LEARNING PRODUCT CODEBOOKS USING VECTOR-QUANTIZED AUTOENCODERS FOR IMAGE RETRIEVAL

Hanwei Wu and Markus Flierl

School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology, Stockholm
{hanwei, mflierl}@kth.se

ABSTRACT

Vector-Quantized Variational Autoencoders (VQ-VAE)[1] provide an unsupervised model for learning discrete representations by combining vector quantization and autoencoders. In this paper, we study the use of VQ-VAE for representation learning of downstream tasks, such as image retrieval. First, we describe the VQ-VAE in the context of an information-theoretic framework. Then, we show that the regularization effect on the learned representation is determined by the size of the embedded codebook before the training. As a result, we introduce a hyperparameter to balance the strength of the vector quantizer and the reconstruction error. By tuning the hyperparameter, the embedded bottleneck quantizer is used as a regularizer that forces the output of the encoder to share a constrained coding space. With that, the learned latent features better preserve the similarity relations of the data space. Finally, we incorporate the product quantizer into the bottleneck stage of VQ-VAE and use it as an end-to-end unsupervised learning model for image retrieval tasks. The product quantizer has the advantage of generating large and structured codebooks. Fast retrieval can be achieved by using lookup tables that store the distance between any pair of sub-codewords. State-of-the-art retrieval results are achieved by the proposed codebooks.

I. INTRODUCTION

Recent advances in variational autoencoders (VAE) provide new unsupervised approaches to learn the hidden structure of data [2]. However, the classic VAEs are prone to the phenomenon of "posterior collapse ". Here, the latent representations are largely ignored by a decoder that is "too powerful ". Vector-quantized variational autoencoders (VQ-VAE) learn discrete representations by incorporating the idea of vector quantization (VQ) into the bottleneck stage. With that, the "posterior collapse "can be avoided [1], and the latent features learned by the VQ-VAE are more meaningful.

In the following, we study the use of VQ-VAE for representation learning for downstream tasks, such as image retrieval. We first describe the VQ-VAE from an information-theoretic perspective by using the so-called variational information bottleneck principle [3], [4]. We show that the regularization term of the latent representation is determined by the size of the embedded codebook, and hence, will potentially affect the generalization ability of the model. Since the regularizer is absent during the training, we introduce a hyperparameter to balance the strength of the vector quantizer and the reconstruction error. In this way, the bottleneck vector quantizer is used as a regularizer that enforces a constrained code space onto the output of the encoder. This is critical for applications such as image retrieval which require learned latent features that preserve the similarity relation of the input data.

We further modify the VQ-VAE by introducing a product quantizer (PQ) into the bottleneck stage such that the product codebook can be learned in an end-to-end fashion. Compared to classic vector quantization, the product quantizer can generate an exponentially large codebook at very low memory cost [5]. In addition, distance calculations between query and database items in the retrieval process can be avoided by using lookup tables which store the distances between codewords.

II. RELATED WORK

Several works have studied the end-to-end discrete representation learning model with different incorporated structures in the bottleneck stage. [6] and [7] introduce scalar quantization in the latent space and optimize the entire model jointly for rate-distortion performance over a database of training images. [8] 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. [9] and [10] introduce the Gumbel-Softmax gradient estimator for non-differentiable discrete distributions. For extended works on VQ-VAE, [11] uses the Expectation Maximization algorithm in the bottleneck stage to train the VQ-VAE and to achieve improved image generation results. We note that the authors in [12] also explore the product quantization idea for the VQ-VAE and use it to parallelize the decoding

process for the sequence model. This approach is known as the decomposed VQ-VAE.

III. QUANTIZER-BASED REGULARIZATION

A. Information Bottleneck and VQ-VAE

In this section, we give a brief description of how the VQ-VAE can be developed by using the concept of information bottleneck. Let I denotes the index of the input data, $\mathbf{X}$ the feature representation of the input data, and Z the index of the latent codeword. The objective is to learn a distribution $p(Z|I)$ from the given data distribution $p(I, \mathbf{X})$. Given certain constraints, the learned representation Z should retain from I as much information as possible about $\mathbf{X}$. The above variables are subject to the Markov chain constraint $\mathbf{X} \leftrightarrow I \leftrightarrow Z$.

Similar to the deterministic information bottleneck (DIB) principle as introduced in [4], we focus on minimizing the *representational cost* $H(Z)$ of the learned latent representation. Since the lossy compression of I incurs information loss when inferring $\mathbf{X}$ from $\mathbf{Z}$, we can formulate the problem as a rate-distortion-like problem that leverages the representational cost and the compression distortion by using the Lagrangian formulation

$$L_{\text{IB}} = d_{\text{IB}}(I, Z) + \epsilon H(Z), \tag{1}$$

where $d_{\text{IB}}(\cdot)$ is the information bottleneck distortion and ϵ the Lagrangian parameter.

Now, consider the case where the information bottleneck distortion is defined as the Kullback-Leibler (KL) divergence between the true data distribution and the data distribution generated by the latent representation [13], *i.e.*,

$$d_{\text{IB}}(I, Z) = \text{KL}(p(\mathbf{X}|I) \| p(\mathbf{X}|Z)). \tag{2}$$

We can decompose $d_{\text{IB}}(I, Z)$ into two terms

$$\text{KL}(p(\mathbf{X}|I) \| p(\mathbf{X}|Z)) = \tag{3}$$

$$\int \sum_z p(\mathbf{x}, z) \log \frac{p(z)}{p(\mathbf{x}, z)} d\mathbf{x} - \int \sum_i p(i, \mathbf{x}) \log \frac{p(i)}{p(i, \mathbf{x})} d\mathbf{x}. \tag{4}$$

The second term of (4) is determined solely by the given data distribution $p(I, \mathbf{X})$ and can be ignored for the propose of minimization. The first term of (4) has an upper bound by replacing $p(\mathbf{x}|z)$ with a variational approximation $q(\mathbf{x}|z)$ [3]

$$\int \sum_z p(\mathbf{x}, z) \log \frac{p(z)}{p(\mathbf{x}, z)} d\mathbf{x} \tag{5}$$

$$\leq - \sum_i p(i) \int p(\mathbf{x}|i) \sum_z p(z|i) \log q(\mathbf{x}|z) d\mathbf{x}. \tag{6}$$

In the VQ-VAE setting, vector quantization is applied to the output of the encoder. Hence, $p(z|i)$ is defined as

$$p(z = k|I = i) = \begin{cases} 1 & \text{for } k = \underset{z \in [K]}{\text{argmin}} \|z_e(\mathbf{x}_i) - \mu^{(z)}\|_2 \\ 0 & \text{otherwise,} \end{cases}$$

where K is the number of codewords of the quantizer, $\mu^{(z)}, z = 1, \ldots, K$, are the codewords, and $z_e(\cdot)$ is the encoder neural network. The approximation distribution can be expressed as

$$q(\mathbf{x}|z) = q(\mathbf{x}|Q(z)) = q\left(\mathbf{x}|\mu^{(z)}\right), \tag{7}$$

where $Q(\cdot)$ is a lookup function that maps the index to the codeword. Ideally, the decoder does not allocate any probability mass to $q\left(\mathbf{x}|\mu^{(z)}\right)$ for $\mu^{(z)} \neq z_q(\mathbf{x})$, where $z_q(\cdot)$ outputs the closest codeword to $z_e(\mathbf{x})$. Hence, we can write

$$\sum_z p(z|i) \log q(\mathbf{x}|z) = \log q(\mathbf{x}|z_q(\mathbf{x}_i)). \tag{8}$$

If we assume that $q(\mathbf{x}|z) = \mathcal{N}(\mathbf{x}|\hat{\mathbf{x}}, \mathbf{1})$ with $\hat{\mathbf{x}} = g\left(\mu^{(z)}\right)$, then $g(\cdot)$ is the decoder neural network. Hence, the log-likelihood of $q(\mathbf{x}|z)$ is proportional to the squared difference between the input and the output of the model. Therefore, (6) is considered as the reconstitution error of the model.

We use a similar variational approximation technique to upper-bound the entropy $H(Z)$ of the latent variable in (1) as proposed in [4].

$$H(Z) \leq - \sum_z p(z) \log r(z) \tag{9}$$

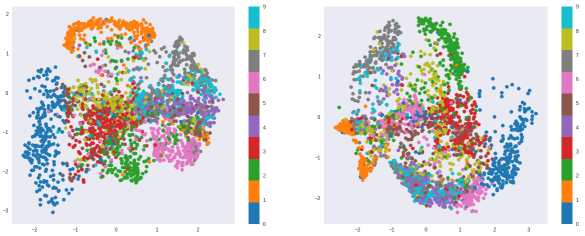
$$= - \sum_i \sum_z p(i) p(z|i) \log r(z) = -H(p(Z), r(Z)) \tag{10}$$

As a standard practice, the marginal $r(Z)$ is set to be an uniform distribution. Then the cross entropy (10) becomes a constant that equals to $\log K$, and hence, can be omitted from the loss function. That is, the constraint on the learned representations is determined by the size of the embedded codebook before the training.

To solve the problem of no gradient flowing through the discrete variables, the stop gradient operator $\text{sg}(\cdot)$ is used to separate the gradient update such that both encoder-decoder and the codebook are trained independently. Therefore, we can recover the empirical loss function of the VQ-VAE [1] from the information bottleneck principle of (6) as

$$L_{\text{VQ-VAE}} = \frac{1}{N} \sum_{i=1}^{N} \left[\log q(\mathbf{x}|z)) + \|\text{sg}\left(z_e(\mathbf{x}_i)\right) - z_q(\mathbf{x}_i)\|_2^2 \right.$$
$$\left. + \beta \|z_e(\mathbf{x}_i) - \text{sg}\left(z_q(\mathbf{x}_i)\right)\|_2^2 \right], \tag{11}$$

where we assume $p(i) = \frac{1}{N}$ and where N is the number of training items. The third term of (11) is the commitment loss which forces the encoder output $z_e(\mathbf{x})$ to commit



(a) $K = 20$ (b) $K = 25$

Fig. 1: Visualization of $z_e(\mathbf{x})$. The colors indicate different digit classes.

to a codeword. β is a constant weight parameter for the commitment loss.

B. Regularization Effects of the Bottleneck VQ

In this section, we investigate the relations among the size of the codebook, the quantization error, and the generalization ability of the model. When we increase the size of the embedded codebook, the output of encoder has more codeword choices. In this case, the latents are more likely to be quantized into codewords that are far away from each other. As a result, lower reconstruction errors can be achieved due to the high discriminability of latent codewords. However, the generalization is poor in this case because the setting does not encourage similar data points to be mapped together in the latent space. On the other hand, a smaller size of the codebook decreases the average discriminability of the input data. In order to reduce reconstruction errors for the low rate setting, the model is forced to ensure that neighboring data points are also represented closely together in the latent space, which potentially leads to a better generalization ability.

In Fig. 1, we plot the two-dimensional latent representation learned from the MNIST dataset with 10 digit classes using MLP encoder-decoder networks. Although the high rate case $(K = 25)$ has lower reconstruction error than the low rate case $(K = 20)$, we can observe from the figures that the latent representation in the high rate case shows more overlap than in the low rate case. This indicates the weak generalization ability of the high rate setting.

In order to solve the problem of the weak generalization ability of the high rate setting, we introduce a hyperparameter to control the strength of the vector quantizer. Specifically, we introduce a multiplicative weight λ to both second and third term of (11) to control the updating power of the vector quantizer. If we increase the value of λ, the vector quantizer becomes "more powerful". This minimizes the quantization error and pushes codewords far away from each other. In this case, the input data is less likely to be updated to another codeword due to the weak encoder-decoder optimization in the first term of (11). This may decrease the reconstruction error, but it leads to a weaker generalization ability. On the other hand, a small value of λ creates a "weaker" vector quantizer and the quantization error increases. This is equivalent to adding noise to the latents. In this case, the input data will be easily swayed away to other codewords due to the increased quantization error. This creates similar effects as the low rate setting of the vector quantizer, where the locality of the data space in the latent space is better preserved. In this way, the bottleneck vector quantizer is used as a regularizer of the latent representation as it enforces a shared coding space on the encoder output.

In Fig. 2, we plot the learned representation using the same layer structure as in Fig. 1, but we use the introduced hyperparameter $\lambda = 0.5$. We can observe that the latent representation preserves the similarity relations of the input data better than the representation as shown in Fig. 1b in terms of using the space more uniformly.

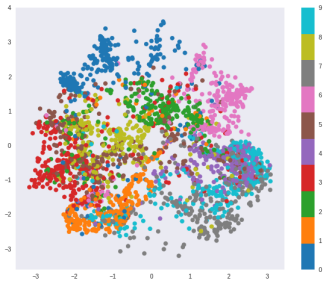


Fig. 2: Representation with $K = 25$ codewords and hyperparameter $\lambda = 0.5$.

C. Tuning of the Hyperparameter λ

In the following, we provide a method for finding appropriate values of λ. We use the ratio of the average distances of the closest codeword to the second closest codeword of the input data as a robust indicator for the "impact" of the vector quantizer. For example, in Fig. 3, we plot the reconstruction error over above distance ratio for varying values of λ. We observe that the reconstruction error decreases for distance ratios between 0.35 and 0.6 as the codewords are updated more easily by the encoder-decoder optimization. However, when the distance ratio exceeds approximately 0.62, the input data is swayed over too easily to other codewords by the encoder-decoder optimization. Therefore, the network optimization becomes sub-optimal and the reconstruction error increases significantly. In order to find an appropriate value for the hyperparameter λ, we limit our search of λ by restricting the permissible range of distance ratios.

IV. APPLICATION: IMAGE RETRIEVAL

A. PQ-VAE

For the PQ-VAE model, the output of the encoder $z_e(\mathbf{x}) \in \mathbb{R}^D$ is fed into a product quantizer. The product quantizer

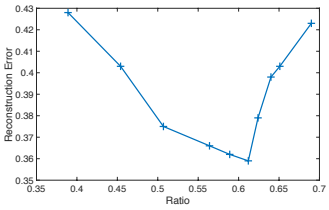


Fig. 3: Reconstruction error over distance ratio for varying λ.

consists of M sub-vector quantizers which handle sub-vectors of dimension D/M. Each sub-quantizer partitions the subspace into K clusters. The sub-clusters are characterized by the sub-codebook $C_m = \left\{\mu_m^{(1)}, \cdots, \mu_m^{(K)}\right\}$, where $m = 1, \cdots, M$ denotes the m-th sub-vector quantizer.

Each sub-vector is quantized to one of the K codewords by the nearest neighbor search

$$z_q^{(m)}(\mathbf{x}) = \mu_m^{(k)}, \text{ where } k = \underset{i}{\operatorname{argmin}} \|z_e^{(m)}(\mathbf{x}) - \mu_m^{(i)}\|_2. \tag{12}$$

The output of the M sub-quantizers is concatenated to the full codeword $z_q(\mathbf{x}) = \left[z_q^{(1)}(\mathbf{x}), z_q^{(2)}(\mathbf{x}), \cdots, z_q^{(M)}(\mathbf{x})\right]$ and then passed as input to the decoder. The decoder then reconstructs the input images $\mathbf{x}$ given the full codewords $z_q(\mathbf{x})$.

The M sub-quantizers are trained independently. Each sub-codeword is simply updated by the average of the latents that have been assigned to it in each iteration.

B. Querying

The discrete encoding $\mathbf{z}$ of images can be generated by using the trained encoder and learned product codebook. The encoder outputs n channel vectors for quantization. Hence, each image is represented by an encoding vector of size $n \times M$, where each element is the index of the closest sub-codeword to the sub-vector of the output of the encoder. The encoding vector of an image can then be compressed into a code of length $R = NM \log_2 K$ bits.

The querying is conducted in the encoding space. Besides the encoding of database images, we store M Lookup Tables (LT) with $K \times K$ entries. Each LT stores the distances between every two sub-codewords of its sub-codebook. The image encoding is used for the indices of the table. When querying, the distance between query $\mathbf{q}$ and database $\mathbf{x}$ is obtained by summing up the distances as given by the LTs

$$d(\mathbf{q}, \mathbf{x}) = LT_1\left(z_u^{(1)}, z^{(1)}\right) + \cdots + LT_M\left(z_u^{(M)}, z^{(M)}\right), \tag{13}$$

where z_u denotes an encoding of the query image, and z an encoding of the database image. Hence, fast retrieval can be achieved because no additional distance calculations are needed.

C. Results

We use the CIFAR-10 dataset which contains 60000 images of size $32 \times 32 \times 3$ to test the performance of PQ-VAE on the image retrieval task. We train the model by using 50000 images from the training set. Further, we treat their discrete encodings as the database items. 10000 test images are used as queries. We use the mean Average Precision (mAP) of the top 1000 returned images as the performance measure.

The encoder consists of 2 strided convolutional layers with stride 2 and filter size 3×3. Each convolution layer is followed by a 2×2 max pooling layer. The number of channels of the first layer is set to be 64 and is doubled for the following layers. All the convolution layers use rectified linear units (ReLU) as activation functions. In this setting, the input images are compressed into $n = 2 \times 2$ discrete encodings. The decoder follows a structure that is symmetric to the encoder. We use the ADAM optimizer [14] with a learning rate of 1e-4 and evaluate the performance after 30000 iterations with a batch-size of 100.

We test the cases of latent codes with compression rates of 32, 48 and 64 bits. To achieve those rates, we set the product quantizer to use $M = 4$ sub-quantizers and the number of codewords of the sub-quantizers to $K = 4, 8, 16$. We compare our model to other unsupervised reference methods in Table I. Our proposed model outperforms the reference methods in the table. Note that we do not compare to results of models that have been pre-trained by ImageNet in a supervised fashion.

	32 bits	48 bits	64 bits
LSH [15]	12.00	12.00	15.07
Spectral Hashing [16]	13.30	13.00	13.89
Spherical Hashing [17]	13.30	13.00	15.38
ITQ [18]	16.20	17.50	16.64
Deep Hashing [19]	16.62	16.80	16.69
PQ-VAE	**21.86**	**22.79**	**23.42**

Table I: Mean Average Precision (mAP) of the top 1000 returned images for compression rates of 32, 48, and 64 bits.

V. CONCLUSIONS

We extended the work of VQ-VAE by embedding a product quantizer into the bottleneck of an autoencoder for the image retrieval task. We formulate the VQ-VAE problem by using the so-called variational information bottleneck principle and show that the regularization term for the learned representations is determined by the size of the embedded codebook. We introduce a hyperparameter to control the impact of the vector quantizer such that we can further regularize the latent representation. With an appropriately tuned hyperparameter, we show that our learned representations improve the mean average precision of our image retrieval task.

VI. REFERENCES

[1] A. Oord, K. Kavukcuoglu, and O. Vinyals, "Neural discrete representation learning," in *NIPS*, 2017.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *ICLR*, 2014.

[3] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *ICLR*, 2017.

[4] D. J. Strouse and D. Schwab, "The deterministic information bottleneck," *Neural Comput.*, vol. 29, no. 6, pp. 1611–1630, 2017.

[5] H. Jegou, M. Douje, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615 –1630, 2011.

[6] L. Theis, W. Shi, A. Cunningham, and F. Huszar, "Lossy image compression with compressive autoencoders," in *ICLR*, 2017.

[7] J. Balle, V. Laparra, and E. P. Simoncelli, "End to end optimized image compression," in *ICLR*, 2017.

[8] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *NIPS*, 2017.

[9] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," in *ICLR*, 2017.

[10] C. J. Maddison, A. Minh, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *ICLR*, 2017.

[11] A. Roy, A. Vaswani, A. Neelakantan, and N. Parmar, "Theory and experiments on vector quantized autoencoders," *arXiv preprint arxiv:1803.03382*, 2018.

[12] L. Kaiser, A. Roy, A. Vaswani, A. Neelakantan, N. Parmar, S. Bengio, J. Uszkoreit, and N. Shazeer, "Fast decoding in sequence models using discrete latent variables," in *Proc. of the 35th International Conference on Machine Learning*, June 2018.

[13] A. Gilad-Bachrach, A. Navot, and N. Tishby, "An information theoretic tradeoff between complexity and accuracy," in *Proceedings of the COLT*, 2003.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[15] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locatlity-sensitive hashing scheme based on p-stable distributions," in *Symposium on Computational Geometry*, 2004.

[16] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *NIPS*, 2008.

[17] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing: Binary code embedding with hyperspheres," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2304 –2316, Nov. 2015.

[18] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[19] V. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2015, pp. 2475 – 2483.