Re-implementing VQ-VAE - Neural Discrete Representation Learning

Vancheeswaran Vaidyanathan
Electrical and Computer Engineering
A59011835

Shrivaths Shyam
Electrical and Computer Engineering
A59019544

Abstract

The main idea of this project is to re-implement VQ-VAE as discussed in the paper "Neural Discrete Representational Learning" by Aaron van den Oord and et. al. We plan to use Pytorch and the CelebA dataset for implementing the Vector Quantised - Variational AutoEncoder (VQ-VAE).

1 Introduction

Generative models have been gaining a lot of attention over the recent years. They have a widespread range of applications such as Data augmentation, art creation, text generation, audio synthesis, dataset synthesis, et cetera. GANs [Generative Adversarial Networks] and VAEs [Variational AutoEncoders] are the most prominent techniques used in Generative Modelling. GANs are highly unstable while training and suffer from mode collapse in several instances. VAEs too have their fair share of issues, one of the most important one being posterior collapse. There has been a lot of research over the years to overcome these issues with generative models by making changes in the architecture. One such variation of VAEs is VQ-VAE. This method tries to encode the data in a discrete manner rather than a continuous feature representation. It uses the principle of Vector Quantisation to avoid the issue of Posterior Collapse. We intend to implement the VQ-VAE model with the CelebA dataset using Pytorch.

2 Related Work

According to [2], GANs are unstable due to mode collapses shown via various theorems and inferences, while VAEs are unstable due to posterior collapse. In order to overcome such issues that are faced by GANs and VAEs, [1] proposes VQ-VAE, a generative model that learns useful representations without supervision. The model differs from VAEs by using discrete codes instead of continuous codes and a learned prior instead of a static one. The model uses vector quantization to learn a discrete latent representation, which helps avoid the issue of posterior collapse commonly seen in VAEs. The model can generate high-quality images, videos, and speech which shows the usefulness of the learned representations. Further, [4] gives an improved VQ-VAE-2 algorithm, where the encoder and decoder have feed-forward networks which improves the speed of the model and it samples an autoregressive model only in the compressed latent space, which is an order of magnitude faster than sampling in the pixel space.

3 Method

VQ-VAE like any other VAE has two major components in the implementation: the encoder part and the decoder part.

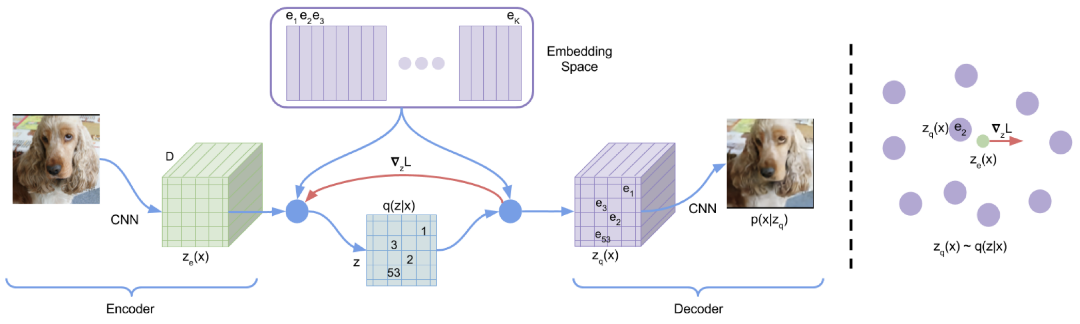

Figure 1: Implementation of VQ-VAE [Source:"Neural Discrete Representational Learning" by Aaron van den Oord and et. al.]

VQ-VAE adds an Embedding Space (which is something like a code-book) to a traditional autoencoder.

$$z_q(x) = e_k, \quad \text{where} \quad k = argmin_j ||z_e(x) - e_j||_2 \tag{1}$$

Here, $z_e(x)$ is the encoder output. The embedding space has e_k vectors that represents various images. Therefore, $z_q(x)$ is basically a quantised vector that happens to be closest to $z_e(x)$ when we use the L2 norm/euclidean distance as the metric. The decoder is then tasked with reconstructing the input from this quantized vector.

The embedding space vectors are learned using gradient descent. The loss function for this is governed by the following equation:

$$L = \text{log}p(x|z_q(x)) + ||\text{sg}[z_e(x)] - e||_2^2 + \beta||z_e(x) - \text{sg}[e]||_2^2 \tag{2}$$

where sg stands for the stop gradient operator. It is defined as identity at forward computation time and has zero partial derivatives. Therefore, it effectively constrains its operand to be a constant that is not updated.

- The decoder optimises the first loss term only,
- The encoder optimises the first and the third (last) loss terms, and
- The embeddings are optimised by the second (middle) loss term.

This is how the VQ-VAE model is learned.

Further, the metrics used to measure the performance of the algorithm were the loss which is calculated as mentioned above and another metric called as perplexity. Perplexity is used to evaluate the quality of the model's latent space representation of the data. The latent space is a compressed representation of the input data, obtained by encoding it using the VQ-VAE encoder. Perplexity measures how well the model can reconstruct the original data from the compressed latent space.

The code-book/Embedding space can be trained based on the loss function that was described earlier or through moving averages as described in the paper in Appendix A.1.

4 Experiments

- Dataset:
  - We have used the CelebA data set to implement the VQ-VAE model
  - The dataset contains 202,599 images of 10,177 celebrities with each image of dimension 178x218. The images are annotated with 40 binary labels indicating various facial features.

- Firstly, we chose 55000 resized images of the entire dataset for tuning the hyper-parameters where the image size we used was 64x64 instead of the original dimension of the images provided by the dataset. Further, we split the chosen images into train, validation and test sets. The reason we used a smaller subset of the dataset was due to the long training time if we used the entire dataset.

- PixelCNN was used in [1] as the encoder and decoder networks while we used a model based on Residual Network for encoding and decoding in the model.

- The experiments were performed on the parameters such as β (commitment loss), learning rate and number of embeddings in the latent space. The values that we used for commitment cost were 0.25, 0.75, 1.25, 1.75, values for learning rates were 10^{-3}, 10^{-4} and 10^{-5} while the number of embeddings were varied from 128, 256 to 512. All experiments were run for 10000 iterations to obtain tangible results.

- Then, we plotted the graphs of the loss versus iterations and perplexity versus iterations. Loss is calculated using the equation (2) where the second term is optimized by the vector quantizer. Perplexity is a measure of how well the model can predict the discrete codes used to represent the input data.

- Finally the model, trained with the best parameters chosen based on the graphs obtained earlier, was trained with the entire dataset. The reconstructed images and the graphs were plotted to visualize the performance of the model.

5 Results

There were many observations from the graphs that were obtained based on the hyper-parameter tuning for the VQ-VAE model. These are described below

- Firstly, as the learning rate is decreased, the perplexity of the algorithm requires more iterations to reach the saturation level. This can be noticed from Figure 7 and Figure 8 below. The error also takes more time to reach the minimum error that is possible for the particular hyper-parameters.

- Secondly, as the number of embeddings increased, it also took more iterations to reach the best perplexity that the model can reach. As we can see from Figure 2, Figure 3 and Figure 4 the number of embeddings increased from 128 to 256 to 512, the number of iterations needed to reach the highest perplexity increased from approximately 2000 iterations to 5500 iterations to 6000 iterations respectively.

- Finally, as the commitment cost increased from 0.25 to all the way 1.75 the number of iterations taken by the model to reach the best perplexity and the lowest error possible also increased.

Based on the observations given above, we can conclude that the best hyper-parameter combinations that were tried is the one with 512 embedding, commitment cost at 1.25 and the learning rate at 10^{-3} where the perplexity reached over 400 and the error drops to almost 10^{-3} roughly at the 4000 iteration mark.

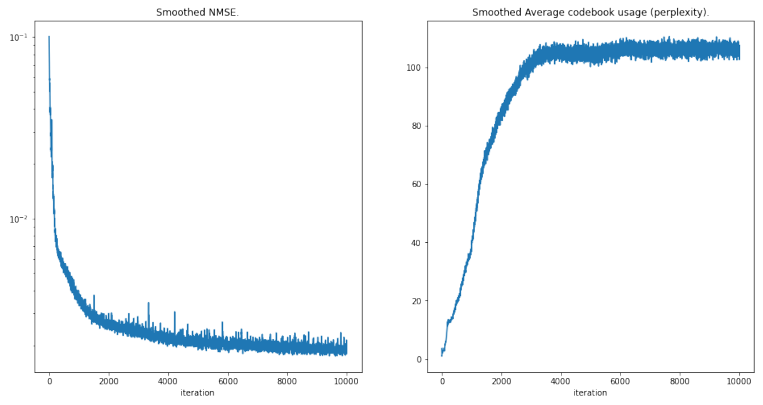

Figure 2: Emb=128, β=1.25, LR=10^{-3}

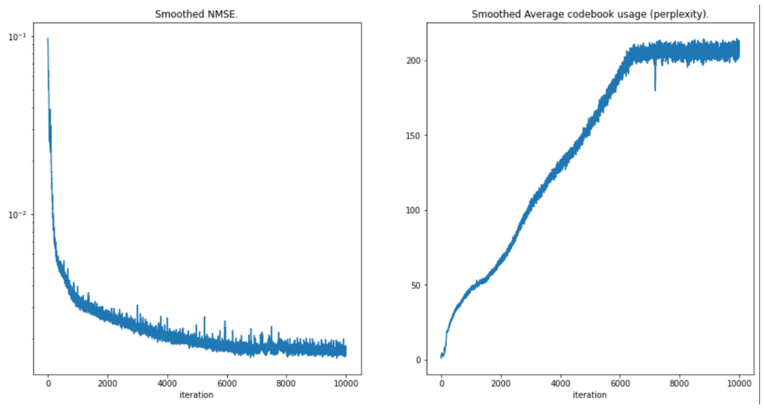

Figure 3: Emb=256, β=1.25, LR=10^{-3}

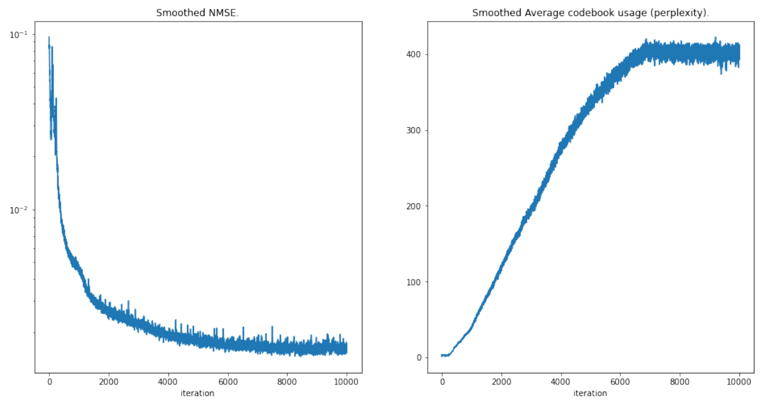

Figure 4: Emb=512, β=0.25, LR=10^{-3}

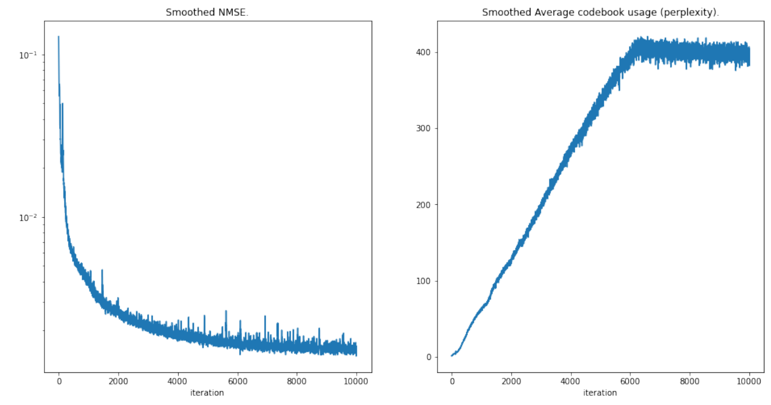

Figure 5: Emb=512, β=0.75, LR=10^{-3}

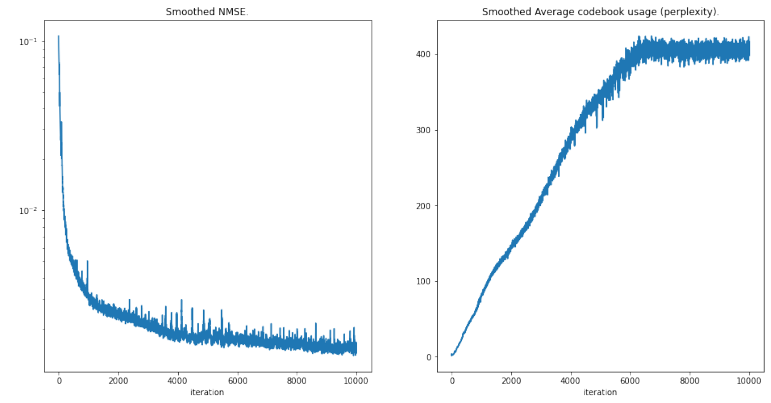

Figure 6: Emb=512, β=1.25, LR=10^{-3}

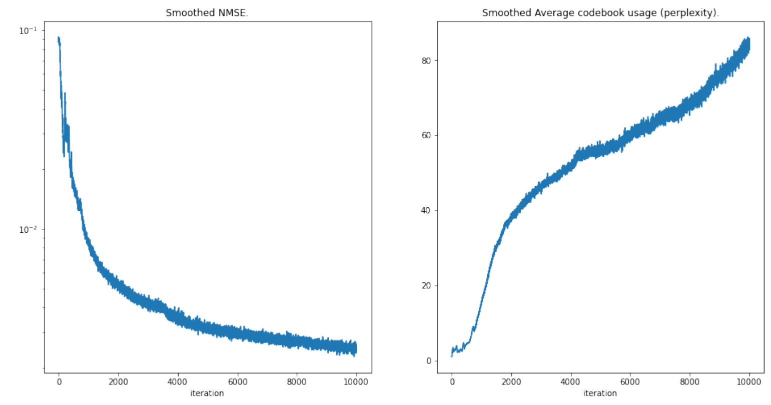

Figure 7: Emb=512, β=1.25, LR=10^{-4}

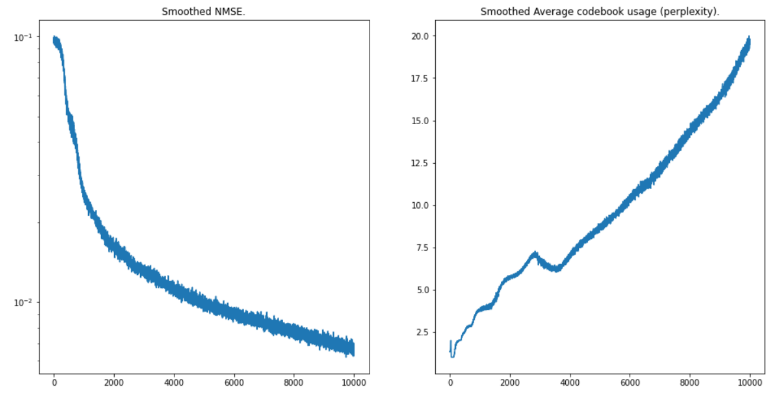

Figure 8: Emb=512, β=1.25, LR=10^{-5}

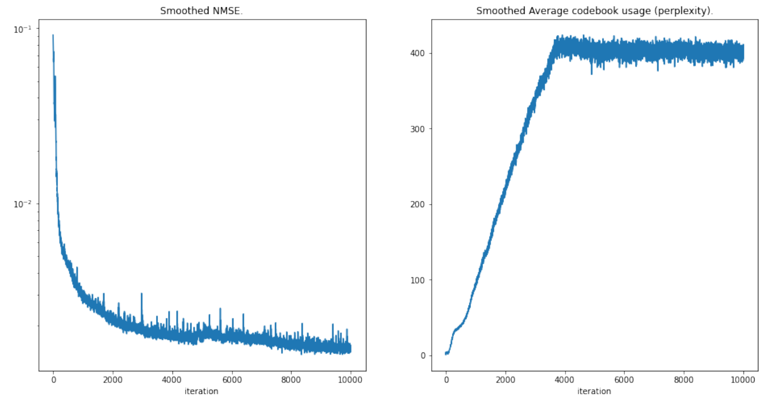

Figure 9: Emb=512, β=1.75, LR=10^{-3}

Now we run the model using the entire dataset with the best hyper-parameters that we have obtained using the smaller batch of the CelebA dataset with commitment cost at 1.25 and the learning rate at 10^{-3} where the perplexity reached over 400 and the error drops to almost 10^{-3}. The reconstructed images was compared with the original images. The graphs also show that the model performed well for the hyper-parameters chosen from the experimentation performed above.

Figure 10: Original dataset - 64x64 resolutions

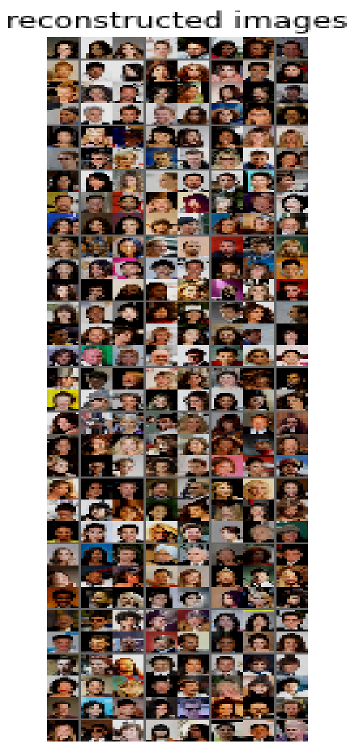

Figure 11: Reconstructed 64x64 Images

By comparing the original images (Figure 10) and reconstructed images (Figure 11) above, we can see that the model is able to generate the images that almost resemble each other. Further we have plotted the error and the perplexity graphs (Figure 12) and it coincides with the plots that we obtained for the best parameters during the hyper-parameter tuning of the model.

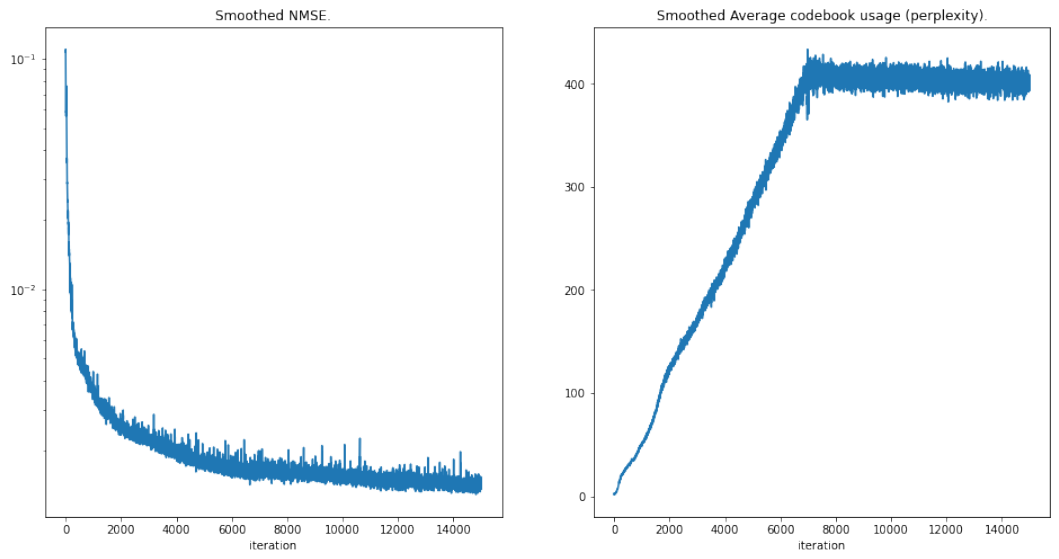

Figure 12: Error and perplexity plots for best hyper-parameters on entire dataset

6 Supplementary Material

Demo of the implementation can be found here: https://bit.ly/3FFF6RC
The code can be found in the following repository: https://github.com/vancheeswaran/VQ-VAE

7 Contributions

The files main.py, model.py, residual.py and encoder.py were contributed by Shrivaths Shyam and the remaining files (dataloader.py, decoder.py, test.py, vqvae.py) were contributed by Vancheeswaran Vaidyanathan.

References

[1] Aaron van den Oord, Oriol Vinyals, Koray Kavukcuoglu, "Neural discrete representation learning", in Advances in neural information processing systems, Volume 30, 2017

[2] Martin Arjovsky, Léon Bottou "Towards Principled Methods for Training Generative Adversarial Networks", arXiv preprint arXiv:1701.04862

[3] S. Yang, P. Luo, C. C. Loy, and X. Tang, "From Facial Parts Responses to Face Detection: A Deep Learning Approach", in IEEE International Conference on Computer Vision (ICCV), Pages 3676-3684, 2015

[4] Ali Razavi, Aaron van den Oord, Oriol Vinyals, "Generating Diverse High-Fidelity Images with VQ-VAE-2", arXiv:1906.00446v1 [cs.LG]