

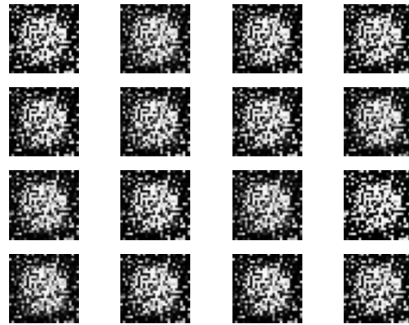
Denali Schlesinger GAN report

Table of Contents

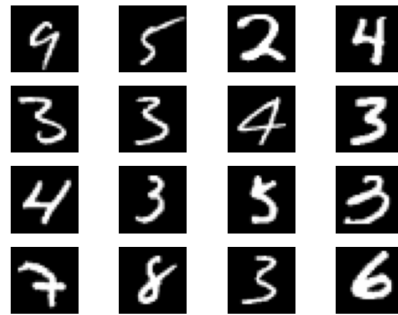
Title	Pg#	links
Introduction	1	link
Data sets	2	link
Specific Gan	3	link
Specific Gan Results	4	link
Increasing Discriminator power	5	link
Changing training demographics	6	link
Single image Gans and uses	7	link
Final implementation	8	link

Introduction:

For my project, I will use open source datasets and our previous generative adversarial network to improve the image quality of the generated images. The model made in class was not very accurate and displayed poor image quality. Over time no general trend in shape appeared and it did not mirror the training data.



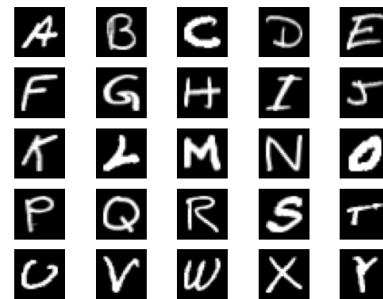
Class model at 1000 epochs(Mnist)



Mnist training data



Class model at 1000 epochs(A-Z)

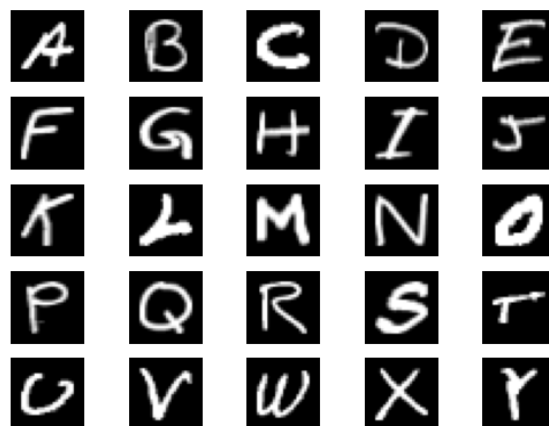


A-Z Handwritten data

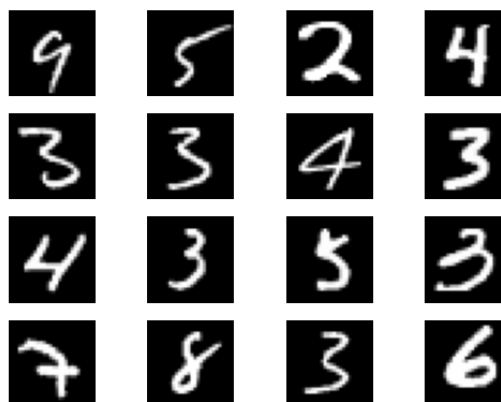
My hypothesis is that these images are a result of conflicting training data, the different structures force the generator to have to adapt to all of the data at once. This causes the generator to continually produce static, to attempt to solve this problem I will experiment with the model's parameters and the data entered into the model.

Data sets:

To train and test the Gan network's image quality I will be using two data sets. First I will use the Mnist data set as its numbers are simpler in structure and it is smaller. Mnist is made of grayscale images of handwritten numbers taken from both high schoolers and census bureau employees. The images are in 28*28 form and organized by the number they represent, there are 60000 training images and 10000 validation images. After I have used the Mnist data set I will use the A-Z Handwritten letters data set which I found on [kaggle](https://www.kaggle.com/sachinpatel123/handwritten-letters), the handwritten data is more complex than the numeric images so it will be my evaluation data. The A-Z data set was posted by Sachin Patel under a creative commons license. It has 370000 samples, each a 28*28 image; there are no images specifically set aside for testing. I downloaded it using pandas as it is a csv and set aside 70000 images for evaluation in case I needed them. I choose to use numeric and alphabetic data as they are both important and well-studied areas of machine learning.



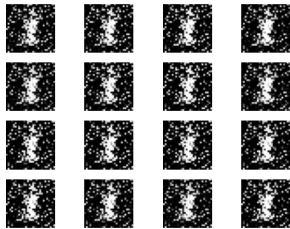
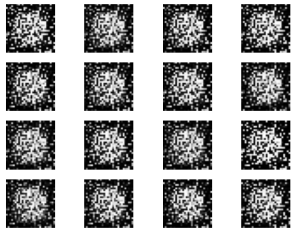
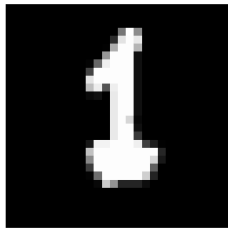
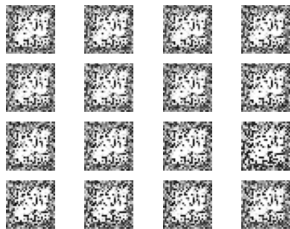
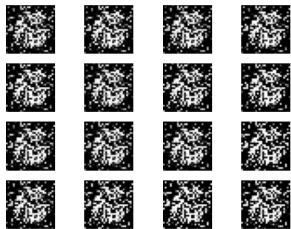
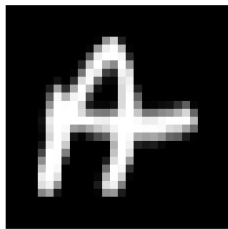
A-Z Handwritten letter examples alphabetic order A-Y



Mnist random assortment of numeric examples

Specific Gan:

My first theory on how to improve my Gan was to only use only one class of images to train the discriminator. I thought that this would allow the generator to focus specifically on the inherent structure of that image type. One article I found from the Computer Vision Foundation¹([link](#)) backed up my theory. The article detailed using multiple selective Gans for multi-class images like the human body and cityscapes. The use of one class of training data with cropped boundaries to prevent multi-class training focuses the generator on producing one clear image. However, this could lead to dissociation from the environment as context would be needed for the generator to fit the generated image into the specific environment, for example, the light level of the room is a reflection of the state of the lamp. Generating a lamp that is on in a darkened room would not accurately model a real-life occurrence. However, I only have uni-class images, meaning I do not need to worry about context between multiple classes. My goal is to improve image quality by narrowing the training data to only the relevant material. To do this I will simply add code to my batch builder that chooses the data to train on based on its class.

Selective Gan image	Nonselective Gan image	Base image
		
		

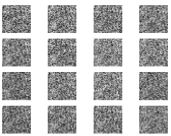
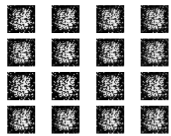
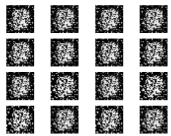
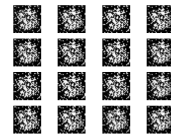
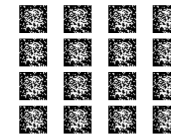
All trained images shown were taken at 1000 epochs and 0.01 lr unless otherwise specified

¹ Li, Yuheng, Yijun Li, Jingwan Lu, Eli Shechtman, Yong J. Lee, Krishna K. Singh, University of Wisconsin-Madison, and Adobe Research. n.d. "Collaging Class-Specific GANs for Semantic Image Synthesis." CVF Open Access. Accessed December 18, 2022. https://openaccess.thecvf.com/content/ICCV2021/papers/Li_Collaging_Class-Specific_GANs_for_Semantic_Image_Synthesis_ICCV_2021_paper.pdf.

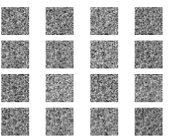
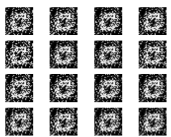
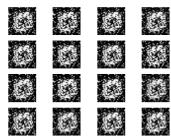
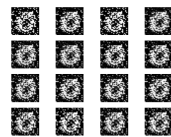
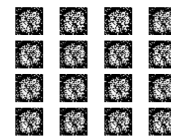
Specific Gan Results:

Using a specific Gan significantly improved the generated image quality. We can see in both data sets that the resulting images somewhat mirror examples from the data set. Over time I saw significant quality progression, unlike in the nonselective model where quality progression stagnated after epoch 1000. These results mean I am on the right path and should continue testing the selective models with other changes.

Nonselective progression over time(Mnist):

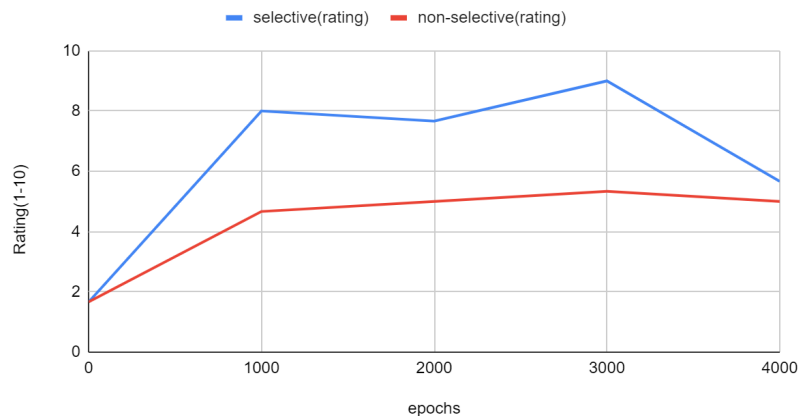
				
Epoch 0	Epoch 1000	Epoch 2000	Epoch 3000	Epoch 4000

Selective progression over time(Mnist):

				
Epoch 0	Epoch 1000	Epoch 2000	Epoch 3000	Epoch 4000

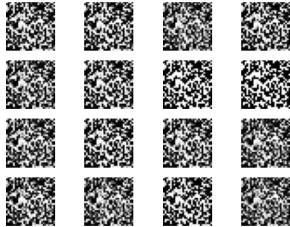
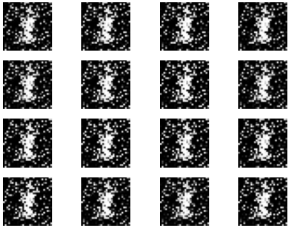
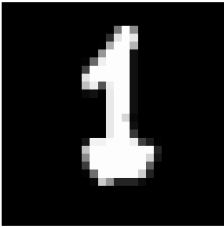
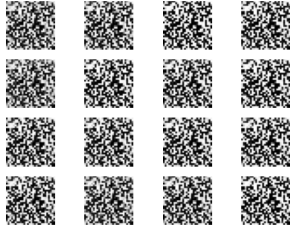
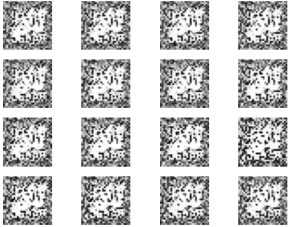
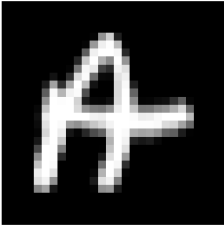
Qualitative analysis of generated images(Mnist)(Sample size of 3):

Qualitative rating over epochs



Increasing Discriminator Power:

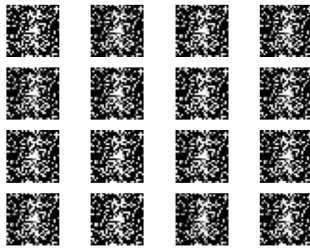
I have noticed a pattern of static surrounding developed numbers when using the specific Gan. To counter this occurrence I will try to increase the discriminator power by adding two 512-unit and two 1024-unit dense layers. My theory is that increased discriminator power will force the generator to drop the surrounding static to increase accuracy, however, this increase in discriminator power could also prevent the generator from gaining adequate feedback on its produced images. If the generator's input never fools the discriminator then no progression can happen. I will be adapting the previous specific Gan because the problem arises from the generated static and to provide a benchmark for any results.

Discriminator Increase	Regular Selective	Base Image
		
		

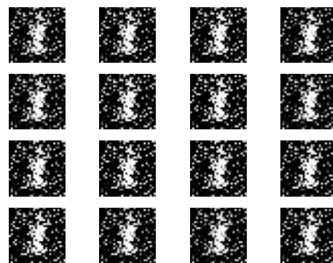
As I thought no generator progression can occur with such a powerful discriminator, I observe no patterns in the generated images indicating that the generator cannot fool the discriminator and therefore can not improve the quality of the image. It would be interesting to play around with model size, however, it would be computationally expensive and hard to train.

Changing training demographics:

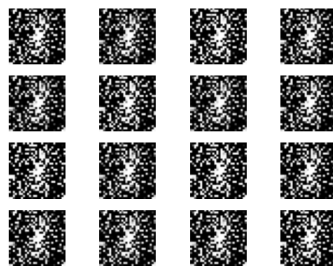
My next idea was to change the percentages of real and fake images in the Gans training data. I theorized that increasing the number of real images would most likely damage the quality of the generated images as the generator would get less feedback. By increasing the amount of generated images I hoped to increase the amount of feedback the generator was receiving and maybe eliminate some of the interference particles seen in the purely specific model.



Selected number 1 25% real images at 1000 epochs



Selected number 1 50% real images at 1000 epochs

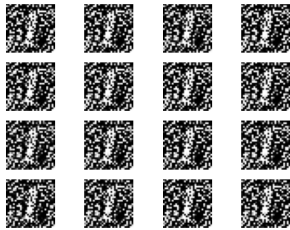
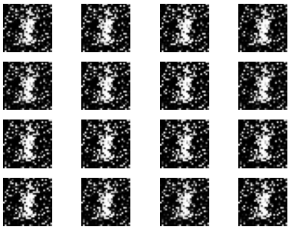
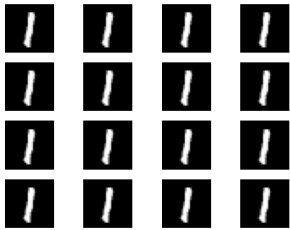
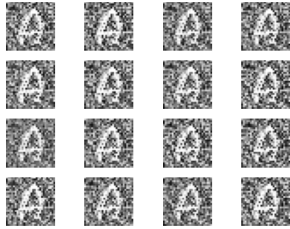
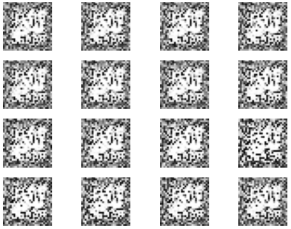
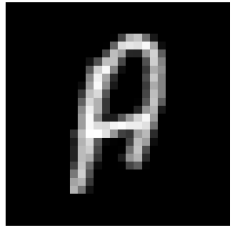


Selected number 1 75% real images at 1000 epochs

Manipulating generated and real image percentages didn't seem to have any positive effect on image quality. In fact, the images produced at these percentages have more particles than the base images. Any more testing with real and generated percentages will most likely have similar results, as it seems 1/1 is the best ratio for real to generated images.

Single image Gans and uses:

Single image Gans are trained on only one image, this makes the discriminator extremely precise and leads to the image almost perfectly mirroring the training data. The Gans usually produce high-quality images but with little variability. Which is not ideal when trying to replicate hand written images. However, this type of training can be useful for the manipulation of data. A group at the Quebec Artificial Intelligence Institute²([link](#)) used an upsampling Gan they called Nu-Gan to generate high-quality versions of a given audio. They did this by upsampling the given input within the generator and training the discriminator with the already high-resolution audio. They then used low-resolution audio to train the Gan as a whole, they were able to successfully reproduce the given sound at a higher frequency. This use for single image Gans could be interesting but as neither the Mnist nor the A-Z datasets have different resolution images I will use a single image Gan to test how my model adapts to the given data.

Single Image Gan	Regular Selective	Single Image
		
		




Using single image Gans works well, image quality is high and most of the generated images are easily recognizable. However, the generator overfits to the training data as there is only one unique image.

² Kumar, Rithesh, Kundan Kumar, Vicki Anand, Yoshua Bengio, and Aaron Courville. n.d. "NU-GAN: High resolution neural upsampling with GAN." arxiv. <https://arxiv.org/abs/2010.11362v1>.

Final implementation:

To show my findings I will build a text-to-handwriting converter. To do this I will train a single image model for all twenty-six letters and let the user choose what to say after the training is finished. I think this is the best course as the single fit images had the best quality results, the A-Z letters are complex and a selective or nonselective model will not be able to adapt to the training data to a high degree due to time and processing power constraints. Although there will be less variability in this path due to there being only one training point per image I think it can create the feel of handwritten letters due to the fact the generator will not completely adapt to the single image in the time allotted. To combine the letters into the shape of the inputted string I will create a matplotlib one-dimensional array and add each generator prediction in sequential order.

Test cases using the A-Z Handwritten Dataset(proof of concept)

"ABC"	
"Hello"	
"Dice"	

Images poor quality due to time constraint