Comparing the interpretability of different GAN architectures

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 DCGAN[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], Maximum Mean Discrepancy (MMD)[4], etc.

In order to analyze these networks qualitatively, we will use methods such as SmoothGrad[5], Latent Space Exploration[6] 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.

Related Works

As detailed in *How to Evaluate Generative Adversarial Networks*[4], the following methods can be used to quantitatively evaluate the performance of the GANs:

1) **C2ST[4]:** Train a binary classifier that predicts if two input images come from the same distribution. Train using real images from the test set, and fake images from the trained models.
2) **Image Quality[4]:** Quantify the structure, sharpness and presence of noise within the outputs from the GANs.
3) **Maximum Mean Discrepancy (MMD[4]:** Measure the dissimilarity between the distribution of the real images, and the distribution of images generated by each model. Take a random sample of each distribution to do so.

We will then analyze both models qualitatively as detailed below:

1) **SmoothGrad[5]:** Method to find pixels in the input image which most greatly affects the output. This is done by performing back propagation to the input image to build a sensitivity map. This process is done multiple times with different noise maps to provide variation in the data. The results are finally averaged out to produce a more general estimate of the sensitivity map.
2) **Latent Space Exploration[6]:** To understand how changes in input correspond to changes in the output image, we will explore the input space of the models. This includes interpolation, sampling with fixed values, and moving along nonlinear functions.
3) **Nearest Neighbours[4]:** Uses distance measurements, such as the Euclidean distance between pixel values, to find the nearest image in the training set. This provides context about how well the model is performing compared to a real image and can be used to detect overfitting.

Method

We will begin by splitting our dataset into training, test and validation sets. Note that we will need a large quantity of test data to accommodate the training of classifiers for our quantitative metrics.

Next we will implement and train both models in PyTorch using the recommended settings from the original papers. The architectures may be modified to match our dataset, and we will tune hyperparameters to get the best qualitative results from the models.

The next step is to move onto scoring the models using quantitative and qualitative metrics as detailed in Related Works. We will score each model using each metric and provide a detailed breakdown of the results along with our interpretation of how each model performs separately.

The qualitative metrics will be used to explain and justify the results of the quantitative metrics, and vice-versa. The results from each model will be compared with the purpose of each metric taken into consideration. We will also analyze if superior quantitative performance is reflected when comparing the qualitative results from both models. For example, if we find that one model produces noisier images, we will look for explanations within our quantitative results to determine the cause.

The results from comparing the models will be used to detail the strengths and weaknesses of each model. We will end the report with a breakdown of possible relationships / hypotheses between metrics we discovered when analyzing our results.

Summary and Reference

(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 https://papers.nips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

(2) Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” (2016). https://arxiv.org/pdf/1511.06434.pdf

(3) Tero Karras, Samuli Laine, Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”, 2018. https://arxiv.org/pdf/1812.04948.pdf

(4) Brownlee, Jason. “How to Evaluate Generative Adversarial Networks.” Machine Learning Mastery, 12 July 2019, https://machinelearningmastery.com/how-to-evaluate-generative-adversarial-networks/.

(5) Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viegas, Martin Wattenberg. “SmoothGrad: removing noise by adding noise”, 2017. https://arxiv.org/pdf/1706.03825.pdf

(6) Tom White. “Sampling Generative Networks”, 2016. https://arxiv.org/pdf/1609.04468.pdf