Thanks for Professor Chen’s HW3, we had an opportunity to practice and study DCGAN code in a deeper level than before.

First thing to note is there are several hundred gray scale image mixed within the huge dataset and cause error at DCGAN origin program. We adopt two approach to run through the code.

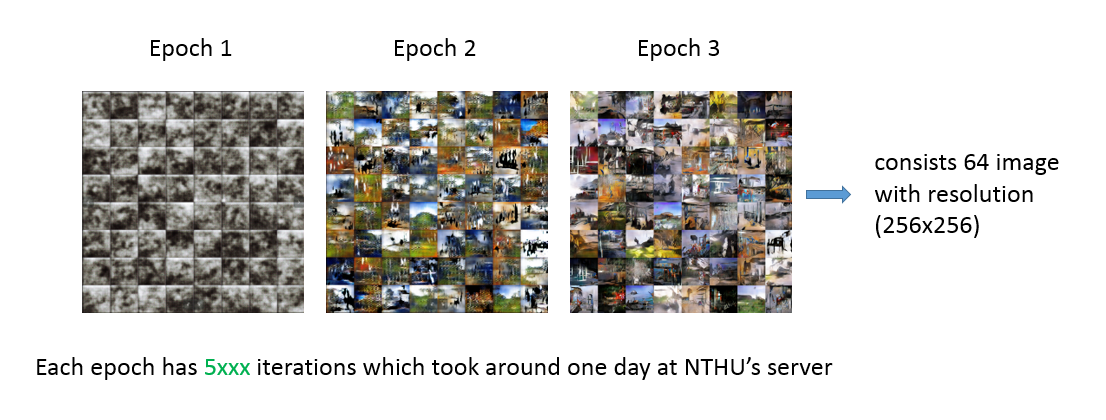
1. Filter out these grayscale images since the numbers of gray image is pretty small compare to original number (700 vs 300000)
2. Or use cv2.cvtColor to transform grayscale image to RGB

After run through the code smoothly, below is our first flow to test DCGAN

1. Directly modified DCGAN code to output 256x256 size image
2. Train the model at full dataset for around one weeks instead of just indoor dataset
3. Then applied transfer learning to continue the training at indoor dataset for another 7 days

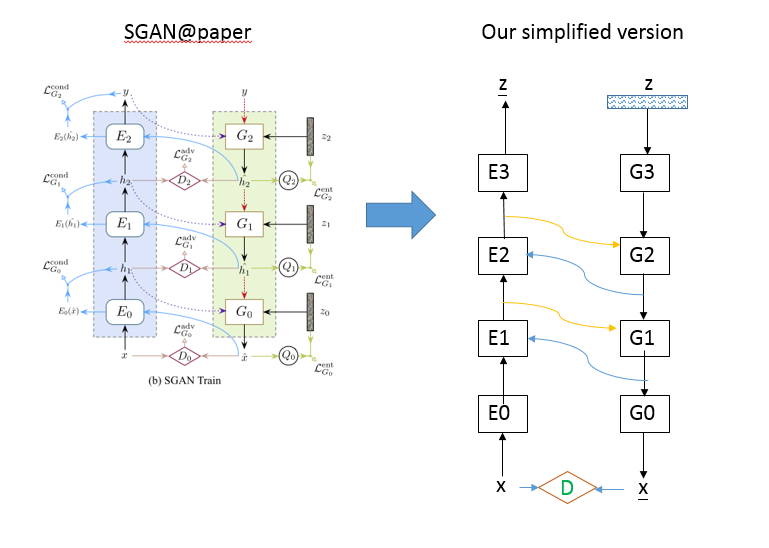
The iterations per epoch is around 5000 and took over around one day to complete

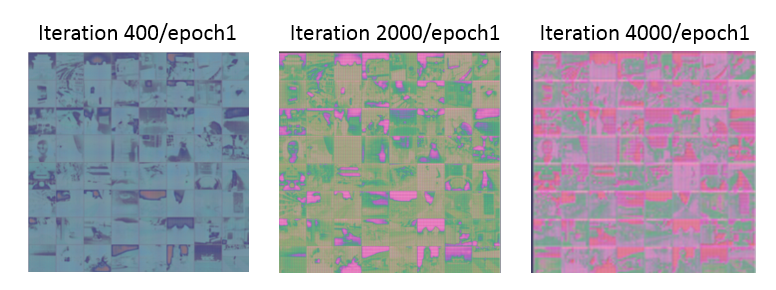
We can see the generated image is becoming real after several epochs while still vague at close look

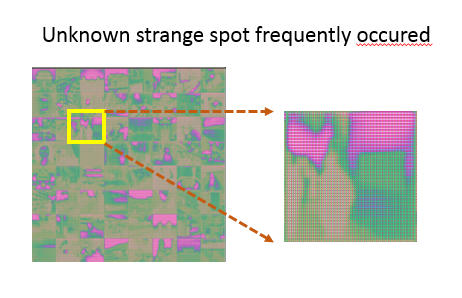


Besides directly modify output dimension, the second flow we tried is to mimic the approach at residual network & stacked GAN which will connect a shortcut between the same dimension layers between discriminator and generator.

To simplify the structure, we skip the cost function between the shortcut







Due to limited GPU resource, the run at CPU is quite slow and still waiting the result.

Adding two-way shortcut seems obviously slow the optimization time

In sum, we found DCGAN not so easy to optimize well and lots hands on needed to tune the framework. The strange spots are both observed at our first flow and second flow. Despite the difficulties to get good result of DCGAN, we really enjoy the experience learned from these raised unknown behavior and challenge