Nicolai Garcia and Sebatianos Hapte-Selasie
Stanford University
nicolaig@stanford.edu | sebhs@stanford.edu

Abstract

The abstract-DCGAN is an AI that generates abstract art that is evocative of the late 20th century Abstract Expressionism movement. Pioneered by the likes of Jackson Pollock and Helen Frankenthaler, the movement disrupted the traditional art world, moving completely away from figurative painting to impressionistic blur. We'd like to make the same impressions with neural networks. The DCGAN has been used in several art generation applications. Most notably, Robbie Barrat used the DCGAN to generate landscapes, portraits, and nude portraits. Using this model, the Paris-based artist collective created *Portrait of Edmond de Belamy*, which sold for $432,500 at the British auction house Christie's in 2018. We were inspired by this project and wanted to follow its implementation with the focus on generating abstract artwork. The issue with abstract art is that it doesn't have common patterns like landscapes and portraits that were used in other projects. By rethinking how we create our dataset we were able to generate new abstract art.

Introduction

The Generative Adversarial Network (GAN) is a machine learning framework and class of generative models that is popular in the Artificial Intelligence field of unsupervised learning. The model architecture the authors proposed includes two neural networks: a **Generator** and a **Discriminator**. At a high-level, the generator is learning to generate new data based on the input data, and the discriminator is learning to distinguish between the input data and the generated data. In this way, the GAN can be used to create realistic images of faces, hand-drawn characters, paintings, and other images the model is trained on.

This paradigm works well for generating figurative images, like faces or pictures with mountains, coastlines and trees. Although the representations of these subjects can change from image to image (female vs. male faces, landscapes in different places in regions have different colors and species of wildlife), there are common representations among all of the images that the GAN is able to recreate.

The challenge with generating abstract artwork is that abstract works have no such distinct visual features, but instead involve a range of colors, textures, and shapes. The common GAN pipeline, from input data to model architecture and output size, fails when trained on a dataset of abstract artwork.

Our solution, therefore, is to adapt the typical DCGAN to generate non-figurative artwork. To do so, we introduce methods of input data augmentation and non-symmetrical convolutional layers in the Generator and Discriminator.

Related work

The GAN was introduced in Ian Goodfellow's paper *Generative Adversarial Nets*[1]. Here the researchers detail the GAN paradigm (described above) and the results of the model on the mnist dataset.

The Deep Convolutional Generative Adversarial Network (DCGAN) is an extension to the GAN, introduced in 2016[2]. The DCGAN incorporates Convolutional Neural Networks (CNN), a popular framework used in many Computer Vision applications, in the existing GAN architecture, and has since been used in many image-generating GAN implementations. The DCGAN uses a deconvolutional generator to generate images and a convolutional discriminator to classify images as real and fake.

Common applications of the DCGAN have been used to generate images of faces, using the celeba dataset (http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html), or handwritten characters, based off of the mnist characters dataset. The latter outputs black and white images, meaning the final output layer has only one channel, while the face generation model (and any other DCGAN generating RGB images) output images with 3 channels—however the output image is typically sized 32x32 or 64x64.

The DCGANs used for art generation differ in their implementations. Most follow the same architecture used for other image generation applications, changing only the input dataset to produce different results. We found that these implementations were not as effective.

Robbie Barrat's implementation adapts the original DCGAN implemented by Chintala et al. but adds several features to optimize the model for art generation. Therefore, we chose to model our DCGAN off of his.

Dataset and Features

For the dataset we needed a large number of abstract art work which we got from this dataset (https://github.com/cs-chan/ArtGAN/tree/master/WikiArt%20Dataset), there we used the images that fell under the category Abstract_Expressionism, which gave us a dataset of 2750 images. The beauty of art is created by the observer and for this project it was important to us that we enjoy the paintings that our algorithm produces. Thus, the primary metric of deciding rather the algorithms works were our subjective opinions, thus there was no need of a test or validation set. After working and testing different approaches, we realized that the dataset was too diverse in order to produce good results, thus we decided to only use a small dataset of 150 paintings that look quite similar (see appendix)

We then used heavy data augmentation like flipping, mirroring, rotating, and cropping in order to create 34,481 images out of the original 150. Each sample images was sized to 128x128.

[1] Goodfellow et al.
[2] Chintala et al.

Methods

Both the Generator and Discriminator are implemented as Convolutional Neural Networks (CNN), using strided convolutions and no pooling or fully-connected layers. The Generator creates images using a randomly-generated noise vector with shape (100,) and passing it through 5 deconvolutional (transpose convolutional) layers. The output is a 128x128 Image with 3 channels for RGB values. The Discriminator essentially reverses this architecture, taking as input 128x128x3 images and passing it through 5 convolutional layers with a one-node output layer. The output of the Discriminator is the raw value or logit of the probability that the input image has the assigned label. (See appendix for Generator & Discriminator architecture)

We use more filters for each convolutional layer in the Generator than we do in the Discriminator. Specifically, we define the variable `ngf` to denote the number of generative filters in the final convolutional layer (128 in the generator graphic above). Therefore, there are `ngf*16` filters in the first convolutional layer (5 layers, number of filters is halved in each layer). `ndf` denotes the number of filters in the first convolutional layer in the discriminator (64 in the discriminator graphic above). There are `ndf*16` filters in the final convolutional layer. We made `ngf > ndf` meaning each convolutional layer in the generator had more filters than its counterpart layer in the discriminator. This is because in the case of abstract artwork, the Generator has a much harder job to learn than the Discriminator. With more filters in each convolutional layer, the Generator learns more parameters to minimize its loss function.

In each step of training, a batch of real images (batch size 64) and generated images (fake images) are input to the Discriminator. The loss for the Discriminator is computed as the sum of the binary cross-entropy loss on the real image predictions, using a vector of 1's as the true labels and the binary cross-entropy loss on the fake image predictions, using a vector of 0's as the true labels. The loss for the Generator is computed as the binary cross-entropy loss on the fake image predictions, using a vector of 1's as the true labels. In this way, the Discriminator is trying to minimize its loss by correctly classifying real and generated images, while the Generator is minimizing its loss by learning to generate images that look real.

We use the binary cross-entropy loss that is included with keras. This loss function computes the following equation:

$$BCE = \frac{1}{N}\sum_{i=1}^{N} y_i log(p(y_i)) + (1 - y_i)\, log(1 - p(y_i))$$

Where y_i is the assigned label for that image (1 for real, 0 for fake) and $p(y_i)$ is the probability that the image has that label, as output by the Discriminator. N represents the number of images in that batch.

When using the algorithm we realized that the outputs kept on being unsatisfying. Even with a relatively large dataset of abstract images - the outputs did not resemble our imagination of what the algorithm should output. We realized that previous approaches of creating art with GANs used input that has common features, which our dataset didn't. In order to change this we went through the dataset and

selected 150 paintings that looked somewhat similar (many colors, no recognizable shapes, see appendix). Then we used data augmentation to create a large enough dataset for our DCGAN to work. For each image we took the original and a 90 degree rotated image, then flipped, mirrored, and flipped+mirrored each image. Then for all these images we divided each image into several 128x128-sized images (imagine passing a 128x128 frame over the image). With this method we could get several hundred images out of each original artwork, which gave us a dataset of 34,481 images. With this dataset we were able to achieve our goal.

Experiments/Results/Discussion

We implemented our original model adapting the popular implementation in Chintala et al. to generate RGB images. We ran the DCGAN on the original, full dataset of abstract artwork (Abstract_Expressionism) without augmentation (see appendix, *first iteration*).

The next thing we tried was to use data augmentation (like flipping and mirroring) to train the model on 10,000 images (up from the original 2,750). (see appendix, *second iteration*).

These were still not what we expected. After several tweaks to our model, such as changing the number of filters and activation of each convolutional layer in the generator from ReLU to Leaky ReLU, changing the activation of the output layer in the generator from tanh to sigmoid, we were not getting any better results. We still didn't know, however, if the issue is with the dataset or with our model. In order to test our model we ran it on a dataset of human faces and on one with landscapes and were satisfied with the results (obviously we didn't tweak the model to work for these kinds of dataset, but we could prove that the model works, see appendix)

This made us realize that there may not be an issue with our model's architecture, but rather the dataset we were using.

We realized that the issue with the traditional GAN implementation we followed was that it will fail on a dataset of abstract artwork, because there are no common visual patterns for the GAN to learn. Therefore, our model was running into the problem of mode collapse, as the generator learned one poor type of output that minimized the discriminator's loss function, but failed to learn various different representations.

Thus, we used a method of data augmentation to make a more "homogenous" dataset. Rather than using a dataset of artwork from different artists and time periods with different styles, we focused on a few artworks, synthesizing new input data from these.

The result of our DCGAN following 100 epochs of training on this new dataset are below:

Conclusion/Future Work

In conclusion, our model differs from the standard DCGAN implementation in both the architecture and dataset used. The model architecture includes an additional convolutional layer to standard DCGANs and each convolutional layer in the Generator has more filters than those in the Discriminator. Further we used a much smaller data set then other implementations and relied on a lot of data augmentation.

The next work to be done with the abstract-DCGAN is in taking the 128x128-pixel output images and enlarging them to be clearer to the human eye (~1024x1024 pixels). The methods for this involve linear interpolation of each pixel value to create more pixels, and choosing the proper method of resampling pixels when enlarging the images.

We would also like to add an evaluation metric for our artworks by showing them to humans in relation to the real abstract artworks we trained on. A simple method for this would be to put an abstract artwork next to an artwork that our model generated and ask the viewer to choose which painting is their favorite. This would give us an idea of how our generated images compare to real artworks.

Contributions

Nicolai built the DCGAN, modelling it after Chintala's and Barrat's implementations. He also used his aws account to train the model and performed the experiments.

Sebastianos compiled the original dataset of artworks and built the data augmentation technique.

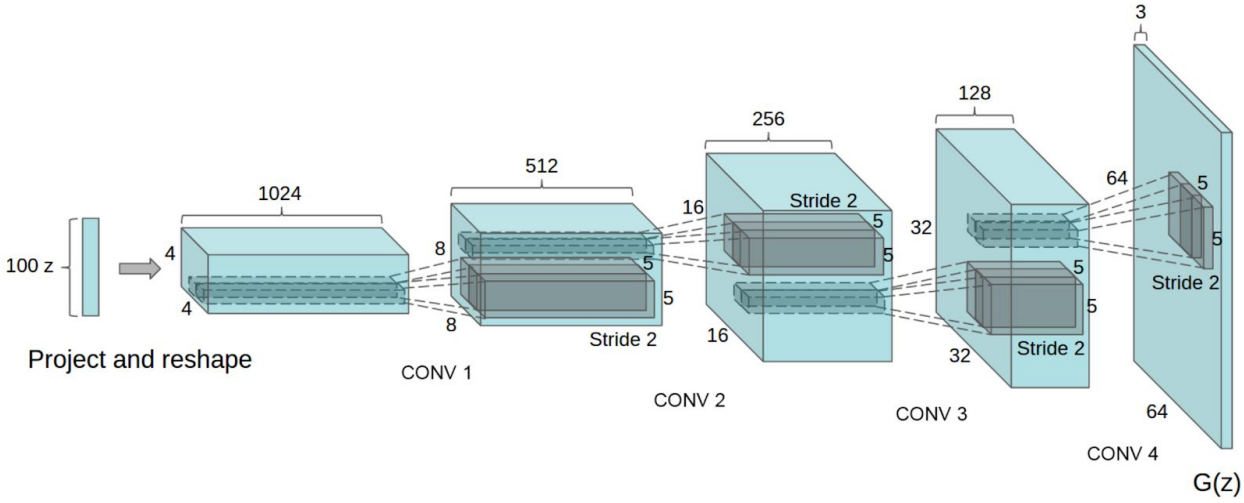
Throughout the project we regularly communicated how to achieve progress on our project and what steps each team member has to take.

References

1. Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Medhi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, Bengio, Yoshua. *Generative Adversarial Nets*. Montreal, 2014.
2. Chintala, Soumith, Radford, Alec, Metz, Luke. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.* 2016.

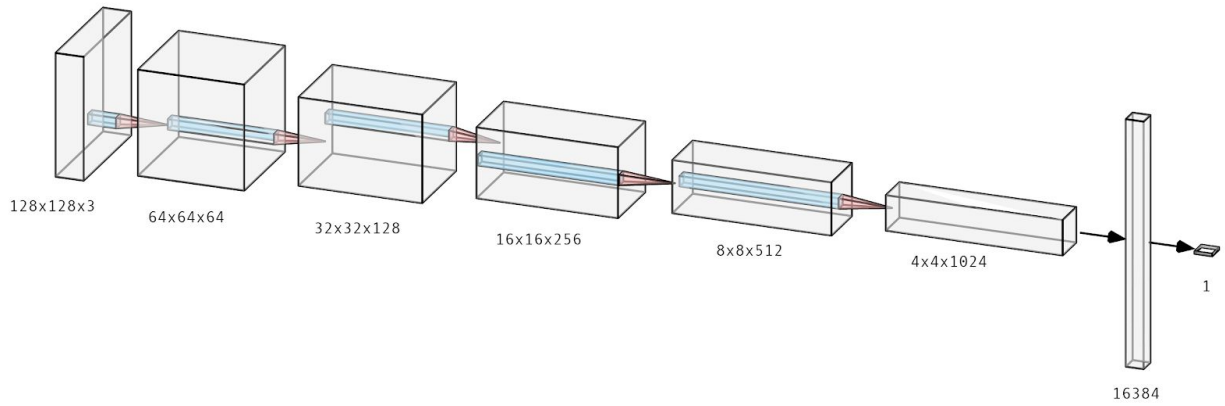
Appendix

Generator architecture



Generator architecture (note that our generator uses 5 convolutional layers which is an extra convolutional layer than the graphic above so that the output image size is 128x128)

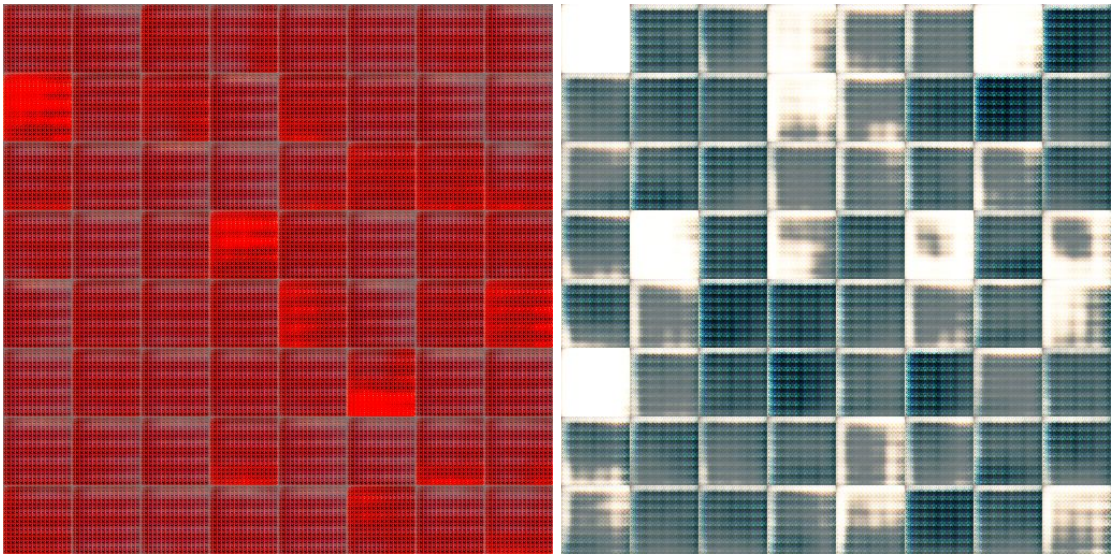
Discriminator architecture



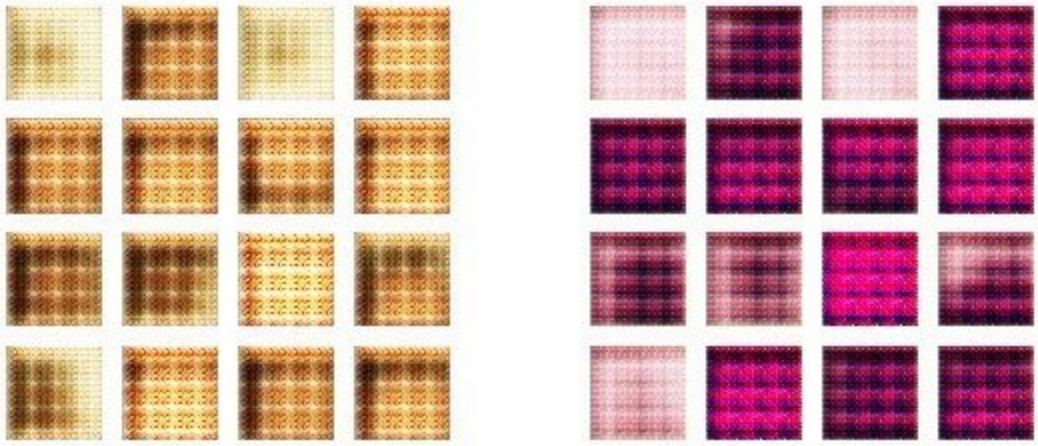
Example images from final dataset (selected 150 images)

Example images from original dataset (~3500 images)

Results from first iteration



Results from second iteration



Testing model using faces dataset

Testing model using landscapes