Environment:

Microsoft Windows 10 Pro

RAM: 8GB

CPU: Intel i5-8265U (64-bit)

GPU: GeForce MX150 (Using Korea University GPU server)

Version: Python 3.7.4

Data collection method:

First, I connected to the GPU server using MobaXterm.

https://github.com/eriklindernoren/Keras-GAN/blob/master/README.md#cyclegan

The data was obtained using the command via GitHub above (based on Ubuntu):

```
$ git clone https://github.com/eriklindernoren/Keras-GAN
$ cd Keras-GAN/
$ sudo pip3 install -r requirements.txt
```

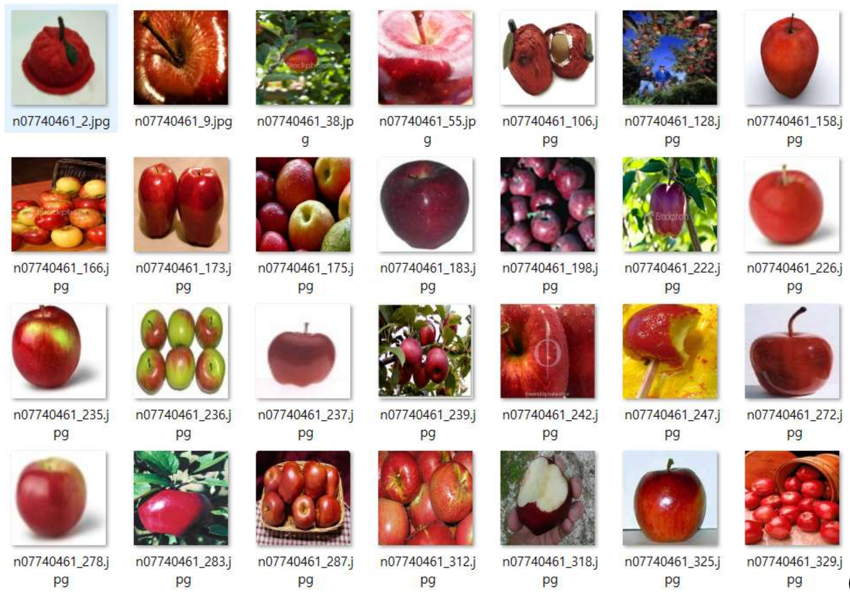
After that,

```
$ cd cyclegan/
$ bash download_dataset.sh apple2orange
$ python3 cyclegan.py
```

After downloading the data set by executing the corresponding command, the model was trained using Python.

The source train set used apple2orrange in the Keras-GAN folder.

A trainset containing images of apples and oranges is stored in that folder.



(Source Train set)

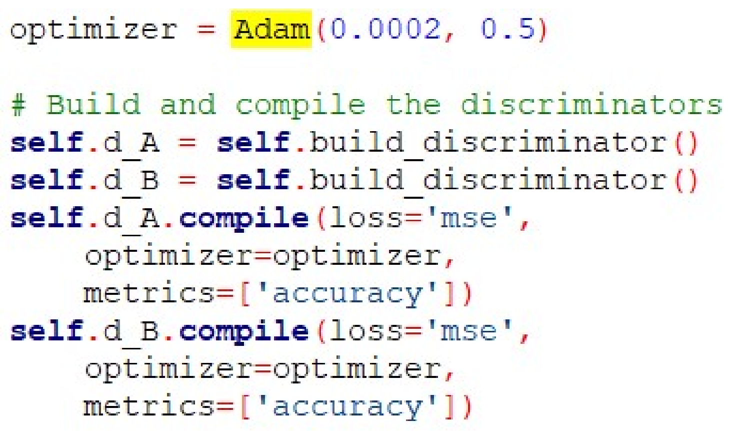
Description of the data used to train the model:

As described above, the source train set used the corresponding GitHub data, and the total number of train images was 995, resulting in a batch value of 995. However, it took too long, so I lowered the batch value and ran it. In the trainset, there are folders Test A and Test B. In A, there are images of apples (with features removed) and in B, images of oranges (with features removed). I went through the process of converting from oranges to apples. The output image of the process is saved as a target train set and shows the process (cycle) of changing apple -> orange -> apple.

Hyperparameter:

Hyperparameter can more accurately find the optimization of each parameter through Random search.

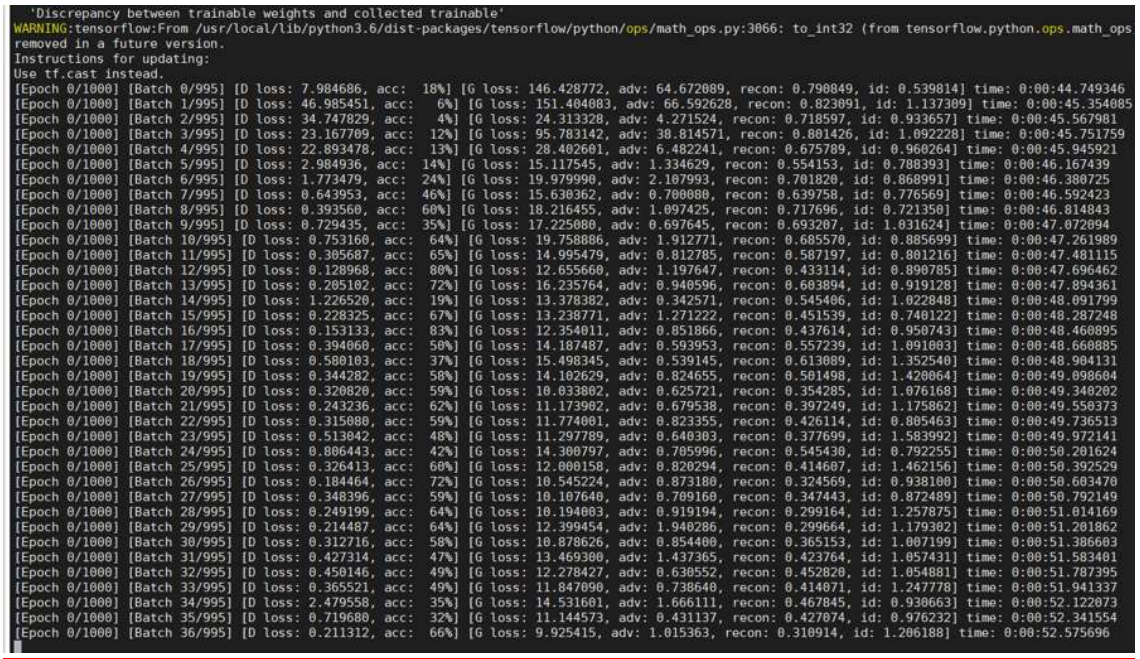
The optimizer settings are as follows:



```
optimizer = Adam(0.0002, 0.5)

# Build and compile the discriminators
self.d_A = self.build_discriminator()
self.d_B = self.build_discriminator()
self.d_A.compile(loss='mse',
    optimizer=optimizer,
    metrics=['accuracy'])
self.d_B.compile(loss='mse',
    optimizer=optimizer,
    metrics=['accuracy'])
```

Training image with the above settings:



```
  'Discrepancy between trainable weights and collected trainable'
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops
removed in a future version.
Instructions for updating:
Use tf.cast instead.
[Epoch 0/1000] [Batch 0/995] [D loss: 7.984686, acc:  18%] [G loss: 146.428772, adv: 64.672089, recon: 0.790849, id: 0.539814] time: 0:00:44.749346
[Epoch 0/1000] [Batch 1/995] [D loss: 46.985451, acc:   6%] [G loss: 151.404083, adv: 66.592628, recon: 0.823091, id: 1.137309] time: 0:00:45.354085
[Epoch 0/1000] [Batch 2/995] [D loss: 34.747829, acc:   4%] [G loss: 24.313328, adv: 4.271524, recon: 0.718597, id: 0.933657] time: 0:00:45.567981
[Epoch 0/1000] [Batch 3/995] [D loss: 23.167709, acc:  12%] [G loss: 95.783142, adv: 38.814571, recon: 0.801426, id: 1.092228] time: 0:00:45.751759
[Epoch 0/1000] [Batch 4/995] [D loss: 22.893478, acc:  13%] [G loss: 28.402601, adv: 6.482241, recon: 0.675789, id: 0.960264] time: 0:00:45.945921
[Epoch 0/1000] [Batch 5/995] [D loss: 2.984936, acc:  14%] [G loss: 15.117545, adv: 1.334629, recon: 0.554153, id: 0.788393] time: 0:00:46.167439
[Epoch 0/1000] [Batch 6/995] [D loss: 1.773479, acc:  24%] [G loss: 19.979990, adv: 2.107993, recon: 0.701820, id: 0.868991] time: 0:00:46.380725
[Epoch 0/1000] [Batch 7/995] [D loss: 0.643953, acc:  46%] [G loss: 15.630362, adv: 0.700080, recon: 0.639758, id: 0.776569] time: 0:00:46.592423
[Epoch 0/1000] [Batch 8/995] [D loss: 0.393560, acc:  60%] [G loss: 18.216455, adv: 1.097425, recon: 0.717696, id: 0.721350] time: 0:00:46.814843
[Epoch 0/1000] [Batch 9/995] [D loss: 0.729435, acc:  35%] [G loss: 17.225080, adv: 0.697645, recon: 0.693207, id: 1.031624] time: 0:00:47.072094
[Epoch 0/1000] [Batch 10/995] [D loss: 0.753160, acc:  64%] [G loss: 19.758886, adv: 1.912771, recon: 0.685570, id: 0.885699] time: 0:00:47.261989
[Epoch 0/1000] [Batch 11/995] [D loss: 0.305687, acc:  65%] [G loss: 14.995479, adv: 0.812785, recon: 0.587197, id: 0.801216] time: 0:00:47.481115
[Epoch 0/1000] [Batch 12/995] [D loss: 0.128968, acc:  80%] [G loss: 12.655660, adv: 1.197647, recon: 0.433114, id: 0.890785] time: 0:00:47.696462
[Epoch 0/1000] [Batch 13/995] [D loss: 0.205102, acc:  72%] [G loss: 16.235764, adv: 0.940596, recon: 0.603894, id: 0.919128] time: 0:00:47.894361
[Epoch 0/1000] [Batch 14/995] [D loss: 1.226520, acc:  19%] [G loss: 13.378382, adv: 0.342571, recon: 0.545406, id: 1.022848] time: 0:00:48.091799
[Epoch 0/1000] [Batch 15/995] [D loss: 0.228325, acc:  67%] [G loss: 13.238771, adv: 1.271222, recon: 0.451539, id: 0.740122] time: 0:00:48.287248
[Epoch 0/1000] [Batch 16/995] [D loss: 0.153133, acc:  83%] [G loss: 12.354011, adv: 0.851866, recon: 0.437614, id: 0.950743] time: 0:00:48.460895
[Epoch 0/1000] [Batch 17/995] [D loss: 0.394060, acc:  50%] [G loss: 14.187487, adv: 0.593953, recon: 0.557239, id: 1.091003] time: 0:00:48.660885
[Epoch 0/1000] [Batch 18/995] [D loss: 0.580103, acc:  37%] [G loss: 15.498345, adv: 0.539145, recon: 0.613089, id: 1.352540] time: 0:00:48.904131
[Epoch 0/1000] [Batch 19/995] [D loss: 0.344282, acc:  58%] [G loss: 14.102629, adv: 0.824655, recon: 0.501498, id: 1.420064] time: 0:00:49.098604
[Epoch 0/1000] [Batch 20/995] [D loss: 0.320820, acc:  59%] [G loss: 10.033802, adv: 0.625721, recon: 0.354285, id: 1.076168] time: 0:00:49.340202
[Epoch 0/1000] [Batch 21/995] [D loss: 0.243236, acc:  62%] [G loss: 11.173902, adv: 0.679538, recon: 0.397249, id: 1.175862] time: 0:00:49.550373
[Epoch 0/1000] [Batch 22/995] [D loss: 0.315080, acc:  59%] [G loss: 11.774001, adv: 0.823355, recon: 0.426114, id: 0.805463] time: 0:00:49.736513
[Epoch 0/1000] [Batch 23/995] [D loss: 0.513042, acc:  48%] [G loss: 11.297789, adv: 0.640303, recon: 0.377699, id: 1.583992] time: 0:00:49.972141
[Epoch 0/1000] [Batch 24/995] [D loss: 0.806443, acc:  42%] [G loss: 14.300797, adv: 0.705996, recon: 0.545430, id: 0.792255] time: 0:00:50.201624
[Epoch 0/1000] [Batch 25/995] [D loss: 0.326413, acc:  60%] [G loss: 12.000158, adv: 0.820294, recon: 0.414607, id: 1.462156] time: 0:00:50.392529
[Epoch 0/1000] [Batch 26/995] [D loss: 0.184464, acc:  72%] [G loss: 10.545224, adv: 0.873180, recon: 0.324569, id: 0.938100] time: 0:00:50.603470
[Epoch 0/1000] [Batch 27/995] [D loss: 0.348396, acc:  59%] [G loss: 10.107640, adv: 0.709160, recon: 0.347443, id: 0.872489] time: 0:00:50.792149
[Epoch 0/1000] [Batch 28/995] [D loss: 0.249199, acc:  64%] [G loss: 10.194003, adv: 0.919194, recon: 0.299164, id: 1.257875] time: 0:00:51.014169
[Epoch 0/1000] [Batch 29/995] [D loss: 0.214487, acc:  64%] [G loss: 12.399454, adv: 1.940286, recon: 0.299664, id: 1.179302] time: 0:00:51.201862
[Epoch 0/1000] [Batch 30/995] [D loss: 0.312716, acc:  58%] [G loss: 10.878626, adv: 0.854400, recon: 0.365153, id: 1.007199] time: 0:00:51.386603
[Epoch 0/1000] [Batch 31/995] [D loss: 0.427314, acc:  47%] [G loss: 13.469300, adv: 1.437365, recon: 0.423764, id: 1.057431] time: 0:00:51.583401
[Epoch 0/1000] [Batch 32/995] [D loss: 0.450146, acc:  49%] [G loss: 12.278427, adv: 0.630552, recon: 0.452820, id: 1.054881] time: 0:00:51.787395
[Epoch 0/1000] [Batch 33/995] [D loss: 0.365521, acc:  49%] [G loss: 11.847090, adv: 0.738640, recon: 0.414071, id: 1.247778] time: 0:00:51.941337
[Epoch 0/1000] [Batch 34/995] [D loss: 2.479558, acc:  35%] [G loss: 14.531601, adv: 1.666111, recon: 0.467845, id: 0.930663] time: 0:00:52.122073
[Epoch 0/1000] [Batch 35/995] [D loss: 0.719680, acc:  32%] [G loss: 11.144573, adv: 0.431137, recon: 0.427074, id: 0.976232] time: 0:00:52.341554
[Epoch 0/1000] [Batch 36/995] [D loss: 0.211312, acc:  66%] [G loss: 9.925415, adv: 1.015363, recon: 0.310914, id: 1.206188] time: 0:00:52.575696
```

Experiment purpose and description:

The Image-to-Image transformation used in that experiment is an experiment that aims to learn to visually demonstrate the mapping between an input image and an output image using a training set of aligned image pairs (A and B). However, if there are too many tasks, pared training data cannot be used. In the absence of paired examples, we present how to transform an image from source domain X to target domain Y. The experiment is to map G: X → Y so that the G(X) image distribution and the Y distribution cannot be distinguished using adversarial loss. This mapping is so restrictive that it uses cycle consistency loss to push F (G (X)) ≈X (or vice versa) to F: (Y → X).

The experiment uses CycleGAN and Unpaired Data. As mentioned above, the CycleGAN model has two mapping functions for G: X → Y and F: Y → X, and Dx for determining F(Y) and Dy for determining G(X). The generator continuously replaces the input images with images from the target domain to generate fake images, and the discriminator learns to discriminate the fake images created by the generator. By repeating the process, it is possible to identify the characteristics of the two objects to be converted and the characteristics to be maintained, and to convert the necessary parts. The more data you have, the more accurate it is.

Description of the code:

- The init() function determines the size of the input image of the model, calculates the size of the discriminator's output image, and sets the structure of CycleGAN.
- build_generator() creates the generator structure.
- build_discriminator() determines whether an image is real or fake.
- train() trains the generator and discriminator built with image data for as many epochs as data for batches.