Comparing the interpretability of different GAN architectures

Henry Tu
University of Toronto

Seel Patel
University of Toronto

1 Abstract

A Generative Adversarial Network (GAN) uses an adversarial process between two models which are simultaneously trained to estimate a generative model.[1] There are many variants of GAN architectures, such as COCO-GAN[2] and StyleGAN[3], which both have the ability to generate synthetic images which mimic real images.

We are exploring methods of comparing the quantitative performance of these architectures with their interpretability. Our quantitative measures include C2ST[4], image quality measures[4], etc.

In order to analyze these networks qualitatively, we will use methods such as Latent Space Exploration[5] and nearest neighbour tests[4] to decipher what the networks have learned.

By combining these two forms of analysis, we hope to gain insight into the relationship between performance metrics and the generated output of the models.

2 Introduction

StyleGAN and COCO-GAN generate images using different techniques to accomplish two different goals: StyleGAN is designed such that high level features of the output image can be finely tuned and adjusted (e.g. lighting, hair colour, etc.)[3]. On the other hand, COCO-GAN generates each part of the image separately before stitching it together in order to simulate the human perception of vision[2]. As a result, we expected each model to generate images with different qualities that relate to their designed tasks.

3 Dataset Generation

Both GAN models were trained on the LSUN Bedroom dataset[6] at a resolution of 256×256. Due to time constraints, we used pretrained models to generate images for our analysis.

3.1 StyleGAN

NVIDIA Research Projects' Official TensorFlow Implementation of StyleGAN pretrained to the LSUN Bedroom dataset[7] was used to generate 5000 images for analysis. 5000 latent code vectors $\mathbf{z} \in \mathbb{R}^{512}$ drawn from a standard normal distribution which each correspond to an output image.

3.2 COCO-GAN

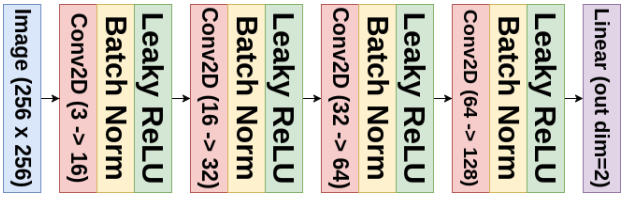
The TensorFlow implementation of COCO-GAN by its authors pretrained to the LSUN Bedroom dataset[2] was used to generate 5000 images for analysis. 5000 latent code vectors $\mathbf{z} \in \mathbb{R}^{69}$ were drawn from a uniform $[-1, 1]$ distribution which each correspond to an output image.

4 Quantitative Analysis

4.1 C2ST

Classifier Two-Sample Tests (C2ST) is used to predict if two samples came from the same distribution[8]. In the context of evaluating GANs, we will use C2ST to quantitatively measure how well the models mimic real images. We trained a deep convolutional neural network to perform binary classification based on the recommended discriminator architecture from *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*[9]. The goal is to compare the margin by which each GAN model is able to trick the classifier to quantify image generation quality.

The classifier was implemented in Pytorch with CE loss, the Adam optimizer ($\alpha = 0.1$), and the following network architecture:



For all Conv2D layers, **Kernel Size: 3, Stride: 2, Padding: 1, Leaky ReLU slope: 0.2**. Argmax is taken across output of linear units for classification.

To avoid introducing bias, all three datasets (training, validation, test) were equally balanced with real and generated images (5000 each). The generated images were further divided in half between images from StyleGAN and COCO-GAN. Out of the combined pool of 10000 images, 5000 were allocated for training, 2500 validation, and 2500 testing.

The model was trained with a batch size of 200 for 1000 epoches. After tuning hyper parameters and performing early stopping with the validation dataset, the following results were attained:

Table 1: Real vs (StyleGAN or COCO-GAN)

Data Set	Accuracy
Training	75.9% of 5000
Validation	67.6% of 2500
Test (Real)	58.2% of 1250
Test (StyleGAN)	60.5% of 625
Test (COCO-GAN)	89.6% of 625

The model was able to correctly identify the origin of an image the majority of the time. This data suggests that StyleGAN is better at tricking the classifier than COCO-GAN.

To confirm this, we trained additional classifiers using the same architecture, data set distribution, and hyper-parameters but only using images from one of the GAN models.

(i.e. all 5000 generated images are all from one GAN model) The goal is to test how well the classifier performs when isolated to a specific GAN.

Table 2: Real vs StyleGAN	
Data Set	Accuracy
Training	77.6% of 5000
Validation	68.1% of 2500
Test (Real)	74.6% of 1250
Test (StyleGAN)	61.4% of 1250

Table 3: Real vs COCO-GAN	
Data Set	Accuracy
Training	87.9% of 5000
Validation	88.2% of 2500
Test (Real)	96.4% of 1250
Test (COCO-GAN)	76.1% of 1250

As seen here, the Real vs COCO-GAN classifier (Table 3) performed significantly better than the Real vs StyleGAN classifier (Table 2). This supports the hypothesis that StyleGAN mimics real images better than COCO-GAN.

Although these results suggest it is more difficult for a classifier to identify a StyleGAN generated image, it by no means shows that it is impossible to train a more accurate classifier. Therefore, we must put the information gathered in the context of the other metrics in order to draw a stronger conclusion.

4.2 Image Sharpness

A common indicator of image quality is sharpness. When an image is sharp, the details within the image are seen more clearly and there is a better distinction between different objects in the picture. Overall, this allows humans to form a better understanding of what they are seeing when they view the image. Using directional image gradients along the x and y axes, we measured the mean image sharpness of the generated images and real images as seen below. (Higher is better)

Therefore the real images are the sharpest and StyleGAN does the better job of producing sharper images than CocoGAN. However, does higher sharpness always mean better image quality? We explore this question in the combined analysis section below.

4.3 Color Distribution Comparison

The purpose of a GAN is to generate new images that match the distribution of the training images. Therefore, we decided to statistically compare the distribution of generated images with the distribution of real images in order to understand how well the GANs have fit to the training data.

In order to properly mimic the real images, the generated images have to closely follow the color distribution of the real images. Each image in our data is represented by RGB values ranging between 0 and 1. We extracted the frequencies of RGB color values using 25 bins for each of the datasets. The distribution of these frequencies are plotted below for the

real images and each set of generated data.

Insert images

As seen in the frequency plots, both StyleGAN and CocoGAN do a good job of modelling the real distribution of colors. However, both generated image frequency plots show many spikes that are not present in the real image frequency plots. We will explore these spikes in our combined analysis section. Additionally, to quantify this color reproduction quality, we also took the KL Divergence between the color distributions of each model to the real images.

Therefore CocoGAN does a statistically better job of modelling the color distribution of the real images. Note that this does not guarantee that CocoGAN is superior because this method of comparison does not take into account the spatial position of the colors. This means that the images would post the same KL Divergence even if the pixels were scrambled. Therefore further analysis is needed to determine which model performs better overall.

5 Qualitative Analysis

5.1 Latent Space Exploration

An important part of the generator is it's ability to combine different features from the real image in order to generate new images. For example, in our dataset, an image could contain a bed, a window or a painting. What about a combination of these? What if the locations of these features is changed? We use latent space interpolation to determine if the model produces robust images in all of these cases.

5.2 Nearest Neighbours

A common problem within GANs is mode collapse. This occurs when the generator consistently outputs the same image for distinct latent vectors. In addition to this, we want to identify if the generator creates multiple images with identical structure or color scheme. To identify these issues, we randomly sampled generated images and visually compared them with the k nearest neighbours from the images produced by the same model. In our search, we found no evidence of mode collapse in either StyleGAN or CocoGAN. Samples from the images used in this analysis are provided below.

6 Combined Analysis

6.1 Colour Frequency Peaks

In our analysis of color frequency distributions above, we noticed that both StyleGAN and CocoGAN exhibited spikes in their frequency graphs. One prominent spike occurs simultaneously in the green and blue channel at intensity level 0.6. This occurs in both the StyleGAN and CocoGAN generated images. Specifically, this spike happened in colors covered by the following gradient.

Insert Gradient

We decided to sample some of these colors and find generated images with color schemes closely matching said colors. The images matching color $(0.3, 0.6, 0.6)$ can be seen below for both StyleGAN and CocoGAN.

Cocogan Image
Stylegan Image

For both models, we are seeing an increase in noise and distortion in the images. This is especially true in the case of CocoGAN, for which every image displays noisy textures or bad structure. What does this trend tell us about CocoGAN? From the samples, we can see that many of these images have windows in them that display the sky. CocoGAN has done a relatively poor job of displaying the sky and natural imagery in the background.

Additionally in the CocoGAN images we can see that much of the color from the sky bleeds into other parts of the image. These sky colors are very similar to the colors in the gradient we are studying. Therefore we theorize that this spike in frequency occurs from CocoGAN being unable to properly generate images that include the sky.

6.2 High Sharpness Images

In our analysis of image sharpness, we found that StyleGAN produced sharper images than CocoGAN. Neither model was able to produce images as sharp as the training data. We want to qualitatively explore what higher sharpness means in the case of these models. Below we have plotted the sharpest images produced by both models.

INSERT IMAGES

As we can see, the StyleGAN images are of high quality, but the CocoGAN images are consistently filled with noisy patterns. Both sets consist of very sharp images, and therefore we have visualized the sharpness below in order to understand why there is such a large difference in quality.

INSERT SHARPNESS IMAGES

By comparing the sharpness plots to the images, we can see that the sharpness of the StyleGAN images comes from it's ability to effectively generate complex patterns. However, the sharpness of the CocoGAN images comes from unwanted noise or messy patterns which are often out of place. Therefore, even though both models produce images with a relatively similar sharpness, the sharper images produced by StyleGAN are vastly superior to those produced by CocoGAN. In conclusion, the qualitative results indicate that the true quality of CocoGAN images may be lower than the numbers indicate.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. (2014) *Generative Adversarial Networks*

[2] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, Hwann-Tzong Chen. (2019) *COCO-GAN: Generation by Parts via Conditional Coordinating*

[3] Tero Karras, Samuli Laine, Timo Aila. (2018) *A Style-Based Generator Architecture for Generative Adversarial Networks*

[4] Brownlee, Jason. (2019)

[5] Tom White. (2016) *Sampling Generative Networks*

[6] Yu, Fisher and Zhang, Yinda and Song, Shuran and Seff, Ari and Xiao, Jianxiong. (2015) *LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop*

[7] NVLabs. (2019) *StyleGAN — Official TensorFlow Implementation*

[8] David Lopez-Paz, Maxime Oquab. (2016) *Revisiting Classifier Two-Sample Tests for GAN Evaluation and Causal Discovery*

[9] Alec Radford, Luke Metz, Soumith Chintala. (2015) *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*