Optimizing Vector Quantization Methods by Machine Learning Algorithms

Machine learning optimization of vector quantization methods used in end-to-end training of neural networks

Mohammad Hassan Vali · Follow

Published in Towards Data Science · 10 min read · May 17, 2023

This post is a short explanation of my paper [1] published at ICASSP 2023 conference. For more details, please look at the paper under this link.

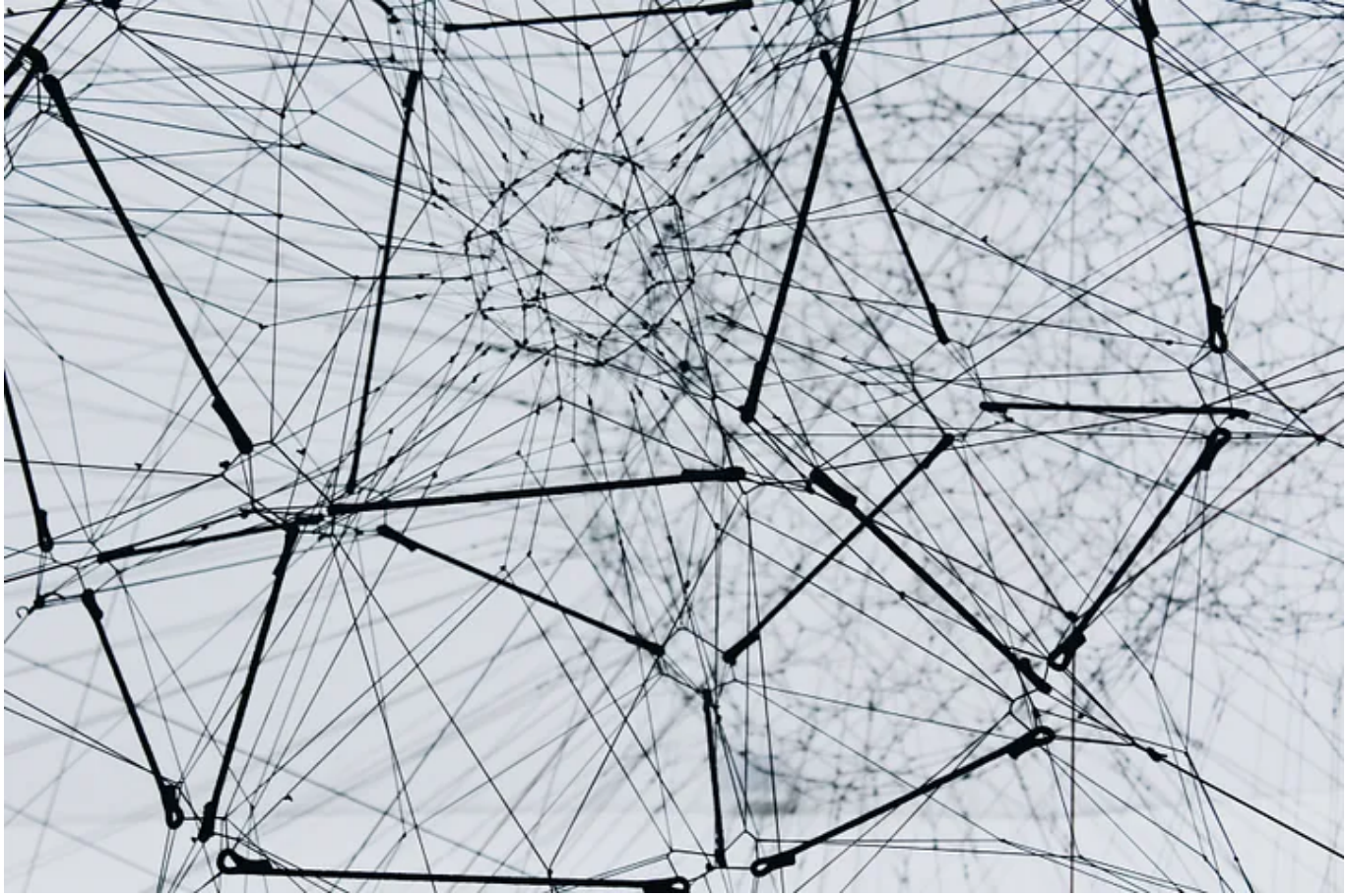
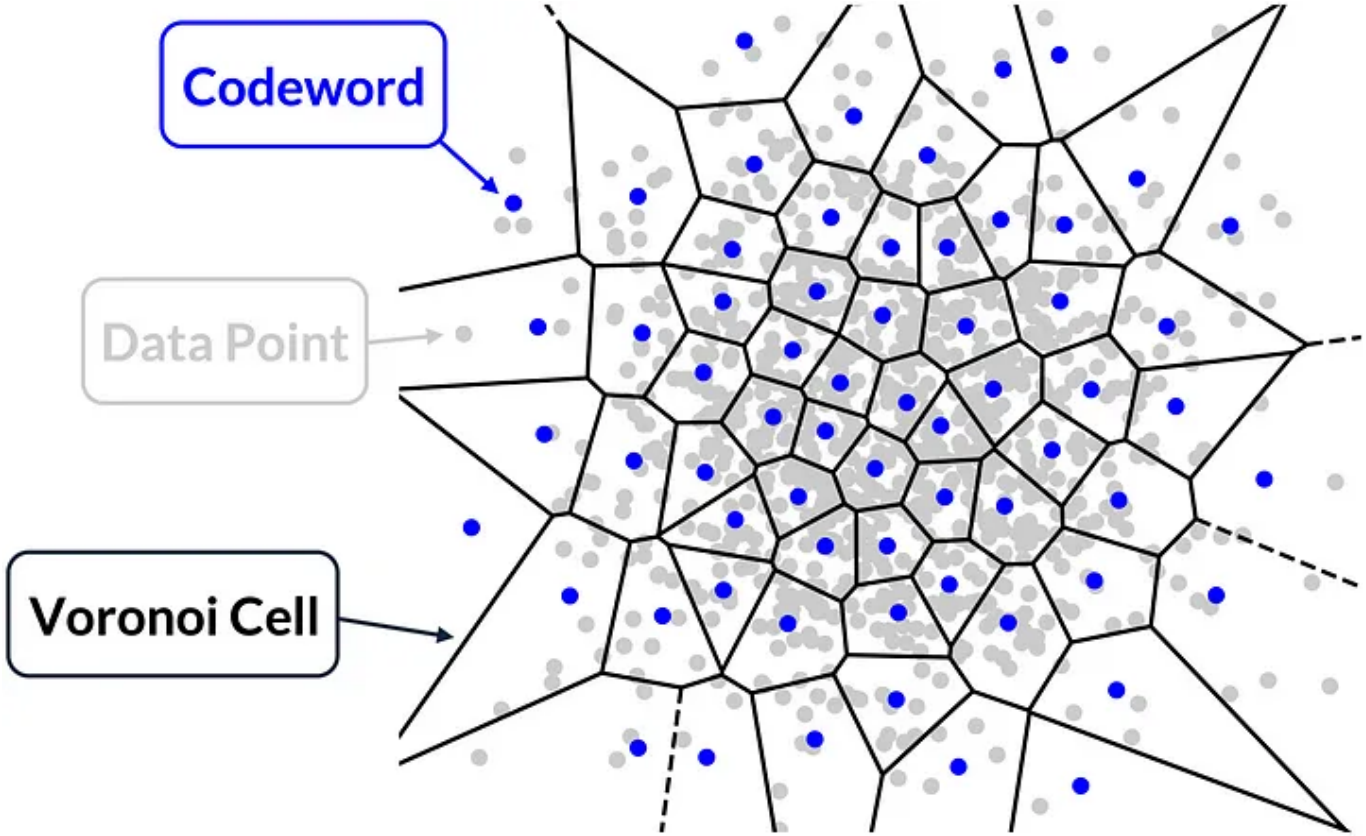


Photo by Alina Grubnyak on Unsplash

Vector Quantization

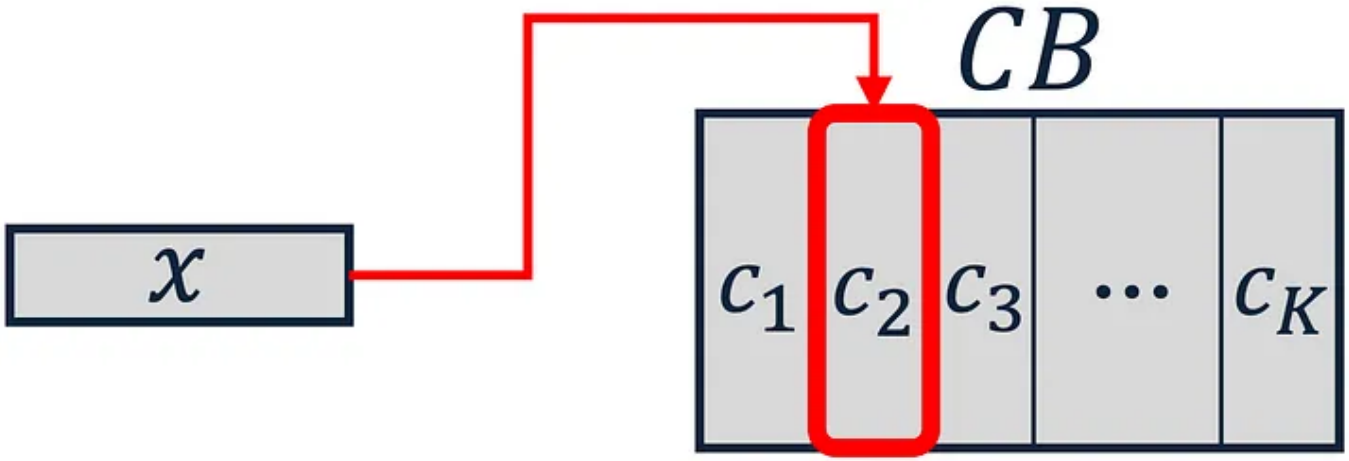
Vector quantization (VQ) is a data compression technique similar to k-means algorithm which can model any data distribution. Vector quantization has been used in a wide range of applications for speech, image, and video data, such as image generation [2], speech and audio coding [3], voice conversion [4,5], music generation [6], and text-to-speech synthesis [7,8]. The figure below shows how vector quantization (VQ) works. For VQ process, we

require a codebook which includes a number of codewords. Applying VQ on a data point (gray dots) means to map it to the closest codeword (blue dots), i.e. replace the value of data point with the closest codeword value. Each voronoi cell (black lines) contains one codeword such that all data points located in that cell will be mapped to that codeword, since it is the closest codeword to data points located in that voronoi cell.



Vector Quantization Operation (image by author)

In other words, vector quantization maps the input vector x to the closest codeword within the codebook (CB) using the following formula:

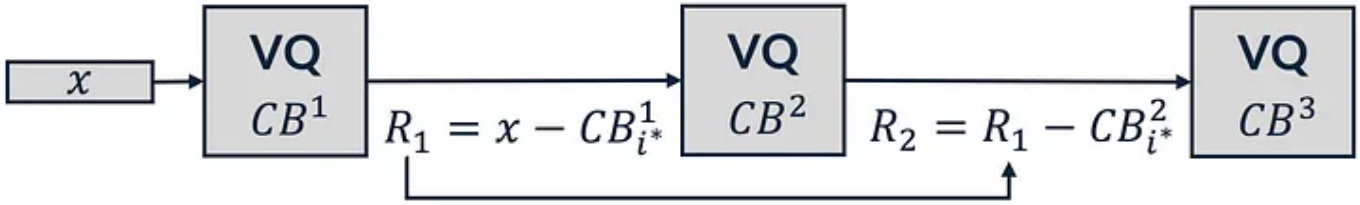


The computational complexity of VQ increases exponentially with the increase in the codebook size (increase in VQ bitrate). Hence, this plain form of VQ is applicable only for limited bitrates (limited codebook sizes). To solve this challenge and apply VQ for higher bitrates and higher dimensional data, we use some variants of VQ such as Residual VQ, Additive VQ, and Product

VQ. These methods considers more than one codebook to apply VQ on the data. We will explain these three VQ methods in the following.

Residual Vector Quantization (RVQ)

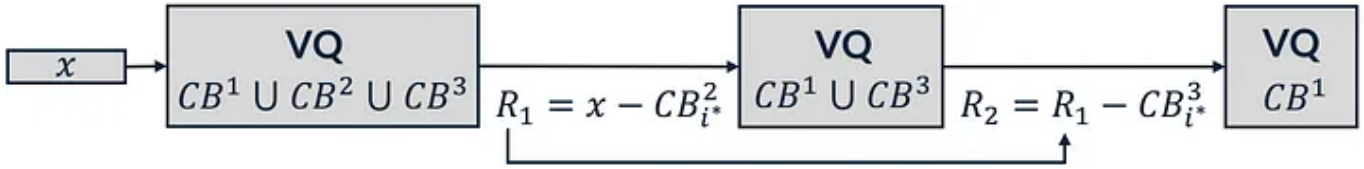
Residual VQ quantizes the input vector x by applying M consecutive VQ modules on it. According to the following figure, suppose M=3. We apply the first VQ module on input vector x using the first codebook (CB^1). Then, after finding the closest codeword form first codebook, we calculate the remainder (R1). Afterwards, we pass R1 as input to the next VQ module using the second codebook (CB^2). This process will continue for M stages where we would find three closest codeword coming from separate codebooks. At the end, we quantize the input vector x as a summation of M closest codewords.



Additive Vector Quantization (AVQ)

In a similar way as Residual VQ, Additive VQ quantizes the input vector x by applying M consecutive VQ modules. However, Additive VQ adopts the complex beam searching algorithm to find the closest codewords for the quantization process (you can find the details of beam searching algorithm in this paper [9]). According to the following figure, we suppose M=3. In Additive VQ, first we search for the closest codeword from the union of all three codebooks (here CB^1, CB^2, CB^3). Then, suppose we find the best codeword from CB^2. After that, we calculate the residual (R1) and pass it as input to the next VQ module. Since the first codeword is selected from CB^2, now we search for the closest codeword from the union of CB^1 and CB^3. After calculating the residual R2, we pass it as input to the last VQ module,

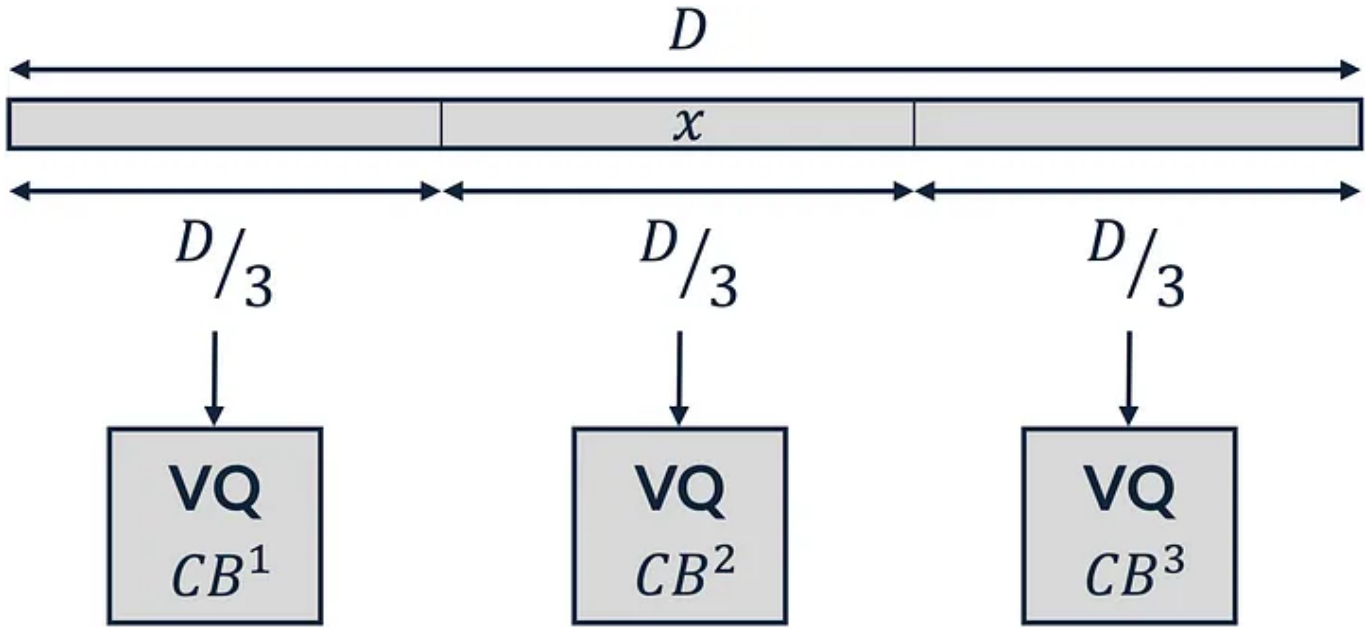
where we do the search using the last codebook (in this case CB^1) which is not yet contributed to the quantization process. At the end, we quantize the input vector x as a summation of M closest codewords.



Product Vector Quantization (PVQ)

Product VQ splits the input vector x of dimension D to M independent subspaces of dimension D/M. Then it applies M independent VQ modules to the existing subspaces. At the end, Product VQ quantizes the input vector x

as a concatenation of M closest codewords (one per each codebook). The figure below shows the Product VQ when M=3.



$$x_{quantized} = concatenate[CB^1_{i^*}, CB^2_{i^*}, CB^3_{i^*}]$$

Codebooks Optimization

Vector quantization (VQ) training means to optimize the codebook(s) such that they model the data distribution in a way that the error of quantization

(such as mean squared error) between data points and codebook elements is minimized. To optimize the codebooks for these three above-mentioned variants of VQ (Residual VQ, Additive VQ, and Product VQ) there are different approaches which we will mention in the following.

1. K-means Algorithm (traditional approach):

Based on the literature review, in most of the papers, codebooks for these three VQ methods have been optimized by k-means algorithm.

2. Stochastic Optimization (machine learning algorithms):

Machine learning optimization algorithms are based on gradient calculation. Therefore, it is impossible to optimize vector quantization methods using machine learning optimization, since the argmin function in vector quantization function (first equation above) is not differentiable. In other words, we cannot pass the gradients over vector quantization function in backpropagation. Here we have mentioned two solutions to solve this problem.

2.1. Straight Through Estimator (STE)

STE [10] solves the problem by simply copying the gradients intactly over VQ module in backpropagation. Hence, it does not consider the influence of vector quantization and leads to a mismatch between the gradients and true behavior of the VQ function.

vector quantization error is simulated by adding noise to the input vector, such that the simulated noise would gain the shape of original VQ error distribution (you can read shortly about NSVQ in this post).

NSVQ technique [11] has some advantages over STE method [10] which are listed in the following. 1) NSVQ yields more accurate gradients for VQ function. 2) NSVQ achieves faster convergence for VQ training (codebook optimization). 3) NSVQ does not need any additional hyper-parameter tuning for VQ training (does not require additional loss term for VQ training to be added to the global optimization loss function).

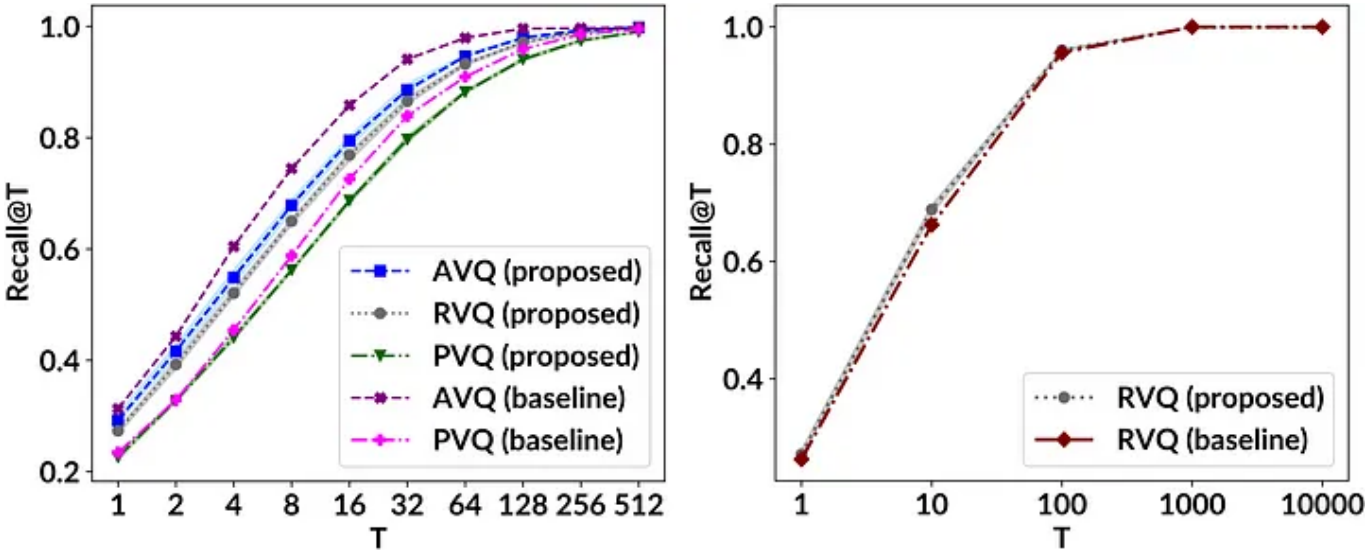
Experiments

In our paper, we have used our recently proposed NSVQ technique [11] to optimize three above-mentioned variants of VQ by machine learning optimization. To evaluate the performance of these three VQ methods and study the trade-offs between accuracy, bitrate, and complexity of them, we conducted four different scenarios for experiments. We will explain all these scenarios of experiments in the following.

1. Approximate Nearest Neighbor (ANN) Search

In this experiment, we modeled the distribution of SIFT1M dataset [12] (128-D image descriptors) by training three VQ methods on its learning set. The SIFT1M image descriptors dataset [12] includes 10^6 base vectors, 10^5 learning

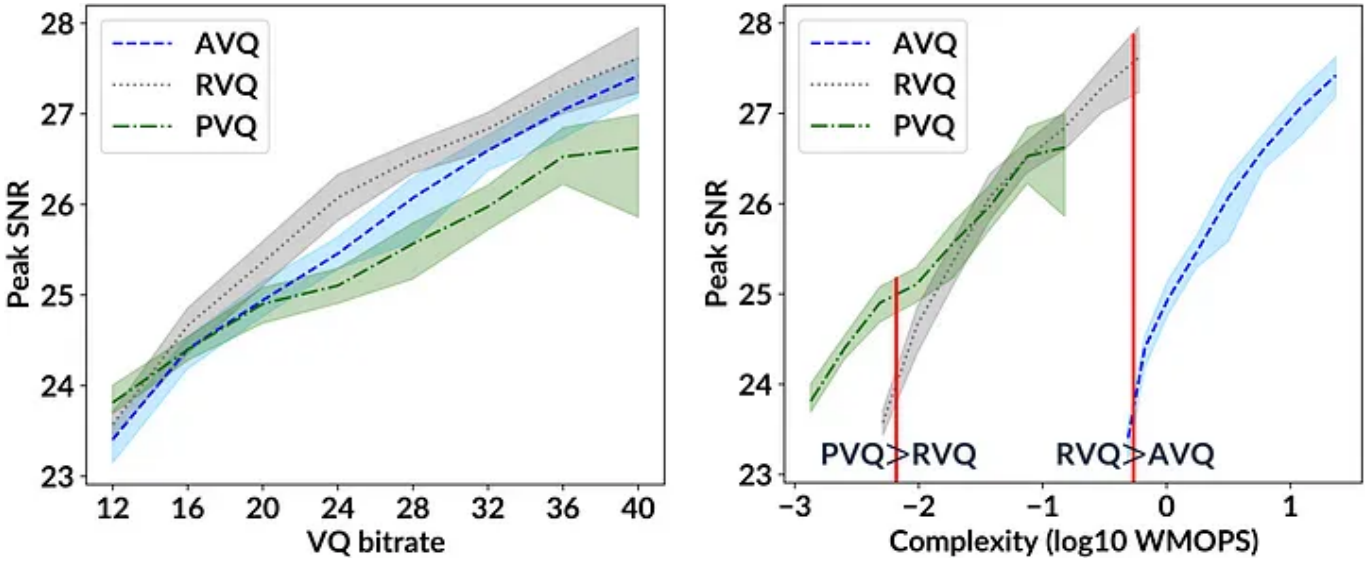
vectors, and 10^4 query vectors for testing purposes. The ground truth contains the set of actual nearest neighbors, from the base vectors to the query vectors. In the ANN search, we first compress the base vectors using the corresponding learnt codebooks trained on the learning set. Then, for each query vector, we find the approximate nearest neighbors from the compressed base vectors by performing an exhaustive search. To assess the quality of data compression, we calculate the *recall metric* at different values for parameter T, which shows whether the actual nearest neighbor (from groundtruth) exists in the first T computed nearest neighbors. The figure below illustrates the comparison of three variants of VQ optimized by our proposed NSVQ technique with the baseline methods under *recall metric*. In general, all three machine learning-based optimized VQ methods achieve comparable (even slightly better in case of RVQ) recall values to the baselines.



Comparison of recall values for compression of SIFT1M dataset by applying our proposed VQ methods and baselines at 64 bits (8 codebooks each with 256 codewords); Recall@T shows whether the actual nearest neighbor (from groundtruth) exists in the T computed nearest neighbors. (image by author)

2. Image Compression using VQ-VAE

In this experiment, we trained a vector quantized variational autoencoder (VQ-VAE) on the training set of CIFAR10 dataset to compress it. To apply the vector quantization in the bottleneck of VQ-VAE, we used each of these three VQ methods. After training, we reconstructed the test images of CIFAR10 using the trained encoder, decoder, and learnt codebooks for each VQ method. To evaluate the quality of reconstructed images, we employ Peak Signal to Noise Ratio (Peak SNR) metric. In addition, we computed the complexity of each VQ method using Weighted Million Operations Per Second (WMOPS) metric, which is under ITU-T standard. The following figure shows the results of this experiment.

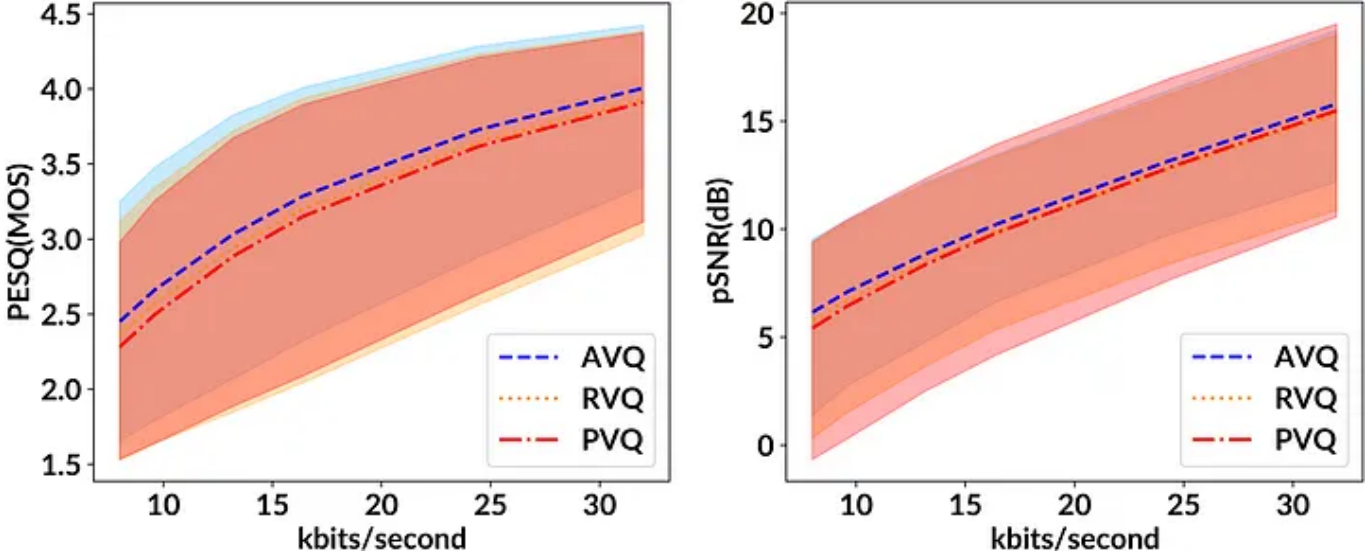


Peak SNR and complexity of proposed VQ methods over 15 k training batches and 10 individual experiments in the image compression scenario; lines refer to the mean values and the corresponding filled areas refer to their 95 % quantiles. For all VQ bitrates we used four codebooks i.e. M=4. (image by author)

According to the complexity figure (in the right), we found that for the same use of computational resources (left vertical red line) and a higher bitrate, Product VQ performs better than Residual VQ. In addition, for the same use of computational resources (right vertical red line) and a higher bitrate, Residual VQ performs better than Additive VQ. Therefore, depending on how much computational resources are available, we can conclude which is the best VQ method to use.

3. Speech Coding

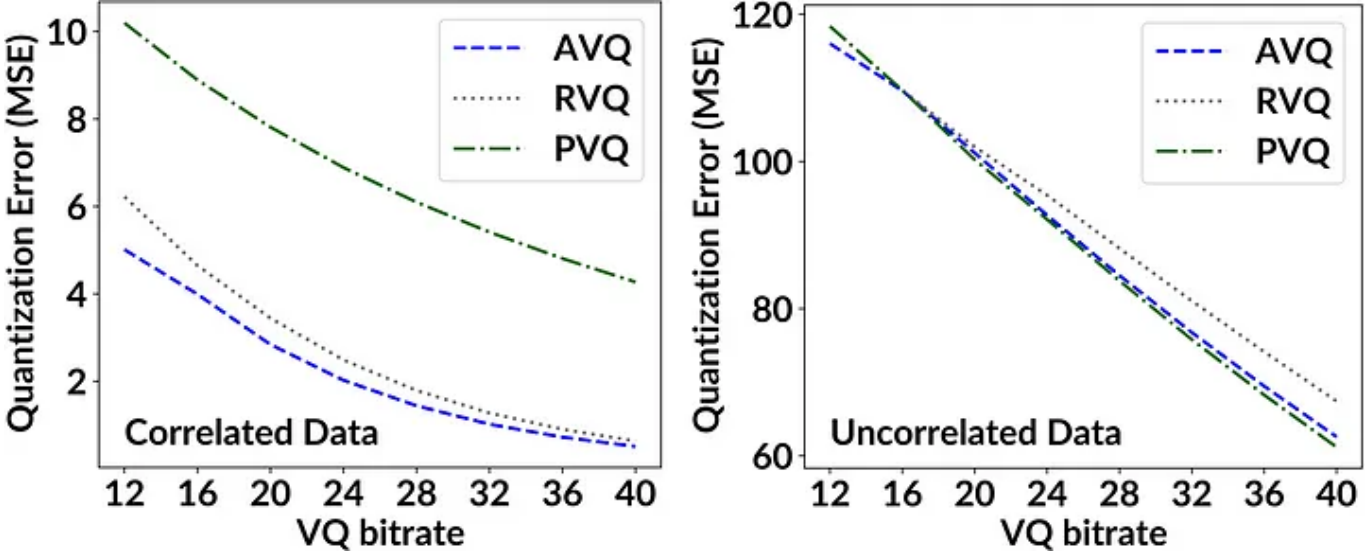
In this experiment, we model the spectral envelope of speech signals by three VQ methods using the speech codec presented in [13]. To evaluate the quality of decoded speech signals, we used perceptual evaluation of speech quality (PESQ) and perceptually weighted signal to noise ratio (pSNR) as objective metrics. The following figure shows the performance of all three VQ methods under PESQ and pSNR criteria. According to the results, we observe that Additive VQ gains higher mean and lower variance than both Residual VQ and Product VQ in both metrics.



Performance of proposed VQ methods in terms of PESQ and pSNR metrics for 16 bit VQ (with 4 codebooks i.e. M=4) at overall bitrates of 8, 9.6, 13.2, 16.4, 24.4 and 32 kbit/s in the speech coding scenario; solid lines refer to the mean values of PESQ and pSNR, and the corresponding filled areas refer to their 95% quantiles. (image by author)

4. Toy Examples

In this experiment, we intend to compare the performance of three VQ methods with respect to the correlation in the data. Hence, we prepared two correlated and uncorrelated datasets of dimension 64. Then, we compressed these datasets using these three VQ methods. To evaluate the performance, we computed the mean squared error (MSE) between each dataset and its quantized version. The following figure shows the results for this experiment.



Vector quantization error for correlated and uncorrelated datasets using all three proposed VQ methods (data dimension=64, and for all VQ bitrates we used four codebooks i.e. M=4). (image by author)

In correlated dataset, since Residual VQ and Additive VQ take the correlation among all data dimensions into account, they have much lower quantization error than Product VQ as expected. On the other hand, Product VQ has better performance than Additive VQ and Residual VQ for uncorrelated data, since there is no correlation among data dimensions and that is exactly what Product VQ presumes.

Conclusions

Using variants of Vector Quantization (VQ) such as Residual VQ, Additive VQ, and Product VQ allows to apply VQ for high bitrates and high dimensional data. These VQ methods have been optimized by classic expectation maximization and k-menas algorithm so far. In this paper, we optimize these

VQ methods by machine learning optimization using our recently proposed Noise Substitution in Vector Quantization (NSVQ) [11] technique. In addition, NSVQ allows end-to-end optimization of VQ methods in Neural Netwroks. We also study the trade-offs between bitrate, accuracy, and complexity of these three VQ methods. Hence, our open-source implementation [14] helps to make the best choice of VQ method for a particular use-case.

GitHub Repository

We provide the PyTorch implementation of these VQ methods in the following webpage.

Acknowledgement

Special thanks to my doctoral program supervisor Prof. Tom Bäckström, who supported me and was the other contributor for this work.

References

[1] M. H. Vali and T. Bäckström, “Stochastic Optimization of Vector Quantization Methods in Application to Speech and Image Processing,” in *Proceedings of ICASSP*, 2023.

[2] A. Razavi, A. van den Oord, and O. Vinyals, “Generating diverse high-fidelity images with VQ-VAE-2,” in *Proceedings of NeurIPS*, 2019.

[3] C. Gârbacea, A. van 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.

[4] B. van Niekerk, L. Nortje, and H. Kamper, “Vector-quantized neural networks for acoustic unit discovery in the zerospeech 2020 challenge,” in *Proceedings of Interspeech*, 2020.

[5] S. Ding and R. Gutierrez-Osuna, “Group latent embedding for vector quantized variational autoencoder in non-parallel voice conversion,” in *Proceedings of Interspeech*, 2019.

[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] 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.

[8] X. Wang, S. Takaki, J. Yamagishi, S. King, and K. Tokuda, "A vector quantized variational autoencoder (VQ-VAE) autoregressive neural F0 model for statistical parametric speech synthesis," *IEEE Transactions on Audio, Speech, and Language Processing*, 2020.

[9] A. Babenko and V. Lempitsky, "Additive quantization for extreme vector compression," in *Proceedings of CVPR*, 2014.

[10] Y. Bengio, N. Léonard, and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," a*rXiv preprint arXiv:1308.3432,* 2013.

[11] M. H. Vali and T. Bäckström, "NSVQ: Noise Substitution in Vector Quantization for Machine Learning," *IEEE Access*, vol. 10, 2022.

[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, “End-to-End Optimized Multi-Stage Vector Quantization of Spectral Envelopes for Speech and Audio Coding,” in *Proceedings of Interspeech*, 2021.

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

Vector Quantization | Machine Learning | Data Compression | Nearest Neighbor Search | Vq Vae

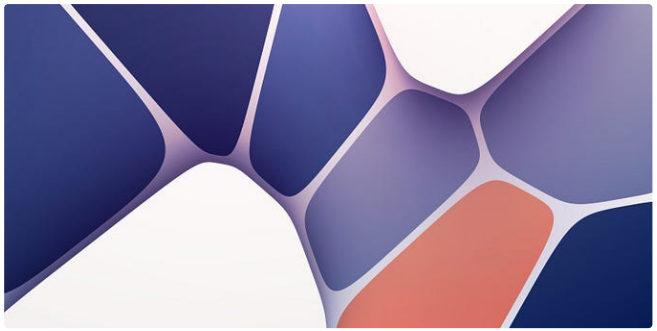
Written by Mohammad Hassan Vali

24 Followers · Writer for Towards Data Science

Doctoral candidate at Speech Interaction Technology team, Aalto University, Finland.

Follow

More from Mohammad Hassan Vali and Towards Data Science



Mohammad Hassan Vali in Towards Data Science

NSVQ: Improved Vector Quantization technique for Neura...

Efficient vector quantization for machine learning optimizations (eps. vector quantize...

8 min read · Jul 20, 2022

48

Iulia Brezeanu in Towards Data Science

How to Cut RAG Costs by 80% Using Prompt Compression

Accelerating Inference With Prompt Compression

✦ · 11 min read · Jan 4

1K 6

Siavash Yasini in Towards Data Science

How to Write Memory-Efficient Classes in Python

Three tricks to prevent your data project from memory overflow

✦ · 7 min read · Jan 13

1.1K 10

Natassha Selvaraj in Towards Data Science

ChatGPT for Data Analysis — A Beginner's Guide

An complete tutorial on using ChatGPT for data analysis.

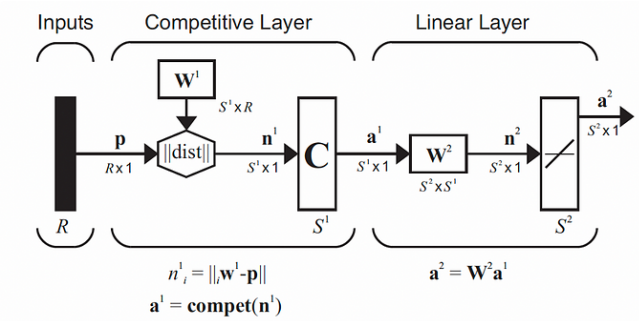
✦ · 12 min read · Dec 23, 2023

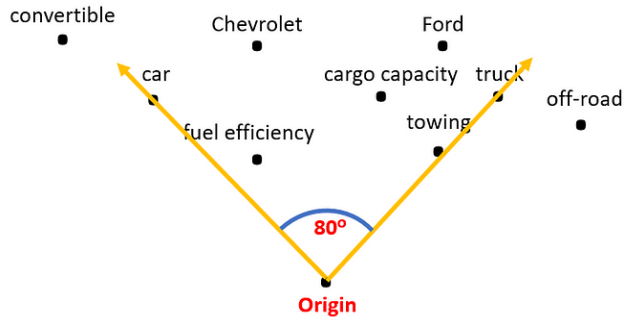
775 9

See all from Mohammad Hassan Vali

See all from Towards Data Science

Recommended from Medium





Udbhav Kush

Learning Vector Quantization (LVQ): A Step-by-Step Guide with...

In the realm of machine learning and pattern recognition, there exists a powerful yet often...

7 min read · Aug 28, 2023

32 4

Arjun Prakash

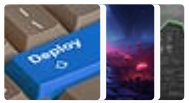
Understanding Cosine Similarity: A key concept in data science

Measuring similarity between 2 vectors

5 min read · Sep 21, 2023

138 1

Lists



Predictive Modeling w/ Python

20 stories · 814 saves

Practical Guides to Machine Learning

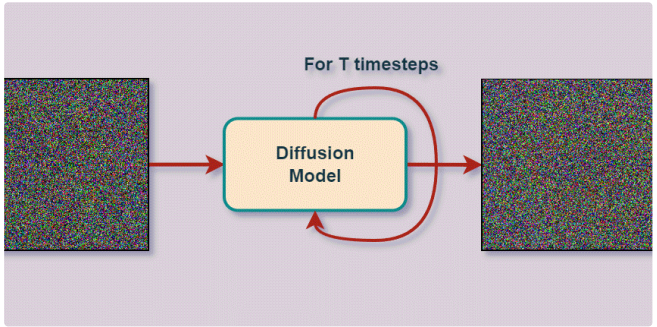
10 stories · 947 saves

Natural Language Processing

1112 stories · 587 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 276 saves



Gabriel Mongaras in Better Programming

Diffusion Models—DDPMs, DDIMs, and Classifier Free Guidance

A guide to the evolution of diffusion models from DDPMs to Classifier Free guidance

28 min read · Mar 13, 2023

616 8

Top Trends World

Transforming Text into Knowledge: Building Conceptual Graphs with...

This sophisticated task blends natural language processing (NLP) and knowledge...

4 min read · Jan 4

60 1

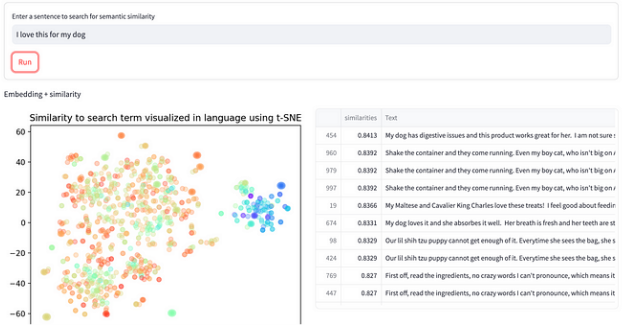
Aguimar Neto

What is Latent Diffusion in AI?

Latent diffusion models are deep learning models that have recently emerged as a...

5 min read · Oct 7, 2023

2



Daniel Avila in LatinXinAI

Visualizing embeddings and semantic similarity with OpenAI...

In this article, we will explore an example of visualizing semantic similarities in language...

4 min read · Aug 7, 2023

46 1

See more recommendations