Team Machine's Journal of Successes and Failures:

(From September 2019 to April 2020)

https://github.com/GrizzlyMagnumJr/GAN_Networks

This is Team Machines GitHub for Generative Adversarial Networks(GANs).

September, 2019:

dcgan.py - - Our code so far in trying to create our own DCGAN based off of the architecture from the medium article (https://medium.com/@liyin2015/dcgan-79af14a1c247)

modified_keras_gan.py - - This is our heavily modified code based off of the sample code given to us. We did a lot of playing around to see how using different optimizers, activations, batch sizes, and epoch values would affect the generator/discriminator loss and the accuracy of the images created.

https://github.com/DIUx-xView/data_utilities xview2.py - November 11th, 2019 - - Trying to import images from the xView dataset and then chipping those images in to smaller images. In the code, we chipped them to 1000x1000 but this can easily be changed. We will most likely go with 224x224 to start and then slowly start to input and generate larger images. Probably the size of 600x600 chip or larger. We are implementing this code in our "Keras-GAN.py" program for now instead of the "Keras_DCGAN.py". This is because the latter program takes a much longer time to run and at this moment, we are mostly going through trial and error with what we have right now. After we can plot generator and discriminator loss and generate a sample of fake images, we will move on to the DCGAN program and alter that architecture. We are currently assuming that at first, we will generate low quality images and the goal after that is to deepen our architecture to improve the quality. If our customer or we are not satisfied with the results, we'll change the architecture completely and go with a different approach. This means we will use a different type of GAN. Later on, we will upload our prototype code that generates images from the xView dataset. That will be uploaded as soon as that happens. From there, we will upload all of our improved GANs as well.

chip_xview.py - - This is the program that was modified from "Keras-GAN.py". wv_util.py - - This is the program that is imported in to chip_xview.py

November 21st, 2019 - chip_script.py - : Just separated the chipping program from the main architecture one. We did this so that our main program does not have to keep chipping the dataset's images every time the code runs. It was okay before because we were using a very small amount (3 to 7) of images and testing the different sizes and such. Since we are switching our direction (Dropping the mindless error fixing of chip_xview.py), we will just run the chip program once on the entire dataset and just use that folder. Our plan from today and forward is to write our own training code and methodology. We still plan to play around with the architecture to see what works well and what does not. The end goal is to generate fake images and train (or not train at first) our discriminator to identify/verify those images. We want our discriminator to be able to know if an image is fake or real and we want it to be accurate. We will have different architectures to see what works well and document that but later on.

November 25th, 2019 - prototype1.py - : Our architecture thus far. Many things have to still be added and modified. Modifications have been made to chip_script.py as of today as well.

December 6th, 2019 - Last day of semester. Added a few lines to prototype1.py but havn't uploaded those changes on this Github file(only like 2 lines of code added). We are currently waiting for the code review of our prototype1.py script from our Subject Matter Expert(SME). We will continue work and research after we recieve feedback from our SME. Will most likely not do much over the winter break until then. We'll most likely be back around January 10th with an update. First day of Spring 2020 semester starts on January 21st.

December 26th, 2019 - prototype2.py - : Added a few lines of code in order to start training and even more lines for debugging those lines. Training not complete yet, almost there. Checking out diffrent batch sizes, epochs, and runtime of the code. Currently having some trouble building the combined model which combines the generator and discriminator. The error is because of the dense layer. Also added a few lines of code for TensorBoard. This isn't ready yet either, mostly because the generation has to be complete first. Still waiting for the SME code review. Added prototype2.py which contains all of these changes.

January 2020:

January 8th, 2020 - setup_environment.sh - : Uploaded this bash scrip to just install all of the required packages needed to run the codes. This is meant for the AWS instance. In case it ever ends and all of the data on it is erased, we can easily just set up the instannce without running it over and over and seeing what to install. This is meant for setting up an environment quickly and in the case our instance is closed and our

environment and data is erased. This script was made for Kali/Ubuntu machines, haven't tried on MAC. It should contain MOST of the packages, not all of them.

January 20th, 2020 - : Finally separated the chipping and training code. Now when we run code that trains the generator and discriminator, it does not chip the dataset. This means that the dataset is not chipped every single time we run the code. When we want to strictly train our dataset, we run prototype2.py. When we want to ONLY chip the dataset, we will run chip_script.py. To chip the dataset, uncomment line 147. The programs that reflect these changes are prototype2.py and chip_script.py.

February 2020:

February 11th, 2020 - : We have fixed our discriminator and training code and combined it with the generator that our SME has given us to create our first traininable GAN. The working training code and chip file plus the needed import files are all inside of the 'working_gan' folder. We also added a small batch of images to show that our discriminator is training and produces more accurate images as the program runs. Our machines are not able to handle the load of the program anymore so we will have to start using the AWS Instance that was created for us by our SME. The plan onward is to observe the generated images with a larger batch, start using Tensorboard to keep track of our loss and accuracy, and also modify our layers to fix our accuracy of the fake images.

February 18th, 2020 - subfolder_train.py/subfolder_chip.py - : These programs contain our up-to-date models. The chip program has been modified to select a number of images equal to the batch size randomly from 12 different folders that contain 10,000 images in each. Currnetly, we have an unstable generator and discriminator. This can be concluded by observing the loss for each model on our graph. The plan right now is to modify the current layers and plot the loss of each model on Tensorbord.

March 2020:

March 12th, 2020 - Different_runs folder - : This folder contains all of the experiment runs we've been conducting. We've been trying to get rid of our mode collapse problem and also improve the quality in the images that we've been creating. To do this, we have tried adding, removing, and modifying the layers of our Generator and Discriminator. We've also tried different batch sizes, epochs, and epoch sizes. By the end of March, we hope to have low losses for both models and have higher quality images that are similar to the xView dataset. SAGE Competition at the end of next month.

April 2020:

April 3rd, 2020 - Progressive_gan.py - : This path month we have been working on desiging a Progressive GAN so that we could avoid mode collapse when we are training our models. Our DCGAN was having this issue and we could not seem to fix it so our SME recommended us to use a Progressive GAN instead. We also modified our chip program. Both of these will be uploaded with a more detailed explanation of the code tomorrow.

April 28th, 2020 - Over the past few weeks, we have been training on the instance all day and all night to get rid of mode collapse. It has been working for most of the models for the Progressive GAN but not for the last one. Modifying the layers for each model and adding and trying new things has been working so far. We are actually really pleased with the images that we've been generating specifically with 128x128 because we've never had so many colors in a single image before. It's upsetting that we can't get more complex than what we've generetated up to this day. We understand it's hard and takes a much longer time but we wish we could have done more. I'm placing a few samples from each model of the Progressive GAN based on shape in the 'Final Product' folder. It's in a .zip file named 'Sample_Generated.zip'. This folder is also where we will be uploading our Produut Specifications, final report, and both of our final presentations. This will probably conclude our work on this. Thanks for reading.

Special thanks to Tim Parker and Jonathan C. Brant for helping us through this whole project. They were our customer and subject matter expert respectively. They provided us with help, resources, and taught us how things work in the real world. Some of those things include what would be expected from us if this was a real contract.