Vector Quantization-Based Regularization for Autoencoders

Hanwei Wu[1, 2] **and Markus Flierl**[1]
{hanwei, flierl}@kth.se
[1]KTH Royal Institute of Technology, Stockholm, Sweden
[2]Research Institutes of Sweden
Stockholm, Sweden

Abstract

Autoencoders and their variations provide unsupervised models for learning low-dimensional representations for downstream tasks. Without proper regularization, autoencoder models are susceptible to the overfitting problem and the so-called posterior collapse phenomenon. In this paper, we introduce a quantization-based regularizer in the bottleneck stage of autoencoder models to learn meaningful latent representations. We combine both perspectives of Vector Quantized-Variational AutoEncoders (VQ-VAE) and classical denoising regularization methods of neural networks. We interpret quantizers as regularizers that constrain latent representations while fostering a similarity-preserving mapping at the encoder. Before quantization, we impose noise on the latent codes and use a Bayesian estimator to optimize the quantizer-based representation. The introduced bottleneck Bayesian estimator outputs the posterior mean of the centroids to the decoder, and thus, is performing soft quantization of the noisy latent codes. We show that our proposed regularization method results in improved latent representations for both supervised learning and clustering downstream tasks when compared to autoencoders using other bottleneck structures.

Introduction

An important application of autoencoders and their variations is the use of learned latent representations for downstream tasks. In general, learning meaningful representations from data is difficult since the quality of learned representations is usually not measured by the objective function of the model. The reconstruction error criterion of vanilla autoencoders may result in the model memorizing the data. The variational autoencoders (VAE) (**?**) improves the representation learning by enforcing stochastic bottleneck representations using the reparameterization trick. The VAE models further impose constraints on the latents by minimizing a KullbackLeibler (KL) divergence between the prior and the approximate posterior of the latent distribution. However, the evidence lower bound (ELBO) training of VAE does not necessarily result in meaningful latent representations as the optimization cannot control the trade-off between the reconstruction error and the information transfer from the data to the latent representation (**?**). On the other hand, the VAE training is also susceptible to the so-called posterior collapse phenomenon where a structured latent representation is mostly ignored and the encoder maps the input data to the latent representation in a "random" fashion (**?**). This is not favorable for downstream applications since the latent representation loses its similarity relation to the input data.

Various regularization methods have been proposed to improve the latent representation learning for the VAE models. (**?**)(**?**) enforce stronger KL regularization on the latent representation in the bottleneck stage to constrain the transfer of information from data to the learned representation. Denoising methods (**?**)(**?**)(**?**) encourage the model to learn robust representations by artificially perturbing the training data. On the other hand, conventional regularization methods may not solve the posterior collapse problem. (**?**) empirically shows that posterior collapse is caused by the original marginal log-likelihood objective of the model rather than the evidence lower bound (ELBO). As a result, modifying the objective function ELBO of VAE as (**?**)(**?**) may have limited effects on preventing the posterior collapse. One potential solution is the vector-quantized variational autoencoder (VQ-VAE) (**?**) model. Instead of regularizing the latent distribution, VQ-VAE provides a latent representation based on a finite number of centroids. Hence, the capability of the latent representation can be controlled by the number of used centroids which guarantees that a certain amount of information is preserved in the latent space.

In this paper, we combine the perspectives of VQ-VAE and noise-based approaches. We inject noise into the latent codes before the quantization in the bottleneck stage. We assume that the noisy observations are generated by a Gaussian mixture model where the means of the components is represented by the centroids of the quantizer. To determine the input of the autoencoder decoder, we use a Bayesian estimator and obtain the posterior mean of the centroids. In other words, we perform a soft quantization of the latent codes in contrast to a hard assignment as used in vanilla VQ-VAE. Hence, we refer to our framework as soft VQ-VAE. Since our focus is on using autoencoders to extract meaningful low-dimensional representations for other downstream tasks, we demonstrate that the latent representation extracted from our soft VQ-

VAE models are effective in subsequent classification and clustering tasks in the experiments.

Bottleneck Vector Quantizer

Vector Quantization in Autoencoders

Here we first give a general description of the autoencoder model with vector quantized bottleneck based on the VQ-VAE formulation. The notational conventions in this work are as follows: Boldface symbols such as $\mathbf{x}$ are used to denote random variables. Nonboldface symbols x are used to denote sample values of those random variables.

The bottleneck quantized autoencoder models consist of an encoder, a decoder, and a bottleneck quantizer. The encoder learns a deterministic mapping and outputs the *latent code* $\mathbf{z_e} = g_{\text{enc}}(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X} = \mathbb{R}^D$ denotes the input datapoint and $\mathbf{z_e} \in \mathbb{R}^d$. The latent code $\mathbf{z_e}$ can be seen as an efficient representation of the input $\mathbf{x}$, such that $d \ll D$. The latent code $\mathbf{z_e}$ is then fed into the bottleneck quantizer $Q(\cdot)$. The quantizer partitions the latent space into K clusters characterized by the codebook $\mathcal{M} = \{\mu^{(1)}, \cdots, \mu^{(K)}\}$. The latent code $\mathbf{z_e}$ is quantized to one of the K codewords by the nearest neighbor search

$$\mathbf{z_q} = Q(\mathbf{z_e}) = \mu^{(\mathbf{c})}, \text{ where } \mathbf{c} = \arg\min_k \|\mathbf{z_e} - \mu^{(k)}\|_2. \quad (1)$$

The output $\mathbf{z_q}$ of the quantizer is passed as input to the decoder. The decoder then reconstructs the input datapoint $\mathbf{x}$.

Bottleneck Vector Quantizer as a Latent Parameter Estimator

In this section, we show that the embedded quantizer can be interpreted as a parameter estimator for the latent distribution with a discrete parameter space. In a vanilla VAE perspective, the encoder outputs the parameters of its latent distribution. The input of the VAE decoder is sampled from the latent distribution that is parameterized by the output of the VAE encoder. For bottleneck quantized autoencoders, the embedded quantizer creates a discrete parameter space for the model posterior. The nearest neighbor quantization effectively makes the autoencoder a generative model of Gaussian mixtures with a finite number of components, where the components are characterized by the codewords of the quantizer (**?**). In contrast, the vanilla VAE is equivalent to a mixture of an infinite number of Gaussians as the latent parameter space is continuous. Furthermore, the variational inference model of the bottleneck quantized autoencoder can be expressed as

$$q(z|x) = \sum_{k=1}^{K} q\left(z|z_q = \mu^{(k)}\right) q\left(z_q = \mu^{(k)}|x\right) \quad (2)$$

$$= q\left(z|z_q = Q(g_{\text{enc}}(x))\right) \delta\left(z_q = Q(g_{\text{enc}}(x))\right), \quad (3)$$

where z is the latent variable and $\delta(\cdot)$ is the indicator function.

As a result, the decoder input z_q can be seen as the estimated parameters of the latent distribution with the discrete parameter space that is characterized by the codebook of the quantizer. That is, the decoder of the bottleneck quantized autoencoders takes the estimated parameter of the latent distribution and recover the parameters of the data generating distribution of the observed variables $\mathbf{x}$. No sampling of $\mathbf{z}$ from the latent distribution is needed during the training of vector-quantized autoencoder models.

Vector Quantizer as a Regularizer

We showcase that the added quantizer between encoder and decoder acts also as a regularizer on the latent codes that fosters similarity-preserving mappings at the encoder for Gaussian observation models. We use visual examples to show that the embedded bottleneck quantizer can enforce the encoder output to share a constrained coding space such that learned latent representations preserve the similarity relations of the data space. We argue that this is one of the reasons that bottleneck quantized autoencoders can learn meaningful representations.

Assume that we have a decoder with infinite capacity. That is, the decoder is so expressive that it can produce a precise reconstruction of the input of the model without any constraints on the latent codes. As a result, the encoder can map the input to the latent codes in an arbitrary fashion while keeping a low reconstruction error (See Fig. 1a).

With the quantizer inserted between the encoder and decoder, the encoder can only map the input to a finite number of representations in the latent space. For example, in Fig. 1b, we insert a codebook with two codewords. If we keep the encoder mapping the same as Fig. 1a, then, both blue and purple nodes in the latent space will be represented by the blue node in the discrete latent space due to the nearest neighbor search. In this case, the optimal reconstruction of the blue and purple nodes at the input will be the green node at the output. This is obviously not the optimal encoder mapping with respect to the reconstruction error. Instead, the more efficient mapping of the encoder is to map similar data points to neighboring points in the latent space (See Fig. 1c).

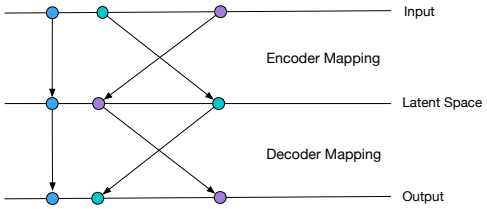
However, we can also observe that the bottleneck quantized autoencoders inevitably hurts the reconstruction due to the limited choices of discrete latent representations. That is, the number of possible reconstructions produced by a decoder is limited by the size of the codebook. This is insufficient for many datasets who have a large number of classes. In our proposed soft VQ-VAE, it increases the expressiveness of the latent representations by using a Gaussian mixture model and the decoder input is a convex combination of the codewords.

Soft VQ-VAE

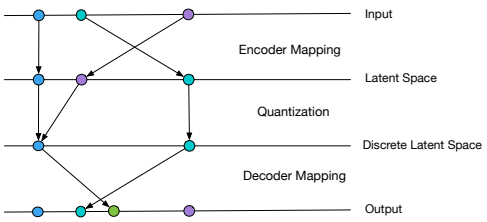
Noisy Latent Codes

Injecting noise on the input training data is a common technique for learning robust representations (**?**). In our paper, we extend this practice by adding noise to the latent space such that the models are exposed to new data samples. We note that this practice is also applied in (**?**) and (**?**) to improve the generalization ability of their models.

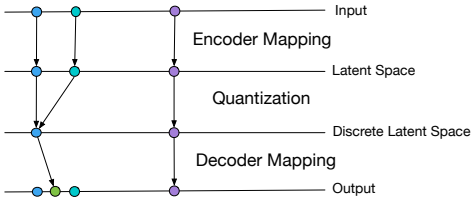
We propose to add white noise $\boldsymbol{\epsilon}$ with zero mean and finite variance on the encoder output $\mathbf{z'_e} = \mathbf{z_e} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \in \mathbb{R}^d$. We assume that the added noise variance σ_ϵ is unknown to the model. Instead, we view the noisy latent code is generated



(a) Vanilla autoencoders



(b) Autoencoders with quantized bottleneck



(c) The quantizer enforces a similarity-preserving mapping at the encoder

Figure 1: The quantizer behaves as a regularizer that encourages a similarity-preserving mapping at the encoder.

from a mixture model with K components

$$p(z'_e) = \sum_{k=1}^{K} p\left(z_q = \mu^{(k)}\right) p\left(z'_e | z_q = \mu^{(k)}\right), \quad (4)$$

where $z_q \in \mathcal{M}$.

We let the conditional probability function of $\mathbf{z'_e}$ given one of the codewords $\mu^{(k)}$ to be a multivariate Gaussian distribution $\mathcal{N}\left(\mu^{(k)}, I^{(k)}\right)$

$$p\left(z'_e | \mu^{(k)}\right) = \frac{\exp\left(\left(-\frac{1}{2}(z'_e - \mu^{(k)})\right)^T I^{(k)-1}\left(z'_e - \mu^{(k)}\right)\right)}{\sqrt{(2\pi)^d |I^{(k)}|}}, \quad (5)$$

where the k-th codeword $\mu^{(k)}$ is regarded as the mean of the Gaussian distribution of the k-th component, $I^{(k)} = \sigma_k^2 I$ and σ_k is the standard deviation of the k-th component.

Bayesian Estimator

We add a Bayesian estimator after the noisy latent codes in the bottleneck stage of the autoencoder. The aim is to estimate the parameters of the latent distribution from noisy observations. The Bayesian estimator is optimal with respect to the mean square error (MSE) criterion and is defined as the mean of the posterior distribution,

$$\hat{z}_q = \mathbb{E}[\mathbf{z_q} | z'_e] = \sum_{k=1}^{K} \mu^{(k)} p\left(\mu^{(k)} | z'_e\right). \quad (6)$$

Using Bayes' rule, we express the conditional probability $p\left(\mu^{(k)} | z'_e\right)$ as

$$p\left(\mu^{(k)} | z'_e\right) = \frac{p\left(\mu^{(k)}\right) p\left(z'_e | \mu^{(k)}\right)}{p(z'_e)}, \quad (7)$$

where we assume an uninformative prior for the codewords $p\left(\mu^{(k)}\right) = \frac{1}{K}$ as there is no preference for single codeword. The conditional probability $p\left(z'_e | \mu^{(k)}\right)$ is given in (5) and the marginal distribution of the noisy observation is given by marginalizing out the finite codebook in (4).

Compared to the hard assignment of the VQ-VAE, we can see that we are equivalently performing a soft quantization as the noisy latent code is assigned to a codeword with probability $p\left(\mu^{(k)} | z'_e\right)$. The output of the estimator is a convex combination of all the codewords in the codebook. The weight of each codeword is determined similar to a radial basis function kernel where the value is inversely proportional to the L2 distance between $\mathbf{z'_e}$ and a codeword with component variance as the smoothing factor. Fig. 2 shows the described soft VQ-VAE.

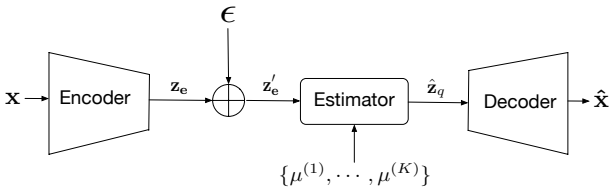


Figure 2: Description of the soft VQ-VAE.

Optimal Estimator

In this section, we show that our added Bayesian estimator is optimal with respect to the model evidence of the bottleneck quantized autoencoders with noisy latent codes. The maximum likelihood principle of generative models chooses the model parameters that maximize the likelihood of the training data (**?**). Similarly, we can decompose the marginal log-likelihood of the model distribution as the model ELBO plus the KL divergence between the variational distribution and the model posterior (**?**),

$$\log p(x) = \mathbb{E}_{q(z)}\left[\log\left(\frac{p(x,z)}{q(z)}\right)\right] + \mathrm{KL}(q(z) \| p(z|x)), \quad (8)$$

where $p(z|x)$ is the model posterior and $q(z)$ is the variational latent distribution that regularizes the model posterior. The maximization of the model ELBO can be seen as searching for the optimal latent distribution q within a variational family $\mathcal{Q}$ that approximates the true model posterior $p(z|x)$. Given a uniform distribution $\hat{p}(x)$ over the training dataset, we can

obtain the optimal estimation of the latent distribution:

$$q^* = \mathbb{E}_{\hat{p}(x)} \arg\max_{q \in \mathcal{Q}} \left[\mathbb{E}_{q(z)} \left[\log \left(\frac{p(x,z)}{q(z)} \right) \right] \right] \tag{9}$$

$$= \mathbb{E}_{\hat{p}(x)}[\log p(x)] - \mathbb{E}_{\hat{p}(x)} \arg\min_{q \in \mathcal{Q}} [\mathrm{KL}(q(z) \| p(z|x))] \tag{10}$$

$$= \mathbb{E}_{\hat{p}(x)} \arg\min_{q \in \mathcal{Q}} [\mathrm{KL}(q(z) \| p(z|x))], \tag{11}$$

Since the first term of (10) is irrelevant with respect to the approximated latent distribution, the maximization of the model ELBO becomes equivalent to finding the latent distribution that minimizes the KL divergence to the model posterior distribution in (11).

For bottleneck quantized autoencoders, the embedded quantizer enforces the model posterior $p(z|x)$ to be the unimodal distribution centered on one of the codewords $\mu \in \mathcal{M}$. In our noisy model, we perturb the encoder output $\mathbf{z_e}$ by random noise. We assume that the noise variance is unknown to the model and the parameter cannot be determined by performing a nearest neighbor search on the noisy bottleneck representation $\mathbf{z}'_\mathbf{e}$. Instead, the introduced Bayeisan estimator (6) outputs a convex combination of the codewords. In the following Theorem, we show that our proposed Bayesian estimator outputs the parameters of the optimal latent distribution for the quantized bottleneck autoencoder models under the condition that the latent distribution belongs to the Gaussian family.

Theorem 1. *Let $\mathcal{Q}$ be the set of Gaussian distributions with associated parameter space Ω. Based on the described noisy model, for one datapoint, the estimator $f\colon \mathcal{X} \to \Omega$ that outputs the parameters of the optimal $q^* \in \mathcal{Q}$ is given by*

$$\hat{z}_q = f(x) = \sum_{k=1}^{K} \mu^{(k)} p\left(\mu^{(k)} | z'_e\right). \tag{12}$$

Proof. For the noisy setting, the expectation of the KL divergence between the model posterior $p(z|x)$ and the approximated q is taken with respect to the empirical training distribution $\hat{p}(x)$ and the noise distribution $\hat{p}(\epsilon)$

$$\mathbb{E}_{\hat{p}(x)} \mathbb{E}_{\hat{p}(\epsilon)} [\mathrm{KL}(q(z) \| p(z|x))]. \tag{13}$$

Since the encoder does not have activation functions in the output layer, we assume that the encoder neural network is a deterministic injective function over the empirical training set such that $\hat{p}(x) = \hat{p}(z_e)$. Also, the injected noise is independent of z_e, we can express the probability distribution of the training data and noise as the following chain of equalities:

$$\hat{p}(x)\hat{p}(\epsilon) = \hat{p}(z_e)\hat{p}(\epsilon) = \hat{p}(z_e, \epsilon) = \hat{p}(z_e, z_e + \epsilon) = \hat{p}(z_e, z'_e) \tag{14}$$

The joint probability $\hat{p}(z_e, z'_e)$ can be further decomposed as

$$\hat{p}(z_e, z'_e) = \hat{p}(z_e)\hat{p}(z'_e|z_e) \tag{15}$$

$$= \hat{p}(z_e) \sum_{k=1}^{K} \hat{p}\left(z_q = \mu^{(k)} | z_e\right) \hat{p}\left(z'_e | z_q = \mu^{(k)}\right) \tag{16}$$

$$= \hat{p}(z_e) \frac{1}{K} \sum_{k=1}^{K} \hat{p}\left(z'_e | \mu^{(k)}\right). \tag{17}$$

where (17) follows from that z_e is considered as unobservable in the model (see Fig. 3), and thus provides no information about $\mu^{(k)}$ such that the conditional probability of $\mu^{(k)}$ given z_e is equal to the prior of the codewords $\frac{1}{K}$.

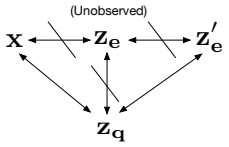


Figure 3: The relation of variables in the soft VQ-VAE model. The symbol $\nleftrightarrow$ is used to indicate that we cannot directly use paths that connected to the unobserved $\mathbf{z_e}$ for probabilistic inference.

Combining the model posterior of bottleneck quantized autoencoders and the above derivations, we can reexpress (13) as

$$\mathbb{E}_{\hat{p}(z_e, z'_e)} [\mathrm{KL}(q(z) \| p(z|z_q)] \tag{18}$$

$$= \mathbb{E}_{\hat{p}(z_e)} \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\hat{p}\left(z'_e | \mu^{(k)}\right)} \left[\mathrm{KL}(q(z) \| p\left(z | \mu^{(k)}\right) \right], \tag{19}$$

where the true model posterior has $p(z|x) = p(z|z_q)$.

Therefore, for each datapoint x, the optimization problem with respect to the latent distribution q (11) for the noisy setting becomes

$$\min_{q \in \mathcal{Q}} \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\hat{p}\left(z'_e | \mu^{(k)}\right)} \left[\mathrm{KL}\left(q(z) \| p\left(z | \mu^{(k)}\right)\right) \right] \tag{20}$$

$$= \min_{q \in \mathcal{Q}} \frac{1}{K} \sum_{k=1}^{K} \hat{p}\left(z'_e | \mu^{(k)}\right) \mathrm{KL}\left(q(z) \| p\left(z | \mu^{(k)}\right)\right). \tag{21}$$

Note that the KL divergence between two exponential family distributions can be represented by the Bregman divergence $d_A(\cdot)$ between the corresponding natural parameters η' and η as

$$\mathrm{KL}(p_{\eta'} \| p_\eta) = d_A(\eta, \eta') \tag{22}$$

$$= -A(\eta') + A(\eta) - \nabla A(\eta)^T (\eta' - \eta), \tag{23}$$

where $A(\cdot)$ is the log-partition function for the exponential family distribution. Furthermore, it has been shown that the minimizer of the expected Bregman divergence from a random vector is its mean vector (**?**). Therefore, we formulate

(20) as a convex combination of the KL divergence

$$\arg\min_{q\in\mathcal{Q}} \sum_{k=1}^{K} \omega_k \mathrm{KL}\left(q(z)\|p\left(z|\mu^{(k)}\right)\right) \tag{24}$$

$$= \arg\min_{\eta} \sum_{k=1}^{K} \omega_k d_A(\eta^{(k)}, \eta), \tag{25}$$

where $\omega_k = \frac{1}{VK}\hat{p}\left(z'_e|\mu^{(k)}\right)$. $V = \sum_{k=1}^{K}\hat{p}\left(z'_e|\mu^{(k)}\right)$ is the introduced normalization constant and the optimal solution of (20) is not affected. In addition, due to the normalization, ω_k becomes $p\left(\mu^{(k)}|z'_e\right)$. Then, the minimizer of (24) is given by the mean of $\eta^{(k)}$

$$\eta = \sum_{k=1}^{K} p\left(\mu^{(k)}|z'_e\right)\eta^{(k)}. \tag{26}$$

The natural parameters for the multivariate Gaussian distribution with known covariance matrix is $\Sigma^{-1}\mu$. Since the $p(z|x)$ is the model posterior of the noiseless bottleneck quantized autoencoders, the covariance matrix is assumed to be the identity matrix for all components $\Sigma = I$. Therefore, we can recover the Bayesian estimator (12) by substituting $\eta^{(k)}$ with $\mu^{(k)}$ in (26), and the proof is complete. □

Related Work

For extended work on VQ-VAE, (**?**) uses the Expectation Maximization algorithm in the bottleneck stage to train the VQ-VAE and to achieve improved image generation results. However, the stability of the proposed algorithm may require to collect a large number of samples in the latent space. (**?**) gives a probabilistic interpretation of the VQ-VAE and recovers its objective function using the variational inference principle combined with implicit assumptions made by the vanilla VQ-VAE model.

Several works have studied the end-to-end discrete representation learning model with different incorporated structures in the bottleneck stages. (**?**) and (**?**) introduce scalar quantization in the latent space and optimize jointly the entire model for rate-distortion performance over a database of training images. (**?**) proposes a compression model by performing vector quantization on the network activations. The model uses a continuous relaxation of vector quantization which is annealed over time to obtain a hard clustering. In (**?**), the softmax function is used to give a soft assignment to the codewords where a single smoothing factor is used as an annealing factor. In our model, we learn different smoothing factors for each component. (**?**) introduces a continuous relaxation training of discrete latent-variable models which can flexibly capture both continuous and discrete aspects of natural data.

Various techniques for regularizing the autoencoders have been proposed recently. (**?**) proposes an adversarial regularizer which encourages interpolation in the outputs and also improves the learned representation. (**?**) interprets the VAEs as a amortized inference algorithm and proposed a procedure to constrain the expressiveness of the encoder. In addition, there is a increasing popularity of using information-theoretic principles to improve autoencoders. (**?**)(**?**) use the information bottleneck principle (**?**) to recover the objective of β-VAE and show that the KL divergence term in ELBO is an upper bound on the information rate between input and prior. (**?**) is also inspired by the information bottleneck principle and introduces the information dropout method to penalize the transfer of information from data to the latents. (**?**) proposes to use encoder-decoder structures and inject noises to the bottleneck stage to simulate binary symmetric channels (BSC). By jointly optimizing the encoding and decoding processes, the authors show that the trained model not only can produce codes that have better performance for the joint source-channel coding problem but also that the noisy latents facilitate robust representation learning.

We also note that the practice of using a convex combination of codewords is similar to the attention mechanism (**?**). The attention mechanism is introduced to solve the gradient vanishing problem that models fail to learn the long-term dependencies of time series data. It can be viewed as a feed-forward layer that takes the hidden state of the at each time step as input and outputs the so-called context vector as the representation which is a weighted combination of the input hidden state vectors.

Experimental Results

Model Implementation

We test our proposed model on datasets MNIST, SVHN and CIFAR-10. All the tested autoencoder models share the same encoder-decoder setting. For the models tested on the SVHN and CIFAR-10, we use convolutional neural networks (CNN) to construct the encoder and decoder. For the MNIST, we use multilayer perceptron (MLP) networks to construct encoder and decoder. All decoders follow a structure that is symmetric to the encoder.

The differences among the compared models are only in the bottleneck operation. The bottleneck operation takes the encoder output as its input, and its output is fed into the decoder. For VAE and information dropout models, the bottleneck input is two separate encoder output layers of d units respectively. One layer learns the mean of the Gaussian distribution and the other layer learns the log variance. The reparameterization trick or the information dropout technique is applied to generate samples for the latent distribution. For the VQ-VAE, the bottleneck performs a nearest neighbor search on the encoder output. Then, the quantized codeword is fed into the decoder. For the soft VQ-VAE, the bottleneck input is also two separate encoder output layers. One layer of size d outputs the noiseless vector z_e. Another layer with size K outputs the log variance of components. The noise injection is performed only on z_e and the estimator uses the noisy samples and the variances of components for estimation. The baseline autoencoder directly feeds the encoder output to the decoder.

The soft VQ-VAE models are trained in a similar fashion as VQ-VAE. Specifically, the loss function for the soft VQ-VAE

mode is

$$L = -\log p(x|\hat{z}_q) + \|\text{sg}\,(z'_e) - \hat{z}_q\|_2^2 + \beta\|z'_e - \text{sg}(\hat{z}_q)\|_2^2. \quad (27)$$

where $\text{sg}(\cdot)$ denotes the stop gradient operator and β is a hyperparameter to encourage the encoder to commit to a codeword. The stop gradient operator is used to solve the vanishing gradient problem of discrete variables by separating the gradient update of encoder-decoder and the codebook. The $\text{sg}(\cdot)$ outputs its input when it is in the forward pass, and outputs zero when computing gradients in the training process. Specifically, the decoder input is expressed as $\hat{z}_q = z_e + \text{sg}(\hat{z}_q - z_e)$ such that the gradients are copied from the decoder input to the encoder output.

Training Setup

For the models tested on the CIFAR-10 and SVHN datasets, the encoder consists of 4 convolutional layers with stride 2 and filter size 3×3. The number of channels is doubled for each encoder layer. The number of channels of the first layer is set to be 64. The decoder follows a symmetric structure of the encoder. For MINST dataset, we use multilayer perceptron networks (MLP) to construct the autoencoder. The dimensions of dense layers of the encoder and decoder are D-500-500-2000-d and d-2000-500-500-D respectively, where d is the dimension of the learned latents and D is the dimension of the input datapoints. All the layers use rectified linear units (ReLU) as activation functions.

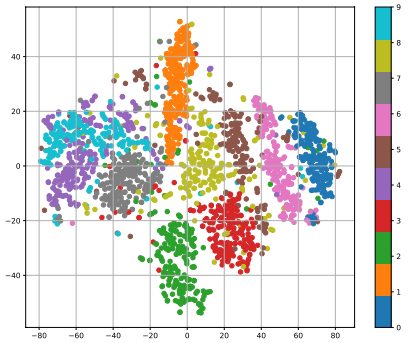
We use the Glorot uniform initializer (**?**) for the weights of encoder-decoder networks. The codebook is initialized by the uniform unit scaling. All models are trained using Adam optimizer (**?**) with learning rate 3e-4 and evaluate the performance after 40000 iterations with batch size 32. Early stopping at 10000 iterations is applied by soft VQ-VAE on SVHN and CIFAR-10 datasets.

Visualization of Latent Representation

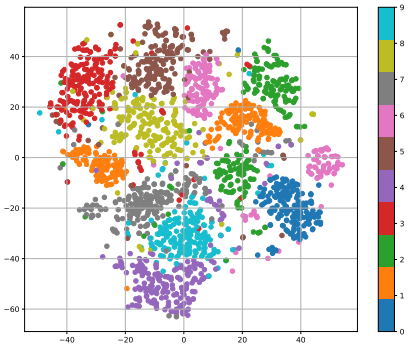
In this section, we use t-SNE (**?**) to visualize the latent representations that have been learned by different autoencoder models, and examine their similarity-preserving mapping ability. First, we train autoencoders with a 60-dimensional bottleneck on the MNIST dataset. After the training, we feed the test data into the trained encoder to obtain the latent representation of the input data. The 60-dimensional latent representations are projected into the two-dimensional space using the t-SNE technique. In Fig. 4, we plot the two-dimensional projection of the bottleneck representation z_e of the trained models with different bottleneck structures. All autoencoder models are trained to have similar reconstruction quality. It is shown that the latent representation of the soft VQ-VAE preserves the similarity relations of the input data better than the other models.

Representation Learning Tasks

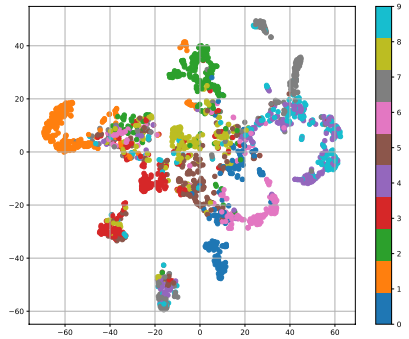
We test our learned latent representation z_e on K-means clustering and single-layer classification tasks as (**?**). The justification of these two tests is that if the learned latents can recover the hidden structure of the raw data, they should become more amiable to the simple classification and clustering



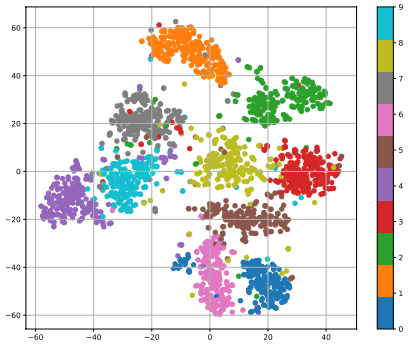
(a) Autoencoder: $\mathbf{z_e}$.



(b) VAE: $\mathbf{z_e}$.



(c) VQ-VAE: $\mathbf{z_e}$.



(d) soft VQ-VAE: $\mathbf{z_e}$.

Figure 4: Two-dimensional learned representations of MNIST. Each color indicates one digit class.

tasks. We first train models using the training set. Then we use the trained model to project the test set on their latent representations and use them for downstreaming tasks.

For the K-means clustering, we use 100 random initializations and select the best result. The clustering accuracy is determined by Hungarian algorithm (**?**), which is a one-to-one optimal linear assignment (LS) matching algorithm between the predicted labels and the true labels. We also test the clustering performance using the normalized mutual information (NMI) metric for the MNIST dataset (**?**) (**?**). The NMI is defined as NMI $(\mathbf{y}, \hat{\mathbf{y}}) = \frac{2I(\mathbf{y},\hat{\mathbf{y}})}{H(\mathbf{y})+H(\hat{\mathbf{y}})}$, where $\mathbf{y}$ and $\hat{\mathbf{y}}$ denote the true labels and the predicted labels, respectively. $I(\mathbf{y}, \hat{\mathbf{y}})$ is the mutual information between the predicted labels and the true label. $H(\cdot)$ is the entropy. For the classification tasks, we use a fully connected layer with a softmax function on the output as our classifier. The single-layer classifier is trained on the latent representation of the training set and is independent of the autoencoders' training.

Table 1: Accuracy of downstream tasks of MNIST.

	MNIST, $d = 64$		
Model	Clustering	Clustering (NMI)	Classification
Raw Data	55.17	0.5008	92.44
Baseline Autoencoder	52.61	0.5301	91.91
VAE	56.44	0.5600	89.10
β-VAE (β = 20)	73.81	0.5760	91.10
Information dropout	58.52	0.4979	91.11
VQ-VAE (K = 128)	51.48	0.3541	81.62
Soft VQ-VAE (K=128)	**77.64**	**0.7188**	**93.54**

Table 2: Accuracy of downstream tasks of SVHN and CIFAR-10.

	SVHN, $d = 256$		CIFAR-10, $d = 256$	
Model	Clustering	Classification	Clustering	Classification
Baseline Autoencoder	11.96	25.95	21.73	40.92
VAE	13.58	26.42	**24.12**	38.83
β-VAE (β = 100)	14.54	49.62	22.80	36.91
Information dropout	12.75	24.46	21.96	39.89
VQ-VAE (K = 512)	12.96	31.57	20.30	33.51
Soft VQ-VAE (K = 32)	**17.68**	**50.48**	23.83	**44.54**

We test 64-dimensional latents for the MNIST and 256 for SVHN and CIFAR-10. We compare different models where only the bottleneck operation is different. The results are shown in Table 1 and 2. We report the means of accuracy results. The variances of all the results are within 1 percent.

For MNIST, soft VQ-VAE achieves the best accuracy for both clustering and classification tasks. Specially, it improves 25 percent clustering accuracy for linear assignment metric and 36 percent clustering accuracy for NMI metric when compared to the baseline autoencoder model. The performance of vanilla VQ-VAE suffers from the small size of the codebook ($K = 128$). All models show difficulties for directly learning from CIFAR-10 and SVHN data as they just perform better than random results in the clustering tasks. Soft VQ-VAE has the best accuracy for classification and has the second best for clustering. One reason for the poor performance of colored images may be that autoencoder models may need the color information to be dominant in the latent representation such that they can have a good reconstruction. However, the color information may not generally useful for clustering and classification tasks.

An interesting observation from the experiments is that we need to use a smaller codebook ($K = 32$) for the soft VQ-VAE for CIFAR-10 and SVHN when compared to MNIST ($K = 128$). According to our experiments, setting a larger K for CIFAR-10 and SVHN will degrade the performance significantly. The potential reason is that we use CNN networks for CIFAR-10 and SVHN to have a better reconstruction of the colored images. Compared to the MLP networks used on MNIST, the CNN decoder is more powerful and can recover the encoder input from more cluttered latent representations. As a result, we need to reduce the codebook size to enforce a stronger regularization of the latents.

Beyond the discussed regularization effects, one intuition of the improved performance by soft VQ-VAE is that the embedded Bayesian estimator removes effects of adversarial input datapoint on the training. The adversarial points of the input data tend to reside in the boundary between classes. When training with ambiguous input data, the related codewords will receive a similar update. On the other hand, only one codeword receives a gradient update in the case of a hard assignment. This causes a problem. Ambiguous input is more likely estimated wrongly and the assigned codeword receives an incorrect update. Furthermore, the soft VQ-VAE model learns the variance for each Gaussian distribution. The learned variances control the smoothness of the latent distribution. The model will learn smoother distributions to reduce the effects of adversarial datapoints.

Conclusion

In this paper, we propose a regularizer that utilizes the quantization effects in the bottleneck. The quantization in the latent space can enforce a similarity-preserving mapping at the encoder. Our proposed soft VQ-VAE model combines aspects of VQ-VAE and denoising schemes as a way to control the information transfer. Potentially, this prevents the posterior collapse. We show the proposed estimator is optimal with respect to the bottleneck quantized autoencoder with noisy latent codes. Our model improves the performance of downstream tasks when compared to other autoencoder models with different bottleneck structures. Possible future directions include combining our proposed bottleneck regularizer with other advanced encoder-decoder structures (**?**)(**?**). The source code of the paper is publicly available.[1]

Acknowledgements

The authors sincerely thank Dr. Ather Gattami at RISE and Dr. Gustav Eje Henter for their valuable feedback on this paper. We are also grateful for the constructive comments of the anonymous reviewers. Sed aliquid animi labore nihil sit atque in odio, totam nisi at numquam eum, veritatis excepturi architecto libero minus necessitatibus quam sunt aliquam a, ipsam rerum unde voluptatem itaque, qui a laudantium.Laudantium aspernatur perspiciatis, maxime perspiciatis sequi id cum sint ipsam?Ratione vel debitis distinctio animi architecto esse possimus iure molestias

[1] https://github.com/AlbertOh90/Soft-VQ-VAE/

repellendus, quam sed beatae.Perferendis aliquam illum adipisci minus optio obcaecati doloribus numquam maxime, perferendis officia culpa perspiciatis esse sint