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

Brian Landy, MS Computer Engineering, Rochester Institute of Technology

Abstract—Deep Learning models are capable of doing many things that humans cannot and one such example of this is their ability to extract features and reproduce fake data. Deep convolutional generative adversarial networks are an interesting part of the deep learning body of knowledge and this experiment and exploration of such models will demonstrate their use in creating believable generated images based on features of Rochester Institute of Technology buildings. Generative adversarial networks are a type of deep learning model which can be used to construct fake data for a number of different mediums. 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 which are then used in fully connected layers. Many images of the Rochester Institute of Technology campus were captured and processed in order to train a deep convolutional generative adversarial network. The highly variational Rochester Institute of Technology image dataset included more than 30,000 images. Several experiments with the model were run with the goal of generating more visually believable output and the quality of each model variation was assessed visually at the end of 800 epochs of training. By varying hyperparameters of the model from the baseline, it is shown that improvements can be made to image quality by altering certain operating characteristics.

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. [2]. In some cases, there aren't enough samples to effectively train a model and so generative approaches are taken to strengthen them. Goodfellow et al. [1] 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 to generative tasks. GANs are computationally expensive and are even more so with the application of convolutional layers for feature extraction on images. In 2012, Krizhevsky et al. [6] worked with a deep convolutional neural net architecture to classify ImageNet entities and achieved an error rate of 15.3% for the top 5 classes of 1000; works such as this strengthen the justification for using convolutional layers as feature extracting methods on images, even for GANs. The application of convolutional layers to GANs is introduced by Radford et al. [3] where the group proposes an architecture of GANs that has convolutional layers for better image feature detection. GANs can have certain parameters altered in order 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 adopted in the experiments of this project. A similar façade 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 to be done, the Proposed Method section will discuss the choices reasoning behind tuning the DCGAN model to generate fake Rochester Institute of Technology (RIT) building images, the results section will cover the experiments outcomes and the conclusion will wrap up the findings and main ideas of the experiments. Supplementary information about GANs and less noteworthy experiments will be in the appendix at the end of this paper.

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 to the generative ability of GANs. The discriminator network functioned similarly to a typical binary classifier convolutional neural network (CNN). To perform the generative task, 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 upsampling. Fig. 1 shows the architecture chosen for the generator network in this experiment from Radford et al. [3].

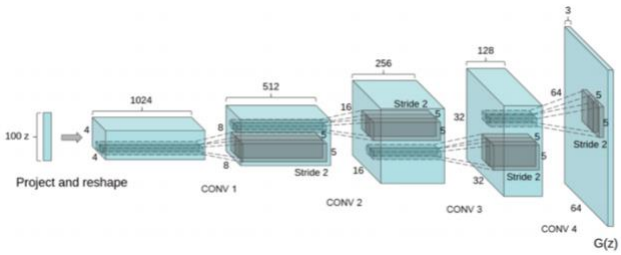

Fig. 1. Architecture for a Deep Convolutional Generator Network. In the figure, transposed convolutional layers and leaky ReLU activations transform random noise vectors into image features. Fig. 1 is from Radford et al. [3].

B. Tools and Chosen Architecture

For this task of generating images based on RIT buildings, DCGANs were chosen as the appropriate method due to their ability to effectively extract and generate 2D image features.

The chosen model to use as a baseline for this was the Pytorch DCGAN example from Radford et al. [7]. 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 type of images allowed in the custom RIT dataset to negate these issues. They are listed in the datasets section of this paper.

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. The goal of this set was to capture a minimally variational image set to work with. By using this, the model should be able to easily overfit the set and produce one type of image. The set size is 11,435. After the small set was collected, a larger highly variational set was also collected. The goal with this set was to capture mostly front facing images of buildings, but to vary the type of buildings as the subject of the photos. This set ended up being made up of 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 performance*

Many variables in the DCGAN training process were modified. Some techniques put to the test were: weakening the discriminator by lowering its learning rate, implementing soft labels, using flipped labels, swapping the activation functions, implementing historical averaging, varying the depth of the feature maps in the discriminator and generator, varying the adaptive moment estimation (adam) optimizer parameters, lengthening the typical number of training epochs with the baseline model and using classical data augmentation to provide a much larger training set to work with. The lowly and highly variational datasets were used in training with these varying parameters.

III. Results

A. *Baseline Results on all image sets*

Loss functions for D and G were plotted after every epoch and images were processed in batches of 64. Using an unmodified DCGAN as a baseline, the LSUN churches dataset ran for **X** epochs and produced the fake images seen in fig. **X**.

Fig. X. 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 simple RIT set was run with the baseline and some generated images are shown in fig. X.

Fig. X. 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 run with the baseline DCGAN and generated images are shown in fig. X.

Fig. X. RIT high variation dataset. The output is more warped and noisy than the input. The goal of the paper is to produce images that look better than this.

B. *Troubleshooting models based on loss plots*

Plotting loss proved to be an effective way to diagnose issues with the model. If the D loss went to 0 way too early, then the discriminator became too good at classifying fakes and needed to be trained more slowly so the generator stood a chance at fooling it. Fig. X shows a loss plot when the discriminator is too capable early on.

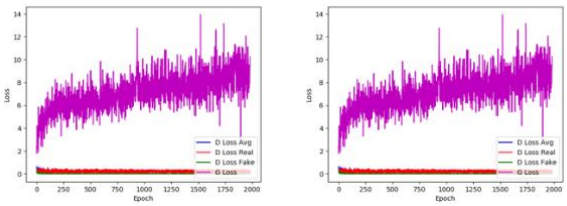

Fig. X. Loss plots of D and G where the discriminator is too capable and no good images can be generated.

Over time the generator should keep increasing because it needs to do more and more work to fool a constantly trained discriminator.

C. *Best Results of modified runs on RIT datasets*

The simple RIT set was run with the baseline and some generated images are shown in fig. X.

Fig. X. RIT low variation dataset. Best output profile info: X,Y,Z,A,B,C,1,2,3

The highly variable set was run with the baseline DCGAN and generated images are shown in fig. X.

Fig. X. RIT high variation dataset. Best output profile info: X,Y,Z,A,B,C,1,2,3

This is where I discuss the final results that led to the best output.

IV. Conclusion

This section will probably change with the results. This paper has shown that when using DCGANs, the output believability and quality benefits greatly from modifying __param__ in order to __something__ during training. This is demonstrated by ___another thing____. DCGANs are a powerful unsupervised learning tool and more than capable of generating some interesting and somewhat believable fake images of RIT.

V. References

[1] https://arxiv.org/pdf/1406.2661.pdf

[2] https://arxiv.org/ftp/arxiv/papers/1803/1803.01164.pdf

[3] https://arxiv.org/pdf/1511.06434.pdf

[4] https://arxiv.org/pdf/1606.03498.pdf

[5] https://arxiv.org/pdf/1907.05280.pdf

[6] https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[7] https://github.com/pytorch/examples/tree/master/dcgan

[8] https://github.com/soumith/ganhacks

VI. Appendix

A. *Generative Adversarial Networks*

Goodfellow et al. [1] 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. [1]. 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.

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

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. 1 is** the architecture for a standard GAN.

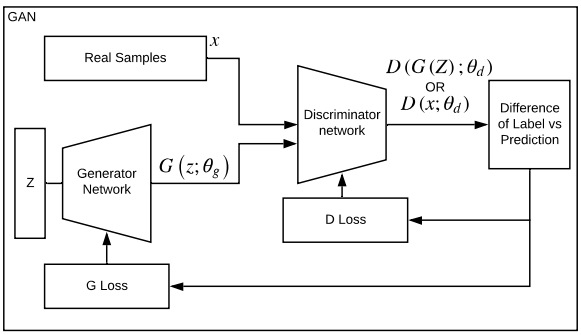

Fig. 1. Architecture for a GAN. In the figure, real samples, *x*, and fake samples, G(z), are passed into a discriminating network that will predict a label, D(G(z)) or D(x). The comparison between real and predicted labels are used to update discriminator and generator loss.

Goodfellow et al. [1] 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. [1].

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

The Generator network (G) is updated one time after updating D. This is shown in (4) which is also provided by Goodfellow et al. [1].

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

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

B. *Best Selection of parameters*

Best quality output
- -visual assessment
- -explain how inception score could be used

- ➔ Input information about starting architecture
- ➔ Soft labels
- ➔ Flipped labels
- ➔ Any filter changes
- ➔ Data augmentation for larger scales
- ➔ Discriminator issues
- ➔ Adjusted learning rate to slow down discrim

- ➔ Play with adam optimizers
- ➔ Ndf is the depth of the convolutional block. Width, height and depth
- ➔ Onesided label smoothing
- ➔ Two sided
- ➔ Historical averagin
- ➔ Epochs
- ➔ Others:

Mode collapse- explain but goal is not to fix it
Noise

- ➔ Mode collapse, occurred, not necessarily fixed

C. What has been submitted

I'm submitting my modified code.
Result files and folders with adequate names

TO do: size figures properly:
-formatting
-proofread
-figure number assignment

-works cited, edit paper, reduce wordiness to be under 2 page limit