Final Project

Deep Learning

Nicolas Vila

1. Introduction and Goal.

VQ-VAEs have grown in popularity in the last few years given the splendid results they have proven in different generating processes. They are mainly based in an encoder-decoder architecture where the latent variables are discretized. Using a discrete space is a more compact way of coding latent variables and it seems reasonable to do so because in real life objects have underlying discrete representations. Nevertheless, discretizing the underlying latent space using a finite codebook as done in VQ-VAE has some limitations when generating new images given that they can only properly represent the underlying structure of the image but misses most of the details. Moreover, sampling from such latent space, will result in a finite set of generated images. To try tackling all these issues, I wondered if there was a possible encoding of the latent space such that it maintains VQ VAEs' discrete latent properties while combining them with the continuous latents given by the encoder architecture.This, may convert the latent space into a more varied and true to reality one. Nevertheless, I haven't found any research trying to use both discrete and continuous latent variables to try to make the generating process better. In this report I study how such a combination could be done and if that's the case, if it results in a better model.

2. Methods.

Firstly, I have to note that all the notation used here, unless explicitly introduced by myself, is the exact same notation used in [1].

2.1 Projected-Latent

In this first part of the extension I try to combine the discrete and continuous latents by projecting the continuous channels over the discrete channels. The intuition behind it is the following. In order to obtain latents that are meaningful, using discrete codes, we need them to be at least close to the discrete codebook space but not exactly equal to them. To do so, and obtain a more variate but still meaningful latent, I projected each pixel's vector in $z_e(x)$ to $z_q(x)$. I also added an additional loss term that brings the projection towards the discrete latents.

$$\overline{z}_q(x) = P(z_e(x), z_q(x)) \ ; \ L = log(p(x|z_q(x)) + ||sg[z_q(x)] - z_e(x)||_2^2 + \beta||sg[z_e(x)] - z_q(x)||_2^2 + ||\overline{z}_q(x) - sg[z_q(x)]||_2^2$$

Where P(a, b) is the projection of a to b. The additional term in the loss function makes sure that the new latent is close to the discrete codes.

2.2 Frequency-Latent

In this second part of the extension, inspired by a discussion with professor Hongyuan Mei, we could argue that the latent space of the image could work like a Huffman Coding, where the more frequent codes are shorter and the less frequent ones are larger. In an analogy to this work, I proposed an adapted model of VQ-VAES where a frequency table $F(z)$ of each encoding is calculated. This table is used then to find the S most frequent codings in the discrete latents. Then, using $q(x|z)$ defined as in the paper, we can obtain the positions in the discrete latent space where the most frequent codes are used. Using such positions, a mask $M(x|z)$ of the same dimensions as $z_q(x)$ is created to keep only those positions with the most frequent embeddings. An equivalent mask $C(x|z) = 1 - M(x|z)$ is used to keep exactly the opposite elements. The final latent variable is computed as following:

$$\overline{z_q}(x) = M(x|z) * z_q(x) + C(x|z) * z_e(x)$$

The idea is that most of $\overline{z_q}(x)$ will be like $z_q(x)$ but some portion of it won't.

3. Results:

I will divide this section into three parts. Firstly I will analyze each method's results independently and then I will compare them with the original VQ-VAE model. From each method, I'll analyze its loss and the reconstructed images.

Another important remark, regarding the model architecture used along this study is that the parameter's values such as the embedding dimensions, latent dimensions, learning rate, batch size, etc, are used as in the code in https://github.com/ritheshkumar95/pytorch-vqvae. Moreover, the encoder, decoder and prior distribution models are also as described in the aforementioned code.

3.1 Projected Latents' Results

This model yielded surprising results. The first thing I noted using this model is that while training, the loss was unstable, we can see high peaks in the last steps of the process in the following figures. This model achieved lower reconstruction losses both during training and testing, which is great. The biggest downside I found when using this model is that the latent space was too large to model using a PixelCNN, at least using Google's GPUs. In comparison, VQ-VAEs discrete latents can be represented as indices to a codebook at each pixel. Therefore, we just have to model those indices distribution over the image. In this modification, that is not the case and therefore I was unable to train a prior model. However I believe that models with a transformer-like architecture could model the larger latent space distribution defined here.

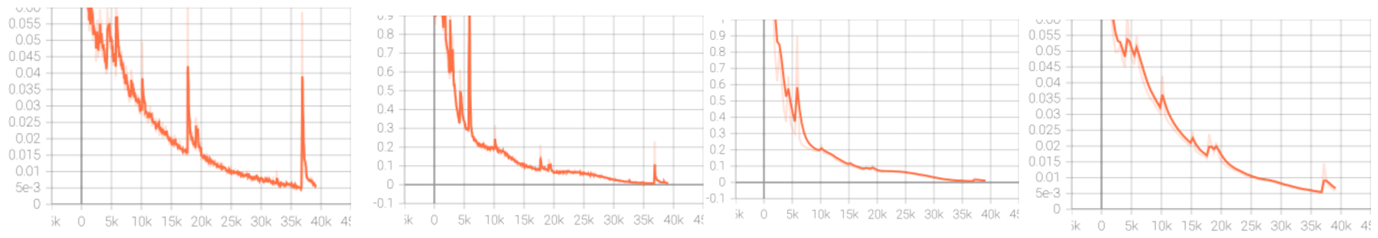

Figure 1. From left to right: test loss quantization, test loss reconstruction, train loss quantization, train loss reconstruction of the Projected-Latent model

We can see that the reconstruction images are very impressive. They look surprisingly well and reproduce almost perfectly the original image.

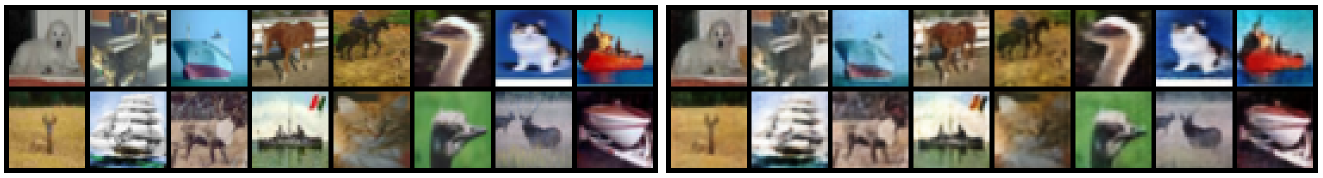

Figure 2. Original images in the left, reconstructions from Projected-Latents model in the right.

3.1 Frequency latents' Results

To test this model, I used values S from 512 (number of codes in the codebook of VQ-VAE), to 502, 412 and 312. In general terms, it can be argued that even though at first, reconstruction loss seemed to increase as S got lower, I found that for S=312 the loss was similar to the VQ-VAE. Therefore, no clear generalization could be made. It would be great to study in depth for a different amount of S how the model's loss would behave, to build a more continuous graph of the evolution of the error with respect to S. In this report, I haven't tested a bigger amount of S values because of the limitations of Google's GPUs usage. As in the previous section, PixelCNNs prior on discrete latents couldn't be used to train a model over the latent space given that with the new modification, the latent space was much larger and it wasn't about just modeling indices of a codebook in every pixel of the image. Given that in some pixels the latent distribution was continuous, the PixelCNN couldn't be used to sample new images.

The reconstructed images looked qualitatively similar to the VQ-VAES. Note that when S = 512, the model is exactly the VQ-VAE.

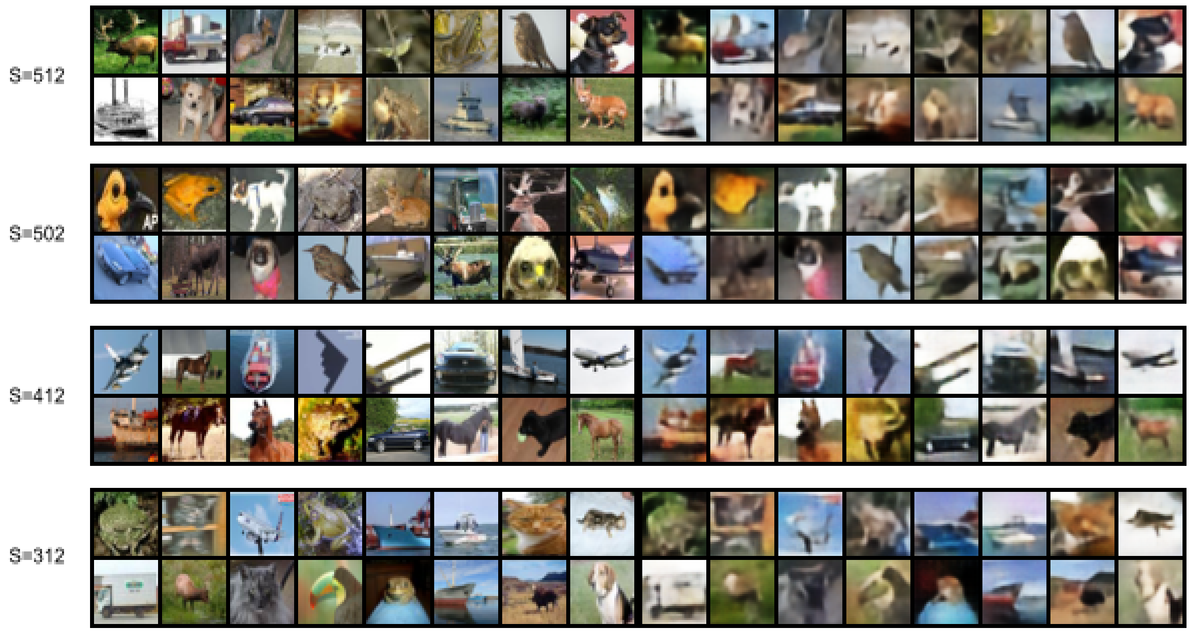

Figure 3. Original images in the left, reconstructions from Frequency-Latent model in the right, for different S parameters.

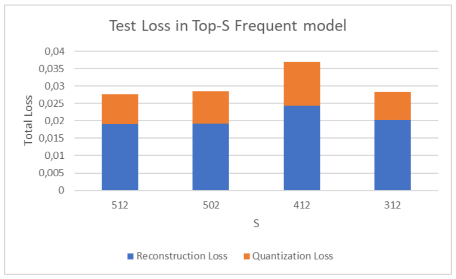

Figure 4. Loss comparison for different S parameters in Frequency-Latent model

3.2 General Remarks

The results presented here showed that the better model was the Projected-Latent model, which obtained the lowest reconstruction error. However, we saw that its learning curve was very unstable and had seemingly random peaks at later stages of training. The Frequency-Model as implemented in this report didn't show any convincing results. However, the study of such observations in the learning curve of the Projected-Latents model, as well as different variations of the Frequency-Latent model that I didn't have time to try is left as future research. An interesting variation combining both models would be the following: instead of directly using the continuous variables in those positions with less frequent codes, one could project the continuous latent into the discrete latents of such positions, which may lay out a more stable and robust model during the learning stage. Such model would be computed as following:

$$\overline{z}_q(x) = M(x|z) * z_q(x) + C(x|z) * P(z_e(x), z_q(x))$$

Furthermore, one could try different modifications of the loss function to try to make these hybrid latent spaces better. Finally, it would also be interesting to train the latent distribution using a model like a transformer, to observe new generated images.

4. Conclusions

We have seen that combining continuous latents with discrete latents is a feasible option when building generative models. However in that combination, discrete latents must be an important part given that they represent the fundamental underlying structure of objects. Nevertheless, continuous latents help to make the resulting hybrid latent space more significant and representative of the underlying structures of objects, which may be very diverse. The findings in this report indicate that in-depth research in this aspect could improve generative models, which are not only applicable to image generation, but any kind of generating processes.

In the particular case of the models described here, I found that the Projected-Latent model performed better than the VQ-VAE. However, it is also true that the model was unstable and it is possible that in some cases it may not even converge to the loss values that I found here. However, different ways to make this process more robust can be studied. Every trained model, as well as the logs of each model's learning process and the modified code files can be found here.

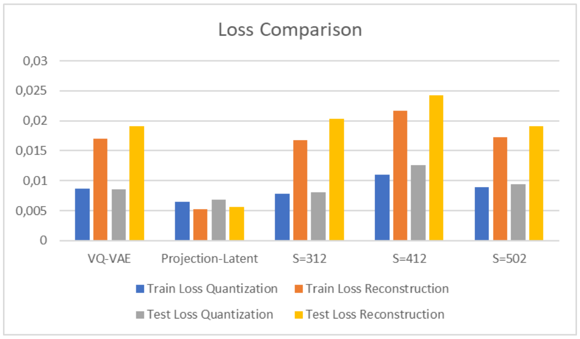

Figure 5. Loss comparison between all models mentioned in the report.

5. References

[1]Aaron van den Oord, Oriol Vinyals, Koray Kavukcuoglu. Neural Discrete Representation Learning. arXiv preprint arXiv:1711.00937v2, 2018