Different Image Style Generation using CNN versus VQGAN

Erik Cikalleshi

Boon-Chung Chi

Abstract

VQGAN and A Neural Algorithm of Artistic Style (Keras Implementation) are cutting-edge technologies in the field of image generation and style transfer. VQ-GAN, short for Vector Quantized Generative Adversarial Network, is a powerful generative model that utilizes vector quantization to generate high-quality and diverse images. It excels in capturing complex patterns and details, making it a versatile tool for various creative applications.

On the other hand, A Neural Algorithm of Artistic Style focuses on artistic style transfer, a technique that involves applying the visual style of one image to another while preserving the content. This technology leverages neural networks to separate and manipulate content and style features, enabling users to create visually appealing and stylized images. It has found widespread use in transforming photographs into artworks inspired by famous artists or specific visual styles.

Both VQGAN and A Neural Algorithm of Artistic Style showcase the remarkable capabilities of deep learning in the realm of computer vision and image processing. They contribute to the democratization of artistic expression and offer powerful tools for creators and researchers to explore new dimensions in visual content generation.

1 Introduction

When looking at art, humans have mastered an unique way of creating visually pleasing paintings and images. Achieved by composing color, objects and scenes in an individual way, thus making art unique. The system uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images. In order to generate the pictures we want to introduce two different approaches. We will look into the architecture and the algorithms which are essential for both types. Moreover, we will compare the results of the two technologies.

2 Methology

In this section we will explain technologies, algorithms and design of these technologies.

2.1 A Neural Algorithm of Artistic Style (Keras Implementation)

This algorithm provides the possibilty to seperate and recombine different styles and content from natural pictures or pre-exitung art, in order to generate new images of perceptual quality. For this type of task, image representations of Convolutional Neural Networks are used, since it is optimised for object recognition and can provide high level of image information. In an ideal scenario, a style

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

transfer algorithm would excel in extracting the semantic image content found in the target image, encompassing objects and the broader scenery. Following this, it would direct a texture transfer procedure to portray the semantic essence of the target image in a manner that aligns with the stylistic characteristics of the source image.

2.1.1 Style-transfer Algorithm

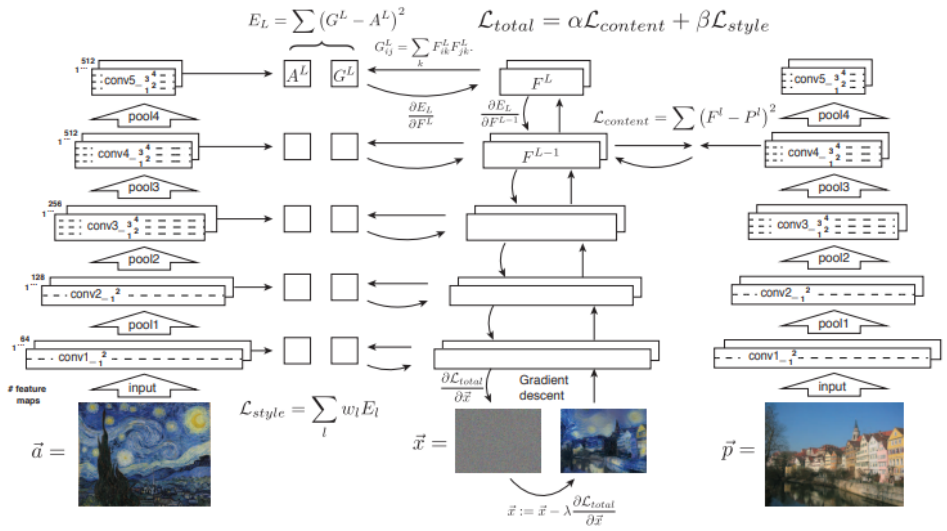

Figure 1: Algorithm and Architecture

Initially, content and style features are extracted and stored. The style image, denoted as $\vec{a}$, undergoes processing through the network, resulting in the computation and storage of its style representation A^t across all layers (depicted on the left). Simultaneously, the content image $\vec{p}$ traverses the network, and its content representation P^t in a single layer is stored (illustrated on the right).

Next, a random white noise image $\vec{x}$ is fed through the network, leading to the computation of its style features G^L and content features F^L. For each layer encompassed in the style representation, the element-wise mean squared difference between G^L and A^L is calculated, forming the style loss $\mathcal{L}_{style}$ (shown on the left). Additionally, the mean squared difference between F^l and P^l is computed to yield the content loss $\mathcal{L}_{content}$ (shown on the right). The total loss, denoted as $\mathcal{L}_{total}$, emerges as a linear combination of the content and style losses.

2.2 Vector Quantized Generative Adversarial Network (VQGAN)

VQGAN help us create high quality images. It's very useful when we need to generate images based on specific conditions, like the type of the object or location.

Transformer are getting more popular for tasks involving language and other areas such as vision or sound. They are also good on understanding complex relationships in data. On the other side we have Convolutional Neural Networks which can capture patterns in nearby pixels in images.

Convolutional VQGAN learn about different parts of an image and then an autoregressive transformer to put these parts together. This helps in making images look realistic while keeping the file size small. Instead of representing images pixel by pixel, we represent them as a combination of different parts from a library of visual elements.

VQGAN is a modified model of the VQVAE (Vectore Quantized Variatonal Autoencoder) and it also combines aspects of GAN (General adversarial Network). The VQVAE consists of a encoder which

maps the images/observations onto a sequence of discrete latent variables and a decoder reconstructs the observations from the discrete variables. They both use a shared codebook.

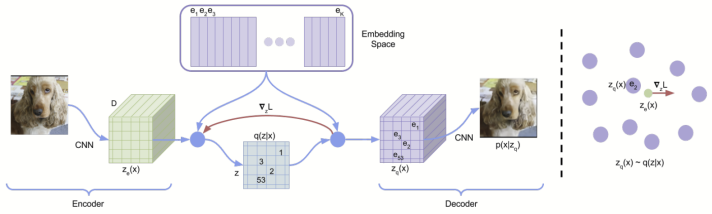

Figure 2: VQVAE

2.2.1 VQVAE

In the Figure 2 we can see that a image of a dog is passed through the encoder. The encoder and decoder are usually CNN's. The encoder than creates a "latent space" which is a space of compressed image data and similar data points are close together. The latent-space in VAE has usually continuous variables. VAEs typically model the latent space as a Gaussian distribution, where each point in the latent space represents a set of parameters that describe the distribution of data. Therefore, the values in the latent space are continuous and can represent a wide range of possibilities.
The continuous variables are then "quantized" which means that continuous variables are converted into discrete ones. This process involves comparing each value in the latent space to a set of code vectors in a codebook, which is essentially a collection of representative vectors. Each value is then replaced with the index of the nearest code vector. During reconstruction, the decoder uses these quantized indices to retrieve corresponding code vectors from the codebook, reconstructing the original continuous values. In simpler terms, the codebook acts like a dictionary that helps translate the compressed representations back into meaningful data.

2.2.2 VQGAN

VQGAN can learn not only the visual parts of an image but also their relationship. Here a table to illustrate what both type of models/variables we need

Table 1: VQGAN

Approach	Model	Example	Analogy
Discrete	Transformer (Long-range dependencies)	Sequence of words, sentences	Perceiving
Continous	Convolutional Neural Networks or Generative Adversarial Networks (Local interactions and visual parts)	RGB channels in a pixel, convolutional filters ...	Sensing

2.2.3 How does a VQGAN work?

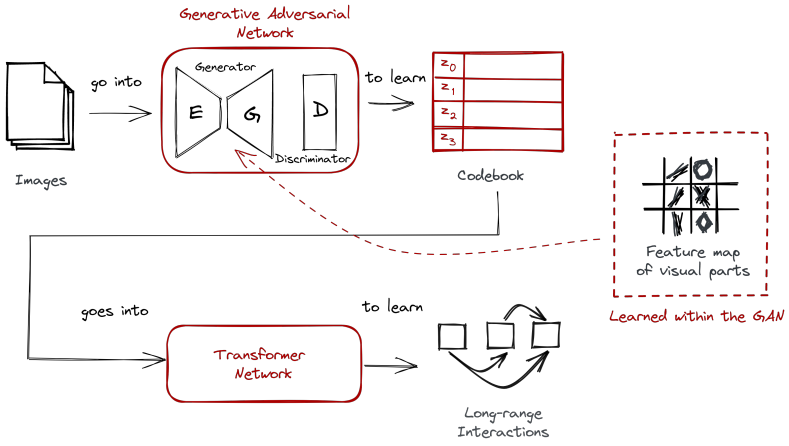

Figure 3: VQGAN

Theory of pereptions suggests that the visual reasoning is very symbolic and the meaning can be described through their discrete representation. We have a two-stage approach implemented by VQGAN. One main component is a convolutional neural network in form of a GAN and the second main component is a Transformer. By using CNN/GAN we can learn highlevel features to construct new images. The convolutional neural network is often replaced with a generative adversarial network and it can still perform convolutional operation but it can caputre more distinct visual parts. Given a sequence to learn, in form of a discrete representation, a transformer can understand relationships across visual parts. A codebook, obtained via vector quantization, consists of discrete codewords that allows us to easily train a transformer on top of it.

Training the GAN When training the GAN from a image dataset we just not want to learn its visual parts of those, but also their codeword representation for the codebook. As we already know from our Lecture the discriminator distinguishes the real input x and the generated sample $\hat{x}$. We use the following loss function for the discriminator:

$$L_{GAN}(N, D) = [\log D(x) + log(1 - D(\hat{x}))]$$

- N is the generator, D is the discriminator
- first term $\log D(x)$ measures probability of discriminator D of the real input being actually real.
- the second term gives the probability of the discriminator D to say that generated instance $\hat{x}$ is real

The generator follows an encoder-decoder architecture, with vector quantization (VQ) happening between the encoder and decoder. The Loss Function for Training Generator and Encoder (LVQ) is the following

$$L_{VQ}(E, G, Z) = ||x - \hat{x}||^2 + ||sg[E(x)] - z_q||_2^2 + ||sg[z_q] - E(x)||_2^2$$

where the first term is the Reconstruction loss. Second term describes the embedding optimization and last the term is the commitment loss.

And to wrap all up the overall training objective is a minmax problem.

$$g^* = argmin_{E,G,Z} max_D \mathbb{E}_{x \sim p(x)}[LVQ(E, G, Z) + \lambda LGAN(N, D)]$$

where E is the encoder, G the decoder, Z is the codebook and λ is an adaptive weight.

And we do need to train also the transformer to predict the next index of an encoded sequence. Images are represented as sequences corresponding to the codebook-indices of their embeddings. This means

each image is converted into a sequence of discrete indices from the codebook Z. The loss function for training the transformer is computed based on the likelihood of predicting the correct next index in the sequence. Given the encoded sequence ss, the transformer predicts the distribution of possible next indices $p(s) = \prod_i p(s_i|s_{<i})$. We want to minimize the transfomer loss

$$L_{Transformer} = \mathbb{E}_{x\sim p(x)}[-\log p(s)]$$

The model learns to predict the next index based on the preceding indices in the sequence. By doing so, the transformer learns the patterns and relationships within the sequence of codebook-indices representing the image. To manage computational resources efficiently, the authors employ a sliding-window mechanism during training. This means that when generating each patch of the image, the transformer only considers information from its neighboring patches.

3 Results

Finally, we can look at the pictures which are generated by both technologies. We tried to apply different prompts using VQGAN and we also looked at the same assignment for the neural algorithm. The target picture is Paris on daylight and for the style picture we used "The Starry Night by Vincent van Gogh".

Figure 4: Reference Picture of Paris

Figure 5: The Starry Night

VQGAN + CLIP We began with following prompt: "Apply Van Gogh Style of The Starry Night" and the Paris picture as target.

Figure 6: At 0 Iterations

Figure 7: After 50 Iteration

Figure 8: After 500 Iterations

A Neural Algorithm of Artistic Style Now we look at the technology implemented using Convolutional Neural Networks. There we can pass a style picture and a target picture, where the style is applied on. Here we also used Paris and Van Gogh's drawing as the input.

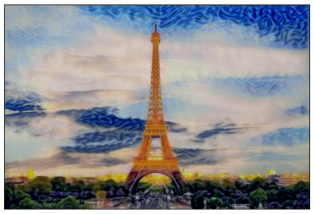

Figure 9: First Conv Block

Figure 10: Second Conv Block

Figure 11: Third Conv Block

As we look at both architectures the results show different outputs. When looking at VQGAN, we can see that it really tried to apply the art of “The Starry Night“ on the Paris picture. While the first 50 iterations look very pleasing, the output of 500 iterations looks like a completely different picture. The cause could be over fitting since it takes information from the iteration before. So it could start interpreting to much on lines and features, which then causes the neural network to add different things. But it is important to note, that the style of Van Goghs Painting still remains.

In comparison the output from the Neural Algorithm is different to the one from VQGAN. We can see that it tried to extract the features and then apply it on the Paris picture. However, it tries to keep the features of Paris without alternating the picture too much. We also notice that the change on the colors is more visible that the “style“ of “The Starry Night“.

4 Summary

In summary, we can say that the VQGAN performed better in the task of applying the style on a certain picture, even though it can change the essence of a target picture very fast. On the other Hand the Neural Algorithm does well in keeping the information of the target picture while converting the style on it. Still it is not perfect but this often lies on the perspective of the user, since the keypoint of art is the interpretation of each individual. Furthermore, we can still improve both technologies with new algorithms or optimization to one’s need.

References

[1] L. A. Gatys, A. S. Ecker and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265. keywords: Image reconstruction;Neural networks;Image representation;Semantics;Neuroscience;Feature extraction;Visualization,

[2] https://ljvmiranda921.github.io/notebook/2021/08/08/clip-vqgan/

[3] https://github.com/Kautenja/a-neural-algorithm-of-artistic-style/

[4] https://arxiv.org/pdf/1508.06576.pdf