Grayson Byrd
HW4 Report
Github: https://github.com/gbbyrd

Project Description:

The purpose of this project is to gain a better understanding of the GAN architecture, training process, and limitations through training multiple different models, employing various optimization techniques and recording and observing the results. For this project, two different GAN models were trained: a DCGAN and a GAN model that uses techniques from both the WGAN and ACGAN papers to improve upon the DCGAN performance. Both of these models were trained to generate fake images using the CIFAR10 dataset.

Technology Review

The basic premise behind generative adversarial networks (GANs) is that you train two models in competition with each other. These two models are labeled the generator and discriminator. The generator takes as input a random noise vector sampled from a gaussian distribution. It uses this random noise vector to generate a fake image in the same domain as the dataset that it was trained on. To put it simply, the generator creates fake images from random noise that look like they could be real images from the dataset that it was trained on. The discriminator takes in an image and outputs a binary prediction of whether it believes the image is a real image from the dataset or a fake image created by the generator. The key feature of GANs is that the generator and the discriminator learn in parallel. The generator learns to fool the discriminator while the discriminator learns to distinguish between the generator's fakes and real images.

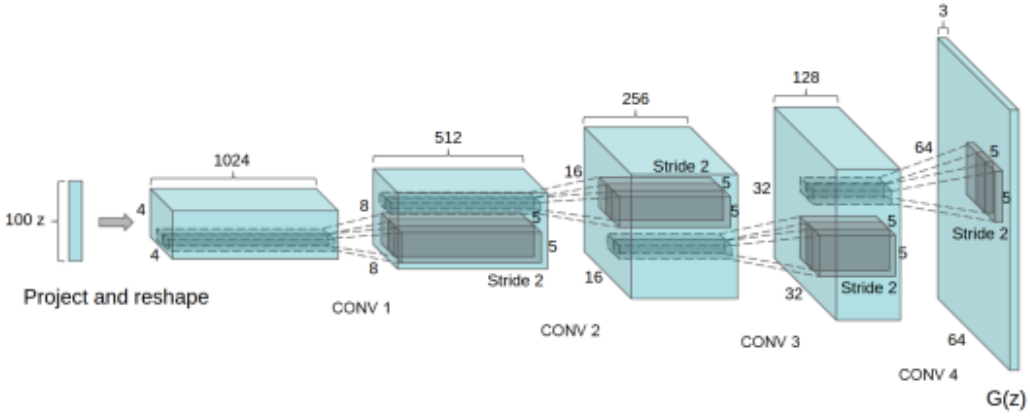
Model 1 - DCGAN

Architecture

Generator

The model architecture for my DCGAN generator is based off of the original paper found here: https://arxiv.org/pdf/1511.06434.pdf. The generator takes in an input of a 100 dim vector of random noise. This vector then goes through multiple transposed convolutional layers to translate that 100 dim vector to a 3 x 64 x 64 image. The original architecture creates layers with 1024, 512, 256, 128, and 3 channels while my architecture creates layers with 512, 256, 128, 64, and 3 channels. This was done to be less computationally expensive and also to see if the DCGAN could work with less parameters.

Figure 1: Example of original DCGAN paper generator architecture.



Discriminator

The model architecture for the discriminator is essentially a reverse of the generator. My discriminator uses regular 2d convolutional layers to downsample the 3 x 64 x 64 image to layers with 64, 128, 256, and 512 channels while decreasing the length and width of the image until finally passing through a 2d convolutional layer that takes 512 channels and outputs 1 channel of a 1 x 1 image. This is then passed through the sigmoid function to normalize between 0 and 1 to obtain a probability. A value of 1 represents a ‘real image’ prediction while a value of 0 represents a ‘fake’ prediction.

Training Equations

For a DCGAN GAN, the discriminator is optimized by maximizing the following equation:

Equation 1: DCGAN Discriminator Loss

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

The first logarithm in the summation refers to the probability that the discriminator will accurately predict a real image while the second term in the summation refers to the probability that the discriminator will accurately predict a fake image.

For a DCGAN, the generator is optimized by minimizing the following equation:

Equation 2: DCGAN Generator Loss

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

By minimizing equation 2, the generator will learn to fool the generator by generating images that are so similar to real images that they are indistinguishable.

It is important when discussing GANs to explain the above equations with respect to *probability distributions* to aptly communicate the limitations of DCGANs and the improvements discussed later in this report. Bayesian probability and concepts from information theory will be used, but not explained in depth for this analysis. The complete training objective functions for GANs with respect to Bayesian statistics is given in equation 3 below:

Equation 3: DGCAN Objective Function

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Essentially, the generator is trying to minimize the Kullback Liebler (KL) divergence between the distribution of the generated images and the distribution of the real images from the training dataset. The KL divergence is a mathematical concept defined by the below equation that measures the difference between two probability distributions.

Equation 4: Kullback Liebler Divergence

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

The KL divergence is not symmetric, meaning it matters if you are comparing P to Q or Q to P. This property is typically undesirable, so the Jenson Shannon (JS) divergence is often used. The JS divergence is just a symmetric equation that uses the KL divergence to measure the difference between probability distributions.

Equation 4: Jenson Shannon Divergence

$$D_{JS}(g||h) = \frac{1}{2} D_{KL}\left(g||\frac{g+h}{2}\right) + \frac{1}{2} D_{KL}\left(h||\frac{g+h}{2}\right)$$

And so begins the battle between discriminator and generator. As the discriminator becomes better and better at determining a real and a fake image, the generator will be forced to adapt by generating better and better images.

Model 2 - WGAN/ACGAN

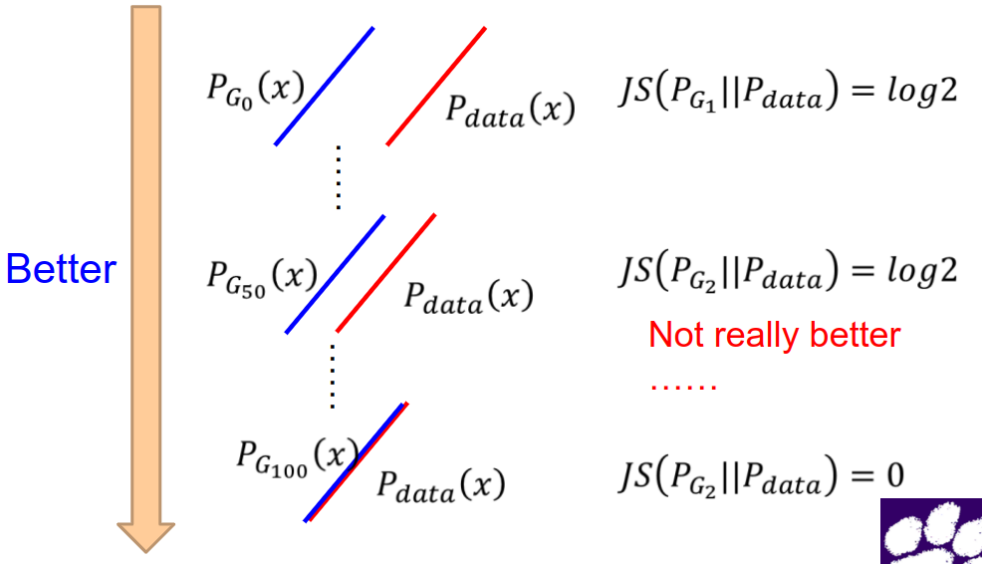
*** Additional Techniques ***

GANs are notoriously difficult to train. As such, the WGAN and ACGAN build upon the previously defined principles of GANs by implementing various techniques to improve the performance and trainability of GAN models. Below I will discuss these techniques.

WGANs

A major issue with training DCGANs emerges when the probability distribution of the training data and the generated fakes do not overlap. When there is no overlap, the JS divergence is always the same (log2). This makes GANs difficult to train, because the distance between distributions could theoretically be improving without any change in the JS divergence.

Figure 2: Improvements Unnoticed by JS Convergence



To improve upon this concept, a different way to measure the difference between probability distributions was used.

For WGANs earth mover's distance was used instead of the JS divergence. Earth mover's distance is described by the below equations. Essentially, it is a measure of the difference between two probability distributions that does not require the distributions to overlap. It is a measure of the best plan to move all of one distribution to another distribution.

Equation 5: Earth Mover's Distance

Average distance of a plan γ:

$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

Earth Mover's Distance:

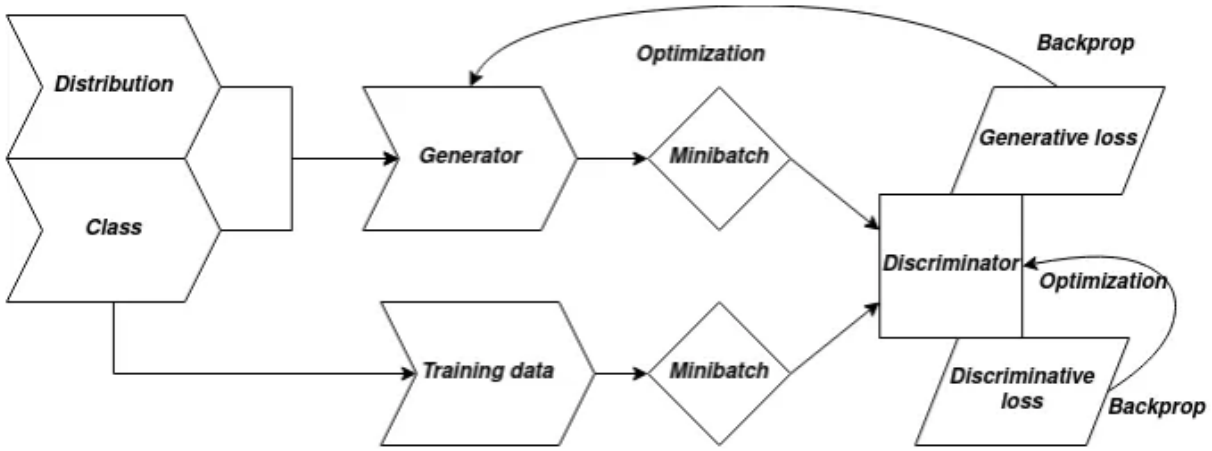
$$W(P, Q) = \min_{\gamma \in \Pi} B(\gamma)$$

In order to optimize using Earth Mover's Distance, the gradient must be smooth enough. To ensure that the gradient is smooth enough, a weight constraint is enforced using a penalty to the gradients of the model calculated during backpropagation.

ACGANs

Auxiliary Classifier GANs (ACGANs) are a specialized GAN that leverage class labels to improve training and allow for additional control when inference with generators. In an ACGAN, the generator takes in noise as input with an additional class label. The discriminator outputs a probability corresponding to if the image is real or fake as well as a predicted class label based on the image. The basic architecture for an ACGAN model is shown below:

Figure 3: ACGAN Model



ACGAN model

The loss function of the ACGAN is divided into two parts:

Equation 6: Log likelihood for the source being checked in ACGAN

$$L_s = E[logP(S = real\,|X_{real})] + E[logP(S = fake\,|X_{fake})]$$

Equation 7: Log likelihood of the class being checked

$$L_c = E[logP(C = c\,|X_{real})] + E[logP(C = \,|X_{fake})]$$

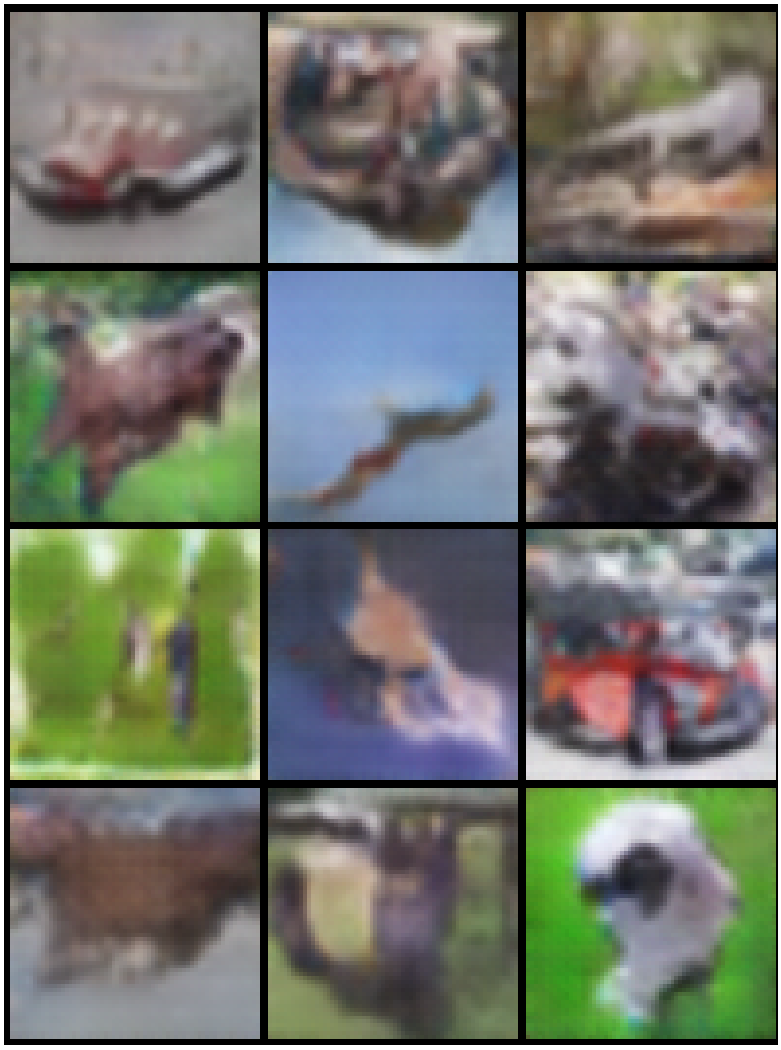
These losses are added together.

So, the generator tries to generate an image from noise that matches its input class description and fool the generator into thinking it is a real image from that class, while the discriminator tries to accurately predict real and fake images as well as their classes.

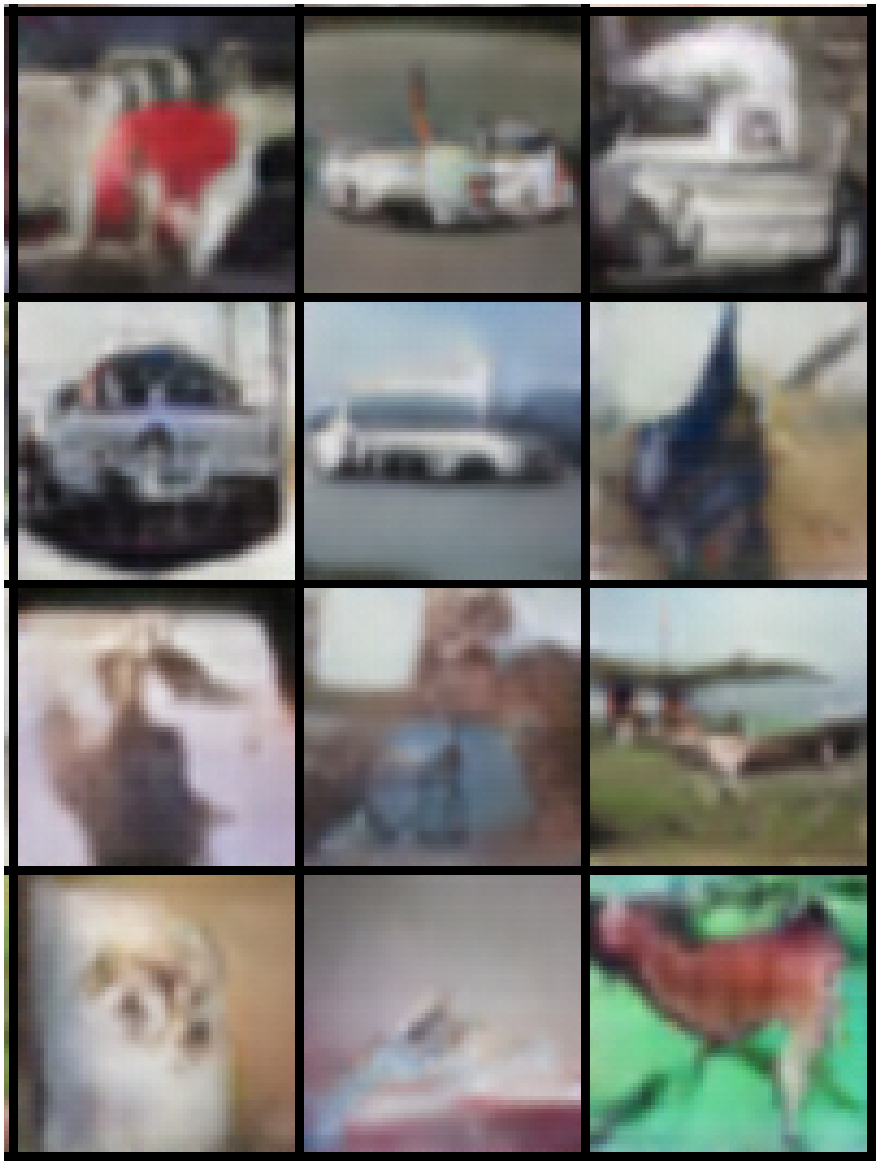
Results

Resulting images from model 1 and 2 are shown below. For model 1, I show the results of the 12 best images after 25 epochs of training and 45 epochs of training for model 1. I trained Model 1 for 250 epochs, but the performance of the model never improved significantly after 45 epochs. In contrast, Model 2 continued to improve as far as 249 epochs. To give a good idea of the best performing model, I included an entire batch of generated images for epoch 249 of Model 2. It seems that Model 1 trained much faster than the Model 2, but Model 2 had a better final performance than Model 1. You can observe that both models seem to have a bit of a “warping effect” on the generated images, but overall the performance is pretty impressive on such simple GAN models.

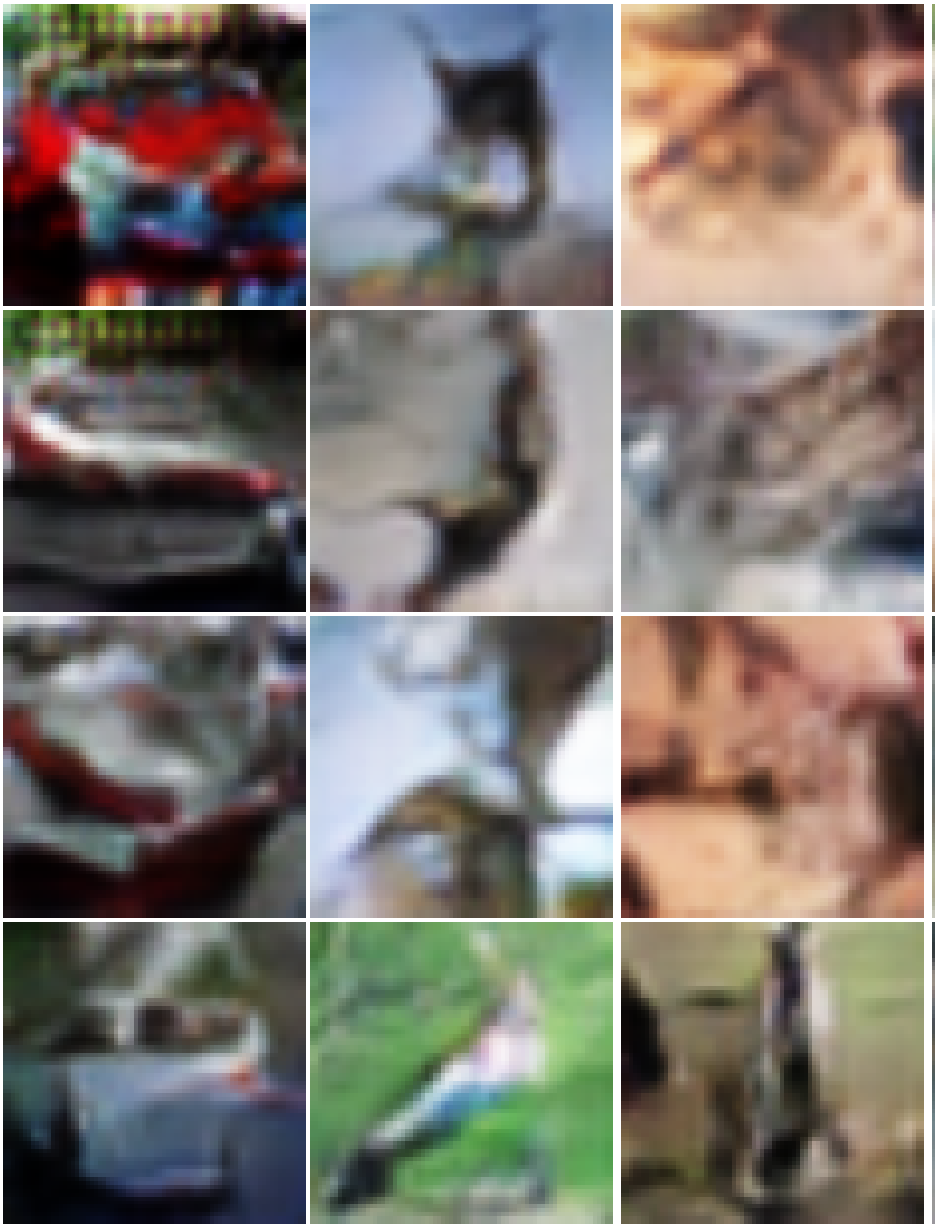
Model 1 - 25 Epochs



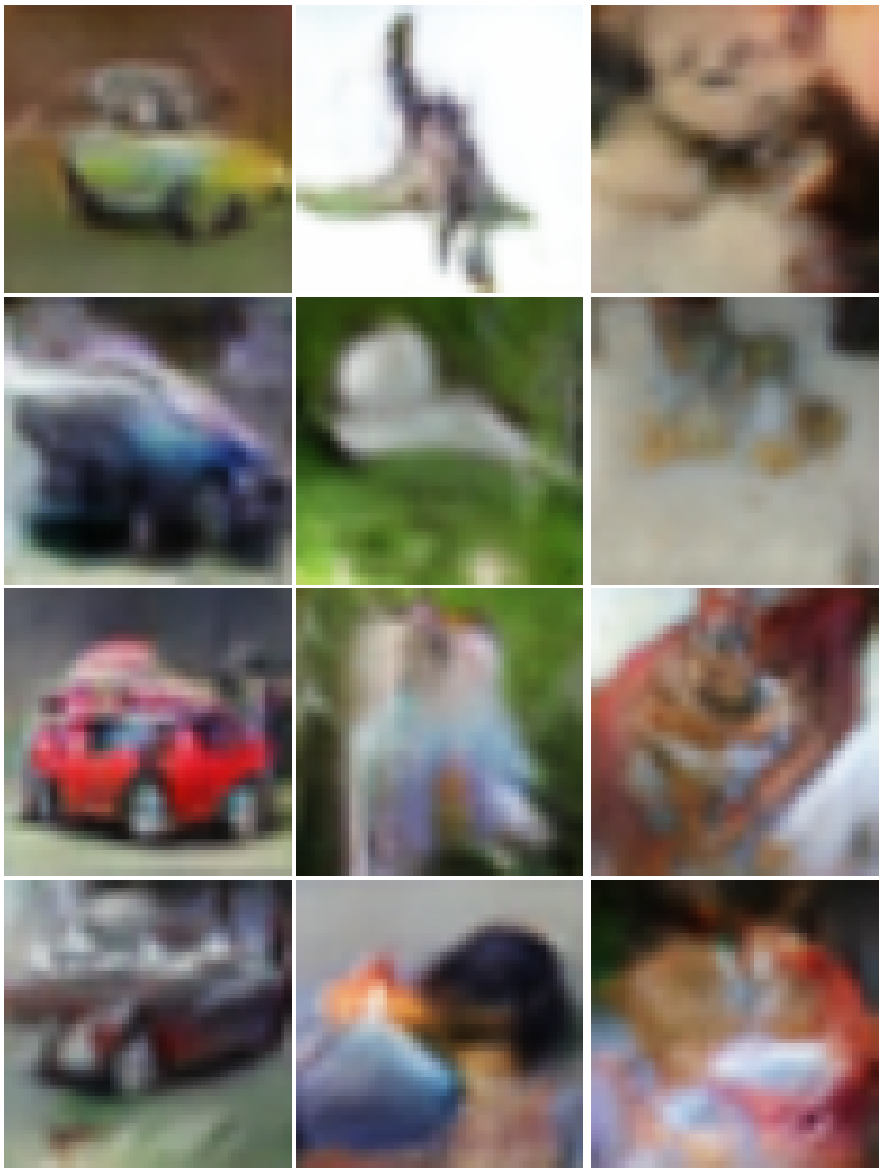
Model 2 - 45 epochs



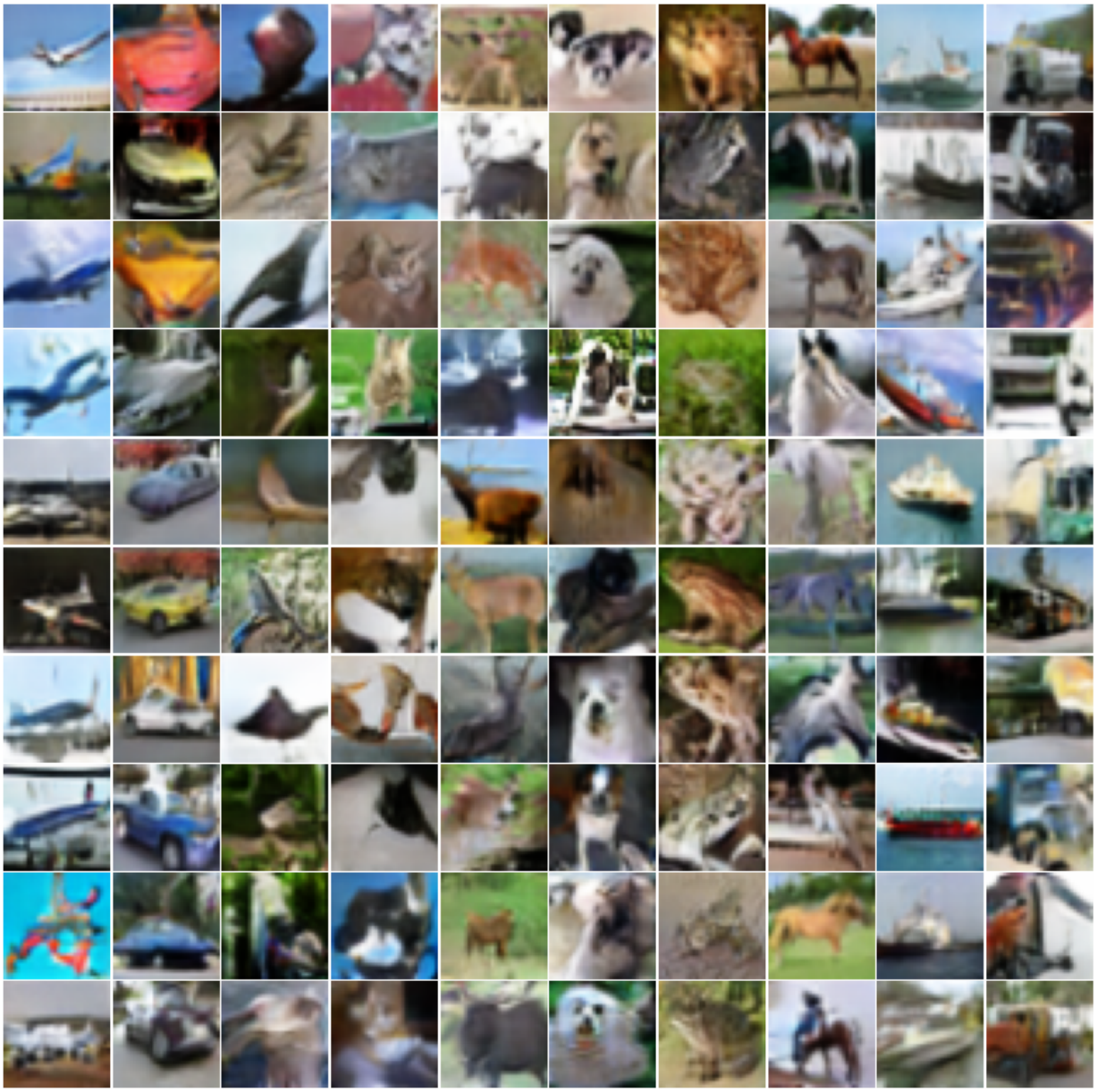
Model 2 - 25 Epochs



Model 2 - 45 Epochs



Model 2 - 249 Epochs



References

Anwar, Aqeel. “What Is Transposed Convolutional Layer?” *Medium*, Towards Data Science, 16 Apr. 2021, https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11.

Arjovsky, Martin, et al. “Wasserstein Gan.” *ArXiv.org*, 6 Dec. 2017, https://arxiv.org/abs/1701.07875.

Bandyopadhyay, Hmrishav. “Understanding Acgans with Code [Pytorch].” *Medium*, Towards Data Science, 23 June 2020, https://towardsdatascience.com/understanding-acgans-with-code-pytorch-2de35e05d3e4#:~:text=ACGAN%20stands%20for%20Auxiliary%20Classifier,learning%20held%20at%20Sydney%2C%20Australia.

Odena, Augustus, et al. “Conditional Image Synthesis with Auxiliary Classifier Gans.” *ArXiv.org*, 20 July 2017, https://arxiv.org/abs/1610.09585.

Radford, Alec, et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.” *ArXiv.org*, 7 Jan. 2016, https://arxiv.org/abs/1511.06434.