

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 MNIST 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

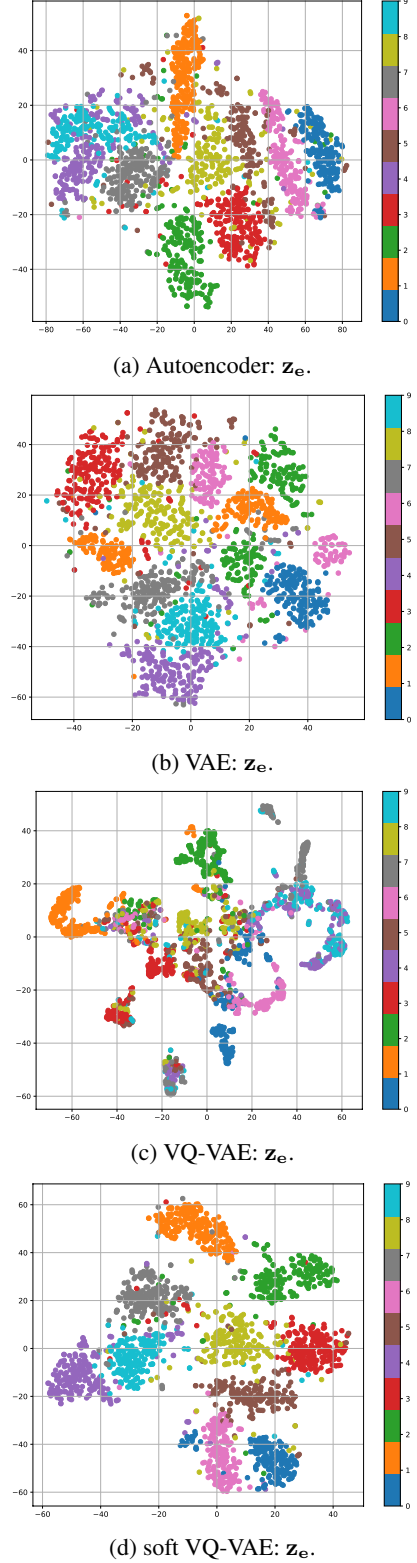


Figure 4: Two-dimensional learned representations of MNIST. Each color indicates one digit class.

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 downstream 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.

| Model | MNIST, $d = 64$ | | |
|-------------------------------|-----------------|------------------|----------------|
| | 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 ($\beta = 20$) | 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.

| 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 |
| β -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 ($K = 32$) | 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 repellendus, quam sed beatae. Perferendis aliquam illum adipisci minus optio obcaecati doloribus numquam maxime, perferendis officia culpa perspiciatis esse sint

¹<https://github.com/AlbertOh90/Soft-VQ-VAE/>