Texture Synthesis Using Adversarial GAN netowrk

April 3, 2025

Raffaele Leo

Abstract

Texture synthesis is vital in computer graphics for generating expansive textures from smaller samples while preserving key visual characteristics. The advent of deep learning, especially through Generative Adversarial Networks (GANs), has markedly improved this process, producing textures with greater realism and variety. In this paper I study the novel approach of Spatial Generative Adversarial Networks, which aim to generate images of any size by taking out the usual fully-connected layer in GANs

1. Introduction

1.1 A Brief History of Texture Synthesis.

Texture synthesis in computer graphics has evolved a lot over time. Early methods worked by repeating small texture pieces or using random methods, but often had issues like visible repeating patterns. In the late 1990s and early 2000s, a large shift in the field came with patch-based methods, like those introduced by (Efros & Leung, 1999). These new methods created better, more natural textures by joining small pieces from a sample image. However, they were slow and didn't always consider the bigger picture, focusing mainly on small areas.

The advancement of deep learning and, in particular, the development of Generative Adversarial Networks (GANs) by (Goodfellow et al., 2014), revolutionized texture synthesis. GANs introduced a new and exciting approach, where a generative network learns to produce new textures, and a discriminative network evaluates their similarity to the original texture. This not only improved the quality and realism of the synthesized textures but also allowed for greater variety in texture generation.

1.2 Overview of the project.

My project leverages Spatial Generative Adversarial Networks (SGANs), as introduced in ”Texture Synthesis with Spatial Generative Adversarial Networks” (Jetchev & Bergmann, 2016), to generate spatially coherent textures. Focused on addressing challenges in SGANs such as mode collapse, my architecture uses upsampling and convolutional layers in the generator for image creation, and an equal amount of convolutional layers in the discriminator for image evaluation.

I take advantage of strategies like the Two Time-Scale Update Rule (TTUR) and noisy labels to attempt and stabilize training while also enhancing output diversity. TTUR manages learning rate disparities between the discriminator and generator, while noisy labels prevent the discriminator from overpowering the generator, ensuring a wider variety of generated textures. This approach combines Jetchev and Bergmann's foundational SGAN structure with advanced techniques for improved texture synthesis.

2. Methodology

I wanted to use a simple, lightweight generator and discriminator so I could focus on hyper parameters and tuning more, so I used a GAN network with just 4 layers and 64 input channels, this architecture can be seen in tables 1 and 2.

Generator
Input: Noise Input
Upsample and Conv2d, BatchNorm, ReLU
Upsample and Conv2d, BatchNorm, ReLU
Upsample and Conv2d, BatchNorm, ReLU
Upsample and Conv2d, Tanh

Table 1. Architecture of SGAN Generator

Discriminator
Input: Image Input
Conv2d, LeakyReLU (0.2)
Conv2d, BatchNorm, LeakyReLU (0.2)
Conv2d, BatchNorm, LeakyReLU (0.2)
Conv2d, Sigmoid

Table 2. Architecture of SGAN Discriminator

For my baseline model I used a batch size of 2 and a learning rate of 0.0003, I found that a lower batch size helped

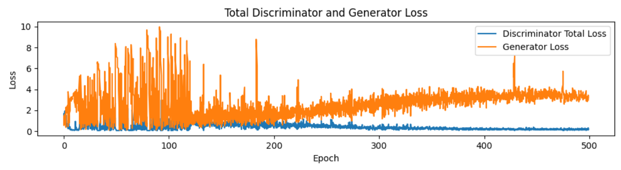

Figure 1. These are my loss outputs before adding handicaps to the discriminator, here we can see the generator start to stagnate and slowly get worse after about epoch 200, this is a classic sign of mode collapse

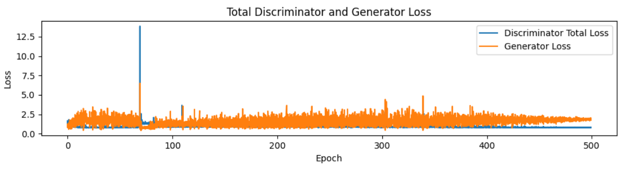

Figure 2. Here are the loss outputs after adding noisy labels and TTUR, it shows a much healthier relationship between the discriminator and generator.

the generator stay on task and generate images closer to the ground truth. My generator input was a simple noise input of size batch_size x noise_channels x 16 x 16. I set the noise channels to 20 after some trial and error as I seemed to generate the best results, and they were of size 16 x 16 so the generator would output images of size 256 after upsampling. This was important as it is imperative that the generator and discriminator use images of a standard size to learn properly.

For training I would take 28 random crops of size 256 of a single texture image, then randomly rotate them along with normalizing the colors. I would then train the model on these 28 images for 500 epochs, outputting the generated images after every 5 epochs to get an idea of how the generator was progressing. At the end I would have the generator output a ”big image” in evaluate mode to see the final output, this big image was 4x larger than the training images and many times larger than the original texture. This output ensured that the models could still generate realistic textures of an arbitrary size.

My code can be found at: `https://colab.research.google.com/drive/1bmqDHxv2QKZ_Vo6BKHL1rZySpNXnQJE1?usp=sharing`.

3. Experimental Results

After training for 500 epochs I noticed the generated images stay relatively stagnant after half way through the training process, and when I plotted the quantitative results in figure 1 I saw that I was most likely dealing with Mode Collapse, where the Discriminator was becoming too powerful and limiting the outputs of the Generator too much.

To fight this I added noisy labels, and the Two Time-Scale Update Rule (TTUR) as introduced by Heusel et al. (Heusel et al., 2017), and also pulled from Salimans et al.’s ”Improved Techniques for Training GANs” (Salimans et al., 2016) . These techniques are to help keep the discriminator guessing and not getting too confidence with its predictions. For the dual learning rates I started the generator at a much lower learning rate than the discriminator (0.0001 and 0.0004 respectively). These learning rates take advantage of the fact that the discriminator has a much easier job than the generator, with one being a binary classifier and the other being a full image generator. After incorporating these two strategies I saw vast improvements in the qualitative results as well as the quantitative, as can be seen in figure 2.

The experimental results can be found in this folder: `https://drive.google.com/drive/folders/1z620RDusSL7XwoN4E5okULJnAP0AaAJl?usp=sharing`

Each folder has all generated images from the 500 epochs, along with the ground truth textures, the loss graphs, and a generated ”big-texture.”

4. Discussion and Conclusions

This paper explored the use of Spatial Generative Adversarial Networks (SGANs) in the realm of texture synthesis, particularly focusing on overcoming common challenges like mode collapse. The implemented SGAN architecture, characterized by a simplified yet effective design, demonstrated significant capabilities in generating spatially coherent textures. Key interventions such as noisy labels and the Two Time-Scale Update Rule (TTUR) were instrumental in stabilizing the training process and ensuring a balanced learning dynamic between the generator and discriminator. These strategies not only mitigated mode collapse but also enhanced the diversity and quality of the generated textures.

These experiments were limited by time and computing power, for future areas of research more layers and computing power would be a great start. Another thing this paper did not touch on is exploring different noise inputs, this GAN architecture is heavily dependent on the noise input and finding new and more purposeful ways to generate noise may be the future in this area of texture synthesis.

References

Efros, A. A. and Leung, T. K. Texture synthesis by non-parametric sampling. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2:1033–1038, 1999.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

Jetchev, N. and Bergmann, U. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in Neural Information Processing Systems*, 29, 2016.