Image synthesis using a convolutional and Transformer architecture

Anonymous author

Abstract

This paper proposes a VQVAE[8] architecture in combination with a transformer[11] trained on the FFHQ dataset. This model implements the works of [4], which utilizes convolutions to learn a context rich vocabulary capturing the local interactions of the image pixels, and uses a transformer architecture to learn complex long term relationships between these visual features to synthesise high resolution images.

1 Methodology

1.1 Learning a context-rich codebook

First step of the model requires learning a codebook, consisting of discrete code words representing visual features appearing in the dataset. To do this we utilise a Variational autoencoder (VAE) [6] architecture, consisting of an encoder (E) and decoder (D). The purpose of the VAE is to learn the visual parts of images with the use of convolutions. This is done by taking advantage of the pixels local interactions through which we learn high level features from the images. Equation 1 shows how the image in the pixel space is encoded into the latent space, where we decrease the dimensions of the image and increase the number of channels (n_z). n_z represents the numbers of convolutional filters learnt, each of which represents a feature regarding the image such as edges.

$$x_{latent} = E(x) \ \ s.t \ \ x_{latent} \in \mathbb{R}^{8\times8\times n_z} \tag{1}$$

To generate the codebook we use vector quantisation (VQ)[8], shown by Equation 2. This is a signal processing technique, to represent the visual parts in quantized form. The aim of this is to make the model less computationally expensive when training the transformer. VQ can be thought of as separating vectors into groups with roughly the same amount of points closest to them, where each group is represented by a centroid as shown with Figure 1[7]. More precisely, in Equation 2 after iterating over the image in the latent space, each

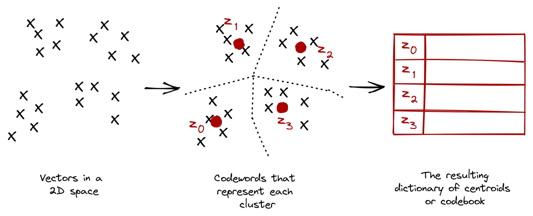
z1
z2
z0
z3
z0
z1
z2
z3
Vectors in a 2D space
Codewords that represent each cluster
The resulting dictionary of centroids or codebook

Figure 1: Visual representation of Vector Quantisation as shown in Equation 2 [7]

latent vector is replaced with the nearest neighbour vector (x_{nn}) in the codebook, where n_z is the number of channels and is the equivalent to the dimension of each vector in the codebook.

$$vq(x_{latent}) = x_{nn} = \underset{k_m \in codebook}{argmin} \ \|x_{latent\ ij} - k_m\| \ \in \mathbb{R}^{h\times\times w\times n_z} \tag{2}$$

The reconstruction of $\hat{x} \approx x$ is done from the quantized form x_{nn}, shown in Equation 3 after the latent vectors replaced by codebook are then transformed into the pixel space using the decoder D.

$$\hat{x} = D(vq(E(x))) \tag{3}$$

To train the VAE and the codebook at the same time, the loss function (Equation 3) is minimised.

$$Loss(E, Codebook) = \|x - \hat{x}\|^2 + \|sg[E(x)] - x_{nn}\| + \|sg[x_{nn}] - E(x)\| \tag{4}$$

In equation 4 the first term refers to the reconstruction loss $\|x - \hat{x}\|$, which is minimised to train the Decoder. With this the network is able to determine how well the input image is reconstructed, when given only its quantized version x_{nn} to the decoder. Note, this is actually the perceptual reconstruction loss [5]. Unlike conventional reconstruction metrics such as Mean absolute error (MAE), which compare the raw pixel values, perceptual loss concentrates on capturing high-level aspects of the image. These are more important to human perception. To calculate perceptual loss we compare the activations of a deep neural network that has already been trained (VGG for our case [10]) on the target image(x) and the output image($\hat{x}$). As high-level elements of the images, such as edges, textures, and forms, are often captured by the activations of the pre-trained network, perceptual loss proves to be a better metric.The second term in equation 4 is minimised to optimise our embeddings to better represent visual features, hence ensuring the codebook learns more perceptually rich features. The third term is 'commitment loss'(coined by authors [4]), which ensures that the encoder E commits to a particular representation of the image. With the minimisation of both of these terms the embeddings move closer to the codebook representation and vice versa. Also, $sg[*]$ denotes the stop-gradient operation, which is an identity function used in the forward pass. Its used to back-propagate through 2 as its non-differentiable. With this stop-gradient operation, we copy the gradients from the E to the D, such that the codebook and the autoencoder can be trained all together via the minimisation loss function shown in Equation 4.

1.2 Training the Transformer

From subsection 1.1 we learn a perceptually rich codebook $\mathbb{C}$, and obtain a discretized representation for our images. To model image synthesis as a auto-regressive problem [2], each latent vector of the image is replaced by its associated codebook index (Equation 5). This image when flattened is equivalent to a sequence $s \in [0, 1, ..., \|\mathbb{C}\|]^{h \times w}$.

$$s_{ij} = k \ s.t \ (x_{nn} = x_k) \ where \ x_k \in \mathbb{C}, \ \mathbb{C} \in \mathbb{R}^{h \times w \times n_z} \tag{5}$$

We then choose some indices, and use the transformer architecture to predict the next-index, hence formulating this as a auto-regressive prediction problem [2]. More precisely, using a image with indices in sequence s_1, we choose the first i indices. With this sub-sequence $s_1[0 : i-1]$ the transformer [11] learns to predict the probability of the next possible indices, i.e. $p(s_1[i]|s_1[0 : i-1])$. Using this probability we compute the likelihood for all next possible indices using Equation 6.

$$p(s_1) = \prod_i p(s_1[i] \mid s_1[0 : i-1]) \tag{6}$$

With this probabilistic approach, we maximise the negative log-likelihood of Equation 6 to find the probability distribution and parameters that best explain the observed indices. This creates the loss function we maximise to train our Transformer shown in Equation 7 below.

$$Loss_{transformer} = \mathbb{E}_{x \sim p(x)}[-\log p(s)] \tag{7}$$

With the maximising of Equation 7 we model long-range dependencies between the visual parts present in the codebook. Through this the transformer learns the complex relationships between these visual parts and is able to predict which features complement each other. For example, learning that a flower feature would be complemented by leaves features in the background.

1.3 Sampling images from transformer

In order to sample images with a greater resolution than the input, the authors employ a sliding attention window with dimension $h_{latent} \times w_{latent}$. The reason for this sliding-attention window was used to restrict context to neighbours and also to increase efficiency for transformers. Although in our case as input and output resolutions are equal we utilise one attention window to generate sequences of visual parts, which are then decoded to output a synthesised image.

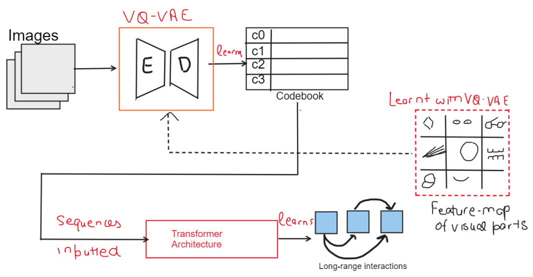

Figure 2: Architecture of the VQVAE with Transformer

1.4 Strategy

Over the span of this project, we initially started with a VAE. The aim was to learn a latent distribution of the images, and sample the VAE for generations. Although, despite our model being very deep the samples were not very good. Next, we tried a latent diffusion model [9], with our trained VAE. The code for this is in the submission file. Although, it seems that with the limited computational power, and with the complexity of the FFHQ dataset the samples were not very good. Therefore, we decided to go with a VQVAE with a transformer for sampling images [4]. To implement this, code from [3] was adapted for FFHQ128 with the change of parameters in the VAE and transformers. Also, some additional functions such as interpolating images were added.

2 Results

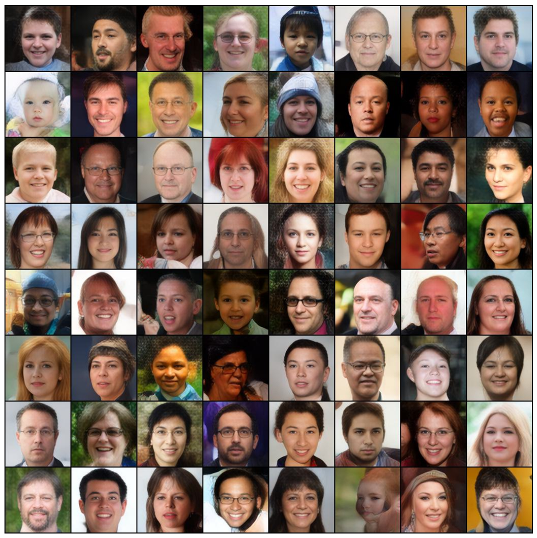

Figure 3: A unique batch of 64 non cherry picked samples

From figure 3 results seem promising, with the transformer generating facial features and other features such as glasses and hair colour very well.

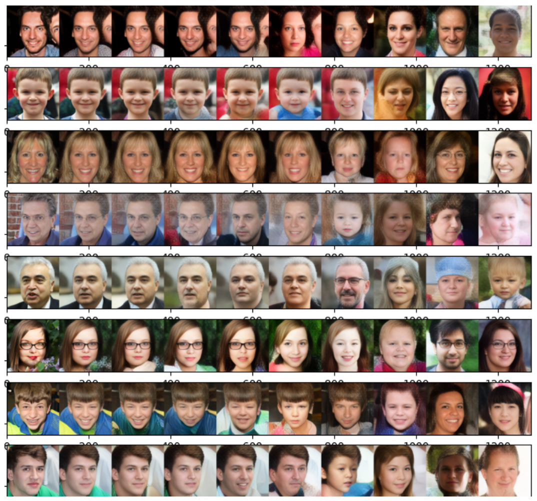

Figure 4: Interpolations between 8 pairs of the samples

Figure 4 shows the images generated by interpolating between the different number of tokens available to the transformer. The leftmost image is the original image, the next image represents the reconstruction of the image with our VAE. The next images show the constructions using our transformer with less tokens available. More precisely, our latent space is $8 \times 8 \times 256$, hence there are 64 tokens for each image. With each image we remove 8 tokens, so firstly the transformer has 56 tokens from the original image and generates the rest 8. Hence, with the rightmost image the transformer generates all 64 tokens, therefore generating a completely new image.

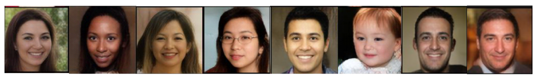

Figure 5: These are 8 best cherry picked generations from the model

Figure 5 consists of some best cherry picked samples generated by our model.

3 Limitations

Their seems to be some artifacts in the results, and it seems that the facial features have been captured well. To add sharpness, a GAN can be utilised in the future when training the VQVAE. However, due to the limits of NCC only a simple VQVAE was trained. Also, the backgrounds for some images seem to be not constructed well. This could possibly the indication that the codebook embeddings do not encompass the background features as well. To improve this in the future, more codebook vectors could be added, increasing from the current 1024. Furthermore, another possible improvement of this model is to sample using a diffusion model implemented by authors [1], which samples images using attention guided diffusion. This would prevent the artifacts in the images, and produce better results.

4 Bonuses

This submission has a total bonus of +8 marks, as the model is trained on the FFHQ 128 dataset.

References

[1] Sam Bond-Taylor et al. "Unleashing Transformers: Parallel Token Prediction with Discrete Absorbing Diffusion for Fast High-Resolution Image Generation from Vector-Quantized Codes". In: *CoRR* abs/2111.12701 (2021). arXiv: 2111.12701. URL: https://arxiv.org/abs/2111.12701.

[2] Murtaza Dalal, Alexander C. Li, and Rohan Taori. "Autoregressive Models: What Are They Good For?" In: *CoRR* abs/1910.07737 (2019). arXiv: 1910.07737. URL: http://arxiv.org/abs/1910.07737.

[3] dome272. *VQGAN repositry adapted for implementation.* 2022. URL: https://github.com/dome272/VQGAN-pytorch (visited on 08/08/2022).

[4] Patrick Esser, Robin Rombach, and Björn Ommer. "Taming Transformers for High-Resolution Image Synthesis". In: *CoRR* abs/2012.09841 (2020). arXiv: 2012.09841. URL: https://arxiv.org/abs/2012.09841.

[5] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *CoRR* abs/1603.08155 (2016). arXiv: 1603.08155. URL: http://arxiv.org/abs/1603.08155.

[6] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *CoRR* abs/1906.02691 (2019). arXiv: 1906.02691. URL: http://arxiv.org/abs/1906.02691.

[7] LJ MIRANDA. *The Illustrated VQGAN.* 2021. URL: https://ljvmiranda921.github.io/notebook/2021/08/08/clip-vqgan/ (visited on 08/08/2021).

[8] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu koray. "Neural Discrete Representation Learning". In: *Advances in Neural Information Processing Systems.* Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf.

[9] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *CoRR* abs/2112.10752 (2021). arXiv: 2112.10752. URL: https://arxiv.org/abs/2112.10752.

[10] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1409.1556.

[11] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.