CPTS 570 Course Project: Stable GAN Training with Multi-Resolution U-GAN

Author: Colin Greeley, 11567185
Instructor: Dr. Jana Doppa

December 13, 2021

Abstract

GANs are the current state-of-the-art model for generating synthetic data. In the past few years, the world's top AI and computer vision researchers have been able to achieve the ability to generate high resolution synthetic images that are nearly indistinguishable from real ones, through the magic of GANs. These generative models are famously difficult to get functioning correctly, due to the instability of the training process, as it is a gradient minimization game between two neural networks. Many papers in recent years even have contradictory information to other publications, stating that some methods are better than others, but no one has seemed to mathematically define what model architecture will definitively work for generating images at any resolution. Most publications change the current best model architecture slightly or introduce a few "tricks" to improve the quality of generated images and the stability of the training process. In this paper, another hypothesised trick to improve model training is proposed to transform any GAN architecture into a U-GAN by creating residual connections.

1 Introduction

Generative Adversarial Networks (GANs) are a paired neural network model that uses game theory ideas to generate synthetic but realistic data [Goodfellow et al., 2014]. GANs where originally proposed as a form of generative model for unsupervised learning, but recent techniques have exposed the flexibility of the system to support semi-supervised learning and reinforcement learning into the framework. Since their invention by Ian Goodfellow in 2014 [Goodfellow et al., 2014], there has been an explosion of contemporary research in the field regarding GANs. The original model architecture was made purely of densely connected neural network layers and was built to synthesise low resolution images such as those in the MNIST handwritten data set [Goodfellow et al., 2014]. The original model was made simply to test the hypothesis that this sort of training setup would work, but was mostly impractical. Many improvements have been made in recent years to allow the images generated to be higher resolution and more realistic. The most notable contribution was the Deep-Convolutional GAN (DCGAN) [Radford et al., 2016], which is also now seen as the baseline model for all further contributions. Many other contributions are focused on improving the loss function, as the training process for GANs is incredibly unstable, especially at higher resolutions.

1.1 Motivation

The problem that is investigate in this paper is building a GAN for high resolution images with stable training. In the field, anything greater than or equal to (256x256) pixels is considered to be high resolution, but due to computational and time limitations, (128x128) is the chosen resolution. The default DCGAN architecture is slow and not very stable for learning to generate higher resolution images. Inspiration for the hypothesis discussed in this paper was acquired

from a paper that published by NVIDIA in 2018 which describes that learns to generate images from low to high resolution incrementally resulting in fast and stable training. The model is called called Progressively Growing GAN (ProGan) [Karras et al., 2018]. The main drawback to this approach in my opinion is that it is incredibly difficult to implement, as it is a complex dual neural network system that grows (i.e., has lots of moving parts). The failed implementation of this model led me to think of a similar architecture without moving parts.

1.2 Challenges

There are many technical challenges when it comes to building stable GANs. First, it is difficult to determine the performance of a GAN purely from the training metrics. The goal of a traditional neural network in the supervised learning paradigm is to simply maximize test accuracy and minimize test loss. In GANs, there are two neural networks, the generator and the discriminator. The goal of the generator is to generate images from random noise that are realistic enough to fool the discriminator, and the discriminator's job is to learn the difference between real and fake images [Goodfellow et al., 2014]. Furthermore, the metrics for both the generator and discriminator are calculated from the output of the discriminator, so the generator accuracy is not even available. It can be seen that the ideal convergence point for the discriminator accuracy would be roughly 50% accuracy, since that is the point when the generated images are indistinguishable from real ones. If the accuracy converges towards 100%, then the system is collapsing as the discriminator is becoming "smart" too quickly. This game of balancing the performance becomes increasingly difficult with deeper neural networks since the gradients used to update the generator are derived from backward propagation through the discriminator. This makes using standard regularization techniques more tricky, since exploding and diminishing gradients for the generator are likely to occur if the architecture is carefully designed.

1.3 Solution Approach

The hypothesized model is analogous to a U-net [Ronneberger et al., 2015], where the generator outputs an image after each upscale-block that would be passed to the discriminator at each respective block resolution. These residual connections would ensure loss for each block in the discriminator would be passed to each block in the generator, at their respective resolutions. Furthermore, the deeper the block is in each network, the more gradient updates it would receive since all losses will pass gradients all the way to the start of the generator. This should make for a stable training process, regardless of the resolution size of the images. For this multi-resolution GAN to train properly, each "real" image for training would also have to be transformed into a set of images at multiple resolution so it can be passed into the discriminator. This is not a problem since it would only have to be down samples n times, where n is the number of down scaling blocks in the discriminator. Of course, the number of up-scaling blocks in the generator would also be n.

Side note: About half way though my research process (after experimenting with working code for weeks), I found a paper published in 2020 called Multi-Scale Gradients for Generative Adversarial Networks (MSG-GAN) [Karnewar and Wang, 2020], which was also inspired by the ProGAN architecture and is literally the exact same model that I hypothesised for this research project. Finding this paper was both disheartening and motivating at the same time, since there was proof that the idea that I came up with would work, but the work had already been done and published. So, I can not say that any of the work done in this paper is truly original, since my architecture was further inspired by the MSG-GAN that already exists.

1.4 Results

The results for the problem are mostly empirical, as I have provided training metrics and sample images for the generators. There are specialized metrics to somewhat give an idea of the success of a GAN such as inception score, but for the results of this project, the main focus is on the image quality by making a direct comparison of a generated image to a real image. So, the reader can make the same conclusion's for themselves as the comparison can only be done with the human eye. The one theoretical results is that training stability is guaranteed as the resolution of the generated image increases, but the training time also increases.

2 Problem Setup

The formal problem for this paper is minimizing training instability in GANs. The problem becomes more relevant as the resolution if the image becomes larger since the linear depth of a the model correlates to quadratic growth in the number of trainable weights, given the model architecture. The gradient update of a standard DCGAN is computed from the loss between the discriminators prediction of the input image being real or fake, computed through binary cross-entropy by default. This means that the gradient computed at each layer in both networks is derived from a single scalar value, which is incredibly sparse learning information for such a large network. This is why things like diminishing or exploding gradients and mode collapse are such common issues in DCGANs. The assumption with the multi-resolution U-GAN is that the residual connections between the generator and discriminator will introduce additional loss to the generator, leading to more stable training and thus, higher synthetic picture quality and less probability of the model failing at higher resolutions.

3 Solution Approach

The problem of generating high quality images via stable gradient updates is an on going problem within the field of generative machine learning. The naive baseline approach to image generation is the DCGAN model. This model was a big advancement from the dense neural network architecture first proposed in 2014, but still suffers in training time and stability as image resolution increases due to the network depth. The multi-resolution U-GAN proposed in this paper addresses this problem by creating the residual connections at each resolution block between the generator and discriminator, there will be more direct gradient updates for each individual resolution block in the generator. This also means that the deeper the block is in the generator, the more gradient updates would be passed through per training step, leading to more diverse loss and faster training. For example, if we are generating images at 128p, the 16x16 block in the generator would get the corresponding gradient update from the 16x16 residual connection to the discriminator, as well as the update from the 32x32, 64x64 and 128x128 blocks in the generator, all in one training step. For what I have experienced, the only drawback to this solution is the slight inconvenience of implementation. All images have to be transformed through an image-pyramid since the discriminator will have n multi-resolution inputs, where n is the number of downscale blocks in the discriminator. This also means that the generator will produce and array of n images, the last in the array will be the highest resolution.

4 Experiments and Results

4.1 Experimental Setup

There are a few standardised data sets in the field of image generation. CelebA, OxfordFlowers102, and LSUN data set. For the experiments, a data set that included easy to use labels was

favorable, so the data set used the for the experiments in this paper is the OxfordFlowers102 data set. This data set is composed of a large amount of flower images, sorted into 102 different classes. It has been shown that labeled data can improve the stability of training while also boosting image quality [Odena et al., 2017]. This is because it introduces an additional loss function to help the model learn. For this reason, the base model The the auxiliary classifier GAN (ACGAN) [Odena et al., 2017] is used as the base model for my setup. Additionally, this model improves as image resolution is increased since the discriminator will be able to identify classes better at higher resolutions. So, the input to the generator will be a class label, and random noise as the latent space. Of course, the input to the discriminator will still only be images, but the output will be the validity of the image as well as a class label. For the loss function, binary cross-entropy will be used as that is the default loss function in DCGAN [Radford et al., 2016] and ACGAN [Odena et al., 2017]. There have been futher improvements to the loss function such as LSGAN [Mao et al., 2017], Wasserstein GAN [Arjovsky et al., 2017], and Wasserstein GP GAN [Gulrajani et al., 2017], but we will be sticking with binary cross-entropy for simplicity. To briefly summarize the model, the architecture is a mix between the ACGAN and ProGAN architectures [Odena et al., 2017, Karras et al., 2018]. Internal residual connections inspired by the ResNet architecture were also added to each model since they have been shown to improve performance in deep models with many layers [He et al., 2016].

4.2 Evaluation Methodology

As stated in the introduction, measuring the performance of a GAN is much more difficult that a traditional neural network in a standard supervised learning setup. For the training metrics, only the loss is available for the generator, but the loss, image validity accuracy, and class label accuracy is available for the discriminator. Since the goal is to have a constant battle between the generator and discriminator until both have reached max performance, studies have shown that the discriminator image validity accuracy hovering around 80% is favorable while the entropy loss hovers around 0.5 [Salimans et al., 2016]. Obviously, the class label accuracy should converge towards 100% since that is a trivial classification problem. We can look at the curves generated across all training iterations and visually see instability. Additionally, we can use general statistics on these curves to see the variance of the training metrics (a measure of stability) and the mean (to compare to the recommended training values). Other than that, the best comparison with be with the human eye, comparing generated images to real images and generated images between models.

For the experiment description, the baseline model described above will be trained which we will just call the ACGAN (ACGAN/ProGAN mix, see Figure 5 and 6 in the Appendix for more details on model description), and compare that to the same ACGAN with the residial connections between the generator and discriminator (multi-resolution U-GAN). The performance metrics described above will be measures head-to-head between the two models, ACGAN (base model) and multi-resolution U-ACGAN (proposed model). The parameters and additional performance tricks for BOTH models are listed below:

1. Optimizer for Generator and Discriminator: Adam(lr=0.0002, β_1=0.5) [Odena et al., 2017]
2. Batch Size: 100 [Odena et al., 2017]
3. Iterations: 100,000 (One Batch per Iteration) [Odena et al., 2017]
4. Latent Space: 128 (Equal to Output Image Size) [Karras et al., 2018]
5. Dropout in last layers if Discriminator: 0.25 [Salimans et al., 2016]
6. Image Validity Loss: Binary Cross-entropy [Odena et al., 2017]

7. Class Label Loss: Categorical Cross-entropy [Odena et al., 2017]
8. Convolution Weight Initializer: He Normal [He et al., 2016]
9. Generator Normalization after Convolution: Batch Normalization [Odena et al., 2017]
10. Discriminator Normalization after Convolution: None [Odena et al., 2017]
11. Generator and Discriminator Activation after Convolution: Leaky ReLU [Karras et al., 2018]
12. Generator Upsampling Method: UpSampling2D [Karras et al., 2018]
13. Discriminator Downsampling Method: AveragedPooling2D [Karras et al., 2018]
14. MiniBatch STD in Discriminator (Improves Image Diversity) [Karras et al., 2018]
15. Instance Noise: 0.1 (Visual Noise Added to Discriminator Input Images) [Salimans et al., 2016]
16. Probability of Randomly flipping label: 0.05 (Prevents Discriminator from learning too fast) [Salimans et al., 2016]

4.4 Results

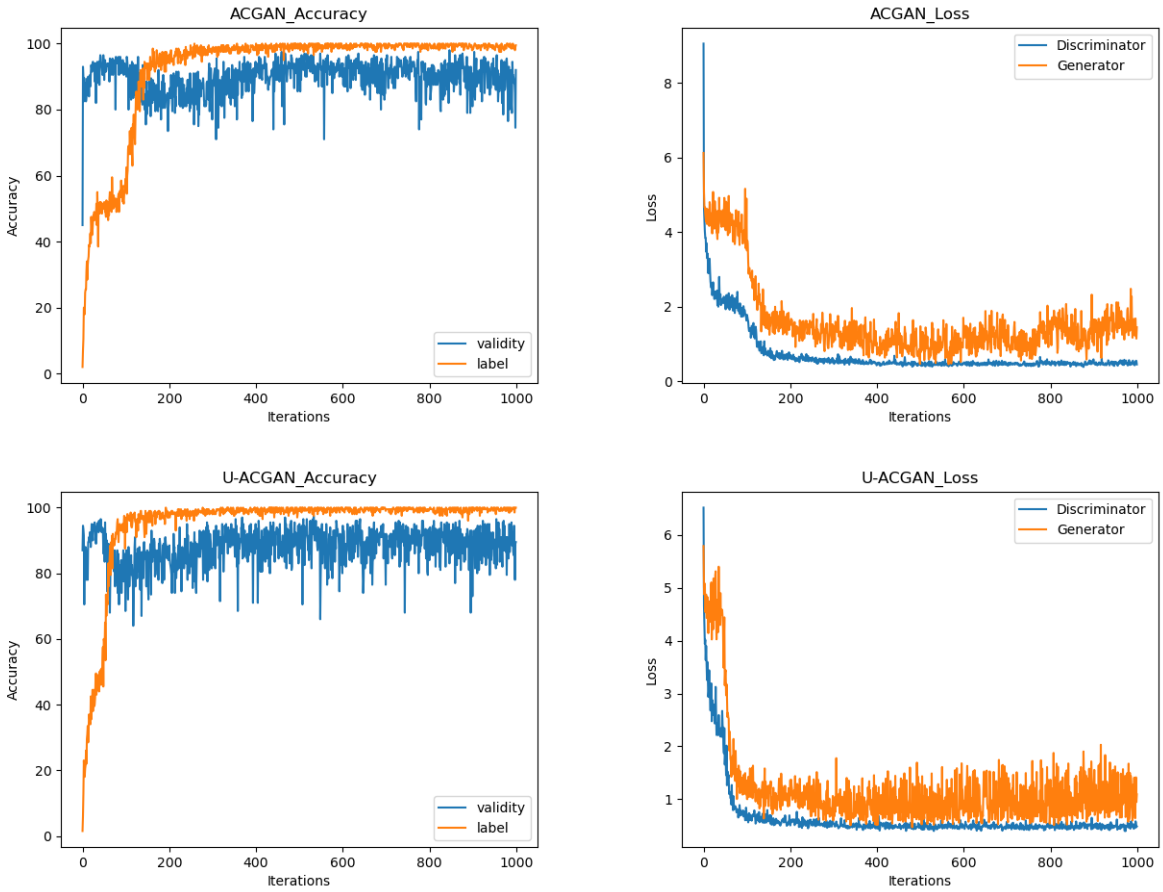

Figure 1: Accuracy and loss results for both the baseline ACGAN model and the multi-resolution U-ACGAN

The training metrics are plotted on the figures above. As described before, the accuracy metrics were only available for the discriminator models, which will be seen on the left side figures. The

loss for both the generator and discriminator across both models is shown on the right side figures. For training, there was a total of 100,000 iterations with one batch of 100 examples for each training iteration. On the plots, the iterations only goes up to 1,000 and that is because the the metrics were only saved every 100 iterations to avoid clutter. Another thing to notice, only the training metrics are plotted since there is no test set for GANs.

From the training graphics shown above, we can conclude that the hypothesis for more stable training holds true with some key observations. It can be seen that U-ACGAN was able to converge towards a local minimum with respect to the loss of both networks. This is due to the fact the there is n times gradients being computed and passed through the U-GAN at each iteration. Recall, n is the number of downscale blocks in the discriminator, so one gradient for each resolution scale. Something interesting to point out is that the sum of losses across training for the U-ACGAN is less than that of the baseline ACGAN. The expectation would be the opposite, since it would seem likely that the sum of losses for all resolution scales would be higher than just the loss at the highest resolution, but it seems that this method has helped to model minimize losses significantly.

For the stability of training, it is hard to tell from the graphics which model performed better. It seems that the U-ACGAN plots contain more violent fluctuations across the accuracy and loss metrics which could be an indicator that the model is in fact less stable. Although, if the loss metrics were smoothed out into a single curve for each individual network, we can see that the U-ACGAN losses would be less bumpy curves. Additionally, the U-ACGAN generator and discriminator entropy losses converged around 1.0 and 0.5, respectively, while the baseline ACGAN generator and discriminator entropy losses converged around 1.5 and 0.8, respectively.

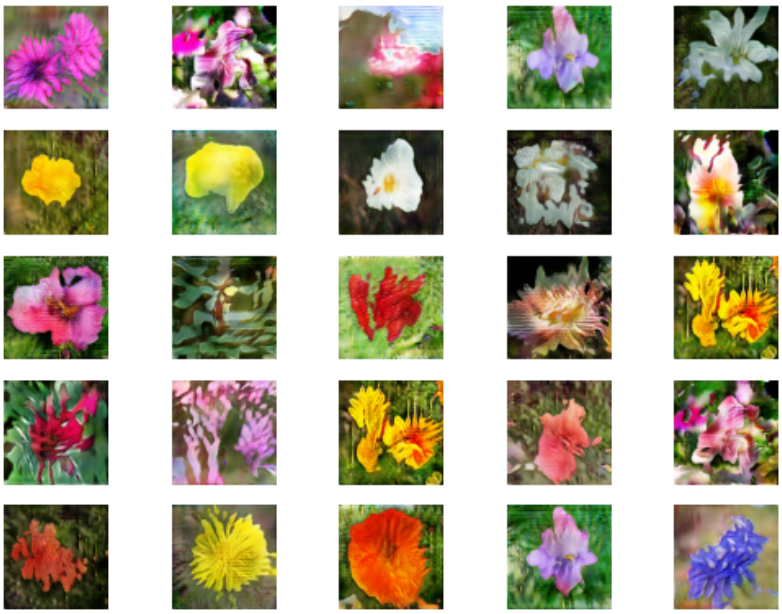

Figure 2: 25 images generated from the baseline ACGAN with random noise and random labels as input.

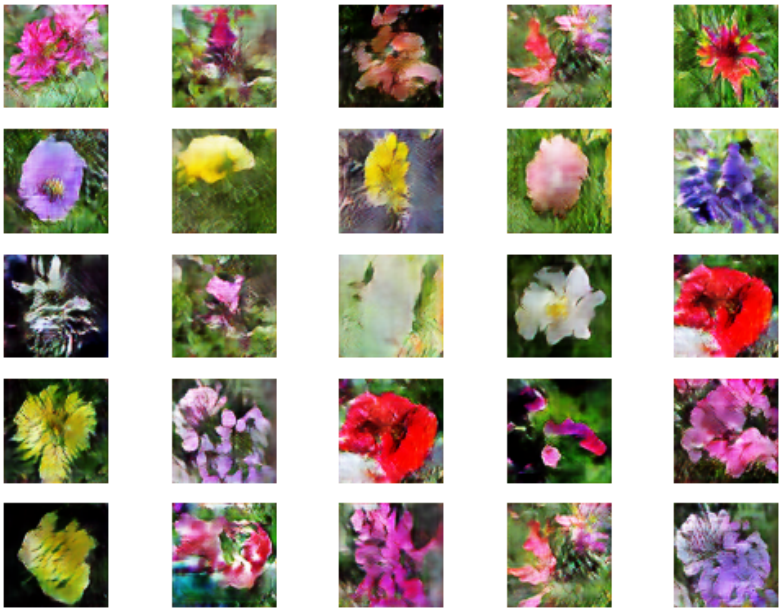

Figure 3: 25 images generated from the multi-resolution U-ACGAN with random noise and random labels as input. Only the at the highest resolution are displayed.

From the images alone, it is actually quite difficult to see which generator can produce higher quality images. The hypothesis was that the U-ACGAN generated images would clearly be more realistic, but that determination can not be made for the output images. At first glance, both sets of images seem to be high quality realistic images of flowers, but at closer inspection, it is obvious that the images are synthetic and contain a lot of nonsense and incoherent pixel groupings.

5 Conclusions and Future Work

The addition of residual connections between the generator and discriminator of GANs is not a revolutionary addition, but just another simple trick to help stabilize GANs and maintain a nash equilibrium in the training process. Although there was no direct correlation between the addition of these connections and the quality of the generated images, we can conclude that there are no bad side effects to the implementation. There are many reasons as to why the training metrics weren't better and why the images weren't higher quality. As seen in 4.2, there is very large list of hyperparamters and combination of neural network layers that goes into building a GAN. The next steps for this project would be to experiment with those to try to improve both the numerical metrics and the quality of the generated images. An adaptive or smaller learning rate might be beneficial for improving the loss metrics. To improve the image quality, the simplest trick would probably be to increase the number of trainable parameters in the neural networks. It is entirely possible that the highest possible images with this model setup were achieved do to a lack of trainable weights in the convolutional layers. Adding more filters to each convolution would increase the complexity and training time but likely lead to higher quality syntetic image generation.

6 Acknowledgments

References

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. volume 1.

[Goodfellow et al., 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. volume 3.

[Gulrajani et al., 2017] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans. volume 2017-December.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. volume 2016-December.

[Karnewar and Wang, 2020] Karnewar, A. and Wang, O. (2020). Msg-gan: Multi-scale gradients for generative adversarial networks.

[Karras et al., 2018] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of gans for improved quality, stability, and variation.

[Mao et al., 2017] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Smolley, S. P. (2017). Least squares generative adversarial networks. volume 2017-October.

[Odena et al., 2017] Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. volume 6.

[Radford et al., 2016] Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. volume 9351.

[Salimans et al., 2016] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans.

7 Appendix

Figure 4: Visual representation of what is output from the multi-resolution generator. At an image resolution of 128 pixels, the output is an array of 5 images (rows). Coincidentally, the graphic above also shows the output of 5 random images (columns). The order of image resolutions is (8x8), (16x16), (32x32), (64x64), 128x128).

[ACGAN] [U-ACGAN]

Figure 5: The graphics above are the layer by layer decomposition of the generators for both models (ACGAN Generator on the left, U-ACGAN Generator on the right). The internal "diamond" shape connections are the ResNet upscale blocks. On the right hand side, you will see each block has an additional 3 channel convolution layer where an image at the respective resolution is the output. You can imaging a residual connection from these layers to those in the discriminator.

[ACGAN] [U-ACGAN]

Figure 6: The graphics above are the layer by layer decomposition of the discriminators for both models (ACGAN Discriminator on the left, U-ACGAN Discriminator on the right). Again, the internal "diamond" shape connections are the ResNet downscale blocks. On the right hand side, you will see each block has an additional 3 channel convolution layer where an image at the respective resolution is the input. You can imaging a residual connection to these layers from those in the generator.