Generating building images with RIT features using DCGANs (November 2019)

Brian Landy, BSMS Computer Engineering, Rochester Institute of Technology

Abstract—Deep Learning models can do many things that humans cannot. They can even extract features and reproduce fake data. Deep convolutional generative adversarial networks (DCGAN)s are an interesting part of the deep learning body of knowledge as they can be used to create synthetic input to strengthen classification model accuracy. This experiment will demonstrate the effectiveness and behavior of DCGANs in creating believable generated images based on features of Rochester Institute of Technology buildings. Deep convolutional generative adversarial networks are an extension of generative adversarial networks with the difference being that they use image convolutions to extract different types of features, yielding artificial output. Many images of the Rochester Institute of Technology campus were captured and processed in order to train a DCGAN. Several experiments with model changes were run and output was assessed visually at points during training. By varying some hyperparameters of the model from the baseline like learning rates and filter dimensions, it is shown much improvement can be made to output image quality. The task of generating truly believable output was much more difficult to deliver on.

I. Introduction

Performance in deep learning tasks continues to increase as more data is collected and used in training, as mentioned by Alom et al. [1]. In some cases, there aren't enough samples to effectively train a model and so generative approaches are used to make more data. Goodfellow et al. [2] proposed a new type of generative network, the generative adversarial network (GAN), which is an unsupervised model where a generator network which tries to fool a discriminator network with fake samples. Since the first introduction of GANs, many variants have been proposed and successfully applied. GANs are computationally expensive and are even more so with the application of convolutional layers for feature extraction on images. The application of convolutional layers to GANs is introduced by Radford et al. [3] where the group proposes an architecture of GANs for better image generation. GANs have many parameters that can be altered to increase output quality, and better train the model, as demonstrated by some of the techniques presented by Salimans et al. [4]. A couple of these techniques were tried in the experiments of this project. A similar facade generation task was performed by Bachl et al. [5] and this paper has some similarities in methodology to the DCGAN attempts Bachl et al. [5] had made. The rest of this paper will cover topics in this order: the Background will explain DCGANs and the scope of the generation task, the Proposed Method section will discuss the choices made to tune the DCGAN model to generate better fake Rochester Institute of Technology (RIT) images, the results section will cover the best case outcomes and parameters and the conclusion will briefly restate findings and outcomes. Background information about GANs and less important points are in the appendix.

II. Background

A. DCGANs and Convolutional Feature Extraction

Radford et al. [3] proposed a DCGAN architecture than applies the image extraction properties of convolutional neural networks (CNN) to the generative ability of GANs. The discriminator network functioned similarly to a binary classifier convolutional neural network. To generate samples, transposed convolutional layers are employed by Radford et al. [3]. Transposed convolutions are opposite to convolutional layers in that they produce an output larger than the input and are meant for up sampling. Figure 1 shows the architecture chosen for the generator network in this experiment from Radford et al. [3].

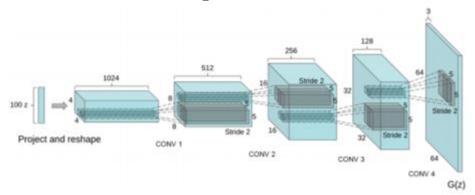

Fig. 1. Architecture for a Deep Convolutional Generator Network. In the figure, transposed convolutional layers and leaky ReLU activations transform random noise vectors into image features. Fig. 1 is from Radford et al. [3].

B. Tools and Chosen Architecture

The chosen model to use as a baseline for this was the Pytorch DCGAN example from Radford et al. [6]. DCGAN was experimented with by Bachl et al. [5], but the goal in that paper was to generate new building images with datasets from many cities. The DCGAN model had trouble with the dataset and so some restrictions have been placed on the profile of building images allowed in the custom RIT dataset to negate these issues. They are listed in the datasets section of this paper. These images were processed in batches of 64 at a time.

C. Datasets and Collection Methods

This project required the collection of images of RIT. Video was taken and the images were broken out frame by frame. This was done on a smartphone camera and sampled down to 64 pixel wide square images. The first captured set was of one side of the bioscience building and it is a minimally variational image set. By using this, the model should be able to easily overfit the set and produce one type of image. A larger, highly variational, set was also collected. The goal with this set was to capture front facing images, and to vary the type of buildings as the subject of the photos. This set has 29,297 images. LSUN churches is an image set of churches and was also used as a preliminary test with this DCGAN codebase.

D. *Modifications to DCGAN to Improve Output Believability*

Many hyperparameters of the DCGAN model were modified. Some techniques used and assessed were: weakening the discriminator by lowering its learning rate, implementing soft labels, swapping the activation functions to all leaky ReLU in both models or changing the negative slope, varying the number of the generator filters, varying the adaptive moment estimation (Adam) optimizer regularization, lengthening the number of training epochs, and adding noisy, flipped, and rotated samples to provide a larger set to train on. Combinations of many of these were also tried. Some of these options were tried and shown to be effective in [4] and others were less justifiable. Nonetheless, many things were evaluated.

III. Results

A. *Baseline Results on All Image Sets*

Loss functions for D and G and prediction output of D for real and fake samples were plotted during training. Using an unmodified DCGAN [6] as a baseline, training on LSUN churches for 75 epochs produced the fake images seen in fig. 2.

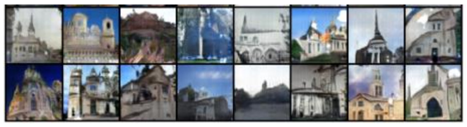
Fig. 2. LSUN churches had fake images resembling churches but when looked at closely, it is still clear that they aren't real. But from a distance and with a lower attention to detail, these are passable.

The low variation RIT set was run with the baseline for 560 epochs and generated images are shown in fig. 3.

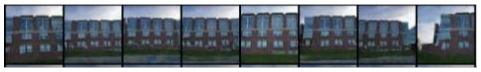
Fig. 3. RIT low variation dataset. The output is very similar to the input and this is as expected due to the model overfitting the one building face.

The highly variable set was trained with, and at 323 epochs, the baseline DCGAN generated the images shown in fig. 4.

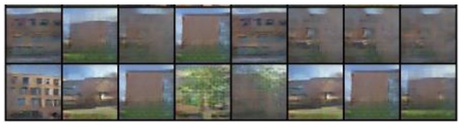
Fig. 4. RIT high variation dataset. The output is noisy, low quality, and clearly fake. The goal of the paper is to produce images that look better than this. G Filter depth: 64 G learning rate: 0.0002, D learning rate: 0.0002 Epoch: 323

B. *Troubleshooting Based on Loss Plots and D Output*

Plotting loss proved to be an effective way to diagnose issues with the model early on ass suggested by Chintala et al[7]. If the D loss went to 0 way too early, then the discriminator became too good at classifying fakes and the generator loss kept increasing rapidly. The fake output samples looked terrible under these conditions. Over time G loss should keep increasing, but not radically, because it needs to do more and more work to fool a constantly trained discriminator until it satisfies its objective. The output of the discriminator was also plotted for real and fake samples to see if D was being fooled. Since the dimensions of the input are so large, it is difficult for G to properly fool D, even when the model produces believable fake images. Due to this, the information was not as useful as hoped for. Ideally though, the output of D would end up moving away from 0 for fakes and away from 1 for real images as the classifier would start to be fooled. Both types of plots, loss and D output, are shown for reference in the appendix.

C. *Best Results of Modified Runs on RIT Datasets*

The samples determined to be the best of each model were handpicked as every model had produced several bad samples. The best output produced came from training with a larger number of filters in the generator, training for more epochs and using a lower discriminator learning rate than the generator. Shown in fig. 5 is a picked example of believable quality output.

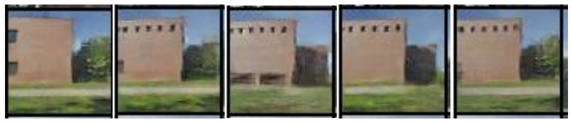
Fig. 5. RIT High Variation Set Fake Samples Best Output. Filter depth: 128 G learning rate: 0.0002, D learning rate: 0.0001 Epoch: 690-700, and One-sided Noisy Labels [4]. Images were taken from a few different epochs to show the emergence and disappearance of RIT building features in generated facades.

These images do a good job of conveying the small feature additions and disappearances that can happen during training. Some samples have trees and windows of a certain style and others do not as training continues. Again, these results had to be hand selected from all the generated images as each trained model still had good and bad outputs. But the more believable images generated in fig. 5 were better than 4, visually speaking.

IV. Conclusion

After training these DCGANs, it is apparent that there is a lot to the process of generating believable fakes. Aside from fig. 5, there were other models with decent fakes too, but they occurred less frequently and there were some models with very obviously fake output. Unfortunately, it seems that some of the more realistic fake images in most cases look very similar to real buildings from the input samples. A final observation is that the best-looking fakes looked like real samples but had small detail differences like missing features or extra features and were not radically different fakes as hoped for. The output quality benefits greatly from modifying certain parameters over others like weakening the discriminator and adding more filters, but when evaluating those changes, much time is spent pouring over large amounts of images that barely scratch the surface of a convincing fake.

V. References

[1] Z. M. Alom, et al.,. “The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches,” arXiv, 09, Sep. 2018 [https://arxiv.org/ftp/arxiv/papers/1803/1803.01164.pdf]

[2] I. J. Goodfellow et al., “Generative Adversarial Networks,” arXiv, 10, Jun, 2014 [https://arxiv.org/pdf/1406.2661.pdf]

[3] A. Radford, et al., “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” arXiv, 19, Nov. 2015 [https://arxiv.org/pdf/1511.06434.pdf]

[4] T. Salimans, et al., “Improved Techniques for Training GANs,” arXiv, 10, Jun. 2016 [https://arxiv.org/pdf/1606.03498.pdf]

[5] M. Bachl, et al., “City-GAN: Learning architectural styles using a custom Conditional GAN architecture,” arXiv, 3 Jul, 2019 [https://arxiv.org/pdf/1907.05280.pdf]

[6] S. Chintala, Pytorch dcgan example, github, [https://github.com/pytorch/examples/tree/master/dcgan]

[7] S. Chintala, Gan Hacks, github, [https://github.com/soumith/ganhacks]

VI. Appendix

A. Generative Adversarial Networks

Goodfellow et al. [2] presented the GAN which is made of two dueling networks. The networks used in the DCGAN are convolutional neural networks. A generator network takes a random input vector and passes a fake generated sample to a discriminator. The discriminator takes in real and generated samples and then determines whether a sample is real or fake. The training process is complex as it relies on the training of two networks, but if the discriminator detection confidence has been reduced to 50% for all samples (real and fake), then the generator has successfully fooled the discriminator. This section will outline important training equations for GANs as first described in Goodfellow et al. [2]. Discriminator networks are meant to maximize (1), which represents properly labeling a real sample *x* with a 1.

$$Goal_D = \max_D \log\left(D(x)\right) \tag{1}$$

Generator networks are meant to minimize (2) which represents properly labeling a fake sample 0.

$$Goal_G = \min_G \log\left(1 - D(G(z))\right) \tag{2}$$

These goals are meant to be met simultaneously by the networks. The general architecture for a GAN includes a real sample set, a random noise generator input, a generator network and a discriminator network. Shown in fig. 6 is the architecture for a standard GAN.

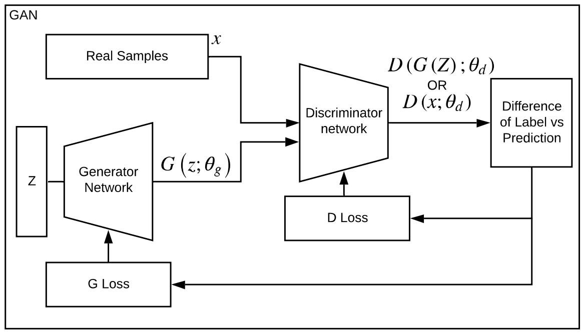

Fig. 6. Architecture for a GAN. In the figure, real samples, *x*, and fake samples, G(z), are passed into a discriminating network that will predict a label, D(G(z)) or D(x). The comparison between real and predicted labels are used to update discriminator and generator loss.

Goodfellow et al. [2] outlines the training process for GANs. Model updates occur in two steps. The discriminator network (D) is trained with *m* real and *m* fake samples. The weights of D are then adjusted using (3) which comes from Goodfellow et al. [2].

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]. \tag{3}$$

The Generator network (G) is updated one time after updating D. This is shown in (4) which is also provided by Goodfellow et al. [2].

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right). \tag{4}$$

The updates are performed with stochastic gradient descent and these are the gradients produced by the cost functions.

B. A Few of the Baseline Samples for Reference

To get an idea of the colors and quality of the regular training samples fig. 7 was included.

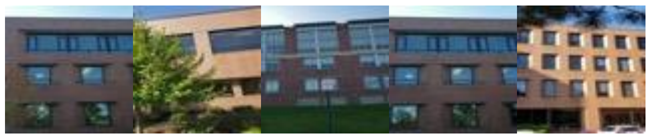

Fig. 7. RIT high variation dataset sample images.

C. Plotted Loss and Discriminator Output

An example of the loss plot mentioned is shown in fig. 8. This specific case has noise added to the real and fake samples as soft labels in order to activate both portions of the binary cross entropy loss function.

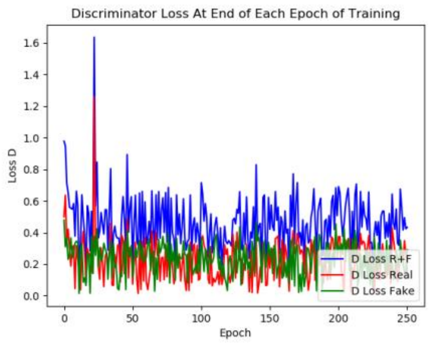

Fig. 8. D, G and the sum of D and G loss can be seen on the plot.

These helped a small amount and were at least interesting for trying to compare training behavior of different models. As for discriminator output, fig. 9. shows a typical discriminator output result during training.

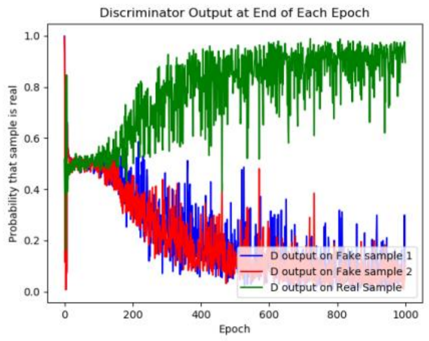

Fig. 9. Discriminator output during training process. Green is the real sample probability and red/blue is the fake.

D. Training Environment and Process

The training took place on a machine with a 1070ti and cuda support. This helped speed up training dramatically and allowed for many different trials and combinations, with varied hyperparameters, to be run.

E. Techniques That Performed Much Worse than Expected

Since deep learning models benefit from more data, simple data augmentation was employed to created rotated, flipped and noisy samples. However, this dramatically worsened the generated images and wasn’t a good approach. The set was augmented by flipping images horizontally, rotating them 45 degrees and adding small amounts of noise. This had the opposite effect and output generated was some of the worst. The image set grew to around 100,000 samples.

F. A Note on Mode Collapse and the Selected Result

The samples presented in the results section look very similar but actually are not a result of mode collapse. They were just similar samples that have been picked from a few separate random selections of generated samples over a couple different epochs. Mode collapse was encountered during this process but it tended to take a lot more training before many of the random samples were all the same image.