Test Plan and Results - ShenaniGANs

Overall Test Plan

For our project, testing will be of critical importance throughout the development of our GAN, as numerous bulk tests of image generation will be required to measure the progress and improvements made to our network. To do this rapidly, we will attempt to run our GAN on the Ohio Super Computer (OSC) using TensorFlow. This will greatly increase our capability to successively test our GAN as code changes are made. In the early phases of testing and development, we expect our tests to primarily be unit and whitebox tests. The internals of our neural network will be the first components we test to ensure basic functionality. Afterwards, we will be observing how modifications to the network influence the realism of our GAN's output when compared to the input images it receives.

Since there are many variables that can influence the results of our tests, most of our testing will be done after a change to only one variable at a time. Using this method, we can pinpoint exactly what is influencing the image output and tweak it to best meet our expectations. A large factor that will come into play during our integration tests (especially those over multiple iterations of our network), will be the amount and variability of the input data we provide. For most of our testing we intend to use the same database of images, all scaled to the same size, and ideally containing a large number of images (several thousand at least).

Test Plan Descriptions

ID	Purpose	Description	Inputs	Outputs	Normal / Abnormal / Boundary	Blackbox / Whitebox	Functional / Performance	Unit / Integration
T1	Verify input image conversion to usable data structure	The beginning portion of the program should be able to take in images and turn them into raw data to be used in other parts of the program consistently.	Images from the keras database	Raw data to pass further into the program	Normal	Whitebox	Functional	Unit
T2	Verify generation of new image data structure in same format	The generator function should be able to take in raw data of an image, and output the same kind of data	Raw image data	Raw image data	Normal	Whitebox	Functional	Unit
T3	Verify domain subject in output images appears similar to an input domain subject	The images created by the program should be recognizable as similar to the original images by anybody	Images from keras database	New images	Normal	Blackbox	Performance	Integration

T4	Backgrounds appear similar to those of input image backgrounds	The backgrounds of output images should also be similar to the original images, recognizable by anybody	Images from keras database	New images	Normal	Blackbox	Performance	Integration
T5	Image colors should be appropriate	The colors of the new images should be colored similarly to the original images, if the original dataset is all black and white, so should all the outputs.	Images from keras database	New images	Normal	Blackbox	Functional	Unit
T6	Discriminator Correctness Percentage	The discriminator should ideally be correct 50% of the time, but realistically correct only 70% of the time.	Images from the generator	Determinatio n proportion	Normal	Whitebox	Performance	Integration
T7	Training two GANs on same dataset and swapping discriminators	Changing the discriminator should produce the same results, as it should show that the entire system is not sensitive to small variances in the original dataset.	Images from keras database	New images	Abnormal	Whitebox	Performance	Integration
T8	Image output is appropriate size and resolution for viewers	Depending on the size of the images in the original dataset, the output images should be of similar size, as they should be as similar as possible.	Images from keras database	New images	Normal	Blackbox	Performance	Integration
T9	Execution time for a single epoch of Neural Network	The execution time can be changed by a number of factors, including computer speed, resolution of the images, layers in the network	Images from keras database	Time of epochs	Boundary	Whitebox	Performance	Integration
T10	Provide GAN with several copies of the same input image	Giving the same image to the GAN several times will test our noise generation, and there should always be random variation with the output due to it.	Multiple of the same image	Slightly altered image	Abnormal	Blackbox	Functional	Unit
T11	Provide GAN with copies of rotated images	Giving the GAN the same image but rotated to different degrees. This should produce wildly variable images from the original image	Multiple of the same image, rotated	Images similar to the original	Abnormal	Blackbox	Performance	Integration

T12	Provide GAN with filtered images	Giving the GAN the same image but with varying filters or color differences should vary the output in many different ways, but overall be the same image with strange colors	Multiple of the same image with color differences	Images similar to the original, but with strange colors	Abnormal	Blackbox	Performance	Integration
T13	Run GAN with varying number of iterations	Varying the number of iterations the GAN runs before outputting the image to the user will vary the quality of the image, but to varying degrees as well. Seeing the upper and lower limits will help with optimization/time	Images from keras database	New Images	Boundary	Blackbox	Performance	Integration

Test Case Matrix

Test ID	Normal / Abnormal / Boundary	Blackbox / Whitebox	Functional / Performance	Unit / Integration
T1	Normal	Whitebox	Functional	Unit
T2	Normal	Whitebox	Functional	Unit
T3	Normal	Blackbox	Performance	Integration
T4	Normal	Blackbox	Performance	Integration
T5	Normal	Blackbox	Functional	Unit
T6	Normal	Whitebox	Performance	Integration
T7	Abnormal	Whitebox	Performance	Integration
T8	Normal	Blackbox	Performance	Integration
T9	Boundary	Whitebox	Performance	Integration
T10	Abnormal	Blackbox	Functional	Unit
T11	Abnormal	Blackbox	Performance	Integration
T12	Abnormal	Blackbox	Performance	Integration
T13	Boundary	Blackbox	Performance	Integration