{z} \\ &= \int_{\mathbf{z}} \mathcal{N}(\mathbf{z}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}^2\mathbf{I}) \log \mathcal{N}(\mathbf{z}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}^2\mathbf{I})\, d\mathbf{z} \end{aligned}$$

According to Lemma 1 in Appendix B, we have:

$$E_{q(\mathbf{z},c|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x})\right] = -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1 + \log\tilde{\sigma}_j^2)$$

- $E_{q(\mathbf{z},c|\mathbf{x})}\left[\log q(c|\mathbf{x})\right]$:

$$\begin{aligned} E_{q(\mathbf{z},c|\mathbf{x})}\left[\log q(c|\mathbf{x})\right] &= \int_{\mathbf{z}} \sum_{c=1}^{K} q(\mathbf{z}|\mathbf{x}) q(c|\mathbf{x}) \log q(c|\mathbf{x})\, d\mathbf{z} \\ &= \int_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \sum_{c=1}^{K} q(c|\mathbf{x}) \log q(c|\mathbf{x})\, d\mathbf{z} \\ &= \sum_{c=1}^{K} q(c|\mathbf{x}) \log q(c|\mathbf{x}) \end{aligned}$$

where μ_{cj}, σ_{cj}, $\tilde{\mu}_j$ and $\tilde{\sigma}_j$ simply denote the j^{th} element of $\boldsymbol{\mu}_c$, $\boldsymbol{\sigma}_c$, $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}$ described in Section 3 respectively. J is the dimensionality of $\mathbf{z}$ and K is the number of clusters.

For all the above equations, $q(c|\mathbf{x})$ is computed by Appendix A and can be approximated by the SGVB estimator and the *reparameterization* trick as follows:

$$q(c|\mathbf{x}) = E_{q(\mathbf{z}|\mathbf{x})}\left[p(c|\mathbf{z})\right] = \frac{1}{L}\sum_{l=1}^{L}\frac{p(c)p(\mathbf{z}^{(l)}|c)}{\sum_{c'=1}^{K}p(c')p(\mathbf{z}^{(l)}|c')}$$

where $\mathbf{z}^{(l)} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}^2\mathbf{I})$, $\left[\tilde{\boldsymbol{\mu}}; \log\tilde{\boldsymbol{\sigma}}^2\right] = g(\mathbf{x};\boldsymbol{\phi})$, $\mathbf{z}^{(l)} = \tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\sigma}} \circ \boldsymbol{\epsilon}^{(l)}$ and $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$.