Exploring Deep Convolutional Generative Adversarial Networks for Image Generation

Haris Javed
COMP3547 - Deep Learning

Abstract

This paper explores the use of Generative Adversarial Networks (GANs) for generating realistic images, specifically using the LSUN Church dataset at a resolution of 128x128. We adapt the original Deep Convolutional GANs (DCGANs) architecture, following the guidelines but modifying the number of channels, filters, and input/output dimensions to suit the dataset's requirements. Using the Adam optimizer with a learning rate of 0.0002 and a batch size of 128, we train the proposed model for 100 epochs. Our results demonstrate that the modified model can produce high-quality images with a range of unique patterns and shapes.

1 Introduction

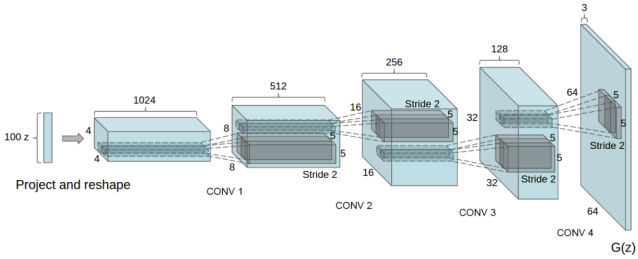
The objective of Generative Adversarial Networks (GANs) [1] is to learn the underlying distribution of a training dataset. The generator G transforms random input vectors z drawn from a simple distribution $\mathcal{P}_z$ into synthetic samples $G(z)$. Meanwhile, the discriminator D aims to differentiate real samples $x \sim P_x$ from the synthetic samples $G(z) \sim \mathcal{P}_G(z)$, resulting in a minimax game. Here, the generator tries to minimize the difference between the generated and real samples, while the discriminator aims to maximize the difference between them. Following from [1, 2], the loss function can be defined as follows:

$$\min_G \max_D \ (\mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))]) \tag{1}$$

The training of GANs has been known to be a challenging task, often resulting in unstable training and nonsensical generator outputs. In order to address this, several methods have been introduced. One such method proposed in [3] is the use of a DCGAN architecture that employs deep convolutional neural networks (CNNs) for both the generator and discriminator.

The DCGAN architecture uses multiple layers of convolutional filters that learn to extract features from images. Specifically, the discriminator uses convolutional layers, while the generator uses convolutional-transpose layers. The key innovation is the use of CNNs for both networks, allowing the model to capture spatial information in input images. Furthermore, the DCGAN model follows specific architectural guidelines [4, 5, 6] that have proven effective for training GANs. These include the removal of fully connected layers to reduce overfitting, the use of batch normalization to stabilize training and reduce initialization dependence, the use of strided convolutions instead of pooling layers to avoid checkerboard artifacts, and the use of transposed convolutions to generate higher resolution images.

In this paper, we build upon the original DCGAN architecture and modify it to train at a 128x128 image resolution. We follow the architectural guidelines of DCGAN and adjust the number of channels, filters, and input/output dimensions to suit the requirements of our chosen dataset.



2 Methodology

2.1 Network Architecture

The Generator architecture is comprised of a series of transpose convolutional layers that progressively upscale the input noise vector to generate a high-resolution image. The model consists of 6 transpose convolutional layers, with each layer being followed by a batch normalization and ReLU activation function, except for the final layer that utilizes a tanh activation function to ensure that the generated image values are within the range of [-1, 1]. The activation function plays a critical role in allowing the model to effectively saturate and cover the color space of the training distribution, thereby accelerating the learning process. A representation of the Generator architecture for a 64x64 resolution is shown above.

On the other hand, the Discriminator takes a 128x128 image as input and outputs a scalar value that indicates whether the input image is real or fake. The Discriminator model comprises of 6 convolutional layers, each followed by a batch normalization, leaky ReLU activation function, and a sigmoid activation function at the final layer to ensure that the output value lies within the range of [0, 1], which can be interpreted as a probability.

Additionally, the weights of both the convolutional and batch normalization layers are initialized using a normal distribution with a mean of 0 and a standard deviation of 0.02.

2.2 Training details

We train our model on the LSUN-Church dataset [7], which is medium-sized (126k images) and reasonably visually complex, using a resolution of 128 $\times$ 128 pixels. Following the approach described in [3], images were resized, center-cropped, and normalized without augmentation. We scaled the images to fit the range of the tanh activation function [-1, 1], used mini-batch SGD with a batch size of 128, and initialized weights from a zero-centered Normal distribution with a standard deviation of 0.02. The LeakyReLU activation function was used with a leak slope of 0.2, and we utilized the Adam optimizer with a learning rate of 0.002 and momentum term beta1 set to 0.5 for stability. Training is performed for 100 epochs.

The binary cross-entropy loss function is used to computer our losses:

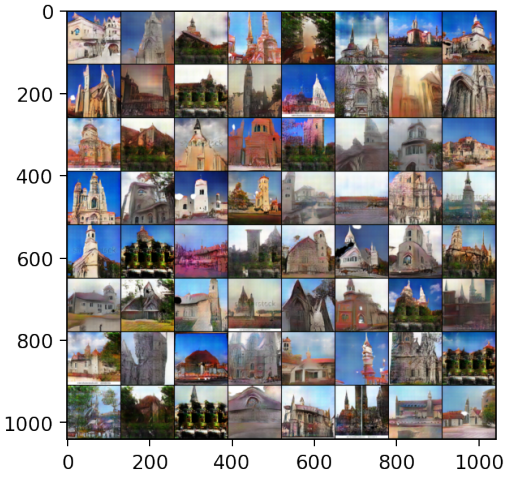
$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)] \tag{2}$$

The discriminator aims to distinguish between real and fake inputs by computing binary cross-entropy loss for batches of real and generated samples. Gradients are accumulated and used to update the discriminator's parameters. The generator seeks to improve the realism of its generated samples by maximizing log(D(G(z))) during training, which provides a stronger gradient signal. This is achieved by classifying the generator's output with the discriminator, computing the loss using real labels as ground truth, accumulating gradients, and updating the generator's parameters.

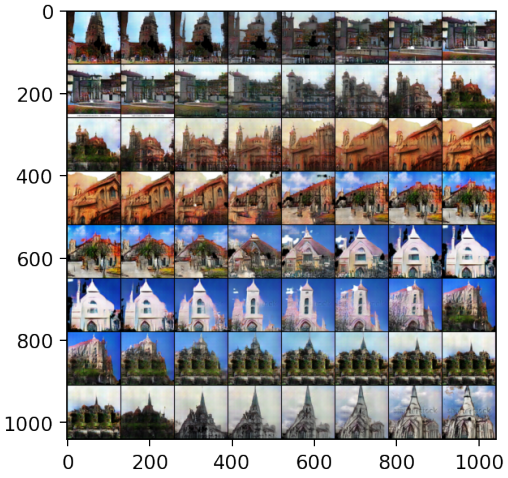
3 Results

Based on the random batch of non-cherry picked samples shown below, we observe that the generative model is capable of separating foreground and background and recognizing

different objects in the generated images. Notably, the absence of mode collapse is a positive indication of the model's capacity to generate diverse samples. However, upon closer inspection, the generated images still exhibit a lack of clarity and photo-realism when compared to real images. Additionally, the generated samples often contain errors in proportions and artifacts, and in some cases, exhibit "white blobs".



The next results show images generated by interpolating between points in the latent space, demonstrating that the space learned has smooth transitions while maintaining a consistent representation of churches. The first row of interpolated images depicts a gothic-style church gradually transitioning into a modern-style church. Similarly, the 8th row displays a church with a domed roof slowly transforming into a church with a pointed roof. However, it is important to note that while the generated images exhibit a level of smoothness in the transitions, they do not possess absolute continuity. Despite this, the generated images retain the overall characteristics of a church and provide a promising demonstration of the model's ability to learn and generate realistic-looking images in the latent space.



The figure below illustrates a selection of generated samples that showcase the most successful outputs produced by the model:

4 Limitations

The methodology used in this paper has several limitations that need to be considered. Firstly, the lack of data augmentation techniques may limit the model's ability to generalize new and unseen data. Secondly, the training time of 100 epochs may not be sufficient for the model to fully converge and achieve optimal performance, thus requiring longer training times. Thirdly, incorporating quantitative evaluation metrics, such as Inception Score or Frechet Inception Distance (FID) would've provided a comprehensive assessment of the model's performance. Finally, while the generated samples show promising results in terms of diversity, they often contain errors in proportions and exhibit "white blobs." Future work could explore recent developments in generative models such as StyleGAN and BigGAN to address these limitations.

References

[1] Ian J Goodfellow et al. *Generative Adversarial Networks*. arXiv.org, 2014. URL: https://arxiv.org/abs/1406.2661.

[2] Axel Sauer et al. "Projected GANs Converge Faster". In: *arXiv:2111.01007 [cs]* (Nov. 2021). URL: https://arxiv.org/abs/2111.01007 (visited on 03/04/2023).

[3] Alec Radford, Luke Metz, and Soumith Chintala. *UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS*. 2016. URL: https://arxiv.org/pdf/1511.06434.pdf.

[4] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv.org, 2015. URL: https://arxiv.org/abs/1502.03167.

[5] Jost Tobias Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *arXiv:1412.6806 [cs]* (Apr. 2015). URL: https://arxiv.org/abs/1412.6806.

[6] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. *Inceptionism: Going Deeper into Neural Networks*. Google AI Blog, June 2015. URL: https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

[7] Fisher Yu et al. "LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop". In: *arXiv:1506.03365 [cs]* (June 2016). URL: https://arxiv.org/abs/1506.03365 (visited on 03/05/2023).