Vali, Mohammadhassan; Bäckström, Tom

Stochastic Optimization of Vector Quantization Methods in Application to Speech and Image Processing

Published in:
International Conference on Acoustics, Speech, and Signal Processing

DOI:
10.1109/ICASSP49357.2023.10096204

Published: 01/01/2023

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

STOCHASTIC OPTIMIZATION OF VECTOR QUANTIZATION METHODS IN APPLICATION TO SPEECH AND IMAGE PROCESSING

Mohammad Hassan Vali, Tom Bäckström

Department of Signal Processing and Acoustics, Aalto University, Finland
mohammad.vali@aalto.fi, tom.backstrom@aalto.fi

ABSTRACT

Vector quantization (VQ) methods have been used in a wide range of applications for speech, image, and video data. While classic VQ methods often use expectation maximization, in this paper, we investigate the use of stochastic optimization employing our recently proposed noise substitution in vector quantization technique. We consider three variants of VQ including additive VQ, residual VQ, and product VQ, and evaluate their quality, complexity and bitrate in speech coding, image compression, approximate nearest neighbor search, and a selection of toy examples. Our experimental results demonstrate the trade-offs in accuracy, complexity, and bitrate such that using our open source implementations and complexity calculator, the best vector quantization method can be chosen for a particular problem.

***Index Terms*—** Complexity, Machine Learning, Rate-Distortion, Vector Quantization

1. INTRODUCTION

Vector Quantization (VQ) is a data compression technique which can be used to model any data distribution. It has been widely used in applications such as image generation [1], speech and audio coding [2, 3], voice conversion [4, 5], music generation [6], and text-to-speech synthesis [7, 8, 9]. All of these applications employ a plain form of VQ, which is applicable only for limited bitrates, because it would be too complex for high bitrates. Regarding this bitrate constraint, plain VQ might not be able to model structurally complicated distributions properly. Hence, using VQ variants such as additive VQ (AVQ) [10], residual VQ (RVQ) [11], and product VQ (PVQ) [12] can lead to a superior performance in some applications, since they allow the design of VQ with high bitrates and high dimensional data.

One of the main applications of these VQ variants is in approximate nearest neighbor (ANN) search [10, 11, 12], in which high dimensional vectors are compressed to another space (with lower or the same dimensionality), where distances among them can be calculated with less computational and storage cost. To optimize the codebooks of the VQ operation, RVQ [11] and PVQ [12] apply k-means, whereas AVQ [10] uses block-coordinate descent optimization algorithm. However, to the best of our knowledge these VQ variants have not yet been optimized by machine learning (ML) optimization algorithms for ANN search, or even for other applications which deal with different types of data. One issue with optimizing VQ codebooks with ML algorithms is that the backpropagation process fails, since VQ is a piecewise constant function and has zero gradients almost everywhere [13]. To solve this issue, most of the applications adopt the straight through estimator (STE) [14], which does not consider the influence of quantization noise [15].

In this paper, we use machine learning optimization for AVQ, PVQ, and RVQ. To optimize their codebooks, we apply our recently proposed noise substitution in vector quantization (NSVQ) technique [13], which provides more accurate gradients and faster convergence than STE. Also in contrast to STE, NSVQ does not need any additional hyper-parameter tuning. We apply these VQ variants in four different scenarios: 1) speech coding, 2) image compression, 3) ANN search, and 4) toy examples, and compare their performances in each scenario. Our results demonstrate that in the ANN search task, our recall values are comparable to the baseline approaches for all three VQ variants. We provide the PyTorch implementation of these VQ variants along with their complexity calculator code in a public webpage[1]. Using our experimental results and prepared implementation codes, one can make informed choices in the trade-off between complexity, accuracy and bitrate for specific applications.

2. VECTOR QUANTIZATION METHODS

Assume that we want to encode a D-dimensional input vector $x \in \mathbb{R}^D$ with a specific vector quantization (VQ) method using M codebooks $(C_1, C_2, ..., C_M)$, each containing K vectors (codewords). The overall bitrate is then $M \log_2 K$.

Additive VQ (AVQ) [10] encodes x to $\hat{x}$ as sum of M codewords, by selecting one codeword from each codebook,

$$\hat{x} = \sum_{m=1}^{M} C_m(j) \; ; \quad j \in [1, K] \; , \tag{1}$$

[1] https://gitlab.com/speech-interaction-technology-aalto-university/vq-variants

where $C_m(j)$ refers to the jth codeword in the mth codebook. In other words, the main objective of AVQ is to find a tuple for the input vector (a vector showing the codeword index per codebook) like $I = [i_1, i_2, ..., i_M]$, such that it minimizes the Euclidean distance of $\|x - \hat{x}\|^2$. To find the best tuple I, AVQ adopts the beam search algorithm [16]. It starts by defining N (so-called beam width) different tuples through finding the N closest codewords to x from $C_1 \cup C_2 \cup ... \cup C_M$ using

$$\arg\min_{C_m(j)} \|x - C_m(j)\|^2; \quad m \in [1, M],\ j \in [1, K], \quad (2)$$

and putting them as the first index in each tuple. These N tuples are preserved as the basis tuples for the rest $M-1$ iterations of the encoding process, until all tuples are filled with M elements. To complete the pth index ($p > 1$) for these N basis tuples, N residual vectors are calculated by subtracting the $p-1$ codewords already included in each tuple from x. Then, for each residual vector corresponding to a basis tuple, beam search considers the remaining $M + 1 - p$ codebooks (which were not already involved in the tuple) as the searching set, and finds the N closest codewords to the residual in this set. Now, N^2 tuples with length p are generated. Among these tuples, the best N unique tuples which renders the best VQ error are selected as the basis tuples for the next encoding iteration. At the end of $M-1$ encoding iterations, when all tuples are filled with M elements, the tuple with the least VQ error is selected as the final best encoding tuple.

Residual VQ (RVQ) [11], in contrast to AVQ, considers only one tuple I for quantization without using beam searching to find the best indices. To determine the first index i_1, RVQ finds the closest codeword to x from the first codebook C_1 using Eq. 2. Then it calculates the residual of the first stage R_1 by subtracting the selected codeword $C_1(i_1)$ from x. The next index i_2 is determined by finding the closest codeword to R_1 from codebook C_2. Accordingly, the best index at each stage is specified by finding the closest codeword of that stage to the residual of the previous stage, until all elements of the tuple I is filled. Similarly to AVQ, at the end RVQ encodes the input vector x as as sum of M codewords (one per codebook), as in Eq. 1.

Product VQ (PVQ) [12] splits the input vector x to M distinct and independent subvectors (subspaces) of the dimension $D^* = D/M$, and encodes x as a concatenation (i.e. Cartesian product) of M codewords (one per codebook) of dimension D^*. PVQ considers only one tuple $I = [i_1, i_2, ..., i_M]$ for the quantization, and performs M independent quantizers for M available subvectors. The index $i_k; k = 1, 2, .., M$ corresponding to the kth subvector is determined by finding the index of the closest codeword from codebook C_k to that subvector. After performing all M quantizers for M subspaces and completing all elements in tuple I, PVQ encodes the input vector x as concatenation of all M subvectors, such that

$$\hat{x} = concatenate[\ C_1(i_1), C_2(i_2), ..., C_M(i_M)\ ]\ . \quad (3)$$

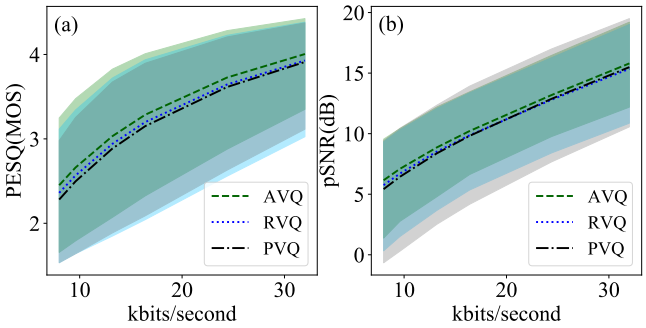

Fig. 1. (a) PESQ and (b) pSNR values for 16 bit VQ at sampling rates of 8, 9.6, 13.2, 16.4, 24.4 and 32 kbit/s in the speech coding scenario; lines refer to the mean values and the corresponding filled areas refer to their 95 % quantiles.

3. EXPERIMENTS

To evaluate the performance of three VQ variants, we conducted the experiments in four different scenarios. In the *speech coding scenario* of [3], we model the 16-dimensional spectral envelopes of speech, using each of three VQ variants trained end-to-end, using the LibriSpeech dataset [17] (containing 100 h of clean English speech) over 5 epochs using the Adam optimizer with the learning rate of 10^{-3}. In the *image compression scenario*, we apply VQ variants to model the discrete latent space of the vector quantized variational autoencoder (VQ-VAE) presented in [18]. The network architecture and hyper-parameters are the same as those in the original paper [18], except the batch size which is shown to be optimum at 32 [19]. The latent space dimension is 64, and VQ variants are trained and tested on CIFAR10 dataset over 15 k training batches using Adam optimizer with the learning rate of 10^{-3}.

In the *approximate nearest neighbor (ANN) search scenario*, we model the distribution of SIFT1M [12] dataset (of 128-dimensional image descriptors) by training three VQ variants on its learning set over 300 epochs using Adam optimizer with the learning rate of 10^{-2}. The batch sizes for AVQ, RVQ and PVQ are selected as 100, 50 and 50, respectively. Finally, in the *toy examples scenario*, we compress correlated and uncorrelated datasets with dimensions of 64, 128, and 256. The correlated dataset was generated by matrix multiplication between one million vectors sampled from a normal distribution ($\mathcal{N}(0, 1)$) and fifteen pre-specified fixed vectors as the generative vectors. To generate the uncorrelated dataset, we sampled one million vectors from a uniform distribution ($U(-20, 20)$). We trained the VQ variants over 100 epochs using the Adam optimizer with the learning rate of 10^{-2}. For all scenarios we use four codebooks (i.e. $M = 4$), and we apply the codebook replacement function during training following [13]. Also, the beam width (N) is 16 for both training and inference phases for all scenarios, except the ANN search in which $N = 16$ for training and $N = 64$ for inference, to have the same setting as the baselines.

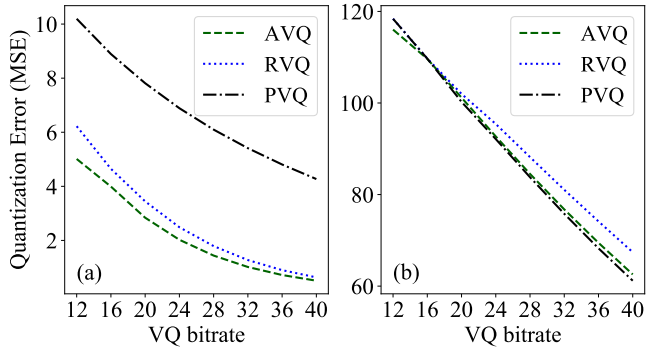

Fig. 2. Quantization error for (a) correlated and (b) uncorrelated datasets using all VQ variants (data dimension = 64).

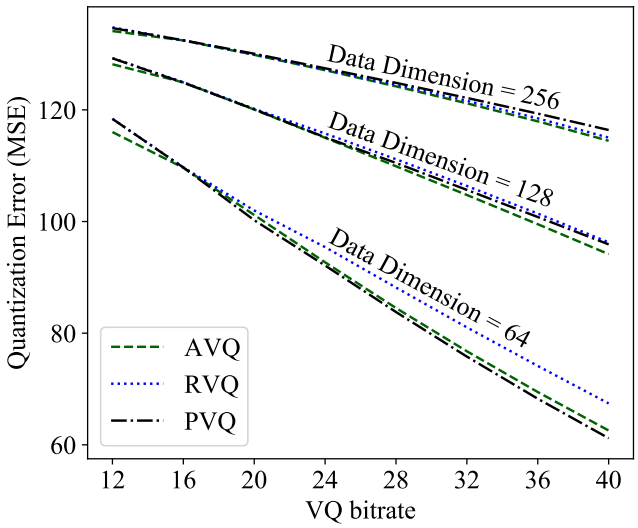

Fig. 3. Quantization error for uncorrelated dataset using all three VQ variants in different data dimensions and bitrates.

4. RESULTS AND DISCUSSIONS

The performance of three VQ variants were evaluated in four different scenarios. In the *speech coding scenario*, we employ perceptual evaluation of speech quality (PESQ) [20] and perceptually weighted signal to noise ratio (pNSR) objective metrics to evaluate the decoded speech quality. We decode the LibriSpeech test set samples [17] with different sampling rates of 8, 9.6, 13.2, 16.4, 24.4 and 32 kbit/s, matching the operating modes of 3GPP EVS codec [21]. Regarding the experiments, the performance of AVQ, RVQ, and PVQ were quite the same for high VQ bitrates. Hence, we prepare the results for a lower VQ bitrate of 16 bits (4 bits per codebook) in Figure 1. The lines and filled areas refer to the mean and 95 % quantiles of the metrics respectively, taken over entire test samples. We observe that AVQ has higher mean and lower variance than both RVQ and PVQ for both metrics, while RVQ and PVQ perform more or less similarly. AVQ gains 78 % and 81 % higher PESQ values than RVQ and PVQ, respectively. In addition, AVQ obtains 77 % and 65 % higher pSNR values than RVQ and PVQ, respectively.

In the *image compression scenario*, we reconstruct the test images of CIFAR10 dataset using the trained encoder, decoder and learnt codebooks of each VQ variant. We employ structural similarity index measure (SSIM) and peak signal to noise ratio (Peak SNR) objective metrics to assess the quality of the reconstructed images. Figure 4.a,b shows mean and 95 % quantiles of SSIM and Peak SNR values for entire CIFAR10 test set over ten individual experiments. Both AVQ and RVQ methods have higher means and lower variances than PVQ for both metrics. In addition, AVQ performs better than RVQ with the SSIM metric, but worse than RVQ for the Peak SNR measure. Since SSIM refers to the interdependencies among neighboring pixels of an image, it can be concluded that AVQ causes less perceptual degradation in structural information. However, AVQ attains the best quality at the expense of the highest computationally complexity. According to the Figure 4.c, for a fixed complexity value (same use of computational resources), RVQ performs far better than AVQ at higher VQ bitrates, and better than PVQ at lower VQ bitrates.

The SIFT1M [12] image descriptors dataset includes 1 M base vectors, 100 k learning vectors, and 10 k query vectors for testing purposes. The ground-truth is the set of actual nearest neighbors, from the base to the query vectors. In the *ANN search scenario*, we first compress the base vectors using the corresponding learnt codebooks trained on the learning set. Then, for the query vectors, we find the approximate nearest neighbors from the compressed base vectors by performing an exhaustive search using asymmetric distance [12]. We assess the precision of this searching process by calculating the recall metric, which denotes the probability that the actual nearest neighbor exists in the T closest compressed base vectors. Figures 5 and 6.a compare the recall values of the baseline AVQ [10], RVQ [11], and PVQ [12] methods with the recall values of the proposed ML optimized versions. Note that for the baseline methods, we extract the recall values from the plots prepared in the original papers as accurate as possible. Since ML optimization may end up in various approximate solutions, for the proposed ML optimized VQ variants, we plot the mean and 95 % quantiles of the recall values over ten individual experiments. For both baseline and ML optimized versions, AVQ performs better than RVQ, and RVQ performs better than PVQ approach. In total, all three ML optimized VQ variants achieve comparable (even slightly better in case of RVQ) recall values than the baselines. According to Figure 6.b, for a fixed complexity value, RVQ can perform far better than AVQ when having higher VQ bitrates, and better than PVQ while using lower VQ bitrates.

In the *toy examples scenario*, we compress the correlated and uncorrelated datasets using all three VQ variants. The mean squared error (MSE) between the original and quantized data then measures the accuracy. Figure 2 shows the mean and 95 % quantiles of MSE values over ten individual experiments, when the data dimension is 64. Since AVQ and

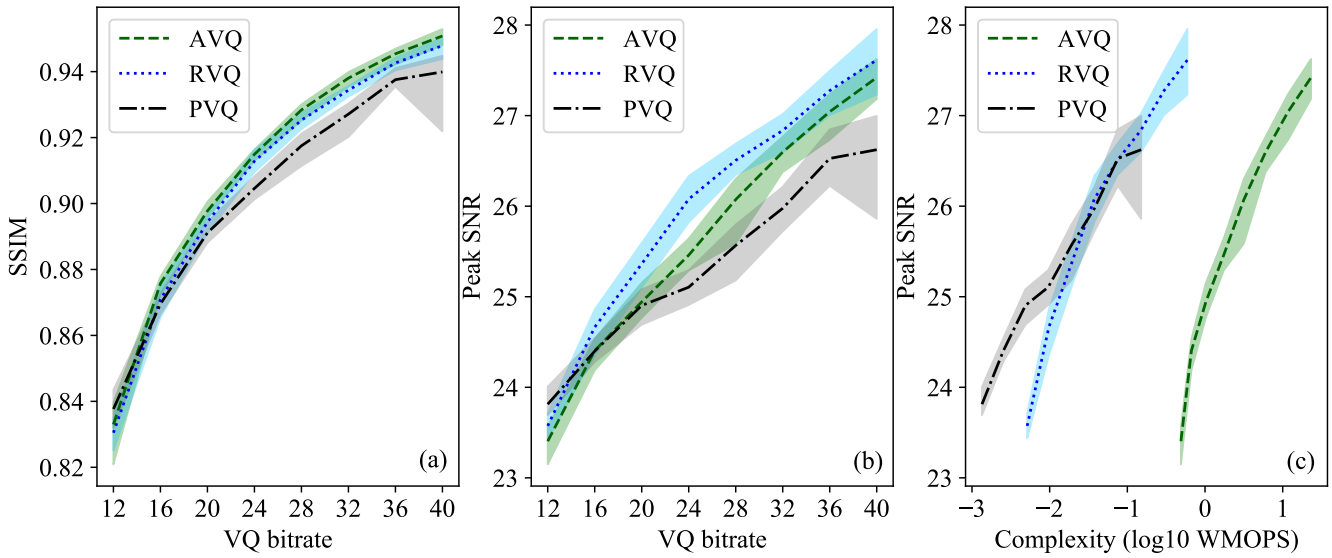

Fig. 4. (a) SSIM, (b) Peak SNR and (c) complexity of VQ variants over 15 k training batches and 10 individual experiments in the image compression scenario; lines refer to the mean values and the corresponding filled areas refer to their 95 % quantiles.

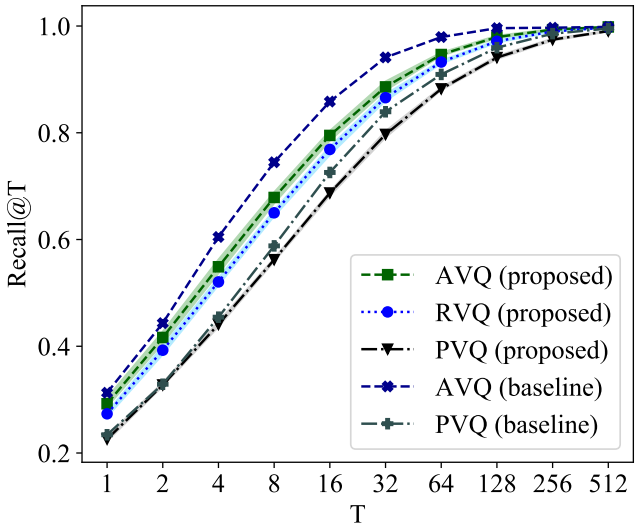

Fig. 5. Recall values for SIFT1M dataset by applying VQ variants at 64 bits (8 codebooks each with 256 codewords).

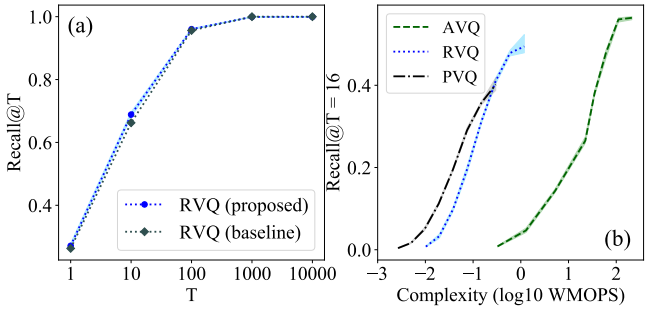

Fig. 6. (a) RVQ recall values. (b) Complexity of VQ variants in weighted million operations per second (WMOPS).

RVQ take the correlation among all data dimensions into account, they have much lower quantization error than PVQ as expected. On the other hand, PVQ has better performance than AVQ and RVQ for uncorrelated data, since there is no correlation among data dimensions and that is exactly what PVQ presumes. Figure 3 shows the MSE values of the uncorrelated dataset for input dimensions of 64, 128, and 256. We observe that the performance of PVQ decreases when increasing the data dimension from 64 to 256 (or increasing the subspace dimension from 16 to 64), such that it can perform worse than AVQ and RVQ, even if there is no correlation in the data. Note that in both Figures 2 and 3, the 95 % quantiles of MSE values are so small that they are not visible.

5. CONCLUSION

Plain form of vector quantization (VQ) has been widely used in many applications even if it is applicable only for limited bitrates, and it was optimized by classic expectation maximization or straight through estimator in neural networks so far. In this paper, we optimize three variants of VQ (including Additive VQ, Residual VQ, and Product VQ) by machine learning (ML) optimization using our recently proposed noise substitution in vector quantization technique, in order to allow quantization for higher bitrates and high dimensional data. We evaluate the performances of these three VQ variants and investigate the trade-offs in accuracy, complexity and bitrate through four different scenarios. Experimental results reveal that the proposed ML optimized methods performs quite comparably to the baseline approaches. Furthermore, our experiments demonstrate strengths and weaknesses of each method, helping in the choice among them, using our open-source implementation to a particular use-case.

6. REFERENCES

[1] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with VQ-VAE-2," in *Proceedings of NeurIPS*, 2019, pp. 14 866–14 876.

[2] C. Gârbacea, A. v. den Oord, Y. Li, F. S. C. Lim, A. Luebs, O. Vinyals, and T. C. Walters, "Low bit-rate speech coding with VQ-VAE and a Wavenet decoder," in *Proceedings of ICASSP*, 2019, pp. 735–739.

[3] M. H. Vali and T. Bäckström, "End-to-end optimized multi-stage vector quantization of spectral envelopes for speech and audio coding," in *Proceedings of Interspeech*, 2021, pp. 2728–2732.

[4] S. Ding and R. Gutierrez-Osuna, "Group latent embedding for vector quantized variational autoencoder in non-parallel voice conversion," in *Proceedings of Interspeech*, 2019, pp. 724–728.

[5] D.-Y. Wu, Y.-H. Chen, and H.-Y. Lee, "VQVC+: one-shot voice conversion by vector quantization and U-Net architecture," *arXiv preprint arXiv:2006.04154*, 2020.

[6] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: a generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[7] G. E. Henter, J. Lorenzo-Trueba, X. Wang, and J. Yamagishi, "Deep encoder-decoder models for unsupervised learning of controllable speech synthesis," *arXiv preprint arXiv:1807.11470*, 2018.

[8] A. Tjandra, B. Sisman, M. Zhang, S. Sakti, H. Li, and S. Nakamura, "VQVAE unsupervised unit discovery and multi-scale code2spec inverter for Zerospeech challenge 2019," in *Proceedings of Interspeech*, 2019, pp. 1118–1122.

[9] X. Wang, S. Takaki, J. Yamagishi, S. King, and K. Tokuda, "A vector quantized variational autoencoder (VQ-VAE) autoregressive neural F_0 model for statistical parametric speech synthesis," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 157–170, 2020.

[10] A. Babenko and V. Lempitsky, "Additive quantization for extreme vector compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 931–938.

[11] Y. Chen, T. Guan, and C. Wang, "Approximate nearest neighbor search by residual vector quantization," *Sensors*, vol. 10, no. 12, pp. 11 259–11 273, 2010.

[12] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

[13] M. H. Vali and T. Bäckström, "NSVQ: Noise substitution in vector quantization for machine learning," *IEEE Access*, vol. 10, pp. 13 598–13 610, 2022.

[14] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[15] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: bridging full-precision and low-bit neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[16] S. C. Shapiro, "Encyclopedia of artificial intelligence," 1988.

[17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Proceedings of ICASSP*, 2015, pp. 5206–5210.

[18] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proceedings of NeurIPS*, 2017, pp. 6309–6318.

[19] R. Kumar, T. Deleu, and E. Racah, "Reproducing neural discrete representation learning," 2018, [Online; accessed 10.10.2022]. [Online]. Available: https://github.com/ritheshkumar95/pytorch-vqvae/blob/master/final_project.pdf

[20] *Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs*, ITU-T Recommendation P.862, 2001. [Online]. Available: http://www.itu.int/rec/T-REC-P.862/en

[21] *TS 26.445, EVS Codec Detailed Algorithmic Description; 3GPP Technical Specification (Release 12)*, 2014.