**Summer 2022 Deep Learning**

**Report of Lab #5**

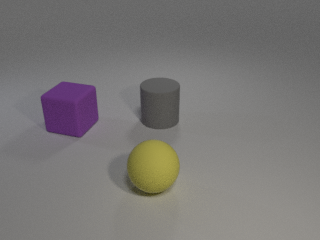
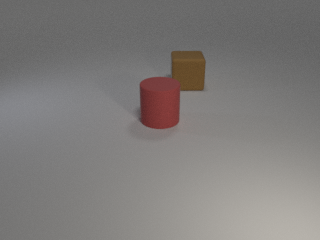
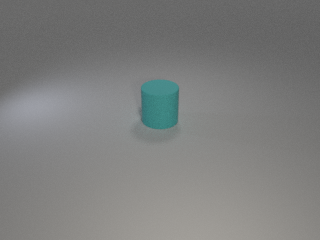
Student name: 劉子齊 Jonathan

Student ID: 311605004

**Part 1: Introduction**

In Lab5, our goal is to implement a conditional GAN to generate synthetic images through the given multi-label conditions. Also, we would have to design our own generator and discriminator, and choose the loss function and the optimizer that fit our model the best.

For the dataset in this lab, we got over 18, 000 images with corresponding labels for the training data, which was recorded in the “train.json”. Each training image got 1 to 3 objects in it. The objects in the image could vary in 3 different shapes and 8 different colors. The following figures are how the input data looked like.

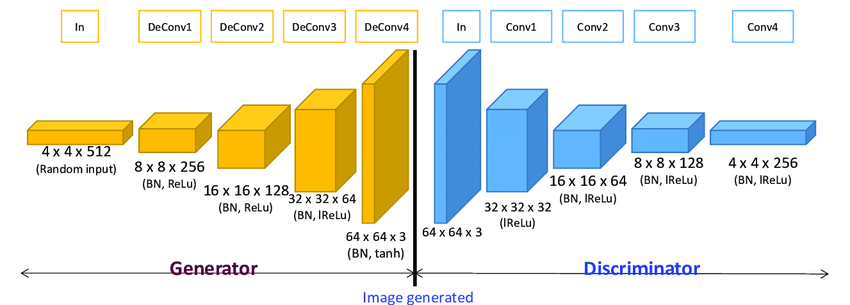
  

With the images and labels fed into the discriminator we designed and trained, the discriminator should be capable to classify the objects and the colors of the objects in the given image. Besides, the generator we constructed should also be capable to generate the image corresponding to the object condition we gave through the initialized noise we provided.

**Part 2: Implementation Details**

**Part 2-A: Architecture of my GAN**

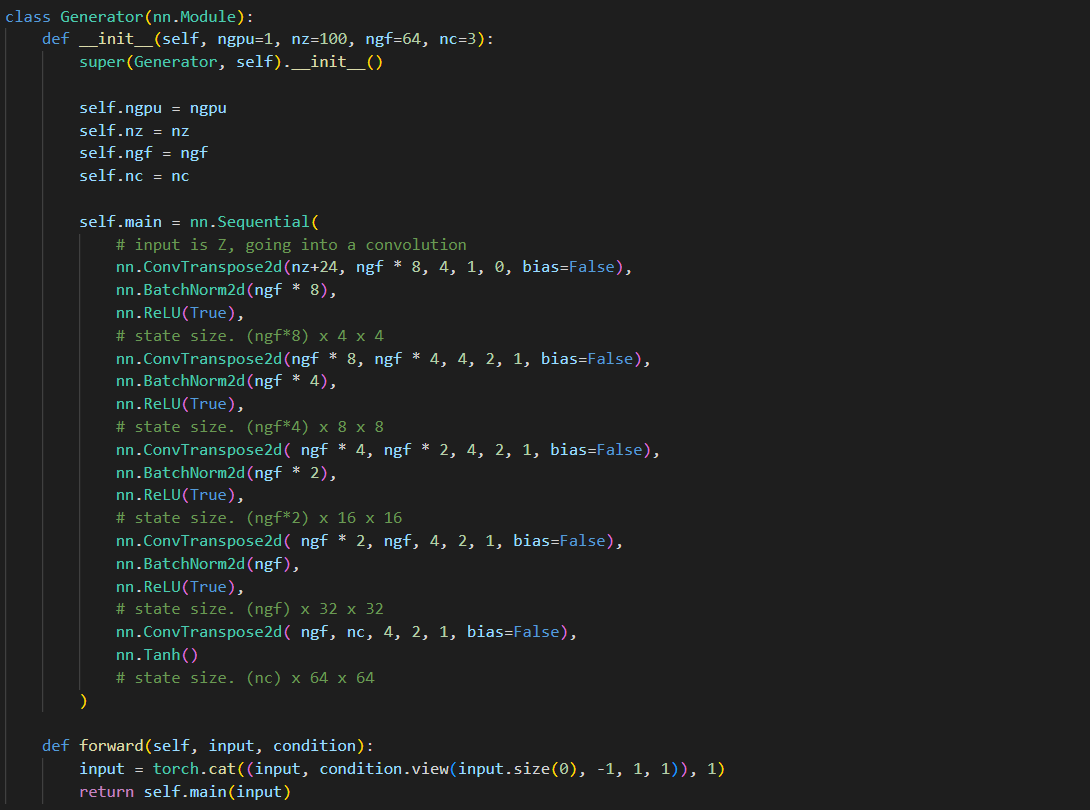
The GAN architecture I chose for this lab is the conditional GAN. Besides, I designed the generator and the discriminator as the architecture as DCGAN, which was shown in the following figure.

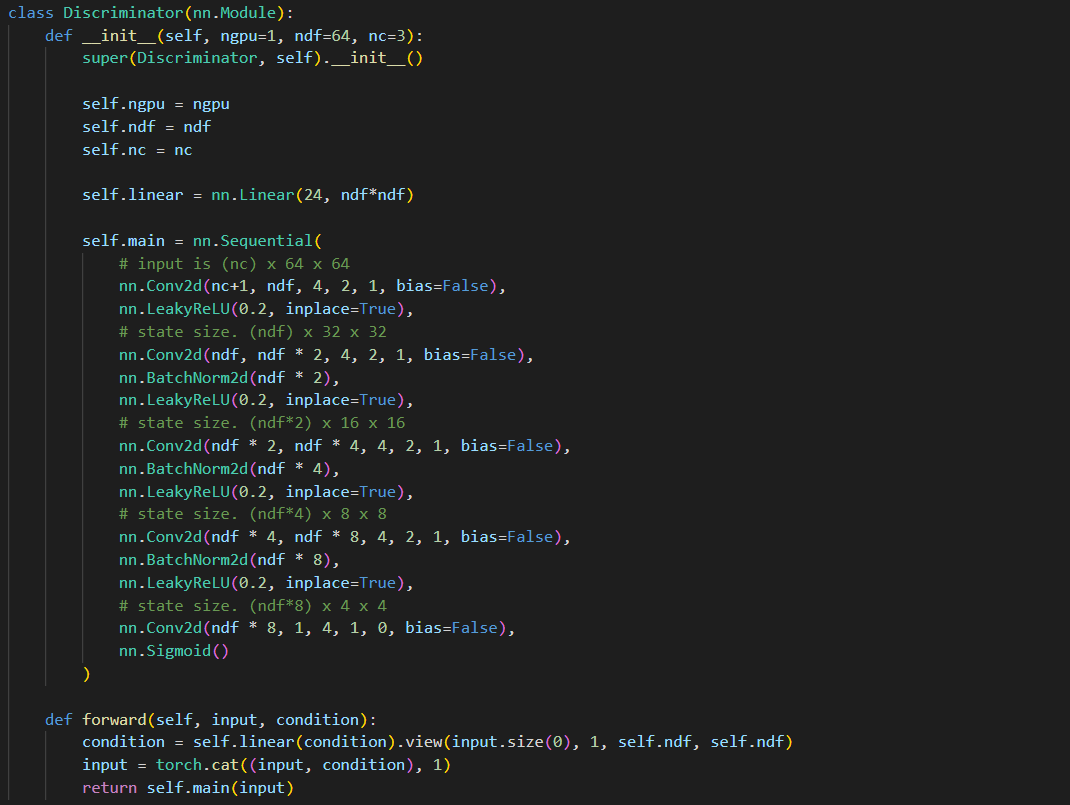


For conditional GANs, we would concatenate the condition data with the images and the latent variables in the form of one-hot vectors. In the “Conditional DCGAN”, the generator was constructed by Deconvolution layers, Batch Normalize layers, and ReLU layers, just as the architecture of the DCGAN. Besides, the conditional data would directly be concatenated into the latent variable in the generator.

Then we would come to the discriminator, which was constructed by Convolution layers, Batch Normalize layers, and Leaky ReLU layers. The conditional data would be passed into a linear layer here in the output size of 64 x 64, which made the condition data turned into an input image sized array. Afterwards, we would concatenate the condition array after the image.

Hence, this made the architecture of our “Conditional DCGAN” different from the “DCGAN” architecture in the figure above, we would have 4 channels for the image. The following figures were how I implemented the generator and the discriminator of the “Conditional DCGAN”:





**Part 2-B: Training Loss Function**

Initially,

**Part 3: Experimental Results & Discussions**

**Part 3-A:**

**Part 3-B:**