CS 5330
YES WE GAN!

Nicholas Wen
Robotics, Electrical and Computer Engineering
Northeastern University
Boston, US
wen.n@northeastern.edu

Sumegha Singhania
Robotics, Electrical and Computer Engineering
Northeastern University
Boston, US
singhania.s@northeastern.edu

***Abstract*—Websites like wombo.art, neuralblender.com, and artflow.ai utilize neural networks in order to generate art based on a text prompt. In this paper, we intend to explore how a neural network generates art based on some input and explore different methods of image generation and modification. We explore several options of image generation chronologically DCGANs, WGANs, and CycleGANs. We evaluate the performance of each empirically and explore options for network optimization.**

***Index Terms*—**

I. Introduction

The idea of having a computer create meaningful pictures in a certain style is an intriguing premise. With the help of neural network we can see how different artists may have painted the same picture. Commercially, companies like Dream and Artflow AI attempting to capitalize on machine produced artwork so that even the least artistically inclined can have their imagination translated into a picture (Figure 16). Most machine learning systems that generate pictures or artwork use Generative Adversarial Networks in order to create artwork. In this paper, we look at a subset of machine generated art, image to image translation, or taking aspects of one type of image and applying it to another image.

Fig. 1. An example of a machine generated image using neuralblender with the prompt "The sky above the port was the color of television, tuned to a dead channel.

II. Related Works

Much of our work is based on the following papers about Generative Adversarial Networks and different approaches that have been used in order to get quicker or more accurate results.

A. Generative Adversarial Nets (GAN) [1]

The key to Generative Adversarial Networks(GANs) is "Adversarial". A GAN network at its core is built upon 2 interconnected neural networks, a generative network (which produces some output meant to mimic the training set) and a discriminative network (which tries to distinguish between training set inputs and the generative network outputs). In essence, the two networks we train are playing a game of forgery, with the generative network attempting to trick the discriminative network until some equilibrium can be found. In concept, these networks are attempting to find a Nash Equalibrium, a state where it is not advantageous for either player to change strategy, however, in practical usage, a gradient descent model is used to adjust the weights of the model.

B. Deep Convolutional Generative Adversarial Nets(DCGAN) [2]

While basic neural networks were effective for Goodfellow et.al, GANs can produce unwanted results such as nonsensical results and training instability. By using a Deep Convolution Generative Adversarial Network(DCGAN), we can introduce more stability to training as well as increase the visual quality of our outputs. In order to acheive these results, several modifications were made to include convolution layers.

Several changes were made to the original GAN model to improve performance using Convolutional Neural Networks(CNN). First replacing spacial pooling functions with strided convolution(for discriminator) and fractional-strided convolutions(for generator). This allows the generator to learn its own down-sampling and up-sampling models. Second, utilizing batch normalization, the process of normalizing the input to 0 mean and unit variance, in order to reduce issues issues due to poor initialization and GANs simplifying all samples to the same output image(mode collapse), which is a common difficulty with GANs. Finally, CNNs allow more flexibility in network usage, including the ability to implement vector arithmetic [3], which allows attributes of one image to be modified into another.

C. Wasserstein Generative Adversarial Networks [4]

One of the issues with using DCGANs is that when the discriminator is not tuned correctly, it cannot provide useful information to the generator to learn, which can make training useless. One cause of this could be traditional GANs attempting to minimize the Jensen-Shannon divergence, which is what is used in the original GAN model [1].In order to avoid this issue, Wasserstein Generative Adversarial Networks(WGAN) uses Wasserstein distance (also known as Earth mover's distance) to calculate minimization. In order for Wasserstein distance to be useful as a metric, our function must be 1-Lipschitz continuous (i.e. if there is a constant c such that $||f(y) - f(x)|| <= Cyx$) . This is done via weight clipping, the process of limiting the effect of errors to some threshold during back propagation. As a result of these changes WGANs tend to be more stable during training, as well as be less likely to suffer from mode collapse.

D. Improved Training of Wasserstein GANs [5]

As discussed in M. Arjovskyi et.al [4], Weight clipping is a very ineffective method of enforcing the 1-Lipschitz constraint. Often, this clipping can cause bad outputs or a failure of the model to converge. In order to rectify this issue, Gradient Penalty is applied to WGAN networks (called WGAN-GP) in order to combat these issues. Gradient Penalty implements a soft constraint to achieve 1-Lipschitz continuity by adding another term to the original clipped loss function to allow the larger values to be seen, but introduce a penalty for being over the threshold.

E. Image to Image Translation [6]

DCGANs are very powerful in their own right, however, the result may look nothing like the original input image. In order to remedy this issue, this paper implements Conditional Generative Adversarial Networks, the process of generating images with some constraint such as a classifier label, in order to generate different types of images. In order to learn these mappings, the network is fed an input and attempts to learn how to create a ground truth output through training. In practice, this allows the network to learn the mapping from input to output images in order to build an output that looks similar to the input.

F. Unpaired Image to Image Translation using Cycle-Consistent Adversarial Networks [7]

Despite Image to Image Translation being a useful tool, the requirement to use paired image data restricts the ability to implement the system in a wide variety of situations. In order to remedy this fact, this paper implements Cycle-consistency loss in order to be able to create generators without image pairs. This is done via 2 generators, one for mapping some domain X to Y($F(x)$), and another that does the inverse, mapping Y to X($G(y)$). When an image is received in the X domain, it is translated into the Y domain, then back to the X domain. We also have 2 discriminators which are trained to determine if an image is real or generated in a given domain. In order to train the model, we feed in some image in one of the 2 domains, and use our generator to convert it to the other domain. We then feed the generated image into our other generator to return the image to its original domain as well as the discriminator for that domain. Finally we calculate the loss of the entire system based on the results of the discriminator and the difference between the original image and the resulting image in the original domain.

III. Methods

Our approach towards understanding the world of machine generated art, was to try implementing the various networks that have been laid out in the aforementioned papers. We began by implementing a DCGAN trained on the Fashion-MNIST dataset to generate grayscale 28x28 images. We then modified the same network to generate 28x28 colour fashion images. After gaining an understanding of how we can generate colour images, we then implemented a WGAN to generate coloured anime faces. At this point, having gained an understanding of how GANs work and enable new image generation from noise, we decided to dive deeper into image to image translation. We started by understanding pix2pix: image to image translation with a conditional GAN for paired data and CycleGAN for paired data. We then built a CycleGAN for converting images to Monet style paintings.

A. Fashion image generation using DCGAN

We start by implementing a DCGAN to generate 28x28 grayscale images. The architecture implemented is the same as mentioned in the paper. It employs strided convolutions with a stride of 2 during the downsampling function and fractional strided convolutions with a stride of 2 for the upsampling function. Additionally, it uses four activation functions: Sigmoid, which squashes the input to 0(fake image) or 1(real image), is used in the last layer of the discriminator; tanh(Hyperbolic Tangent), which squashes the input value to [-1,1], is used in the last layer of the generator; ReLU which returns 0 when the input is negative otherwise returns the input value, is used in all layers of the generator except the last layer; LeakyReLU, which is similar to ReLU except when the input value is negative, it uses a constant alpha value (0.2 in this case) to give a small slope, is used in all layer of the discriminator except the last layer. The only difference to the model for generating colour 64x64 images, is to change the number of upsampling functions, to alter the generator output shape and accordingly change the input image shape of the discriminator.

B. Anime face generation using WGAN

We implemented the anime face generator using two variants of GAN: Wasserstein GAN (WGAN) and Wasserstein GAN with Gradient Penalty (WGAN-GP). WGANs help us address the training instability challenge that GANs face. Training instability refers to the difficulty in keeping the generator and the discriminator balanced in an effort to reach an equilibrium. This is done by introducing a Wasserstein

Fig. 2. Generated gray-scale fashion images

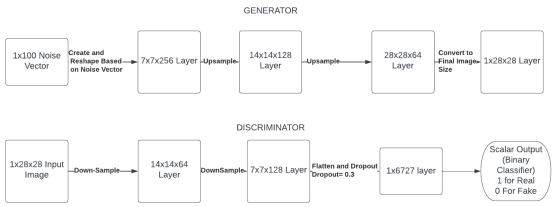

Fig. 3. The Structure of our 2 sub-networks

Fig. 4. Generated colour fashion images

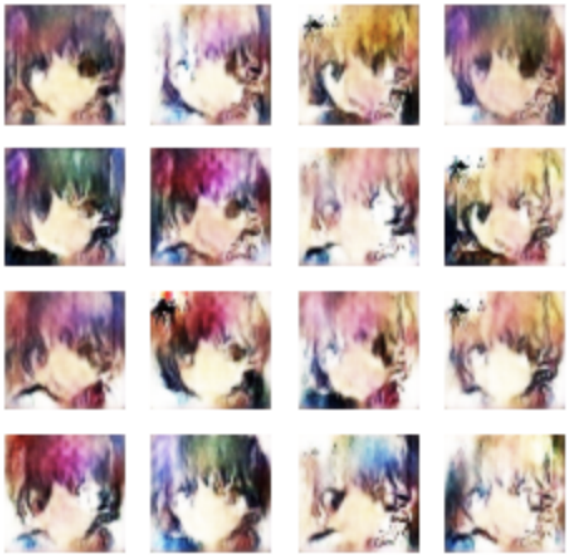

Fig. 5. Anime faces generated using WGAN

Fig. 6. Anime faces generated using WGAN with Gradient Penalty

loss function in place of a binary cross-entropy or modified min-max loss function. Additionally, it uses a linear activation function in its last layer as opposed to a sigmoid function and a weight clipping function to enforce the 1-Lipschitz constraint in the discriminator. The WGAN with gradient penalty model uses gradient penalty in place of the weight clipping function. We can see that the WGAN-GP model performed slightly better than the WGAN model, in terms of achieving more realistic generated images.

C. Rendering images to Monet painting using CycleGAN

This model for rendering images to monet style paintings is based on the GAN models discussed in the image to image translation papers mentioned above. We started by gaining an

Fig. 7. Image generation with conditional GAN

Fig. 8. Horses to Zebras using CycleGAN

understanding of a conditional GAN called pix2pix which uses paired images as the database. The architecture contains a generator with a U-Net based classifier and a discriminator represented by a convolutional PatchGAN classifier, with a modified min-max loss function or the binary cross entropy loss function.

This was followed by diving into the CycleGAN model which was aimed at image to image translation for unpaired images. The main difference, apart from the unpaired image data set, was the introduction of an additional loss function, cycle consistency loss function.

The CycleGAN model could not be directly implemented to achieve the goal of converting images to monet paintings. We had to experiment with various parameters and modify the upsampling and downsampling functions to accommodate for the varying size of the dataset.

IV. Experiments and Results

Implementing the CycleGAN model as is with the new data set did not seem to behave as well as expected. After 50 epochs, we would often get completely white or completely black images. One of the features we added was to add random normal initialization, which generates tensors with normal distribution, to the discriminator. We also experimented with various batch sizes, given the processing capacity of our systems, but the only value that seemed to give a good result was a batch size of one. Additionally, the model took about 10-20 minutes per epoch and did not produce very satisfactory results for lower values such as 10. We ended up using an epoch value of 40. Fig. 9-16 show some of the images and their generated monet painting.

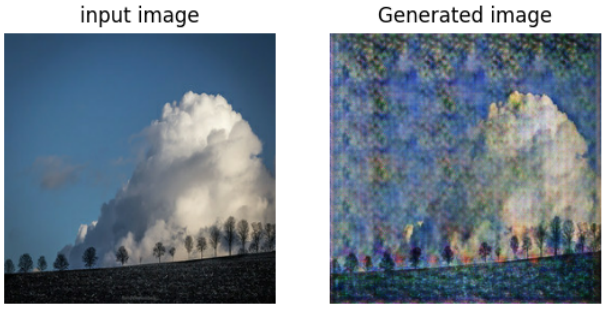

Fig. 9. Generated Monet painting

Fig. 10. Generated Monet painting

V. Future Works

Now that we have experimented with several different models using different data sets, we plan to compare each model against the others using the same data set for all data. While it would have been ideal to do this, limited computing power made this task non-viable for this paper. In addition, we hope to investigate in full how these models compare to each other in terms of training time and result quality.

VI. Conclusion

We have presented several different techniques used to develop GAN networks with various depths of training. Based on these results, we observed discovered that CycleGAN is the

Fig. 11. Generated Monet painting

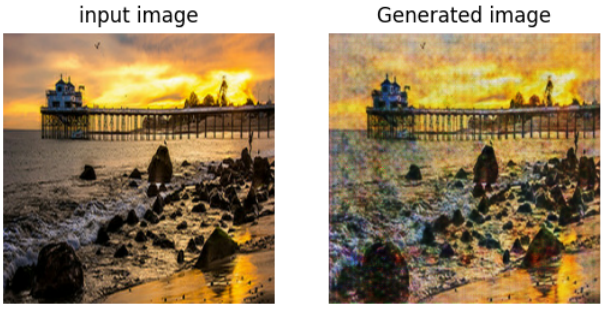

Fig. 13. Generated Monet painting

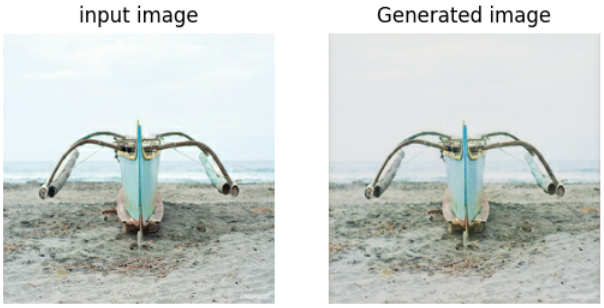

Fig. 12. Generated Monet painting

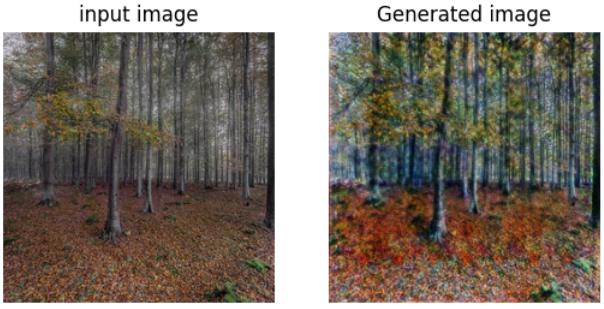

Fig. 14. Generated Monet painting

most effective at generating images, especially under significant constraints due to time and computing power. Empirically observing our results, while still visually identifiable as computer generated, are only subtly different from a human created work. In terms of computation time, many of the insights we learned from our related works were confirmed through testing. Most notably, when using a WGAN vs WGAN-GP, we found that using Gradient Penalty drastically improved the quality of images produced. Overall all of the methods we tried had the similar results to what we expected given the literature avaliable.

REFERENCES

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.06434

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[4] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[5] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," 2017. [Online]. Available: https://arxiv.org/abs/1704.00028

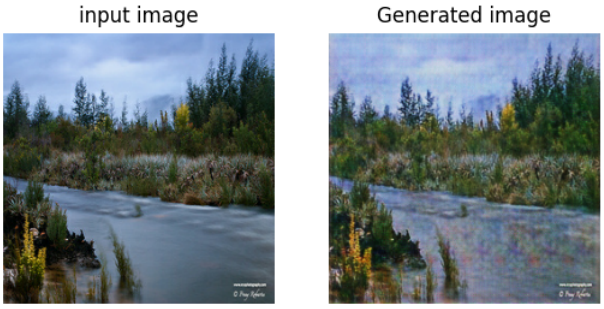

Fig. 15. Generated Monet painting

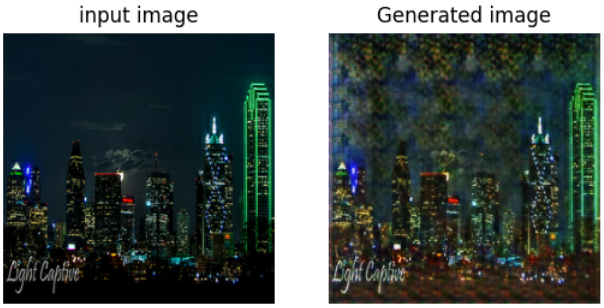

Fig. 16. Generated Monet painting

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2016. [Online]. Available: https://arxiv.org/abs/1611.07004

[7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017. [Online]. Available: https://arxiv.org/abs/1703.10593