Quantization-Based Regularization for Autoencoders

Hanwei Wu  
Department of Intelligent System  
KTH Royal Institute of Technology  
Stockholm, Sweden  
hanwei@kth.se

Ather Gattami  
RISE AI  
Research Institutes of Sweden  
Stockholm, Sweden  
ather.gattami@ri.se

Markus Flierl  
Department of Intelligent System  
KTH Royal Institute of Technology  
Stockholm, Sweden  
mflierl@kth.se

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 schemes of neural networks. We interpret quantizers as regularizers that constrain latent representations while fostering a similarity mapping at the encoder. Before quantization, we impose noise on the latent variables 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 latent variables. 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.

1 Introduction

An important application of autoencoders and their variations is the use of learned latent representations for downstream tasks. However, learning meaningful representations from data is challenging since the quality of learned representations is usually not measured by the objective function that is optimized. The reconstruction error criterion of classical autoencoders may lead to latent representations that memorize the data while suffer from model overfitting. On the other hand, it has also been shown that maximum likelihood training of variational autoencoders (VAE) [17] does not result in good latent representations [3]. The reason is that the maximum likelihood criterion alone cannot control the trade-off between the reconstruction error and the information transfer from the data to the latent representation through the Kullback–Leibler (KL) divergence between inference and latent distribution.

Various methods have been proposed to add constraints on the latent representation in the bottleneck stage to regularize the transfer of information. The use of hyperparameters that enforce stronger KL regularization [14, 8] and noise-based methods [1, 21, 15] are examples of effective regularizers. However, such methods may fail due to the so-called posterior collapse phenomenon. This can

Preprint. Work in progress.
happen if the model is equipped with a powerful decoder (such as PixelCNN). 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. A solution to this problem is a vector-quantized variational autoencoder (VQ-VAE) \cite{van2017neural}. Instead of regularizing the encoder output 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.

In this paper, we regard the objective of learning meaningful representations as a problem of finding the adequate amount of information to transfer from the data to the latent representation. We combine the perspectives of VQ-VAE and noise-based approaches. We inject noise into the latent variables before the quantization in the bottleneck stage. We assume that the noisy observations are generated by a Gaussian mixture model where the mean of the components are 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 variables in contrast to a hard assignment as used in classic VQ-VAE. Hence, we refer to our framework as soft VQ-VAE.

2 Quantizers as Regularizers

2.1 VQ-VAE

Here, we give a brief description of the architecture of VQ-VAE. The model of VQ-VAE consists of an encoder, a decoder, and a bottleneck quantizer. The encoder learns a deterministic mapping and outputs the latent representation \( z_e(x) \in \mathbb{R}^d \), where \( x \in X = \mathbb{R}^D \) denotes the input datapoint. The learned latent \( z_e(x) \) can be seen as an efficient representation of the input \( x \), such that \( d \ll D \). The latent \( z_e(x) \) is then fed into the bottleneck quantizer. The quantizer partitions the latent space into \( K \) clusters characterized by the codebook \( \mathcal{M} = \{ \mu^{(1)}, \ldots, \mu^{(K)} \} \). The latent \( z_e(x) \) is quantized to one of the \( K \) codewords by the nearest neighbor search

\[
z_q(x) = \mu^{(z)}, \text{ where } z = \arg\min_k \| z_e(x) - \mu^{(k)} \|_2.
\]

The output \( z_q(x) \) of the quantizer is passed as input to the decoder. The decoder then reconstructs the input datapoint \( x \). For simplicity, we will use \( z_e \) and \( z_q \) to denote the values of \( z_e(x) \) and \( z_q(x) \) respectively.

To solve the problem of no gradient flowing through the discrete variables, the stop gradient operator is used to separate the gradient update such that the encoder-decoder and the codebook are trained independently. The loss function of the VQ-VAE is given by

\[
L = - \log g(x|z_q) + \| z_e - z_q \|_2^2 + \beta \| z_e - \text{sg}(z_q) \|_2^2,
\]

where \( \text{sg}(\cdot) \) denotes the stop gradient operator, \( g(\cdot) \) is the decoder network and \( \beta \) is a hyperparameter to encourage the encoder to commit to a codeword.

2.2 Enforced Similarity Mapping

We showcase that the added quantizer between the encoder and the decoder is used as a regularizer on the latent variables. We use visual examples to show that the embedded bottleneck quantizer can enforce the output of the encoder to share a constrained coding space such that learned latent features preserve the similarity relations of the data space. We argue that this is one of the reasons that VQ-VAE 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 variables. As a result, the encoder can map the input to the latents in an arbitrary fashion while keeping a low reconstruction error (See Fig. \[\text{1a}\]).

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. \[\text{1b}\] we insert a codebook with two codewords. If we keep the encoder mapping the same as Fig. \[\text{1a}\] 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).

![Diagram of classical autoencoders](a) Classical autoencoders

![Diagram of autoencoders with quantizer](b) Autoencoders with quantizer

![Diagram of the quantizer forces a similarity mapping](c) The quantizer forces a similarity mapping

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

However, we can also observe that the VQ-VAE inevitably hurts the reconstruction due to the limited choices of quantized 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 the latents by using a Gaussian mixture model and the input of the decoder is a convex combination of the codewords.

3 Improved Regularization by Noise Injection

3.1 Noisy Latents

The generalization ability of a model can also be improved if the size of the training dataset increases. However, in practice, we can only access a subset of the data of a domain for training. We assume that the expansion of the training dataset has similar effects of adding noise on the latent representation directly. The corrupted latents can thus enable the autoencoder to learn from outside of the training data \[19\]. Hence, in order to improve the generalization ability based on the limited training data, we propose to add white Gaussian noise on the latents \( z_e' = z_e + \epsilon \), where \( \epsilon \sim \mathcal{N}(0, \sigma^2 I) \), \( I \) is the identity matrix.

In the classical VAE perspective, the noisy \( z_e' \) is the latent variable with mean parameter \( z_e \) and covariance matrix \( \sigma^2 I \). However, an imposed quantizer on the latents will only output a quantized codeword to the decoder. Further, we assume that the added noise variance \( \sigma^2 \) is unknown to the model. Hence, instead, we can view the codewords as the parameters of the latent distribution that \( z_e' \) is sampled from. Then the decoder recovers the parameters of the distribution over the observed variables \( x \) based on the estimated parameters of the latents. This setting is similar to the test channel model that is used to prove rate-distortion theory \[10\]. Specifically, we assume that the noisy latent is generated from a mixture model with \( K \) components

\[
p(z_e' | Z_q) = \sum_{k=1}^{K} p(k)p \left( z_e' | \mu^{(k)} \right),
\]

(3)
where \( Z_q \in \mathcal{M} \). In contrast, the classic VAE is equivalent to a mixture of an infinite number of Gaussians where the \( Z_q \) is continuous.

We let the conditional probability function of noisy observation of \( z_e' \) given one of the codewords \( \mu^{(k)} \) to be a multivariate Gaussian distribution \( N(\mu^{(k)}, I_k) \)

\[
p(z_e' | \mu^{(k)}) = \frac{\exp\left(\left(-\frac{1}{2}(z_e' - \mu^{(k)})^T I_k^{-1} (z_e' - \mu^{(k)})\right)\right)}{\sqrt{(2\pi)^d|I_k|}},
\]

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 \( \sigma_k \) is the standard deviation of the \( k \)-th component.

### 3.2 Soft VQ-VAE

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

\[
\hat{z}_q = \mathbb{E}[Z_q | z_e'] = \sum_{k=1}^{K} \mu^{(k)} p(\mu^{(k)} | z_e').
\]

Using Bayes’ rule, we express the conditional probability \( p(\mu^{(k)} | z_e') \) as

\[
p(\mu^{(k)} | z_e') = \frac{p(\mu^{(k)}) p(z_e' | \mu^{(k)})}{p(z_e')},
\]

where we assume an uninformative prior for the codewords \( p(\mu^{(k)}) = \frac{1}{K} \). The conditional probability \( p(z_e' | \mu^{(k)}) \) is given in (4) and the marginal distribution of the noisy observation is given by

\[
p(z_e') = \sum_{k=1}^{K} p(\mu^{(k)}) p(z_e' | \mu^{(k)}) = \sum_{k=1}^{K} \frac{\exp\left(\left(-\frac{1}{2}(z_e' - \mu^{(k)})^T I_k^{-1} (z_e' - \mu^{(k)})\right)\right)}{K \sqrt{(2\pi)^d|I_k|}}.
\]

Compared to the hard assignment of the VQ-VAE, we can see that we are equivalently performing a soft quantization as the noisy latent is assigned to a codeword with probability \( p(\mu^{(k)} | z_e') \). The output of the estimator is a convex combination of all the codewords in the codebook. On the other hand, the VQ-VAE model can be seen as having a nearest neighbor estimator which only outputs a single codeword in the codebook. Fig. 2 shows the described soft VQ-VAE.

\[
\epsilon \sim N(0, \sigma^2
\]

\[
\text{Encoder} \quad \text{Estimator} \quad \text{Decoder}
\]

\[
\{\mu^{(1)}, \ldots, \mu^{(K)}\}
\]

Figure 2: Description of the soft VQ-VAE.

### 4 Theoretical Analysis

#### 4.1 Information Bottleneck Principle

For the VQ-VAE setting, the objective of the information bottleneck method can be formalized as

\[-I(Z; X) + \beta I(I; Z).\]
Here, with a slight abuse of notation, we denote $X$ as the input data representation, $Z$ as the index of the codewords and $I$ as the index of the input datapoints. The information bottleneck method maximizes $I(Z; X)$ to increase the prediction power of the learned $Z$ on $X$, while minimizes $I(I; Z)$ to avoid learned representations that memorize the data. The idea of limiting the information transfer from training data into the latents has been studied in recent theoretical works. Specifically, [27] shows that the generalization error can be upper bounded in terms of the mutual information between the input dataset and the output hypothesis of learning algorithms.

Using a similar technique as in [4], the mutual information between the input data and its latents of the VQ-VAE with codebook size $K$ can be upper bounded by

$$I(I; Z) \leq \text{KL}(p(Z|I = i|| m(Z))) = \sum_z p(z|x_i) \log \frac{p(z|x_i)}{m(z)}$$

$$= -\sum_z p(z|x_i) \log \frac{1}{K} + \sum_z p(z|x_i) \log p(z|x_i)$$

$$= \log K - H(p(Z|x_i)),$$

where the prior $m(Z)$ is set to be a simple uniform distribution. The nearest neighbor search of classical VQ-VAE creates a hard assignment for the input data. Hence, the codeword assignment is deterministic and the conditional entropy of the latents given the input datapoint is zero $H(Z|x_i) = 0$. The classical VQ-VAE yields a fixed KL divergence equal to $\log K$.

On the other hand, the soft VQ-VAE model enables a soft assignment for the input datapoint. This results in a positive conditional entropy $H(Z|x_i) \geq 0$ and decreases the upper bound of $I(I; Z)$. Therefore, compared to the classic VQ-VAE, the soft VQ-VAE further limits the information transfer from data to the latents.

### 4.2 Optimal Estimator

The maximum likelihood principle of generative models chooses the model parameters that maximize the likelihood of the training data [12]. The marginal log-likelihood of the model distribution can be decomposed as the evidence lower bound (ELBO) plus the KL divergence between the variational distribution and the model posterior [28],

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

where $q(z)$ is the variational latent distribution and $p(z|x)$ is the model posterior. The maximization of the 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:

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

$$= \arg \max_{q \in \mathcal{Q}} \mathbb{E}_{\hat{p}(x)} [\log p(x)] - \mathbb{E}_{\hat{p}(x)} [\text{KL}(q(z)||p(z|x))].$$

Since the first term of (14) is irrelevant with respect to the approximated latent distribution, the maximization of the ELBO becomes equivalent to finding the latent distribution that minimizes the KL divergence to the model posterior distribution in (15).

For classical VQ-VAE models, the inserted quantizer creates a discrete parameter space for the model posterior. The model posterior can be seen as a unimodal distribution centered on one of the codewords $\mu \in \mathcal{M}$. That is, for each datapoint $x$, we have

$$p(z|x) = p(z|z_q)p(z_q|x) = p(z|z_q)\delta(z_q = \mu),$$

where $\delta(\cdot)$ is the indicator function, $z_q$ is the parameter of latent distribution and obtained by the hard quantization. In our noisy model, we perturb the latents $z_q$ 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 latents $z'_e$. Instead, we introduce a Bayesian estimator (5) and the output is a convex combination of the codewords. The weight of each codeword is determined similar to a radial basis function kernel where the value is inversely proportional to the $L^2$ distance between $z'_e$ and a codeword with component variance as the smoothing factor. In the case that the latent distribution belongs to the Gaussian family, we show that our proposed Bayesian estimator outputs the parameters of the optimal latent distribution for the noisy VQ-VAE setting in Theorem 1.

**Theorem 1.** Let $Q$ be the set of Gaussian distributions with associated parameter space $\Omega$. Based on the described noisy VQ-VAE model, for one datapoint, the estimator $f: X \rightarrow \Omega$ that outputs the parameters of the optimal $q^* \in Q$ is given by

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

(17)

### 5 Related Work

For extended work on VQ-VAE, [20] 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. [13] 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 classical VQ-VAE model.

Several works have studied the end-to-end discrete representation learning model with different incorporated structures in the bottleneck stages. [22] and [5] introduce scalar quantization in the latent space and optimize jointly the entire model for rate-distortion performance over a database of training images. [2] 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 [2], 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.

Various techniques for regularizing the autoencoders have been proposed recently. [7] proposes an adversarial regularizer which encourages interpolation in the outputs and also improves the learned representation. [21] interprets the VAEs as an 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. [4, 3] use the information bottleneck principle [23] to recover the objective of $\beta$-VAE and show that the KL divergence term in ELBO is an upper bound on the information rate between input and prior. [11] 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. [9] proposes to use encoder-decoder structures to simulate the binary symmetric channel (BSC) to solve the joint source-channel coding problem while at the same time learn robust representations.

### 6 Experimental Results

#### 6.1 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 consists of two separate layers of $d$ units respectively. One layer learns the mean of the Gaussian distribution and the other layer learns the log variance. Then, the reparameterization trick or the information dropout technique is applied to the output of the layer. For the VQ-VAE, the bottleneck performs a nearest neighbor search on the encoder output.
Figure 3: Two-dimensional learned representations of MNIST. Each color indicates one digit class.

Then, the quantized codeword is fed into the decoder. For the soft VQ-VAE, the bottleneck also has two separate layers. One layer of size $d$ outputs the noiseless vector. Another layer with size $K$ outputs the log variance of components. The noise injection and estimation are performed afterwards. The baseline autoencoder directly feeds the encoder output to the decoder. All layers in the bottleneck do not have activation functions.

VQ-VAE and soft VQ-VAE models are trained in a similar fashion as described in Section 2.1. Specifically, the loss function for the soft VQ-VAE model is

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

### 6.2 Visualization of Latent Representation

Instead of learning high-dimensional latents and then projecting them into two-dimensional representations using t-SNE [25], we directly learn two-dimensional latents using autoencoder models. The reason is that the t-SNE is designed to display the data through clusters, hence t-SNE visualization may distort the real clusterability of representations. We use $z_e$ as our learned representation. In Fig. 3 we plot two-dimensional latent representations of the test set of MNIST learned by different autoencoder models. All autoencoder models are set to have similar reconstruction quality. It shows that the latent representation of soft VQ-VAE preserves the similarity relations of the input data better than the other models.

### 6.3 Representation Learning Tasks

We test our learned latent representation on K-means clustering and single-layer classification tasks as [7]. 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 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 the Hungarian algorithm [26], which is a one-to-one optimal assignment matching algorithm between
the predicted labels and the true labels. For the classification tasks, we use a fully connected layer with a ReLU activation function as our single-layer classifier. The single-layer classifier is trained on the latents of the training set and is independent of the autoencoders’ training.

Table 1: Accuracy of downstream tasks of MINST.

| Model                | MNIST, d = 64 |
|----------------------|---------------|
|                      | Clustering    | Classification |
| Raw Data             | 55.17         | 92.44          |
| Baseline Autoencoder | 52.61         | 91.91          |
| VAE                  | 56.44         | 89.10          |
| $\beta$-VAE ($\beta = 20$) | 73.81         | 91.10          |
| Information dropout  | 58.52         | 91.11          |
| VQ-VAE (K = 128)     | 53.48         | 81.62          |
| Soft VQ-VAE (K = 128)| **77.64**     | **93.54**      |

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

| Model                | SVHN, d = 256 | CIFAR-10, d = 256 |
|----------------------|---------------|-------------------|
|                      | Clustering    | Classification    | Clustering    | Classification    |
| Baseline Autoencoder | 11.96         | 25.95             | 21.73         | 40.92             |
| VAE                  | 13.58         | 26.42             | **24.12**     | 38.83             |
| $\beta$-VAE ($\beta = 100$) | **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) | 14.28         | **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 when compared to the baseline autoencoder model. The performance of classical 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 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 networks preserve more information from the input. Hence, we need to set a smaller $K$ to upper bound the information transfer between input and latents as suggested in ([11]).

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.

7 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 mapping at the encoder. Our proposed
The 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 noisy VQ-VAE model. 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 [7, 18]. The source code of the paper is publicly available.1

References

[1] Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(12):2897–2905, Dec 2018.

[2] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In Advances on Neural Information Processing Systems (NIPS), Long Beach, CA, December 2017.

[3] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a broken ELBO. In Proceedings of the 35th International Conference on Machine Learning, pages 159–168, Stockholm Sweden, 10–15 Jul 2018.

[4] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, April 2017.

[5] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End to end optimized image compression. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, April 2017.

[6] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. Journal of Machine Learning Research, 6:1705–1749, 2005.

[7] David Berthelot, Colin Raffel, Aurko Roy, and Ian J. Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. CoRR, 2018. URL http://arxiv.org/abs/1807.07543

[8] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-VAE. arXiv preprint arXiv:1804.03599, 2018.

[9] Kristy Choi, Kedar Tatwawadi, Tsachy Weissman, and Stefano Ermon. NECST: neural joint source-channel coding. CoRR, abs/1811.07557, 2018. URL http://arxiv.org/abs/1811.07557

[10] Thomas. M. Cover and Joy. A. Thomas. Elements of Information Theory. Wiley, New York, NY, USA, 2nd edition, 2006.

[11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, May 2010.

[12] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. CoRR, abs/1701.00160, 2017. URL http://arxiv.org/abs/1701.00160

[13] Gustav Eje Henter, Jaime Lorenzo-Trueba, Xin Wang, and Junichi Yamagishi. Deep encoder-decoder models for unsupervised learning of controllable speech synthesis. CoRR, 2018. URL https://arxiv.org/abs/1807.11470

[14] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, April 2017.

[15] Daniel Jiwoong Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising criterion for variational auto-encoding framework. In AAAI Conference on Artificial Intelligence, pages 2059–2065, 2017.

1https://github.com/AlbertOh90/Soft-VQ-VAE/
[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, May 2015.

[17] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Proceedings of the International Conference on Learning Representations (ICLR), Banff, Canada, April 2014.

[18] Ali Razavi, Aaron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. CoRR, 2019. URL http://arxiv.org/abs/1901.03416

[19] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Proc. of the 31st International Conference on Machine Learning, June 2014.

[20] Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar. Theory and experiments on vector quantized autoencoders. arXiv preprint arxiv:1803.03382, 2018.

[21] Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. Amortized inference regularization. In Advances on Neural Information Processing Systems (NIPS), December 2018.

[22] Lucas Theis, Andrew Cunningham Wenzhe Shi, and Ferenc Huszar. Lossy image compression with compressive autoencoders. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, April 2017.

[23] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In IEEE Information Theory Workshop (ITW), pages 1–5, 2015.

[24] Aaron van den Oord, Koray Kavukcuoglu, and Oriol Vinyals. Neural discrete representation learning. In Advances on Neural Information Processing Systems (NIPS), Long Beach, CA, December 2017.

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9:2579–2605, 2008.

[26] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In Proc. of the 33rd International Conference on Machine Learning, June 2016.

[27] Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. In Advances on Neural Information Processing Systems (NIPS), Long Beach, CA, December 2017.

[28] Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. Advances in variational inference. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1–1, 2018.
A Proof of Theorem 1

![Diagram](image)

Figure 4: The relation of variables in the soft VQ-VAE model. The symbol $\leftrightarrow$ is used to indicate that we cannot directly use paths that connected to the unobserved $z_e$ for probabilistic inference.

Proof. Here, we follow a similar proof structure of Theorem 1 in [21]. For the noisy VQ-VAE setting, the expectation of the KL divergence between the model posterior $\hat{p}(z|x)$ and the approximated $q$ is taken with respect to the empirical training distribution $\bar{p}(x)$ and the noise distribution $\hat{p}(\epsilon)$.

$$E_{\bar{p}(x)}E_{\hat{p}(\epsilon)}[KL(q(z_e + \epsilon)||p(z|x))]$$ (19)

Since the encoder neural network does not have a sigmoid or softmax function in the final 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:

$$\hat{p}(x)\hat{p}(\epsilon) = \hat{p}(z_e, \epsilon) = \hat{p}(z_e, z_e + \epsilon) = \hat{p}(z_e, z'_e)$$ (20)

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

$$\hat{p}(z_e, z'_e) = \hat{p}(z_e)\hat{p}(z'_e|z_e) = \hat{p}(z_e) \sum_{k=1}^{K} \hat{p}(\mu(k)|z_e) \hat{p}(z'_e|\mu(k))$$ (21)

$$= \hat{p}(z_e) \frac{1}{K} \sum_{k=1}^{K} \hat{p}(z'_e|\mu(k))$$ (22)

where (22) follows from that $z_e$ is considered as unobservable in the model(see Fig. 4), 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 $\frac{1}{K}$.

Following the above derivations, we can reexpress (19) as

$$E_{\hat{p}(z_e,z'_e)}[KL(q(z'_e)||p(z|\mu))] = \frac{1}{K} E_{\hat{p}(z_e)}E_{\hat{p}(z'_e|\mu(k))} \left[ KL(q(z'_e)||p(z|\mu(k))) \right].$$ (23)

Therefore, for each datapoint $x$, the optimization problem with respect to the latent distribution $q$ (15) for the noisy VQ-VAE setting becomes

$$\min_{q \in \mathcal{Q}} \frac{1}{K} E_{\hat{p}(z_e)}E_{\hat{p}(z'_e|\mu(k))} \left[ KL \left( q(z'_e) || p \left( z | \mu(k) \right) \right) \right] = \min_{q \in \mathcal{Q}} \frac{1}{K} \sum_{k=1}^{K} \sum_{z'_e} \hat{p}(z'_e|\mu(k)) KL \left( q(z'_e)||p \left( z | \mu(k) \right) \right).$$ (24)

Note that the KL divergence between two exponential family distributions can be represented by the Bregman divergence between the corresponding natural parameters $\eta'$ and $\eta$ as

$$KL(p_{\eta'}||p_{\eta}) = -A(\eta') + A(\eta) - \nabla A(\eta')^T(\eta' - \eta) = d_A(\eta, \eta'),$$ (25)

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 $\mathbb{E}[\eta]$. Therefore, we formulate (24) as a convex combination of the KL divergence

$$\arg \min_{q \in \mathcal{Q}} \sum_{k=1}^{K} \omega_k KL \left( q(z'_e)||p \left( z | \mu(k) \right) \right) = \arg \min_{\eta} \sum_{k=1}^{K} \omega_k d_A(\eta(k), \eta),$$ (26)

where $\omega_k = \frac{1}{K} \hat{p}(z'_e|\mu(k))$, $V$ (defined by (7)) and $K$ are the introduced normalization constants and the optimal solution of (24) is not affected. Hence, the minimizer of (26) is given by the mean of $\eta(k)$

$$\eta = \sum_{k=1}^{K} \omega_k \eta(k).$$ (27)
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 VQ-VAE, the covariance matrix is assumed to be the identity matrix for all components $\Sigma = I$. Therefore, we can recover the Bayesian estimator [17] by substituting $\eta^{(k)}$ with $\mu^{(k)}$ in (27), and the proof is complete.

B More training details

For the models tested on the CIFAR-10 and SVHN datasets, the encoder consists of 4 convolutional layers with stride 2 and filter size $3 \times 3$. 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, 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 [11] for the weights of encoder-decoder networks. The codebook is initialized by the uniform unit scaling. All models are trained using Adam optimizer [16] with learning rate 3e-4 and evaluate the performance after 40000 iterations with batch size 64. Early stopping at 10000 iterations is applied by soft VQ-VAE on SVHN and CIFAR-10 datasets.