AI6126: Advanced Computer Vision

Li Xiaochen (G2102142D)

Li0029en@e.ntu.edu.sg

Project 2: Blind Face Super-Resolution

Background

In this mini challenge, our goal is to generate high-quality (HQ) face images from the corrupted low Quality (LQ) ones. The mini-dataset for this task comes from the FFHQ, which contains of 4000 HQ training images, 400 LQ-HQ images pairs for validation and another 400 images for testing the model.

During the training, the corresponding LQ images with by generated using degradation pipeline (see Figure 1), which contains 4 types of degradations: Gaussian blur, Downsampling, Noise, and Compression. The code of degradation functions are provided in the baseline config in train_pipeline.

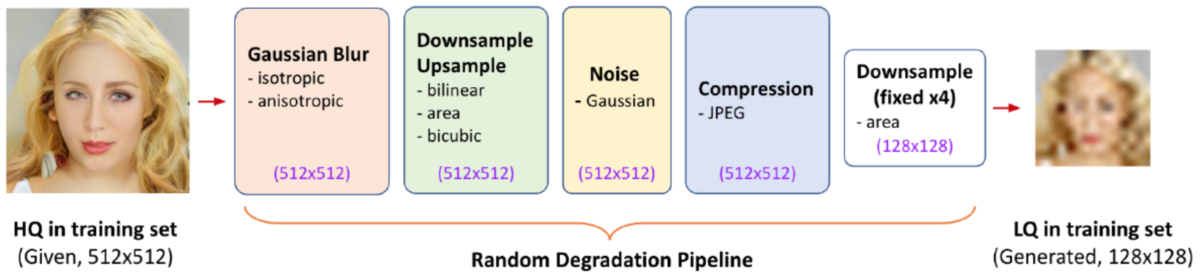


Figure 1. Illustration of degradation pipeline during training

During validation and testing, the trained model will generate the HQ image for each LQ face image. The quality of the output will be evaluated based on the PSNR metric, PSNR computes the peak signal-to-noise ratio between two images, it is a quality measurement between the original and a compressed image. The higher the PSNR, the better quality of the compressed or reconstructed image.

Model

During the experiment stage, three models have been tried to compute higher PSNR results. They are SRGAN (baseline), ESRGAN, and ESRGAN with discriminator loss.

SRGAN stands for Super Resolution Generative Adversarial Networks, which is one of the first techniques that allows the model to achieve an upscaling factor of almost 4x for most image visuals. The core of this GAN network is composed of a Generator Network and A discriminator Network (see Figure 2). In general, the Generator has been told how to train and pass the trained data to the Discriminator. The Discriminator compares the trained results with the ground truth to obtain the difference (Loss). Then the Discriminator tells the Generator the error condition and Generator will modify the training accordingly. Repeat the training until the generator delivers the results with the smallest difference compare with ground truth.

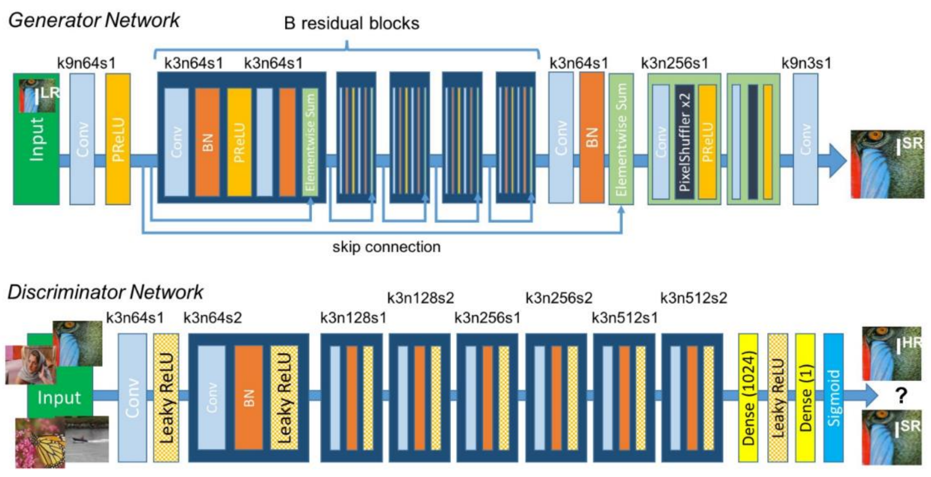
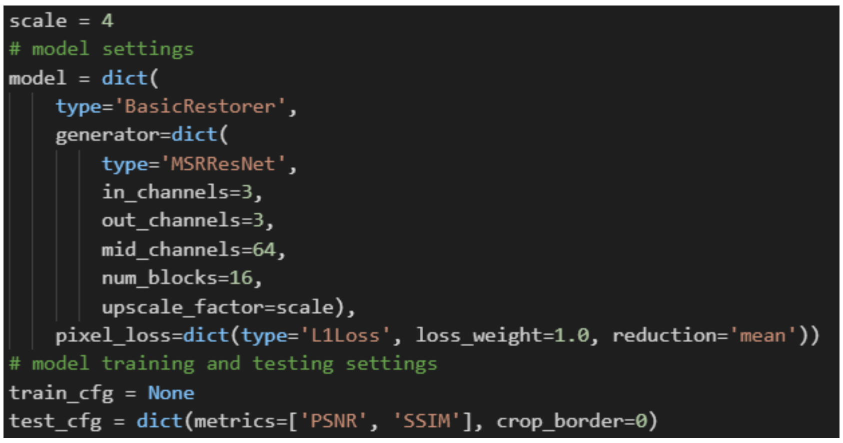


Figure 2. Architecture of Generator and Discriminator Network of SRGAN

The SRGAN model setting in this project is shown as below in the config file



```
scale = 4
# model settings
model = dict(
    type='BasicRestorer',
    generator=dict(
        type='MSRResNet',
        in_channels=3,
        out_channels=3,
        mid_channels=64,
        num_blocks=16,
        upscale_factor=scale),
    pixel_loss=dict(type='L1Loss', loss_weight=1.0, reduction='mean'))
# model training and testing settings
train_cfg = None
test_cfg = dict(metrics=['PSNR', 'SSIM'], crop_border=0)
```

SRGAN is a seminal work that is capable of generating realistic textures during single image super-resolution. ESRGAN follows the baseline ResNet-style architecture of SRGAN but replaces the residual block with the RRDB block. The RRDB block is inspired by the DenseNet architecture and connects all layers within the residual block directly with each other. The architecture of ESRGAN is show as below in figure 3

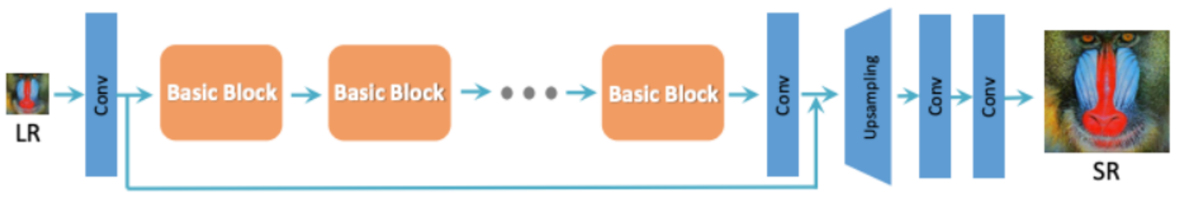
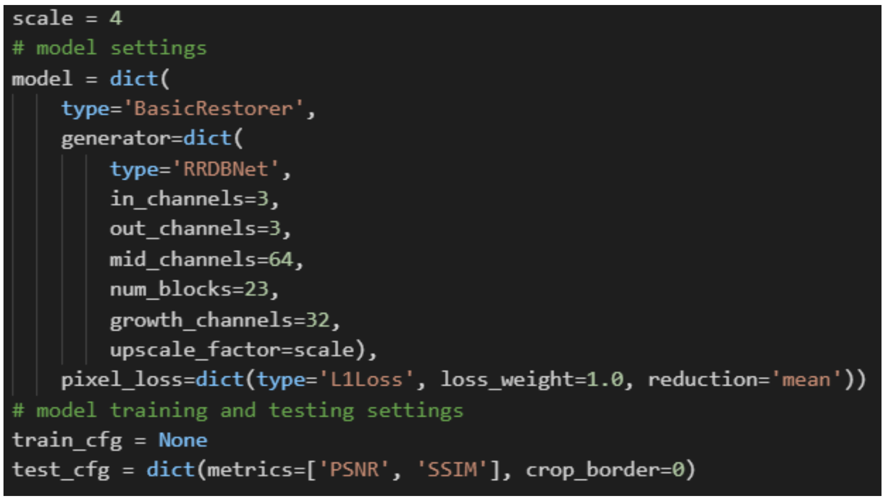


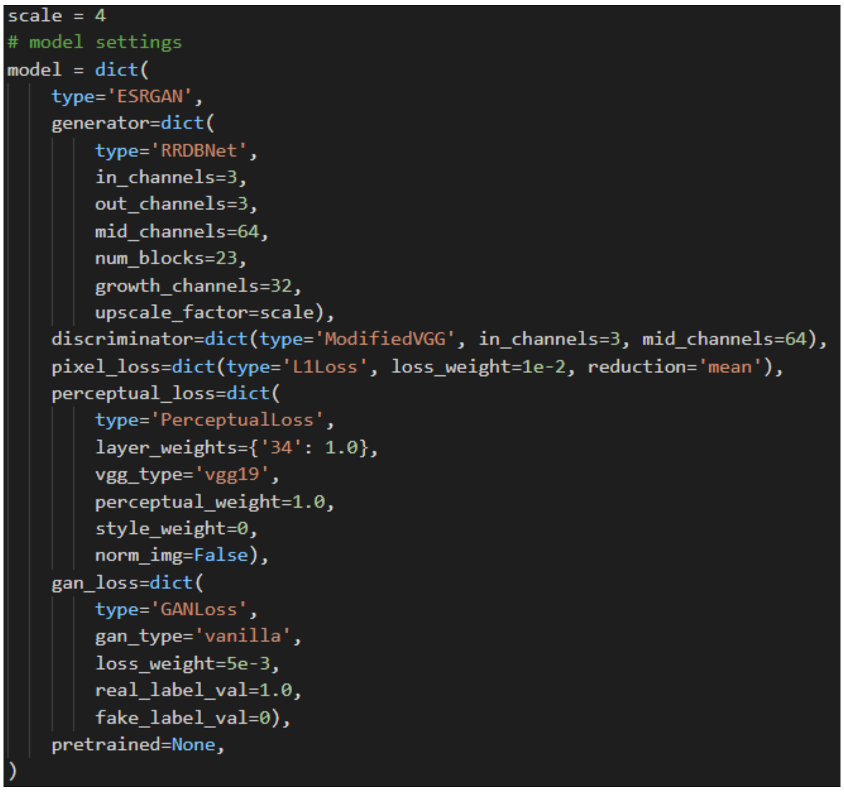
Figure 3. Architecture for ESRGAN

The ESRGAN model setting in this project is shown as below in the config file



```
scale = 4
# model settings
model = dict(
    type='BasicRestorer',
    generator=dict(
        type='RRDBNet',
        in_channels=3,
        out_channels=3,
        mid_channels=64,
        num_blocks=23,
        growth_channels=32,
        upscale_factor=scale),
    pixel_loss=dict(type='L1Loss', loss_weight=1.0, reduction='mean'))
# model training and testing settings
train_cfg = None
test_cfg = dict(metrics=['PSNR', 'SSIM'], crop_border=0)
```

To obtain more natural HQ face images, ESRGAN with discriminator was attempted in this project as well. The model setting in this project is shown as below in the config file.



```
scale = 4
# model settings
model = dict(
    type='ESRGAN',
    generator=dict(
        type='RRDBNet',
        in_channels=3,
        out_channels=3,
        mid_channels=64,
        num_blocks=23,
        growth_channels=32,
        upscale_factor=scale),
    discriminator=dict(type='ModifiedVGG', in_channels=3, mid_channels=64),
    pixel_loss=dict(type='L1Loss', loss_weight=1e-2, reduction='mean'),
    perceptual_loss=dict(
        type='PerceptualLoss',
        layer_weights={'34': 1.0},
        vgg_type='vgg19',
        perceptual_weight=1.0,
        style_weight=0,
        norm_img=False),
    gan_loss=dict(
        type='GANLoss',
        gan_type='vanilla',
        loss_weight=5e-3,
        real_label_val=1.0,
        fake_label_val=0),
    pretrained=None,
)
```

This method uses RRDB as the generator network and Modified VGG as the discriminator network. The Modified VGG model can classify the input image to two categories, actual image and restorer images. ESRGAN uses three loss functions, which will be discussed in the next session.

Loss Functions

For loss function, L1 loss is used for all three models as the penalty for the pixel difference between restorer images and ground truth.

$$L_1(\hat{y}, y) = \sum_{i=0}^{m} |y^{(i)} - \hat{y}^{(i)}|$$

Besides the L1 loss, ESRGAN with discriminator uses another two losses. One is perceptual loss which is calculated via the last conv layer (34th layer) of VGG19 network. VGG loss is a type of content loss introduced in the Perceptual losses, VGG loss attempts to be closer to perceptual similarity, the loss function is shown as below:

$$l_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y} \right)^2$$

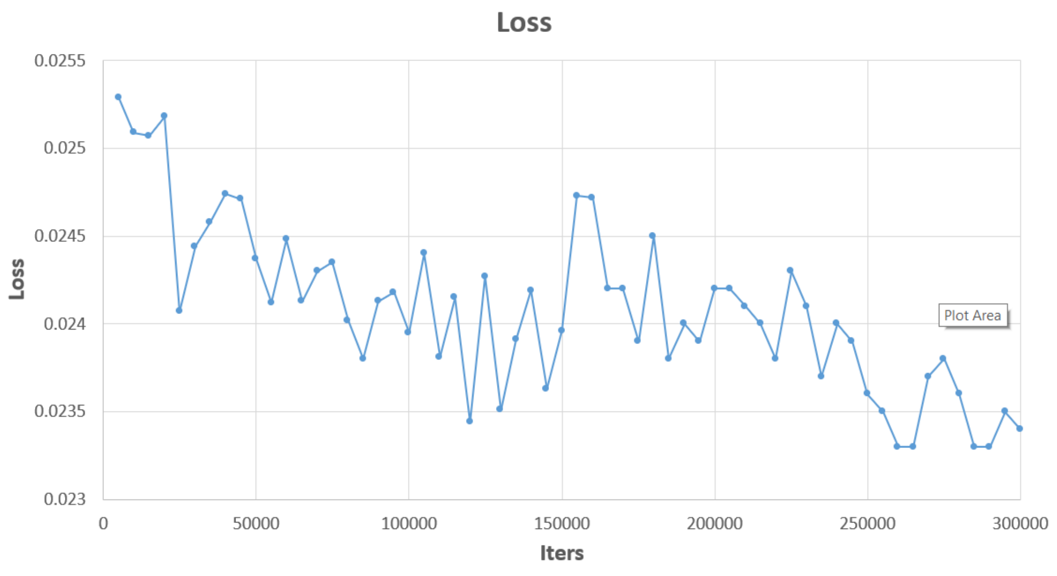
Another one is the Gan loss, when the discriminator is trained, it classifies both the real data and the fake data from the generator. It penalizes itself for misclassifying a real instance as fake, or a fake instance (created by the generator) as real, by maximizing the function below. [3]

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

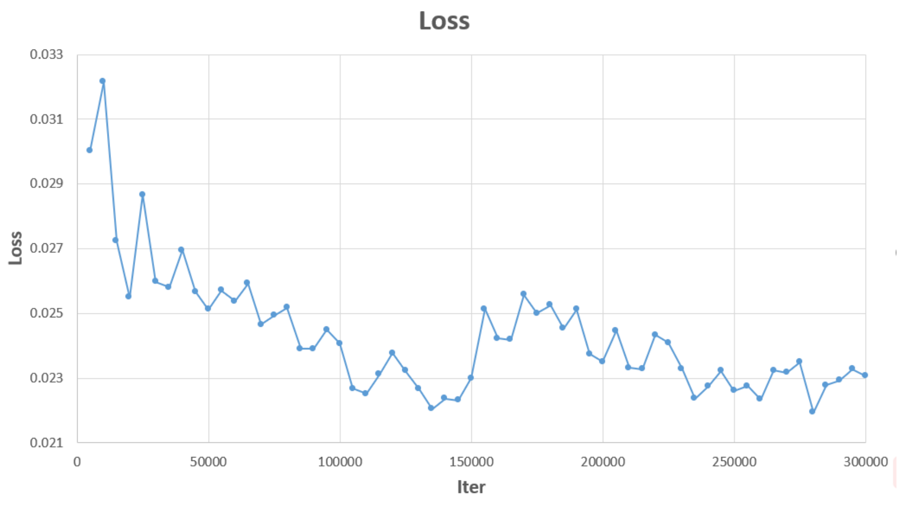
- log(D(x)) refers to the probability that the generator is rightly classifying the real image,
- maximizing log(1-D(G(z))) would help it to correctly label the fake image that comes from the generator.

Training Curves

The training curve of loss is shown as below for SRGAN

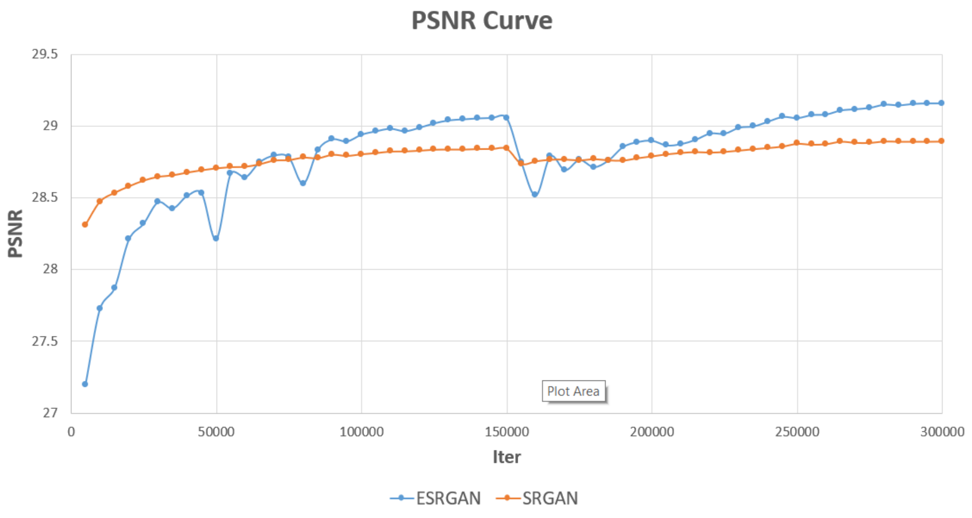


The training curve of loss is shown as below for the best model ESRGAN.



PSNR

The PSNR curve for each model on the validation set is shown as below. (Baseline only ran for 150000 iterations)



Model	Best PSNR on Val Set
SRGAN	28.8984
ESRGAN	29.1578

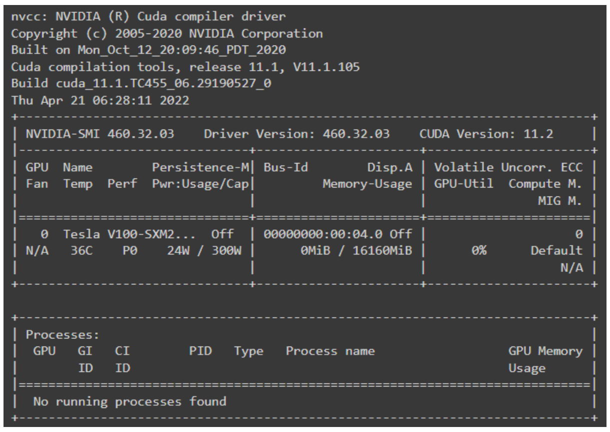
No. of Parameters

Due to the network complexity, the number of parameters for ESRGAN network is much larger than SRGAN

Model	No. of Parameters
SRGAN	1517571
ESRGAN	16697987
ESRGAN (discriminator)	51221772

Training Machine

This project was running on Google Colab with the Tesla V100 GPU.

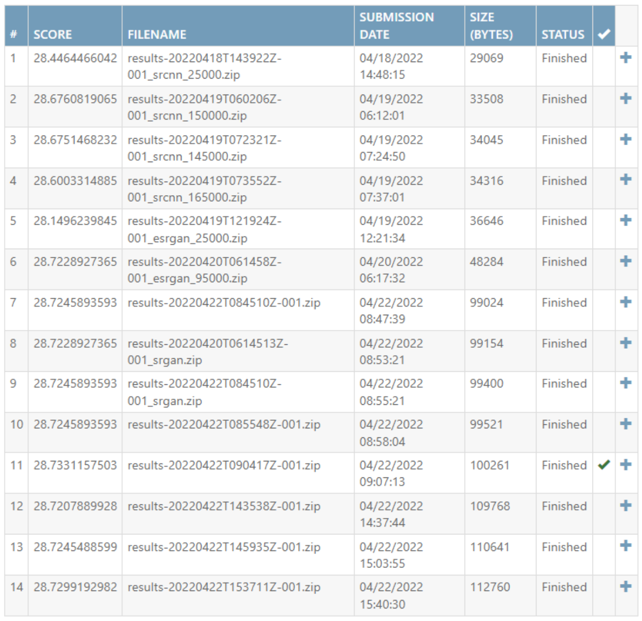


```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Mon_Oct_12_20:09:46_PDT_2020
Cuda compilation tools, release 11.1, V11.1.105
Build cuda_11.1.TC455_06.29190527_0
Thu Apr 21 06:28:11 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   36C    P0    24W / 300W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Test Results

Due to the fairness maintain, the model with number of parameters larger than 1,821,085 is not allowed, so the submitted test results on CodaLab is trained with the SRGAN model (see figure below). Please take note that the best result should be **#14** (PSNR ~28.7299) which run with SRGAN. Some of other results were running with ESRGAN.



#	SCORE	FILENAME	SUBMISSION DATE	SIZE (BYTES)	STATUS	✓	
1	28.4464466042	results-20220418T143922Z-001_srcnn_25000.zip	04/18/2022 14:48:15	29069	Finished		+
2	28.6760819065	results-20220419T060206Z-001_srcnn_150000.zip	04/19/2022 06:12:01	33508	Finished		+
3	28.6751468232	results-20220419T072321Z-001_srcnn_145000.zip	04/19/2022 07:24:50	34045	Finished		+
4	28.6003314885	results-20220419T073552Z-001_srcnn_165000.zip	04/19/2022 07:37:01	34316	Finished		+
5	28.1496239845	results-20220419T121924Z-001_esrgan_25000.zip	04/19/2022 12:21:34	36646	Finished		+
6	28.7228927365	results-20220420T061458Z-001_esrgan_95000.zip	04/20/2022 06:17:32	48284	Finished		+
7	28.7245893593	results-20220422T084510Z-001.zip	04/22/2022 08:47:39	99024	Finished		+
8	28.7228927365	results-20220420T0614513Z-001_srgan.zip	04/22/2022 08:53:21	99154	Finished		+
9	28.7245893593	results-20220422T084510Z-001_srgan.zip	04/22/2022 08:55:21	99400	Finished		+
10	28.7245893593	results-20220422T085548Z-001.zip	04/22/2022 08:58:04	99521	Finished		+
11	28.7331157503	results-20220422T090417Z-001.zip	04/22/2022 09:07:13	100261	Finished	✓	+
12	28.7207889928	results-20220422T143538Z-001.zip	04/22/2022 14:37:44	109768	Finished		+
13	28.7245488599	results-20220422T145935Z-001.zip	04/22/2022 15:03:55	110641	Finished		+
14	28.7299192982	results-20220422T153711Z-001.zip	04/22/2022 15:40:30	112760	Finished		+

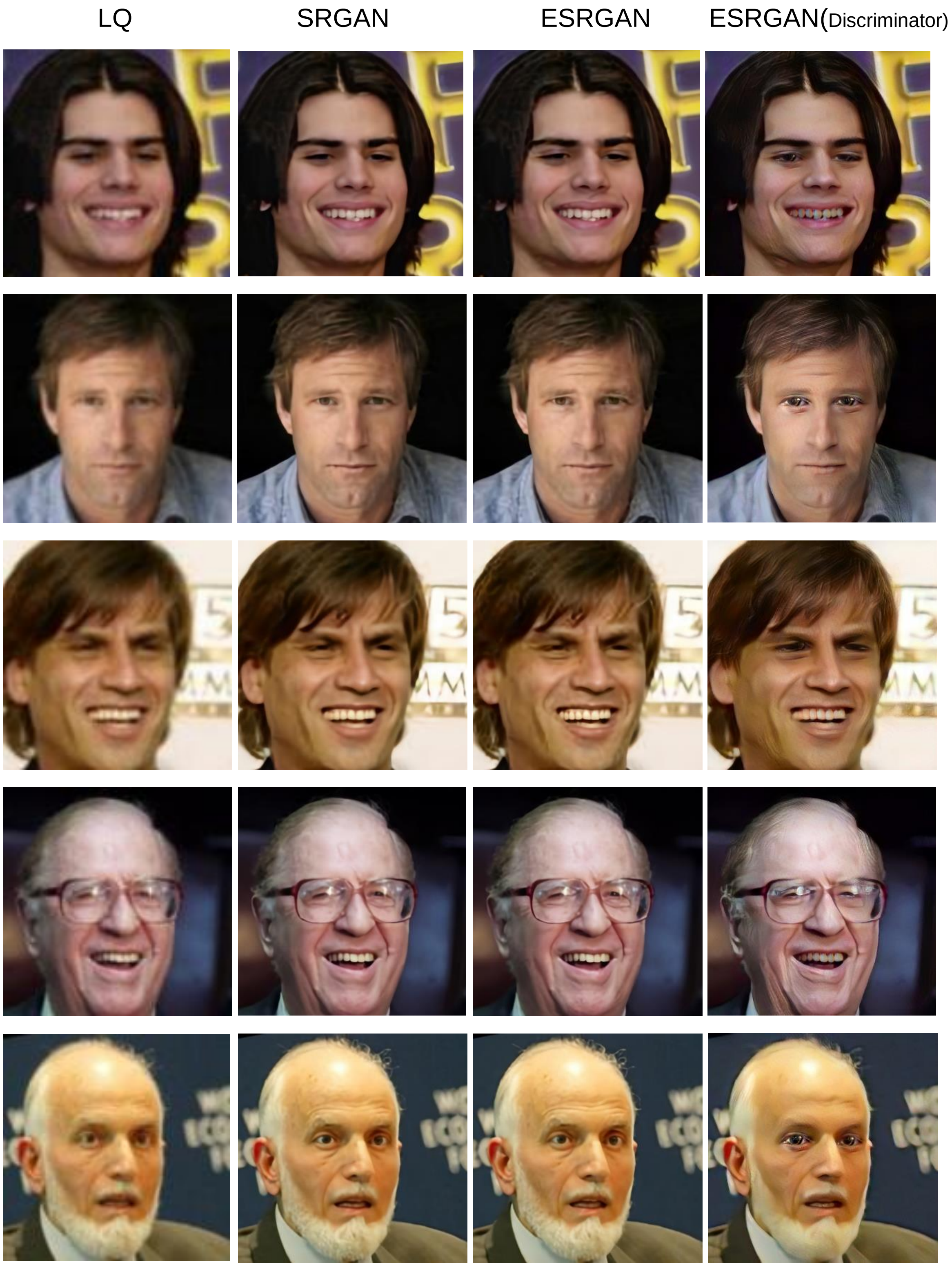
The test results for ESRGAN are not submitted into CodaLab, based on the validation results, the PSNR for ESRGAN suppose to be more than 28.9 dB.

Real-world HQ Images Results

The real-world HQ images results are shown as below for each model. ESRGAN provides us the best outcomes during the experiment.

Both SRGAN and ESRGAN seems more focus on the super-resolution process for eyeballs and tooth, which makes the results not natural enough. In order to solve this problem, ESRGAN with discriminator method has been tried. As it is a time-consuming process, so only 150000 iterations have been run. The results are shown below in the 4th column.

The result seems improved the resolution on the hair part. However, it worse off other parts, especially the eyeball and tooth. The reason might because the PairedRandomCrop function applied in the train pipeline. More studies and researches will be needed to understand this problem.



References:

[1] MMEditing: Super-Resolution Models https://mmediting.readthedocs.io/en/latest/_tmp/restorers_models.html#

[2] VGG Loss: https://paperswithcode.com/method/vgg-loss

[3] Understanding GAN Loss Functions https://neptune.ai/blog/gan-loss-functions#:~:text=In%20practice%2C%20it%20saturates%20for,Discriminator%20loss%20and%20Generator%20loss.