Unsupervised Clustering through Gaussian Mixture Variational AutoEncoder with Non-Reparameterized Variational Inference and Std Annealing

Zhihan Li[†], Youjian Zhao[‡], Haowen Xu[†], Wenxiao Chen[†], Shangqing Xu[†], Yilin Li[†], Dan Pei[‡*]
Tsinghua University
Beijing National Research Center for Information Science and Technology (BNRist)
Beijing, China
[†]{lizhihan17, xhw15, chen-wx17, xsq17, liyilin16}@mails.tsinghua.edu.cn
[‡]{zhaoyoujian, peidan}@tsinghua.edu.cn

***Abstract*—Clustering has long been an important research topic in machine learning, and is highly valuable in many application tasks. In recent years, many methods have achieved high clustering performance by applying deep generative models. In this paper, we point out that directly using $q(z|y,x)$ instead of resorting to the mean-field approximation (as is adopted in previous works) in Gaussian Mixture Variational Auto-Encoder can benefit the unsupervised clustering task. We improve the performance of Gaussian Mixture VAE, by optimizing it with a Monte Carlo objective (including the $q(z|y,x)$ term), with non-reparameterized Variational Inference for Monte Carlo Objectives (VIMCO) method. In addition, we propose *std annealing* to stabilize the training process and empirically show its effects on forming well-separated embeddings with different variational inference methods. Experimental results on five benchmark datasets show that our proposed algorithm NVISA outperforms several baseline algorithms as well as the previous clustering methods based on Gaussian Mixture VAE.**

***Index Terms*—Unsupervised Clustering, Gaussian Mixture Variational Auto-Encoder, Std Annealing**

I. Introduction

Clustering is a fundamental and important research topic in machine learning and artificial intelligence, which aims at grouping similar examples together in an unsupervised manner. Traditional clustering methods like *k*-means [1] and Gaussian Mixture Models (GMMs) [2] represent the clusters using simple similarity measure and statistical distributions in data space or hand-crafted feature space, which need careful feature designs for specific tasks. However, it's quite challenging to manually design such features for complex data such as images, resulting in poor clustering performance.

Related Work. In recent years, with the help of deep neural networks (DNN), many researches have shown that learning good representations plays an important role in improving the accuracy of clustering on complex datasets. For example, Deep Embedded Clustering (DEC) [3] proposed a method to simultaneously learn feature representations and clustering centroids using DNNs, then applied simple *k*-means clustering on the low-dimensional feature space derived from DNN with the learned centroids. However, the clustering accuracy of DEC is also poor, since the training objective of the model does not match so well with the goal of classical clustering technique.

Another approach, quite different from DEC, is deep generative models, including variants of Variational Auto-Encoder (VAE) [4] and Generative Adversarial Network (GAN) [5]. They are able to model the clustering assignment as a latent variable, which benefits the unsupervised clustering task [6]–[8]. *i.e.*, the clustering assignment is also a part of the latent representations in the deep generative model based methods.

Along this direction, several methods for unsupervised clustering were proposed. Gaussian Mixture Generative Adversarial Network (GM-GAN) [6] proposed a variant of GAN where the probability distribution over the latent space is a mixture of Gaussians. However, for the GAN-based models, the constraints on the relationship between the clustering assignment latent variable and other latent variables is not very strong, which may downgrade the clustering performance on datasets more complicated than MNIST (see Section III-B). As for VAE-based models, Gaussian Mixture Variational Auto-Encoder (GMVAE[1]) [7] and Variational Deep Embedding (VaDE) [8] both proposed variants of Gaussian Mixture VAE, which have better constraints on the relationship between the clustering assignment latent variable (categorical latent variable) and other latent variables. Both of them applied mean-field approximation, in order to more easily deduce their training methods, based on Stochastic Gradient Variational Bayes (SGVB) estimator and *reparameterization* trick [4].

Our Approach. Gaussian Mixture VAE is appealing for clustering, since among all models, the VAE-based models can better characterize the constraints on the relationship between different types of latent variables, useful in the unsupervised clustering task which involves disentangling the categorical and other information of the input. However, the performance of the previous works (GMVAE [7] and VaDE [8]) seems not satisfactory. We noticed that both these works used the mean-

* Dan Pei is the corresponding author.

[1]We use Gaussian Mixture VAE to denote the VAE-based model with Categorical latent variable y and continuous Gaussian latent variable $\mathbf{z}$ depending on y, while GMVAE specifically denotes the previous work [7].

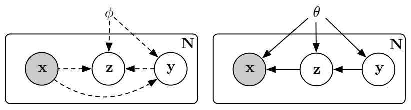


Fig. 1. Architecture of NVISA. The Categorical latent variable y as well as the continuous Gaussian latent variable $\mathbf{z}$ are regarded as part of the generative model (right), the left one denotes the variational model.

field approximation, which we advocate should be replaced by directly using $q(\mathbf{z}|y, \mathbf{x})$ (as shown in Fig. 1). In this paper, we propose NVISA (Gaussian Mixture VAE with **N**on-reparameterized **V**ariational **I**nference and **S**td **A**nnealing), an unsupervised clustering method based on Gaussian Mixture VAE without mean-field approximation, using Variational Inference for Monte Carlo Objectives (VIMCO) [9] as training method and the standard deviation (std) annealing trick (*i.e.*, gradually shrink the lower bound of std of the posteriors during training) proposed by us.

The main contributions of the paper are:

- We point out that directly using $q(\mathbf{z}|y, \mathbf{x})$ instead of using the mean-field approximation in previous works can benefit the unsupervised clustering task with Gaussian Mixture VAE.
- We experiment with three possible variational inference methods (EnumY, NVIL, VIMCO) to train NVISA, and finally use the non-reparameterized method VIMCO to achieve the best performance. Besides, we propose *std annealing* to stabilize the training of NVISA with VIMCO, and empirically show that the *std annealing* can help form well-separated embeddings and improve clustering accuracy with different variational inference methods.
- We evaluate NVISA with VIMCO and *std annealing* on five benchmark datasets. Our results show that NVISA outperforms several baseline clustering algorithms as well as the previous clustering methods based on Gaussian Mixture VAE.

II. NVISA FOR UNSUPERVISED CLUSTERING

In this section, we first present the architecture of NVISA. Unlike the previous works [7], [8], NVISA does not adopt the mean-field approximation. We then show how to optimize NVISA with three candidate variational inference methods (EnumY, NVIL and VIMCO). Finally, we propose the *std annealing*, which helps stabilize the training process of NVISA when VIMCO is used, resulting in better embeddings in the latent space with different variational inference methods.

A. Architecture

As in [7], [8], we use a Categorical latent variable $y \in \{1 \dots K\}$ to represent the clustering assignment of a given input $\mathbf{x}$, and a continuous Gaussian latent variable $\mathbf{z}$, which depends on y, to capture other information of $\mathbf{x}$. The whole generative model is thus composed of three variables, namely $\mathbf{x}$, y and $\mathbf{z}$, with parameters denoted as θ. We factorize the generative model as follows:

$$p_\theta(\mathbf{x}, y, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})\, p_\theta(\mathbf{z}|y)\, p_\theta(y) \tag{1}$$

where each variable is derived as:

$$p_\theta(y) = \mathrm{Cat}(\boldsymbol{\pi}) \tag{2a}$$

$$p_\theta(\mathbf{z}|y) = \mathcal{N}\big(\boldsymbol{\mu}_{\mathbf{z}}(y;\theta), \boldsymbol{\sigma}_{\mathbf{z}}{}^2(y;\theta)\,\mathbf{I}\big) \tag{2b}$$

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}\big(\boldsymbol{\mu}_{\mathbf{x}}(\mathbf{z};\theta), \boldsymbol{\sigma}_{\mathbf{x}}{}^2(\mathbf{z};\theta)\,\mathbf{I}\big) \tag{2c}$$

We let the prior distribution $p_\theta(y) = \mathrm{Cat}(\boldsymbol{\pi})$ assign equal probability to each possible choice of $y \in \{1 \dots K\}$. $p_\theta(\mathbf{z}|y)$ is assumed to be a Gaussian distribution, with $\boldsymbol{\mu}_{\mathbf{z}}(y;\theta)$ as its mean and $\boldsymbol{\sigma}_{\mathbf{z}}(y;\theta)$ as its standard deviation (std). $p_\theta(\mathbf{x}|\mathbf{z})$ is also assumed to be a Gaussian distribution, derived in a way similar to $\mathbf{z}$. $\boldsymbol{\mu}_{\mathbf{z}}(y;\theta)$ and $\boldsymbol{\sigma}_{\mathbf{z}}(y;\theta)$ are just learnable vectors for each y, respectively. $\boldsymbol{\mu}_{\mathbf{x}}(\mathbf{z};\theta)$ and $\boldsymbol{\sigma}_{\mathbf{x}}(\mathbf{z};\theta)$ are derived by neural networks, with parameters θ. Note that we add a constant lower-bound to the std of prior $p_\theta(\mathbf{z}|y)$:

$$\boldsymbol{\sigma}_{\mathbf{z}}{}'(y;\theta) = \mathbf{SoftPlus}(\boldsymbol{\sigma}_{\mathbf{z}}(y;\theta)) + 1$$

where $\mathbf{SoftPlus}(a) = \log(\exp(a) + 1)$. This is because we want the prior $p_\theta(\mathbf{z}|y)$ (which is learnable) to be at least as large as a unit Gaussian.

To apply variational inference on $p_\theta(\mathbf{x}, y, \mathbf{z})$, we need a separated $q_\phi(\mathbf{z}, y|\mathbf{x})$ to approximate the true posterior $p_\theta(\mathbf{z}, y|\mathbf{x})$. We factorize the variational posterior as:

$$q_\phi(\mathbf{z}, y|\mathbf{x}) = q_\phi(\mathbf{z}|y, \mathbf{x}) q_\phi(y|\mathbf{x}) \tag{3}$$

where each variable is derived as:

$$q_\phi(y|\mathbf{x}) = \mathrm{Cat}(\boldsymbol{\pi}(\mathbf{x};\phi)) \tag{4a}$$

$$q_\phi(\mathbf{z}|y,\mathbf{x}) = \mathcal{N}\big(\boldsymbol{\mu}_{\mathbf{z}}(\mathbf{x}, y;\phi), \boldsymbol{\sigma}_{\mathbf{z}}{}^2(\mathbf{x}, y;\phi)\,\mathbf{I}\big) \tag{4b}$$

$q_\phi(y|\mathbf{x})$ is assumed to be a Categorical distribution, while $q_\phi(\mathbf{z}|y, \mathbf{x})$ is assumed to be a Gaussian distribution. $\boldsymbol{\pi}(\mathbf{x};\phi)$, $\boldsymbol{\mu}_{\mathbf{z}}(\mathbf{x}, y;\phi)$ and $\boldsymbol{\sigma}_{\mathbf{z}}(\mathbf{x}, y;\phi)$ are all derived by neural networks with parameters ϕ. Note that, we also add a lower bound to the $\boldsymbol{\sigma}_{\mathbf{x}}(\mathbf{z};\theta)$ and $\boldsymbol{\sigma}_{\mathbf{z}}(\mathbf{x}, y;\phi)$ to stabilize the training of our model, see Section II-D for more details.

The whole architecture of our model, including $p_\theta(\mathbf{x}, y, \mathbf{z})$ and $q_\phi(\mathbf{z}, y|\mathbf{x})$, is shown as Fig. 1. In the generative network, $\mathbf{x}$ conditions on $\mathbf{z}$ and $\mathbf{z}$ conditions on y, hence the latent variable $\mathbf{z}$ should encode the label information. Thus, it's natural to let $\mathbf{z}$ condition on both $\mathbf{x}$ and y, *i.e.*, directly calculating $q_\phi(\mathbf{z}|y, \mathbf{x})$ rather than using the mean-field approximation $q_\phi(\mathbf{z}, y|\mathbf{x}) = q_\phi(\mathbf{z}|\mathbf{x}) q_\phi(y|\mathbf{x})$.

B. Using $q_\phi(\mathbf{z}|y, \mathbf{x})$ instead of Mean-Field Approximation

The previous works that use Gaussian Mixture VAE on unsupervised clustering [7], [8] both adopt mean-field approximation $q_\phi(\mathbf{z}, y|\mathbf{x}) = q_\phi(y|\mathbf{x})\, q_\phi(\mathbf{z}|\mathbf{x})$. By doing this, they managed to train their model without sampling y, the categorical latent variable, which is impossible to directly train[2] with SGVB and *reparameterization* trick. However, this mean-field approximation $q_\phi(y|\mathbf{x})\, q_\phi(\mathbf{z}|\mathbf{x})$ can only represent joint distributions where $\mathbf{z}$ and y are conditionally independent

[2]Although it's possible to use SGVB and categorical reparameterization with Gumbel-Softmax [10] to train Gaussian Mixture VAE with categorical latent variable, this method can only get around 80% clustering accuracy on MNIST in our experiments. Thus we do not discuss this method in our paper.

given $\mathbf{x}$, which may not hold in the real-world datasets in many cases. On the contrary, the factorization $q_\phi(\mathbf{z}|y,\mathbf{x})q_\phi(y|\mathbf{x})$ can represent all possible joint distributions $q_\phi(\mathbf{z},y|\mathbf{x})$, regardless of whether or not $\mathbf{z}$ and y are conditionally dependent given $\mathbf{x}$. Thus we choose the factorization $q_\phi(\mathbf{z}|y,\mathbf{x})q_\phi(y|\mathbf{x})$ deliberately. We shall see in Section III-C that, our model, which directly calculates $q_\phi(\mathbf{z}|y,\mathbf{x})$, significantly outperforms GMVAE, VaDE, and a variant of our model which uses mean-field approximation instead.

C. Three Candidate Variational Inference Methods for NVISA

Since we decide not to use the mean-field approximation, we need a new variational inference method, different from the previous work, to train NVISA. In order to deal with the non-reparameterized Categorical latent variable y, we found three candidate methods: the EnumY, which enumerates y and uses SGVB [4], [11], and two *non-reparameterized* method, NVIL [12] and VIMCO [9]. These methods are briefly described as follows.

EnumY In Gaussian Mixture VAE, if we enumerate $y = 1 \dots K$ and keep the $\mathbf{z}$ variable reparameterized, the evidence lower bound (ELBO) can be written into a summation over y and an expectation over $\mathbf{z}$ (5), which can be optimized by SGVB with *reparameterization* trick [4], [11]:

$$\begin{aligned}\mathcal{L}_{\text{ELBO}}(\mathbf{x};\theta,\phi) = \sum_{y=1}^{K} q_\phi(y|\mathbf{x})\,\mathbb{E}_{q_\phi(\mathbf{z}|y,\mathbf{x})}\big[\\ \log p_\theta(\mathbf{x},y,\mathbf{z}) - \log q_\phi(y|\mathbf{x}) - \log q_\phi(\mathbf{z}|y,\mathbf{x})\big] \quad (5)\end{aligned}$$

Unfortunately, this simplest method does not work as well as the non-reparameterized method, and is likely to get "mixture models" (see Section III-D).

NVIL Mnih and Gregor [12] proposed Neural Variational Inference and Learning (NVIL) to estimate the gradient of ELBO, when latent variables are NOT *reparameterized*. Let $f(\mathbf{x},y,\mathbf{z}) = \log p_\theta(\mathbf{x},y,\mathbf{z}) - \log q_\phi(\mathbf{z},y|\mathbf{x})$, and $\mathcal{L}_{\text{ELBO}}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z},y|\mathbf{x})}[f(\mathbf{x},y,\mathbf{z})]$, NVIL then estimates the gradient $\nabla\mathcal{L}_{\text{ELBO}}(\mathbf{x};\theta,\phi)$ by (6):

$$\begin{aligned}\nabla\,\mathcal{L}_{\text{ELBO}}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z},y|\mathbf{x})}\big[\nabla f(\mathbf{x},y,\mathbf{z})\\ + (f(\mathbf{x},y,\mathbf{z}) - C_\psi(\mathbf{x}) - c)\,\nabla\log q_\phi(\mathbf{z},y|\mathbf{x})\big] \quad (6)\end{aligned}$$

$C_\psi(\mathbf{x})$ is derived by another trainable neural network (which is called "baseline net") with parameter ψ, while c is a learnable constant. The gradient (6) is directly used to train the parameters θ, ϕ and ψ with stochastic gradient descent, instead of computing the gradient of a certain objective function.

In our experiments (Section III-D), the variance of NVIL gradient estimator is too large (which is theoretically possible), such that when training with NVIL, the loss of the model would diverge.

VIMCO Burda et al. [13] proposed the Monte Carlo objective for training VAEs. It is a tighter variational lower bound than ELBO, computed from multiple samples of latent variables. The original method of [13] requires the latent variables to be reparameterized, such that the gradient can be estimated by an SGVB-like algorithm. Mnih and Rezende [9] then extended this lower bound to train *non-reparameterized* latent variables. They suggested to take advantage of the multiple samples, in order to eliminate the need of using the separated learnable network $C_\psi(\mathbf{x})$ and constant c (as in NVIL). This *non-reparameterized* variational inference method is called VIMCO. In NVISA, the Monte Carlo objective can be formulated as (7).

$$\begin{aligned}&\mathcal{L}^M(\mathbf{x};\theta,\phi) = \\ &\mathbb{E}_{(y,\mathbf{z})^{(1:M)}\sim q_\phi(\mathbf{z},y|\mathbf{x})}\left[\log\frac{1}{M}\sum_{m=1}^{M}\frac{p_\theta(\mathbf{x},y^{(m)},\mathbf{z}^{(m)})}{q_\phi(\mathbf{z}^{(m)},y^{(m)}|\mathbf{x})}\right] \quad (7)\end{aligned}$$

where $(y,\mathbf{z})^{(1:M)}$ are M independent joint samples of $(y,\mathbf{z})$ from $q_\phi(y,\mathbf{z}|\mathbf{x})$, and each $(y,\mathbf{z})$ is sampled by first taking $y \sim q_\phi(y|\mathbf{x})$, then $\mathbf{z} \sim q_\phi(\mathbf{z}|y,\mathbf{x})$. The details of VIMCO gradient estimator can be found in [9].

Although more stable than NVIL, the VIMCO method may still cause loss divergence in a few tens or hundreds of epochs in our clustering task (see Section III-D). We developed the *std annealing* trick to stabilize training with VIMCO. With this trick, training with VIMCO converges and gives us better models than EnumY. We thus choose VIMCO as the training method for NVISA.

D. The Std Annealing Trick

We developed the *std annealing* trick (*i.e.*, add and gradually shrink the lower bound of std of the Gaussian posteriors during training) to stabilize the model training with VIMCO and Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$. Moreover, we empirically show that, *std annealing* can help learn better embeddings and lead to higher clustering performance on the raw image datasets like MNIST and fashion-MNIST, either with EnumY or VIMCO training method. The insights of developing this trick is shown below.

In our experiments, we observed that the loss may diverge only if 1) we assume $p_\theta(\mathbf{x}|\mathbf{z})$ to be a Gaussian distribution; *and* 2) we use NVIL *or* we use VIMCO method with just a small number of samples. If we assume $p_\theta(\mathbf{x}|\mathbf{z})$ to be a Bernoulli distribution (which is more stable than Gaussian in our context), the loss will not diverge. However, the Bernoulli $p_\theta(\mathbf{x}|\mathbf{z})$ is only applicable to a few datasets (*e.g.*, MNIST), and in our experiments, it does not work as well as our final model using Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$ even on such datasets. Thus we have to develop tricks that make the training with Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$ converge. Unfortunately, we cannot find a trick that can stabilize NVIL with Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$. For VIMCO with Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$, using a large number of samples may help the loss converge, but we found that it takes too long for training and its performance is not as good as VIMCO with a small number of samples and the stabilization trick that we managed to propose below, which turns out to also work better than EnumY (see experimental results in Section III-D).

Our proposed stabilization trick is based on the following observations and analysis. At the beginning of training, we observe huge loss and gradients with large variance, and then the loss diverges. We then try to clip the gradients by L2-norm, but the loss still gradually increases, and eventually reaches ∞. This suggests that the loss divergence is not because the gradients are too large, but because, when the loss has large variance, the gradients often guide the optimization to wrong directions. Looking closer into the Monte Carlo objective (7), we can see that it is just a lower-bound of the importance sampling based estimator for $\log p_\theta(\mathbf{x})$:

$$\begin{aligned}
&\mathbb{E}_{(y,\mathbf{z})^{(1:M)}\sim q_\phi(\mathbf{z},y|\mathbf{x})}\left[\log \frac{1}{M}\sum_{m=1}^{M}\frac{p_\theta(\mathbf{x},y^{(m)},\mathbf{z}^{(m)})}{q_\phi(\mathbf{z}^{(m)},y^{(m)}|\mathbf{x})}\right] \\
&\leq \log \mathbb{E}_{(y,\mathbf{z})^{(1:M)}\sim q_\phi(\mathbf{z},y|\mathbf{x})}\left[\frac{1}{M}\sum_{m=1}^{M}\frac{p_\theta(\mathbf{x},y^{(m)},\mathbf{z}^{(m)})}{q_\phi(\mathbf{z}^{(m)},y^{(m)}|\mathbf{x})}\right] \\
&= \log \mathbb{E}_{q_\phi(\mathbf{z},y|\mathbf{x})}\left[\frac{p_\theta(y,\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z},y|\mathbf{x})}\,p_\theta(\mathbf{x})\right] = \log p_\theta(\mathbf{x})
\end{aligned}$$

The theory of importance sampling states that, the variance of this estimator will be large if $q_\phi(\mathbf{z},y|\mathbf{x})$ does not approximate $p_\theta(\mathbf{z},y|\mathbf{x})$ well enough [14]. This is exactly what we will encounter at the beginning of training, when $p_\theta(\mathbf{z},y|\mathbf{x})$ are still rapidly evolving. Thus $q_\phi(\mathbf{z},y|\mathbf{x})$ can never approximate the true posterior well. The large variance on the importance sampling estimator may cause large variance on the objective function, and may further cause the gradients go towards wrong directions. When using Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$, for NVIL, the large variance may be caused by the incapability of baseline net. The EnumY does not use Monte Carlo objective, thus will not have this problem. When using VIMCO with a large number of samples, the variance is reduced, thus the training converges. However, as discussed above, these two method does not lead to good performance. We then try to find additional tricks to prevent the loss divergence when using Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$ and VIMCO with a small number of samples.

Since $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|y,\mathbf{x})$ are Gaussian distributions, adding large lower-bounds to their stds actually limit the scale of the probability densities, which prevent the scale of Monte Carlo objective from becoming too large when the importance sampling estimator still has large variance (which causes the loss diverges). Based on these observations and analysis, we propose our *std annealing* trick as follows: We add large lower-bounds on the standard deviation (std) of the Gaussian distributions $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|y,\mathbf{x})$ at the beginning of the training, gradually shrink the lower-bounds during training process, and finally set them to small constants at the end of training. The lower-bounds are added as:

$$\begin{aligned}
\boldsymbol{\sigma}_{\mathbf{x}}{}'(\mathbf{z};\theta) &= \mathbf{SoftPlus}(\boldsymbol{\sigma}_{\mathbf{x}}(\mathbf{z};\theta)) + \epsilon_x \\
\boldsymbol{\sigma}_{\mathbf{z}}{}'(\mathbf{x},y;\phi) &= \mathbf{SoftPlus}(\boldsymbol{\sigma}_{\mathbf{z}}(\mathbf{x},y;\phi)) + \epsilon_z
\end{aligned}$$

where ϵ_x and ϵ_z are annealing parameters, initialized to 1, shrunk by 0.5 after every 100 epochs, and set to 10^{-4} at the end. With the *std annealing* trick, the loss never diverges when training NVISA with VIMCO in our experiments in Section III-B. The resulting model achieves much higher clustering accuracy than the previous Gaussian mixture VAE-based methods on several benchmark datasets.

Moreover, we found that the *std annealing* can help learn better embeddings on the raw image datasets like MNIST and fashion-MNIST, either with EnumY or VIMCO training method. In NVISA, the ELBO can be written as:

$$\begin{aligned}
\mathcal{L}^M(\mathbf{x};\theta,\phi) =& \mathbb{E}_{q_\phi(\mathbf{z},y|\mathbf{x})}p_\theta(\mathbf{x}|\mathbf{z}) \\
&- D_{KL}(q_\phi(y,\mathbf{z})\|p_\theta(y,\mathbf{z})) - \mathbb{I}((Z,Y);X)
\end{aligned}$$

where the first term is called reconstruction loss, the second term is KL divergence between posterior and prior, the last term (mutual information) acts as a regularizer [15]. During training, the model pays more attention to optimize the reconstruction loss (which is much larger than the other two terms) rather than the KL term, while the latter one actually makes the posterior match the Gaussian mixture prior and helps form better embeddings for clustering. Thus, for the raw image data where some images from different classes might be similar, it's easy to get "mixture" clusters on the latent space. However, at the early stage of training, a large lower bound on the std of $p_\theta(\mathbf{x}|\mathbf{z})$ actually slows down the optimization of the reconstruction loss, and the regularizer (whose value is very small) only changes slightly, pushing the model to optimize the KL term and make the posterior match the Gaussian mixture prior. Note that, the *std annealing* is applied during training process, rather than modify the training objective (which may harm the final reconstruction). We empirically show that this leads to a better-separated embeddings, which significantly improves the clustering performance with different variational inference methods in Section III-D.

III. Experiments and Analysis

A. Setup

Datasets We evaluate NVISA on five benchmark datasets and compare its performance with several baseline clustering methods to show the superiority of our algorithm. The datasets include: MNIST [16], fashion-MNIST [17], STL-10 [18], CIFAR-10 and CIFAR-100 [19]. MNIST and fashion-MNIST are grayscale image datasets about hand-written digits and fashion clothes, while the other three datasets are color natural images. Previous works [8], [20] have found that raw pixels of these complex color images may not be suited for our clustering task, since the color information would be dominant, but unrelated to clustering. Thus, following previous works [6], [8], [21], we use raw pixel values for MNIST and fashion-MNIST, 2048-dimensional features[3] extracted by pre-trained Resnet-50 [22] (which is trained on ImageNet [23]) for STL-10, CIFAR-10 and CIFAR-100. These configurations are used for all the compared methods.

Evaluation Metric It is a common practice to use a K-class labeled classification dataset to evaluate the clustering performance of a model, by using the *unsupervised clustering accuracy* (ACC for short hereafter) [3], [6], [8], [21]:

$$ACC = \max_{f\in\mathcal{F}}\frac{\sum_{i=1}^{N}\mathbb{I}(f(c_i)=l_i)}{N}$$

[3]The raw image is resized through a bilinear filter and then passed through the pre-trained Resnet-50. A 7×7 average pooling is applied on the last feature map of Resnet-50 to get the 2048-dimensional features.

N is the total number of samples. c_i is the clustering assignment for the i-th testing data, produced by the model, while l_i is the ground-truth class label. $\mathcal{F}$ is the set of all possible one-to-one mappings between cluster assignments and labels. The best mapping f (as required by $\max$) can be obtained using KuhnMunkres algorithm [24]. Following previous work, we always use K, the number of ground-truth classes, as the number of clusters to fit. Note that $K = 100$ for CIFAR-100, and $K = 10$ for other datasets.

Other Configurations The network structure is shown in Fig. 5(a), while the hidden layers consist of several convolutional and deconvolutional layers. We use Adam [25] optimizer to optimize our model. The initial learning rate is set to 0.001, with a decay rate of 0.5 every 300 epochs. The number of VIMCO samples M is set to 50 for CIFAR-100, and 15 for all other datasets. According to [26], a uniform random noise is added on the raw pixel images for training, and then normalize the pixel value to $[0, 1]$ (by dividing 256). For other datasets, the features are standardized to 0 mean and 1 variance as input. According to [27], the latent $\mathbf{z}$ dimension should not be very large, in order to get more disentangled representations for clustering task. In this paper, we use $\mathbf{z}$ dimension $L = 10$ for MNIST, CIFAR-10 and CIFAR-100, $L = 16$ for more complex fashion-MNIST dataset, and $L = 5$ for the simplest STL-10 features. The cluster assignment of each input $\mathbf{x}$ is obtained by finding the y with the maximum $q_\phi(y|\mathbf{x})$.

B. Overall Performance

We evaluated our method NVISA on five benchmark datasets, and compared it with several baseline clustering methods, including AE+GMM (AutoEncoder + Gaussian Mixture Model), VAE+GMM, DEC [3], GM-GAN [6], GMVAE [7], VaDE [8], LTVAE-GMM [21] and LTVAE-full [21] (LTVAE with structure learning in latent space). The results are shown in Table I. We report the means and standard deviations in ten runs for our method NVISA and highlight the top two clustering accuracy on each dataset.

Our NVISA overall performs the best among all the compared methods on five benchmark datasets. Specifically, NVISA significantly outperforms GMVAE and VaDE, which are based on variants of Gaussian Mixture VAE. This highlights the benefits of our modifications ($q_\phi(\mathbf{z}|y, \mathbf{x})$, VIMCO and *std annealing*) to the Gaussian Mixture VAE. Note that, GMVAE uses an additional regularization term to help improve its clustering performance, which is not very applicable to our model. However, even without using this regularization term, NVISA still outperforms GMVAE on all the five datasets. Thus, we decide not to use any clustering specific regularization term in NVISA. Besides, the result of NVISA on MNIST is slightly worse than GM-GAN. However, NVISA outperforms GM-GAN on the more complicated fashion-MNIST dataset, which suggests NVISA should be at least no worse than GM-GAN. LTVAE [21] (Latent Tree Variational AutoEncoder) is a Variational AutoEncoder with a superstructure of discrete latent variables (*i.e.*, y) on top of the latent features (*i.e.*, z). LTVAE-GMM uses a fixed Gaussian mixture structure with one single y variable, while LTVAE-full performs structure learning to learn the superstructure through a latent tree model. The use of multiple latent variables and the learned structure help LTVAE-full achieves good clustering performance on several datasets. However, although not using the complex superstructure, NVISA achieves clustering results better than or comparable to LTVAE-full on all the five datasets. Each experiment of NVISA takes 3-12 hours to train on one GTX 1080 Ti graphical card for every dataset except CIFAR-100. The training time of CIFAR-100 is around 30 hours, since more VIMCO samples are needed for this 100-class dataset. Overall, such training overhead is comparable with other deep generative model based methods.

One thing to mention is that, the large standard deviation (2.64%) of our method on MNIST dataset is mainly caused by one run with around 90% accuracy. This might be caused by the bad initialization which makes the model mix up similar numbers like “4” and “9”. However, inspired by [8], we find that using parameters of a simple pretrain model as an initialization of NVISA can help tackle the above problem. Specifically, we pretrain an AutoEncoder who has the same network structure as NVISA, then perform GMM on the latent space obtained by AE. The network parameters are used to initialize the neural networks in NVISA, while the means of GMM are used as the initial means of $\mathbf{z}$ priors in NVISA (instead of random initialization). With this better initialization, NVISA is able to get 98.04%($\pm$0.17%) clustering accuracy on MNIST (evaluated in ten runs).

C. $q_\phi(\mathbf{z}|y, \mathbf{x})$ *vs Mean-Field Approximation*

Among the three variants of Gaussian Mixture VAE, NVISA significantly outperforms GMVAE and VaDE, as shown in Table I. Unlike the previous works, we deliberately use $q_\phi(\mathbf{z}|y, \mathbf{x})$, instead of the mean-field approximation, as discussed in Section II-B. Our experiments on MNIST (Table II) shows that the model using $q_\phi(\mathbf{z}|y, \mathbf{x})$ (adopted in NVISA with VIMCO and *std annealing*) outperforms the models using mean-field approximation $q_\phi(\mathbf{z}, y|\mathbf{x}) = q_\phi(\mathbf{z}|\mathbf{x})q_\phi(y|\mathbf{x})$. This is verified not only by the results of previous works, but also by the controlled experiment on NVISA with mean-field approximation. Fig. 4 also shows that the embeddings learned by NVISA are better separated than the controlled experiment with mean-field approximation. Thus, we stick to using $q_\phi(\mathbf{z}|y, \mathbf{x})$ in NVISA.

D. Comparison of Different Variational Inference Methods and Std Annealing

Table II shows the unsupervised clustering accuracy on MNIST of previous works, and NVISA trained with three variational inference methods (EnumY, NVIL and VIMCO) and other configurations. “(Bernoulli)” indicates that a model uses Bernoulli $p_\theta(\mathbf{x}|\mathbf{z})$, otherwise it uses Gaussian. “(mean-field)” indicates that a model uses the mean-field approximation, otherwise it uses $q_\phi(\mathbf{z}|y, \mathbf{x})$. “(1000 samples)” indicates that a model uses 1000 samples in VIMCO, otherwise it uses 15. “*” indicates the *std annealing* is applied. Note that,

TABLE I
COMPARISON OF UNSUPERVISED CLUSTERING ACCURACY (%) ON FIVE BENCHMARK DATASETS.

Method	MNIST	Fashion-MNIST	STL-10	CIFAR-10	CIFAR-100
AE+GMM	84.27	53.83	76.18	43.66	29.41
VAE+GMM	76.87	53.43	82.42	59.4	30.81
DEC [3]	84.30*	57.58	84.08	**70.31**	31.22
GM-GAN [6]	**99.24***	58.16*	-	-	-
GMVAE [7]	88.54*	50.15	65.72	49.79	14.46
VaDE [8]	94.46*	56.12	84.45*	59.69	23.58
LTVAE-GMM [21]	86.31	55.69	81.66	63.51	16.98
LTVAE-full [21]	86.32*	**61.32**	**90.00***	69.16	**38.52**
NVISA(best)	**98.40**	**66.14**	**91.22**	**76.96**	**38.79**
NVISA(avg±std)	**96.55±2.64**	**62.63±2.92**	**89.15±2.36**	**71.37±2.79**	**37.32±0.70**

Results marked by $*$ are excerpted from their original paper.
"-" means there is no published result and we didn't find any released code to produce it.

TABLE II
THE UNSUPERVISED CLUSTERING ACCURACY ON MNIST.

Variational Inference	Method	ACC (%)
	GMVAE	88.54
	VaDE	94.46
EnumY	NVISA (Bernoulli)	95.12
	NVISA	78.43
	NVISA*	93.81
NVIL	NVISA	N/A
	NVISA*	N/A
VIMCO	NVISA	N/A
	NVISA (Bernoulli)	95.74
	NVISA (1000 samples)	71.42
	NVISA*(1000 samples)	91.3
	NVISA* (mean-field)	95.25
	NVISA*	**98.40**

"*" indicates the *std annealing* is applied.
"N/A" means the loss diverges and we cannot get a meaningful result.

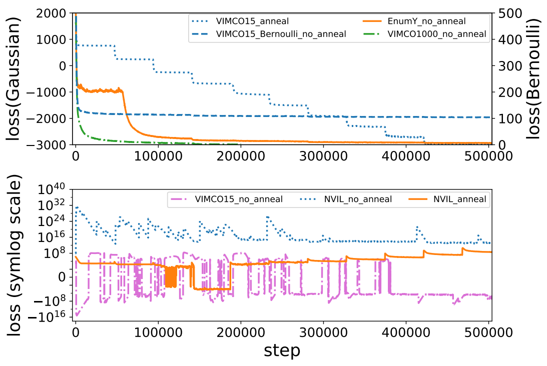


Fig. 2. Training loss of NVISA with different Variational Inference methods on MNIST. The loss curves are smoothed for better visualization. Only 200,000 steps are used for VIMCO-1000 since it takes too long to train. (Top) The right y-axis corresponds to the experiment with Bernoulli $p_\theta(\mathbf{x}|\mathbf{z})$, while the left y-axis corresponds to others with Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$. (Bottom) shows the diverged models discussed in Section III-D.

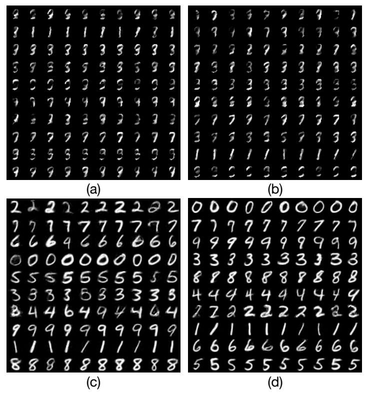
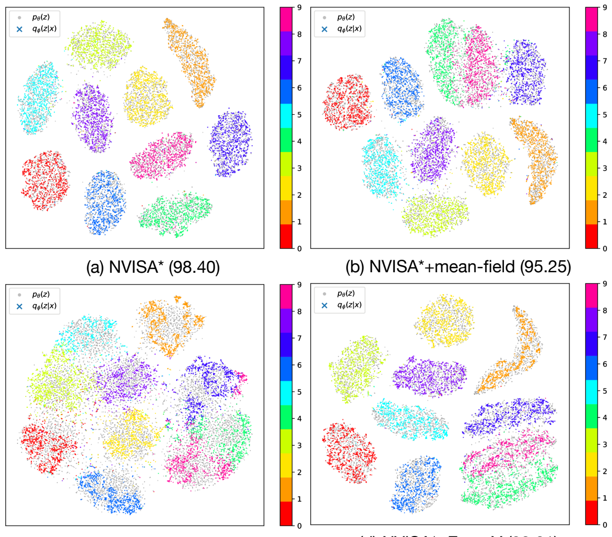


Fig. 3. Generated samples using NVISA with different variational inference methods on MNIST. (a) EnumY, (b) VIMCO with 1000 samples, (c) EnumY with *std annealing*, (d) VIMCO with 15 samples and *std annealing*

as discussed in Section II-D, although it's applicable to use Bernoulli $p_\theta(\mathbf{x}|\mathbf{z})$ for MNIST, we can only achieve 95.74% accuracy (with VIMCO) and 95.12% (with EnumY). Besides, Bernoulli assumption is not applicable for most datasets. Thus, we stick to use Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$ in NVISA for all datasets.

As shown in Fig. 2, training NVISA using VIMCO 15 samples without *std annealing*, and using NVIL with or without *std annealing*, all suffer from loss divergence. This is because the importance sampling estimators have large variance, as discussed in Section II-D. However, with the *std annealing*, NVISA with VIMCO 15 samples can be properly trained and achieve the best clustering performance.

One thing to notice is that, directly using Gaussian with EnumY or with VIMCO 1000 samples leads to poor performance. We found that, this is likely because these two models degenerate into true "mixture models" from the early stage in training: only the marginal distribution $p_\theta(\mathbf{z}) = \sum_y p_\theta(\mathbf{z}|y)p_\theta(y)$ is meaningful, while each individual component $p_\theta(\mathbf{z}|y)$ does not correspond to any particular $\mathbf{x}$. This phenomenon is shown as Fig. 3, where we obtain 100 $\mathbf{x}$ samples from each model by ancestral sampling, *i.e.*, each $\mathbf{x}$ is produced by first sampling a $y \sim p_\theta(y)$, then a $\mathbf{z} \sim p_\theta(\mathbf{z}|y)$, and finally $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$. Moreover, although the above two models can be trained without *std annealing*, we find that equipped with *std annealing* trick can also help them learn better embeddings in the latent space and alleviate the "mixture model" problem. As shown in Fig. 4, for the 10,000 test-set images on MNIST dataset, we use $\mathbf{z} \sim q_\phi(\mathbf{z}) = \mathbb{E}_{p^*(\mathbf{x})}\Sigma_y q_\phi(\mathbf{z}|y,\mathbf{x})q_\phi(y|\mathbf{x})$ to



(a) NVISA* (98.40)
(b) NVISA*+mean-field (95.25)
(c) NVISA+EnumY (78.43)
(d) NVISA*+EnumY (93.81)

Fig. 4. 10,000 encoded test-set digits and 10,000 samples from the prior on MNIST, dimensionally reduced by t-SNE, colored by model predicted labels. "*" indicates the *std annealing* is applied. Values in parenthesis are the test-set clustering accuracy.

get the encoded $\mathbf{z}$ ($p^*(\mathbf{x})$ denotes the true data distribution on test set). Besides, another 10,000 samples are obtained from the prior $\Sigma_y p_\theta(\mathbf{z}|y)$. These 20,000 samples are dimensionally reduced using t-SNE [28]. As shown in Fig. 4 (d), EnumY with *std annealing* learns better-separated embeddings and generates clear digits (Fig. 3 (c)). This is in sharp contrast with Fig. 4 (c) and Fig. 3 (a), which does not apply *std annealing*. Similar results are obtained when using VIMCO 1000 samples. It empirically shows that, with the *std annealing* trick, the model learns well-separated embeddings and matches the posteriors to the priors better. The *std annealing* help improve the accuracy of EnumY from (78.43%/51.41%) to (93.81%/60.56%) on (MNIST/fashion), and improve the accuracy of VIMCO 1000 samples from 71.42% to 91.30% on MNIST.

Fig. 4 (a) and Fig. 3 (d) shows the learned embeddings and generated digits by NVISA with VIMCO 15 samples and *std annealing*, which achieves 98.40% clustering accuracy on MNIST. It clearly shows that the embeddings are well-separated in the latent space, each corresponds to a category of digits in data space. According to the experiments and analysis above, we conclude that it is adequate to use the *non-reparameterized* method VIMCO with small number of samples and *std annealing* to train NVISA, targeted for an unsupervised clustering task.

E. Hyper-Parameter Selection

We did some extra experiments on MNIST to analyze the sensitivity of *std annealing* to the hyper-parameter selection, as shown in Fig. 5(b). Although the model is somewhat sensitive to the initial std lower bound and annealing frequency, it's easy to choose these parameters based on the following intuitions. Similar to the prior, we set the initial std lower bound to 1 to make each posterior at least a unit gaussian at

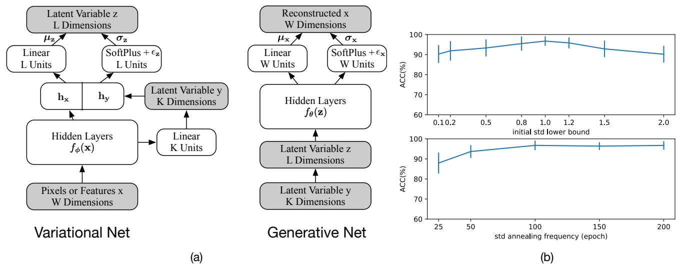


Fig. 5. (a) Network structure of NVISA. Gray nodes are random variables and white nodes are layers. (b) ACC of NVISA with different parameters on MNIST. Each configuration is evaluated in five runs.

the beginning of the training. During training process, we want the model to converge under each intermediate std lower bound, which empirically helps make the posterior match the Gaussian mixture prior, and leads to a better separated embedding for clustering. Thus, we choose to anneal the std lower bound every 100 epochs at the rate of 0.5. These configurations empirically works well on all our five datasets.

IV. Related Work

Recently, deep learning methods are widely used to learn better latent representations from the training data for clustering task. To further improve the clustering accuracy, recent methods [3], [20], [29] often jointly refine the latent features and clustering assignments, through carefully designed combination of loss functions or additional regularization terms. These methods achieve higher clustering performance than directly apply conventional clustering methods on raw data or features, especially for the complex datasets.

Apart from the above methods, several generative model based clustering methods [6]–[8], [21] have been proposed. These methods jointly train the continuous latent variables and the clustering assignments through a deep generative model, where the models are able to achieve clustering in the latent space, as well as generate meaningful samples for each class. Some of them are based on GANs [5], *e.g.*, GM-GAN [6] assumed a mixture of Gaussians prior on the input noise of GAN, while training a K-way multi-class classifier with the samples generated from GAN model to do clustering.

Other methods are based on VAEs [4], which are the most relevant works to our approach NVISA in this paper. LTVAE [21] assumed the dependency among latent variables (*i.e.*, $\mathbf{z}$s and $\mathbf{y}$s) follow a tree structure model. It used a latent tree model to perform structure learning to learn the superstructure among latent variables, and iteratively improved the structure while learning the representations of data in a greedy manner. GMVAE [7], VaDE [8] and our method NVISA can be regarded as different variants of Gaussian Mixture VAE, but have significant differences. Both GMVAE and VaDE applied mean-field approximation and trained their model with SGVB [4], [11] and *reparameterization* trick. Specifically, GMVAE used an additional noise input variable w to derive the GMM prior, as well as an information theoretic regularization term to help training. VaDE made an additional approximation, *i.e.*, assume $q(y|x) = p(y|z)$ in order to train its model and derive the cluster assignments. However, in NVISA, we point out that directly using $q_\phi(\mathbf{z}|y, \mathbf{x})$ instead of using the mean-field approximation can benefit the clustering task, and carefully designed the corresponding model structure. In order to deal with the non-reparameterized Categorical latent variable y, NVISA uses VIMCO as the variational inference method. Moreover, NVISA proposes the *std annealing* trick to help the model learn better-separated embeddings, which benefits the unsupervised clustering task. The $q_\phi(y|\mathbf{x})$ term is directly trained and derived by neural networks in NVISA, and then the cluster assignment of each input $\mathbf{x}$ is obtained by directly finding the y with the maximum $q_\phi(y|\mathbf{x})$.

V. Conclusion

In this paper, we present the analysis and experiments on three variational inference methods to train our proposed NVISA model, targeted for the unsupervised clustering task. We showed that directly using $q_\phi(\mathbf{z}|y, \mathbf{x})$, instead of using the mean-field approximation, can benefit the unsupervised clustering with Gaussian Mixture VAE. Moreover, we proposed *std annealing* trick to stabilize the training process and help form better embeddings in the latent space with different variational inference methods.

NVISA overall outperforms all the baseline clustering algorithms on the five benchmark datasets. Specifically, we focus our discussions and the combinations of $q_\phi(\mathbf{z}|y, \mathbf{x})$, *non-reparameterized* variational inference method VIMCO and *std annealing* in the context of clustering problem, make NVISA significantly outperforms the previous Gaussian Mixture VAE-based clustering methods on all the five datasets, which highlights the benefit of our modifications to the Gaussian Mixture VAE. Additional experiments and analysis are taken to show the effectiveness of each technique used in NVISA.

One direction of future works is to theoretically understanding why *std annealing* can help learn separated embeddings in the latent space. Another direction is to incorporate more powerful priors in NVISA, rather than using simple Gaussian mixtures (*e.g.*, priors with complex structure like in LTVAE [21], or mixture of complex distributions derived by flow-based models [30]). This can enhance the representation capability of our model and may potentially further improve the clustering accuracy of NVISA on complex real-world datasets.

VI. Acknowledgements

This work has been supported by National Key R&D Program of China 2019YFB1802504, and Beijing National Research Center for Information Science and Technology.

References

[1] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[3] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International conference on machine learning*, 2016, pp. 478–487.

[4] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *2nd International Conference on Learning Representations (ICLR)*, 2014.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[6] M. Ben-Yosef and D. Weinshall, “Gaussian mixture generative adversarial networks for diverse datasets, and the unsupervised clustering of images,” *arXiv preprint arXiv:1808.10356*, 2018.

[7] N. Dilokthanakul, P. A. Mediano *et al.*, “Deep unsupervised clustering with gaussian mixture variational autoencoders,” *arXiv preprint arXiv:1611.02648*, 2016.

[8] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, “Variational deep embedding: an unsupervised and generative approach to clustering,” in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 1965–1972.

[9] A. Mnih and D. Rezende, “Variational inference for monte carlo objectives,” in *ICML*, 2016, pp. 2188–2196.

[10] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” *arXiv preprint arXiv:1611.01144*, 2016.

[11] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in *Proceedings of the 31st ICML*. JMLR. org, 2014, pp. II–1278.

[12] A. Mnih and K. Gregor, “Neural variational inference and learning in belief networks,” in *ICML*, 2014, pp. 1791–1799.

[13] Y. Burda, R. Grosse, and R. Salakhutdinov, “Importance weighted autoencoders,” *arXiv preprint arXiv:1509.00519*, 2015.

[14] C. Robert and G. Casella, *Monte Carlo statistical methods*. Springer Science & Business Media, 1999.

[15] M. D. Hoffman and M. J. Johnson, “Elbo surgery: yet another way to carve up the variational evidence lower bound,” in *Workshop in Advances in Approximate Bayesian Inference, NIPS*, 2016.

[16] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[18] A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the 14th international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[19] A. Torralba, R. Fergus, and W. T. Freeman, “80 million tiny images: A large data set for nonparametric object and scene recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[20] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, “Learning discrete representations via information maximizing self-augmented training,” in *ICML*, 2017, pp. 1558–1567.

[21] X. Li, Z. Chen, L. K. M. Poon, and N. L. Zhang, “Learning latent superstructures in variational autoencoders for deep multidimensional clustering,” in *7th ICLR*, 2019.

[22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[24] H. W. Kuhn, “The hungarian method for the assignment problem,” *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[25] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *The International Conference on Learning Representations (ICLR)*, 2015.

[26] L. Theis, A. van den Oord, and M. Bethge, “A note on the evaluation of generative models,” in *International Conference on Learning Representations (ICLR 2016)*, 2016, pp. 1–10.

[27] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, “beta-vae: Learning basic visual concepts with a constrained variational framework,” in *International Conference on Learning Representations*, vol. 3, 2017.

[28] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[29] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, “Towards k-means-friendly spaces: Simultaneous deep learning and clustering,” in *Proceedings of the 34th ICML*. JMLR. org, 2017, pp. 3861–3870.

[30] D. P. Kingma and P. Dhariwal, “Glow: Generative flow with invertible 1x1 convolutions,” in *Advances in Neural Information Processing Systems*, 2018, pp. 10 236–10 245.