Data Analytics and Artificial Intelligence
Vol: 3(6), 2023
REST Publisher; ISBN: 978-81-948459-4-2
Website: http://restpublisher.com/book-series/daai/

Generative Adversarial Network Based Text to Art AI Model

[*]Anuja. T, Dani Julian Bennet, Abdul Hakeem A K, Vignesh S
Jeppiaar Engineering College, Chennai, Tamil Nadu, India.
[*]Corresponding author Email: anuanuja2013@gmail.com

Abstract: *Generative Adversarial Networks (GANs) are a type of deep learning model that has become popular in recent years due to its ability to generate high-quality synthetic data in various domains such as images, audio, and video. The GAN architecture consists of two main components: a generator network and a discriminator network. The generator network is trained to generate synthetic data samples that are similar to real data, while the discriminator network is trained to distinguish between real and fake data. The two networks are trained together in an adversarial process, where the generator tries to fool the discriminator by producing realistic data, while the discriminator tries to correctly identify the generated data as fake. This process continues until the generator produces data that is indistinguishable from real data, and the discriminator is unable to identify the difference. This technique can produce images that appear authentic to human observers. For example, a synthetic image of a cat that manages to fool the discriminator (one of the functional parts of the algorithm) is likely to lead some people to accept it as a real photograph. The difference of VQ-GAN with previous GAN networks is that it allows high resolution outputs. CLIP (Contrastive Language Image Pretraining) is another artificial intelligence that allows you to transform texts into images. To adapt the final image to a specific shape, starting images with color masks can be used by making that shape.*

1. INTRODUCTION

VQ-GAN+CLIP is a powerful technique that has been used to generate a wide variety of images, including realistic photographs, abstract art, and even memes. It has also been used in applications such as image editing and style transfer, as well as in creative projects such as digital art and music videos.

2. LITERATURE SURVEY: IMAGE GENERATION

Most works use CLIP as a source of guidance for image manipulation. Specifically, in CLIP, a text encoder and an image encoder are pre-trained to measure the consistency between text and image. CMA uses CLIP, measuring the distance between generated and the reference image, as supervision for image manipulation during the inference phase, which requires several minutes to perform a single manipulation. To handle this issue, The method proposed in this paper follows the line of visual synthesis research based on auto-regressive models. Visual auto-regressive models performed visual synthesis in a “pixel-by-pixel” manner.

Style CLIP uses CLIP to train three latent mappers to achieve image manipulation in a single forward pass. In contrast, recent Diffusion CLIPS uses CLIP to align the direction between the encoded vectors of the reference and generated images with the direction between the encoded vectors of a pair of reference and target texts, finetuning the latent space for image editing. However, all these models depend on pre-trained CLIP as supervision in the training andinference phase. Recently, VQ-VAE-based [67]visual auto-regressive models were proposed for visual synthesis tasks. By converting images into discrete visual tokens, such methods can con-duct efficient and large-scale pre-training for text-to-image generation (e.g., DALL-E [54] and Cog View [17]), text-to-video generation (e.g., GODIVA [72]), and video prediction(e.g., LVT[53] and Video GPT[75]), with higher resolution of generated images or videos

Image to Image translation has been around for some-time before the invention of Cycle Gans. One really interesting one is the work of Phillip Isola et al in the pa-per Image-to-Image Translation with Conditional Adversarial Networks where images from one domain are translated into images in another domain. The dataset for this work consists ofan aligned pair of images from each domain. Generative image models are well studied and fall into two categories: parametric and nonparametric. The nonparametric models often do match from a database of existing images, often matching patches of images, and have been used in texture synthesis, super-resolution and in-painting. Parametric models for generating images have been explored extensively (for example on MNIST digits or for texture synthesis). However, generating natural images.

Copyright@ REST Publisher

Another approach generates images using an iterative forward diffusion process. Generative Adversarial Networks generated images suffering from being noisy and incomprehensible. The maximum mean discrepancy (MMD) [23] is a measure of the difference between two distributionsP and Q given by the supremum over a function space for differences between the expectations withregard to two distributions. MMD has been used for deep generative models [19,34,35] and model criticism [63]. WGAN [4] conducted a comprehensive theoretical analysis of how the EarthMover (EM) distance behaves in comparison with popular probability distances and divergences suchas the total variation (TV) distance, the
Kullback-Leibler (KL) divergence, and theJensen-Shannon (JS) divergence utilized in the context of learning distributions. Based on W-Div, process, reach a larger applicant pool, andimprove the candidate experience.

3. THEORETICAL STUDY OF GENERATIVE ADVERSARIAL NETWORK

Introduction to GAN:
A Generative Adversarial Network (GAN) is a type of deep neural network architecture that is used for generative modeling. The GAN framework consists of two separate neural networks: a generator and a discriminator.
The generator network takes random input noise and generates output data, such as images, sounds, or text, that mimics the patterns in a given training dataset. The discriminator network then evaluates theoutput data and decides whether it belongs to the realdataset or not. The generator is trained to produce outputs that are similar to the real data, while the discriminator is trained to correctly identify which outputs are real and which are fake.The generator in VQ-GAN consists of several components, including:encoder , vector quantization layer and decoder.

Encoder:
The encoder in a generative model is responsible for mapping an input image to a set of latent codes that represent the content of the image. The process of the encoder can be broken down into the following steps:

Preprocessing: The input image is preprocessed to prepare it for input into the encoder. This typically involves resizing the image to a fixed size and normalizing the pixel values to be within a certain range.

Convolutional Layers: The preprocessed image is then passed through a series of convolutional layers in the encoder. The convolutional layers extract features from the image at different spatial scales andcreate a feature map that encodes the content of the image.

Pooling Layers: After each set of convolutional layers, pooling layers are often used to reduce the spatial dimensions of the feature map while retainingimportant features.

Dense Layers: The output of the final set of convolutional and pooling layers is flattened and passed through one or more dense layers, which further encode the content of the image into a fixed-size vector.

Output: The final output of the encoder is a set of latent codes that represent the content of the input image. These codes are used as input to the decoder in a generative model to produce a new image that is similar to the input image.

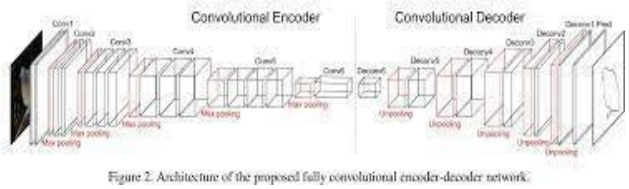

Figure 2. Architecture of the proposed fully convolutional encoder-decoder network.
FIGURE 1. Encoder to decoder flow Vector Quantization Layer:

The vector quantization (VQ) layer is a key component of Vector Quantized Generative Adversarial Networks (VQGANs), which is a variant of Generative Adversarial Networks (GANs) that use a vector quantization technique to improve the quality and stability of the generated images. TheVQ layer is responsible for quantizing the continuous latent vectors output by the encoder into a discrete set of codes, which are used as input to thedecoder.

The process of the VQ layer can be described as follows:
Codebook Initialization: A fixed number of code vectors (usually referred to as "codebook") are initialized randomly or based on some prior knowledge about the data. Each code vector represents a distinct region in the latent space.

Finding the Closest Code: For each latent vector output by the encoder, the VQ layer finds the code vector in the

codebook that is closest to it (in terms of Euclidean distance or other distance metrics). The index of the closest code vector is then used as the discrete code for that latent vector.

Gradient Estimation: The VQ layer computes a gradient estimate that encourages the encoder to produce latent vectors that are close to the code vectors in the codebook. This is achieved by backpropagating the gradient of a distortion loss function (e.g., mean squared error) between the latent vectors and their corresponding closest code vectors.

Codebook Update: The codebook is updated based on the encoded data. This is typically done by updating each code vector to be the average of the latent vectors that were quantized to it.

4. METHODOLOGY PROPOSE WORKFLOW OF THE APPLICATION

To demonstrate our method's effectiveness, we applyit using VQ-GAN [11] and CLIP [37] as pre-trained models, and so refer to our approach as vq-gan-clip. We stress, however, that our approach is not specific to either model and that subsequent work has alreadyshown success that builds on our work using other models [5, 27, 46, 12], and even in other modalities [18, 50]. We start with a text prompt and use a GAN to iteratively generate candidate images, at each step using CLIP to improve the image. We optimize the image by treating the squared spherical distance between the embedding of the candidate and the embedding of the text prompt as a loss function and differentiating through CLIP with respect to the GAN's latent vector representation of the image, which we refer to as the "z-vector" following Oord, Vinyals, and Kavukcuoglu [35]. This process is outlined in Fig. 1

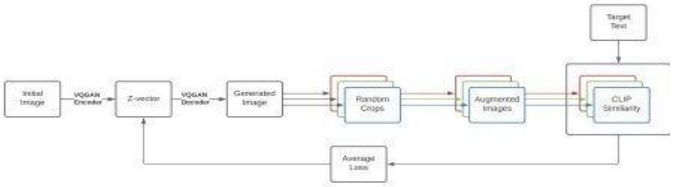

FIGURE 2. Regression layer

Figure 2: Diagram showing how augmentations are added to stabilize and improve the optimization. Multiple crops, each with different random augmentations, are applied to produce an average loss over a single source generation. This improves the results with respect to a single latent Z-vector.

To generate an image, the "initial image" contains random pixel values. The optimization process is repeated to alter the image, until the output image gradually improves such that it semantically matchesthe target text. We can also edit existing images by starting with the image-to-edit as the "initial image". The text prompt used to describe how we want the image to change is used identically to the text prompt for generating an image, and no changes to the architecture exist between generation and manipulation besides how the 'initial image" is selected. We use Adam [20] to do the actual optimization, a learning rate of 0.15, $\beta = (0.9, 0.999)$,and run for 400 iterations for the experiments in this paper.

Discrete Latent Spaces for Images: Unlike the naturally discrete nature of text, the space of naturally occurring images is inherently continuous and not trivially discretized. Prior work by Oord, Vinyals, and Kavukcuoglu [35] borrows techniques from vector quantization (VQ) to represent a variety of modalities with discrete latent representations by building a codebook vocabulary with a finite set of learned embeddings. Given a codebook of vocabulary size K with embedding dimension nk, Z= {zi} K k ∈ R nk. This is applied to images by constructing a convolutional autoencoder with encoder E and decoder G. An input image x ∈ I is first embedded with the encoder z = E(x). We can then compute the vector quantized embedding x aszq = argmin zk∈Z ‖zi,j − zk‖. which we can then multiply back through the vocabulary in order to perform reconstruction. We can then use a straight-through estimator on the quantization step inorder to allow the CNN and codebook to be jointly trained end-to-end. We use the popular VQ-GAN
[11] model for the experiments in this paper.

Contrastive Text-Image Models:
To guide the generative model, we need a way to adjust the similarity of a candidate generation with the guidance text. To achieve this, we use CLIP, [37],a joint text-image encoder trained by using contrastive learning. We use CLIP to embed text prompts and candidate generated images independently and measure the cosine similarity between the embeddings. This similarity is then reframed as a loss that we can use gradient descent tominimize.

Augmentations: One challenge of using vq-gan-clipis that gradient updates from the CLIP loss are quite noisy if calculated on a single image. To overcome this, we take the generated candidate image and modify it many times, producing a large number of augmented images. We take random crops of the candidate image and then apply further augmentations such as flipping, color jitter, noising, etc. [39] Most high-level semantic features of an

image are relatively invariant to these changes, so averaging the CLIP loss with respect to all of the augmented images reduces the variance of each update step. There is a risk that a random crop mightdramatically change the semantic content of an image (e.g., by cropping out an important object), butwe find that in practice this does not cause any issues. For the results presented in this paper we used an augmentation pipeline consisting of random horizontal flips, random affine projections, random perspective projections, random color jitter, and adding random Gaussian noise.

Regularizing the Latent Vector:
When using an unconstrained VQ-GAN for image generation, we found that outputs tended to be unstructured. Adding augmentations helps with general coherence, but the final output will often still contain patches of unwanted textures. To solve this problem, we apply a weighted L 2 regularization to the z-vector. This produces a regularized loss function given by the equation Loss = LCLIP + α · 1 N X N i=0 Z 2 i where α is the regularization weight.This encourages parsimony in the representation, sending low information codes in VQ-GAN's codebook to zero. In practice we note that regularization appears to improve the coherence of the output and produces a better structured image. We decay the regularization term by 0.005 over the course of generation.

Our methodology is highly flexible and can be extended straightforwardly depending on the use-case and context due to the ease of integrating additional interventions on the intermediate steps of image generation. Researchers using our framework have introduced a number of additional components, ranging from using ensembles [6], to using B´ezier curves for latent representations [13, 5], to using perturbations to make the results more robust to adversaries [25]. Although they aren't used in the main experiments of this paper, we wish to call attention to two in particular that we use frequently: "prompt addition" and masked image editing. We give an overview of both here and provide additional experiments and information in Appendix G Prompt Addition: We have found that our users are often interested in applying multiple text promptsat the same time. This can be achieved by computingthe loss against multiple target texts simultaneously and adding the results. In Appendix G.1 we use this tool to explore the semantic cohesion of vqgan-clip's generations. Masking: A common technique in image generation and editing is masking, where a portion ofan image is identified ahead of time as being where amodel should edit5 vq-gan-clip is compatible with masking by zeroing out the gradients in parts of the latent vector that one wishes to not change. However,vq-gan-clip can also leverage the semantic knowledge of CLIP to perform self-masking without any non-textual human input.

5. CONCLUSION

The ability to generate novel art from a seemingly mundane source will always be an interesting one. Today, we looked at how a user could generate pixel art with nothing more than a GPU and text input. The use cases for such a technique are nearly infinite,but we will list a few: Automated content generation (marketing and design): teams in the future will be able to use simple text inputs to generate interesting and new artworks for their content. This could save hundreds of hours for branding and artistic asset creation. Automated content generation (artists): This also extends to creatives, as artists can leverage AI art generation as a new tool at their disposal. Thecreative potential for text to image technology cannot be understated. Auto-mated photo processing:A user can selectively choose the input photo. If the starting photo already has notable designs/patterns instead of random noise, that photo may be used as a guide template for the photo processing. in theory, once this has been sufficiently optimized, we could use a similar architecture to make appropriate edits to existing images rather than doing them manually. For example, if we gave the model an image of clowns with the text prompt "clowns without hats" we could expect the model to try and remove those hats. This is a long way off though.

REFERENCES

[1]. Jonas Adler and Sebastian Lunz. Banach wasserstein gan. InNeural Information Processing Systems.
[2]. Ivan Anokhin, Pavel Solovev, Denis Korzhenkov, AlexeyKharlamov, Taras Khakhulin, Gleb Sterkin, Alexey Silvestrov, Sergey Nikolenko, and Victor Lempitsky. High-resolution daytime translation without domain labels.
[3]. Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay.Face aging with conditional generative adversarial networks.In 2017 IEEE International Conference on Image Processing
[4]. Martin Arjovsky, Soumith Chintala, and L´eon Bottou.Wasserstein generative adversarial networks. In Interna-tional Conference on Machine Learning, pages 214–223,2017.
[5]. Susan Athey, Guido Imbens, Jonas Metzger, and EvanMunro. Using wasserstein generative adversarial networks for the design of monte carlo simulations. arXiv preprintarXiv:1909.02210, 2019.
[6]. Samaneh Azadi, Deepak Pathak, Sayna Ebrahimi, andTrevor Darrell. Compositional gan: Learning image-conditional binary composition. International Journal ofComputer Vision, 128(10):2570–2585, 2020