Github Link：https://github.com/yqh022/Coding3/tree/main

Project code link：

https://github.com/yqh022/Coding3/blob/main/%E2%80%9Cfinal_2.ipynb

Video Link：https://youtu.be/djkX0nX78c4
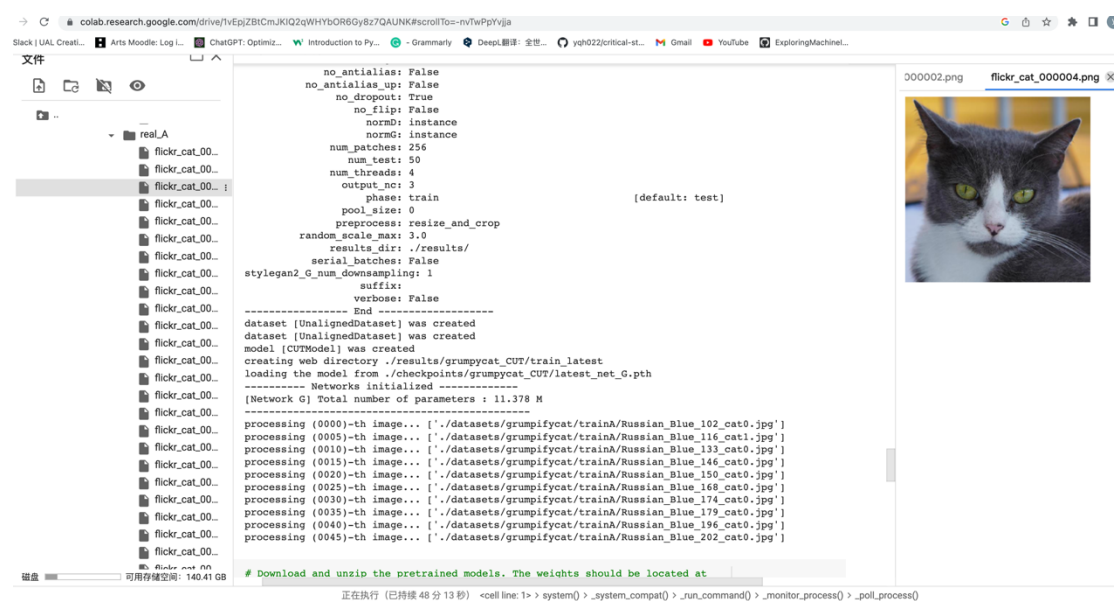
Algorithm Resources: https://github.com/taesungp/contrastive-unpaired-translation

I consulted the CUT GANS code, I start trying to change its database, dataset resources: https://github.com/clovaai/stargan-v2

but I found that after running the high-resolution images of the dataset, when I set running epochs to 60 result is like this:

```
                    no_antialias: False
                 no_antialias_up: False
                      no_dropout: True
                         no_flip: False
                           normD: instance
                           normG: instance
                     num_patches: 256
                        num_test: 50
                     num_threads: 4
                       output_nc: 3
                           phase: train                [default: test]
                       pool_size: 0
                      preprocess: resize_and_crop
                random_scale_max: 3.0
                     results_dir: ./results/
                   serial_batches: False
    stylegan2_G_num_downsampling: 1
                          suffix:
                         verbose: False
    ----------------- End -------------------
    dataset [UnalignedDataset] was created
    dataset [UnalignedDataset] was created
    model [CUTModel] was created
    creating web directory ./results/grumpycat_CUT/train_latest
    loading the model from ./checkpoints/grumpycat_CUT/latest_net_G.pth
    ---------- Networks initialized -------------
    [Network G] Total number of parameters : 11.378 M
    -----------------------------------------------
    processing (0000)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_102_cat0.jpg']
    processing (0005)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_116_cat1.jpg']
    processing (0010)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_133_cat0.jpg']
    processing (0015)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_146_cat0.jpg']
    processing (0020)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_150_cat0.jpg']
    processing (0025)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_168_cat0.jpg']
    processing (0030)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_174_cat0.jpg']
    processing (0035)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_179_cat0.jpg']
    processing (0040)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_196_cat0.jpg']
    processing (0045)-th image... ['./datasets/grumpifycat/trainA/Russian_Blue_202_cat0.jpg']
    # Download and unzip the pretrained models. The weights should be located at
```
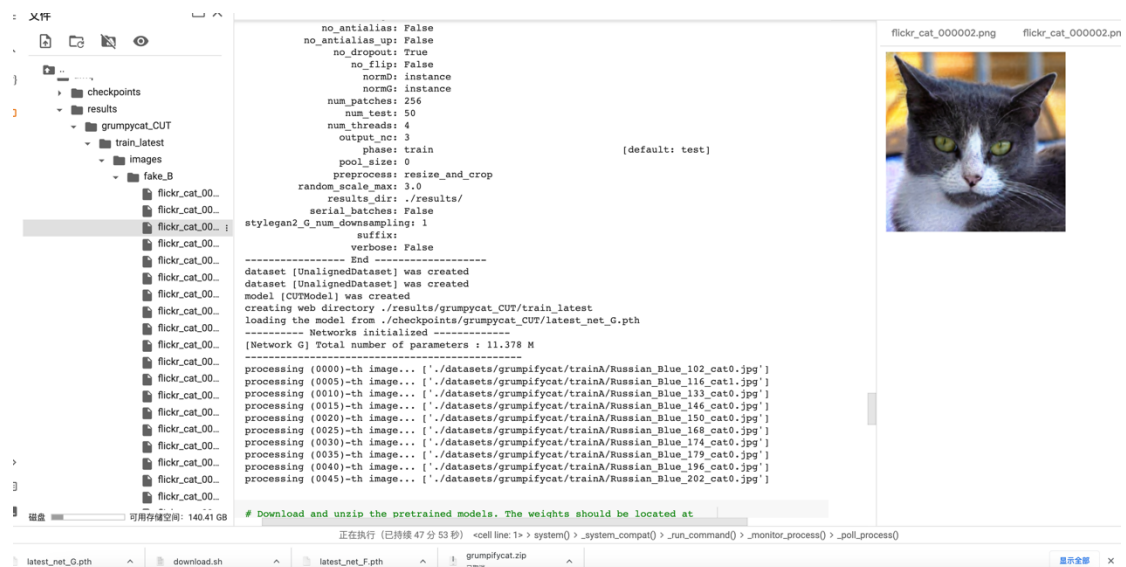
After I tried the animal images and found that it did not generate images with significant results. Because it consumes 8 hours to run 60 epochs, and colab has a time limit to run the GPU, I changed the dataset.

dataset resources: https://github.com/taesungp/contrastive−unpaired−translation/blob/master/datasets/download_cut_dataset.sh

"monet2photo"

```
[ ] %%bash
    cd datasets
    mkdir m2p_mini
    mkdir m2p_mini/trainA
    mkdir m2p_mini/trainB
    cd monet2photo/trainA
    echo $(pwd) #Create datasets folders
    find . -maxdepth 1 -type f -name "*" | head -100 | xargs -I{} cp {} ../../m2p_mini/trainA
    cd ../trainB
    echo $(pwd)
    find . -maxdepth 1 -type f -name "*" | head -100 | xargs -I{} cp {} ../../m2p_mini/trainB
    #Use the first 100 images in both folders of the dataset

    /content/CUT/datasets/monet2photo/trainA
    /content/CUT/datasets/monet2photo/trainB
```

To reduce training time, the first 100 files in the "trainA" and "trainB" directories of the "monet2photo" dataset were selected and copied into the "m2p_mini" directory to create a smaller dataset.

then I set steady_epoch=5, decline_epoch=1 and steady_epoch=20,

decline_epoch=2 to train the model and test it respectively. Then I imported the matplotlib library, defined five empty lists g_gan, d_real, d_fake, g and nce for visual plotting, converted the values to floating point numbers and added them to the corresponding lists. Use the plt.show function to display the plotted image. The images only show 5–11 comparison images from the training set, and all the image generation results are stored in photo2monet1. The image below is a comparison of the running accuracy of the two cases:
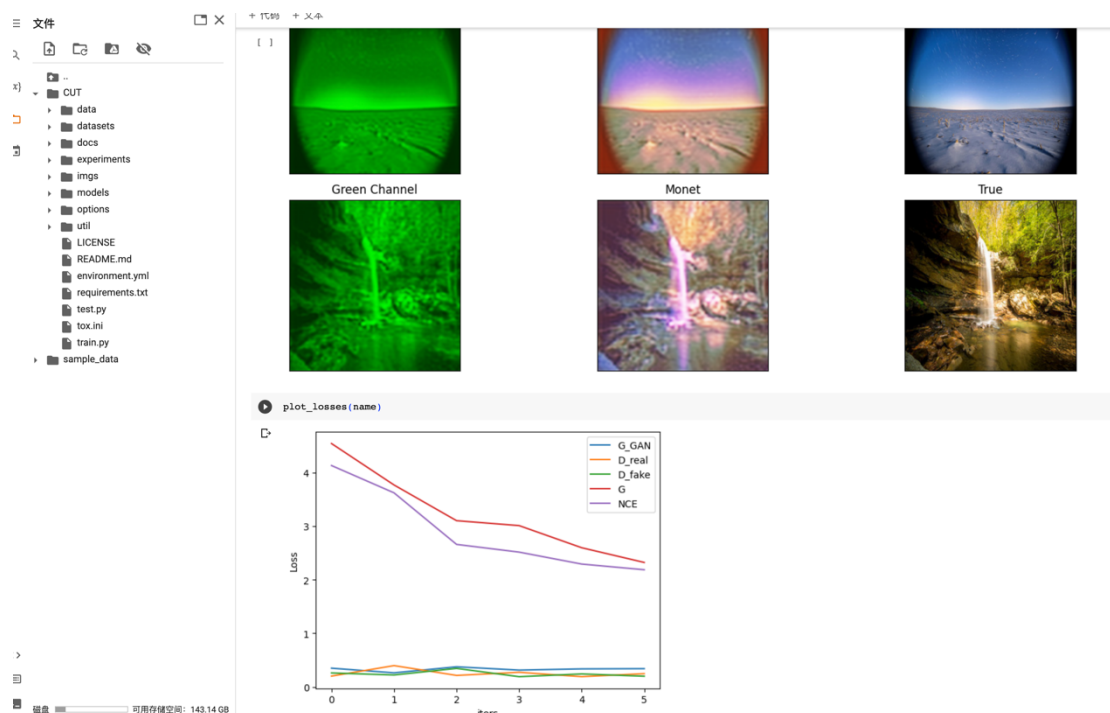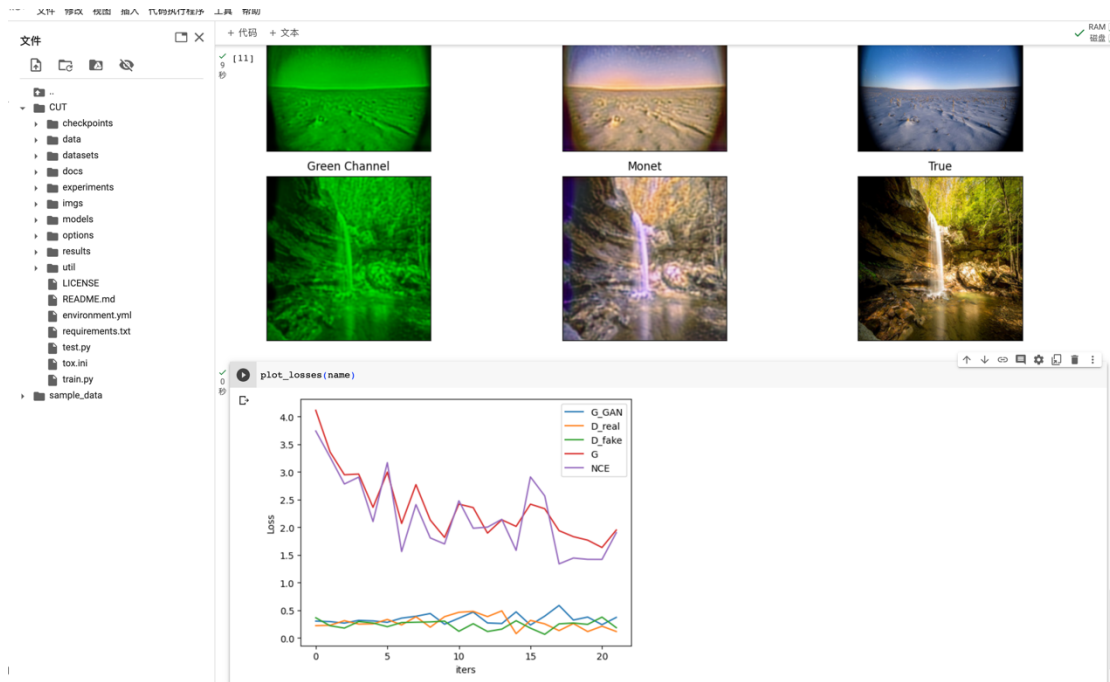
When extracting the colour channels I found that it only resulted in black and white images, then I asked chatgpt. by first converting the image to RGB mode, converting it to a NumPy array, creating a new array with the same size and number of channels as the original image, and assigning the green channel values to the green channels in it, converting the NumPy array back to a PIL image in order to generate the green channel image.

I wanted to add the CBAM attention module to the model, which can be used in various layers of the network to improve feature extraction. The original model training code, train.py, was changed to redefine the training function of the model. Reference code link for how the CBAM module can be used:

https://blog.csdn.net/weixin_41790863/article/details/123413303?spm=1 001.2101.3001.6650.2&utm_medium=distribute.pc_relevant.none–task–

```
opt = Namespace(**opt_dict)

%env name = photo2monet2
name = 'photo2monet2'

opt_dict['n_epochs'] = 20
opt_dict['n_epochs_decay'] = 2
opt_dict['name'] = name
opt = Namespace(**opt_dict)
model = create_model(opt)        # create a model given opt.model and other options


# 将CBAM加入模型
count=0
for m in model.netG.model:
    if m.__class__ == ResnetBlock:
        m.conv_block.add_module(f'cbam{count}',CBAM(m.conv_block[-2].out_channels).to('cuda'))
        count+=1

train(opt, model)
```

```
outchannels:torch.Size([1, 256, 64, 64])
torch.Size([1, 256, 1, 1])
outchannels:torch.Size([1, 256, 64, 64])
torch.Size([1, 256, 1, 1])
```

Set the same steady_epoch=20 and decline_epoch=2 .Find the ResnetBlock section in networks.py. Test the model after adding the CBAM training module in its penultimate position and save the results to photo2monet2. The following is a comparison image of the training results before and after adding the module：