Texture Synthesis with Self-Attention GANs

October 19, 2023

Chiara Giacanelli

Abstract

This report focuses on the analysis and implementation of a texture synthesis model.
For the sake of the project, I built a **SAGAN** (Self-Attention GAN), able to focus on capturing long-range dependencies and building good reconstructions of regular textures in input. For variability purposes, I also introduced a **diversity regularization** loss term to the objective.

1. Introduction

Texture synthesis has gained a lot of attention in the latest years, especially with the advancements of Computer Vision applied to the cinematic and gaming fields. It is the process of constructing a large digital image from a small digital sample image by taking advantage of its structural content. The fundamental goal of example-based texture synthesis is to generate a texture, usually larger than the input, that faithfully captures all the visual characteristics of the exemplar, yet is neither identical to it, nor exhibits obvious unnatural looking artifacts (Zhou, 2018). The evaluation criterion for the quality of the synthesised texture is usually human inspection and textures are successfully synthesised if a human observer cannot tell that the generated texture is the combination of multiple observations taken from the smaller texture in input (in, fact, we want to avoid the *tiling effect*). The result is an acceptable texture image if it is not too repetitive and does not contain too many artifacts.
For the sake of the project, I decided to develop a Self-Attention GAN, which introduces a self-attention mechanism into the generator and discriminator of convolutional GANs (Zhang, 2019). Unlike traditional GANs, SAGANs have the self attention module that is complementary to convolutions and helps with modeling long range, multi-level dependencies across image regions. Armed with self-attention, the generator can draw images in which fine details at every location are carefully coordinated with fine details in distant portions of the image, that is what we are aiming at when dealing with non-tiling texture synthesis.
Due to computational limitations, I managed to synthesize regular textures with very compact and visible pattern details at zoomed image, since the processed images have size 64x64 and low clarity brought not too appealing results. For this reason, in order to improve variability of generated samples, I added a diversity regularization term (Sushko, 2021) to the original adversarial loss of the benchmark code.

2. Related Works

When building the project, I mainly focused on two important reports, which are briefly described below.

Non-Stationary Texture Synthesis by Adversarial Expansion. This has been the baseline project which I started from. This paper proposes a new approach for example-based non-stationary (stochastic) texture synthesis, using adversarial training to double the spatial extent of texture blocks extracted from a specific texture exemplar. The fully convolutional generator learns to expand k × k texture blocks into 2k × 2k ones using a combination of adversarial loss, L1 loss and style loss (Gatys, 2015). The discriminator D is trained to classify whether a 2k × 2k texture block is real (a crop from the input exemplar) or fake (synthesized by G), through the comparison of the extracted features by a pre-trained VGG network.

Self-Attention Generative Adversarial Networks. In this paper, the authors proposed Self-Attention Generative Adversarial Networks (SAGANs), which incorporates a self-attention mechanism into the GAN framework. The self-attention module is effective in modeling long-range dependencies. In addition, they showed that spectral normalization applied to the generator stabilizes GAN training.

3. Method

This project adopted the *self-supervised learning (SSL)* technique. Inference is performed using one-shot learning,

Email: Chiara Giacanelli
<giacanelli.1801145@studenti.uniroma1.it>.

Deep Learning and Applied AI 2023, Sapienza University of Rome, 2nd semester a.y. 2022/2023.

since the self-supervised adversarial training takes place for each specific texture exemplar.

Code. The code of the project is available at the following github repository: `https://github.com/ChiaraGiaca/TextureSAGAN`.

Tools used. The first step has been the collection and organization of the benchmark code that is available at *this* link. I deveolped the entire project with the Pytorch framework and I used Google Colaboratory with a NVIDIA T4 GPU to execute it.

3.1. Network architecture

The GAN is composed by a Generator, a Discriminator each with a Self-Attention Layer and the Generator is trained to minimize a loss function that tries to stimulate variability over repetitiveness. The network is given a *nxm* image in input, and resizes it to 128x128 pixels during training. A 64x64 pixels sample is extracted randomly from the image, it's normalized and it's used by the generator as the base texture from which it generates the extended texture. The output will be of 128x128 pixels.

Generator. The Generator consists of ResNet blocks between a few downsampling and upsampling operations. It's trained in order to produce a 128x128 output from a 64x64 texture in input (actually, the input is of size 128x128 for convenience, but it resembles a random block of its half size).
The input X is initially upsampled by a Padding layer, which pads the input tensor using the reflection of the input boundary. The result is then processed by a Convolutional layer and a ReLU activation function. After, a **Self-Attention layer** is applied to the model to enhance its ability to focus on important regions. Then, there are a couple of downsampling stages that reduce the spatial dimensions of the feature maps and a series of residual blocks (ResnetBlock) are added. After the residual blocks, there are upsampling stages that increase the spatial dimensions of the feature maps. A final Convolutional layer followed by a Tanh activation function produce the output.
The Generator is trained to minimize the following loss function:

$$\mathcal{L}_{GEN} = \mathcal{L}_{adv} + w * \mathcal{L}1 - \mathcal{L}_{DR}$$

Where $\mathcal{L}_{adv}$ is the classical adversarial loss, $L1$ is the L1 loss (that measures the mean absolute error (MAE) between each element in the generated texture and target texture), w is set to 10.0, and $\mathcal{L}_{DR}$ is the introduced **diversity regulation loss** (Sushko, 2021).

$$\mathcal{L}_{DR} = \lambda * \mathbb{E}_{z1,z2}[\frac{1}{L}\sum_{i=1}^{L}\mathcal{L}1(G^l(z1) - G^l(z2)]$$

In the equation, the $L1$ loss computes the distance between samples in the feature space, enabling more meaningful diversity of the generated images, as different generator layers (l in L) capture various image semantics. In this case, λ is the regularization strength and is set to 0.15.

Discriminator. The discriminator network adopts a PatchGAN architecture (Isola, 2016), composed of multiple modules of the form convolution-BatchNorm-LeakyReLU. After the first sequence, a Self-Attention layer is introduced (as for the Generator). This discriminator is run convolutionally across the image, averaging all responses from the local image patches to provide the ultimate output.

Self-Attention Layer. As said before, the proposed attention module has been applied to both the generator and the discriminator, which are trained in an alternating fashion. Query, key and value are implemented as 1×1 convolutions. The final output of the module is the sum of input feature map and the product between the attention formula and a value γ, which is a learnable scalar initialized to 0 that represents the influence of the Self-Attention module on the outputs.

4. Experimental results

I trained the network for 5000 epochs for each texture in input. The learning rate was set to 0.0002 (I tried to add a decay mechanism but it didn't produce great results). I used the *Adam* optimizer for both the Generator and Discriminator.
Unfortunately, for computational limitations, I didn't manage to produce images bigger than 128x128 pixels, that is double the extent of the processed input. I tested the model on various textures of different nature such as water, brick, grid, flowers, dotted and lined (more results can be observed in the GitHub repository). The most appealing results were given by regular and near-regular textures, since cropping a the very small quadratic region of 64x64 pixels didn't cause a huge information loss. On the other hand, stochastic textures like water or lines of variable shape and colors were very difficult to be synthesized properly by the model.

5. Discussion and Conclusions

My project focused on the implementation of a Texture Synthesis SAGAN. The network was trained using one-shot learning, and to mitigate memorization a diversity regulation loss was added to the optimization procedure of the Generator. The best results were given by regular textures, and future works may try to find ways to use the Attention Layer together with higher-dimensional images to be processed for even better results.

References

Gatys, L. Texture synthesis using convolutional neural networks. *Advances in Neural Information Processing Systems*, 2015.

Isola, P. Image-to-image translation with conditional adversarial networks. *CVPR 2017*, 2016.

Sushko, V. One-shot gan: Learning to generate samples from single images and videos. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

Zhang, H. Self-attention generative adversarial networks. *Proceedings of the 36 th International Conference on Machine Learning*, 2019.

Zhou, Y. Non-stationary texture synthesis by adversarial expansion. *ACM Trans. Graph*, 2018.