FPT Education
FPT UNIVERSITY

AIP391 Project

Enhance image resolution using a Generative Adversarial Network

Tran Nguyen Phuc Vinh; Dinh Hoang Lam; Tran Duy Ngoc Bao

Abstract

Based on the idea of improving the quality of images we launched in the proposal. In this report, our team will introduce and implement SRGAN, a Generative Adversarial Network for image super-resolution. Data used is collected from many reputational resources. They had been trained with two categories of low resolution and high resolution for 4x scaling factors. The algorithm is an application of deep convolutional neural networks (CNN - [1]). Our model includes a Generator Network with residual blocks, skip-connections, mean squared error (MSE), and a Discriminator Network for distinguishing between fake and real images. A perceptual loss function consists of adversarial loss and content loss. MSE for peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) are used to measure and compare the result of the SR algorithm [2].

Keywords: SRGAN, 4x scaling factors, CNN, Generator Network, Discriminator Network, perceptual loss function, PSNR, SSIM.

1. Introduction

We can preserve priceless moments in life by preserving images. Creating compelling, high-definition photos is a challenge with rapidly changing cultures and the constant advancement of photographic technology. Using algorithms to improve image quality is the most effective way to solve this problem. Thanks to technology, we can now enlarge old, blurry photos without losing any features and this is why our team chose this research topic.
With our approach, we enhance image quality by using the SRGAN model. The single image super-resolution (SISR) method, which tries to estimate a high-resolution picture (HR) counterpart from a low-resolution image (LR), will be the subject of our attention.
In the picture below are the input and output images of the image before and after being processed for image quality by SRGAN, we can see that the image is noticeably enhanced in sharpness.

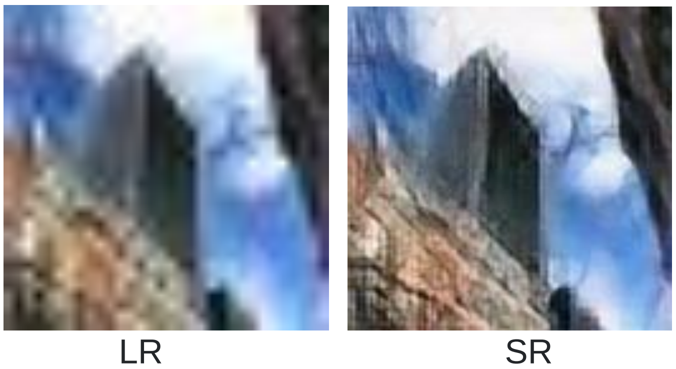


Figure 1: from LR image to SR image

2. Related work

2.1 Image super-resolution:

Create a higher resolution (sharper and larger) image for a LR image. Another name for this issue is the Image Super Resolution (SR) issue.
This is a machine learning regression problem.

Historically, the information from several low-resolution photos was typically combined to create an HR image (Multiple Image Super Resolution) (MISR). High-tech cameras or image-editing software like Photoshop have both been used to perform these tactics.

The drawback of this approach is the requirement for numerous, aligned, low-resolution photographs, and the requirement that the subject of each image remains still throughout the photography process. Additionally, the processing speed is likewise not very fast.

If only one low-resolution image is used, precise alignment can be avoided. On the other hand, because so much new information must be created based on the extremely limited amount of input image information, it is challenging to ensure the quality of the output image. Then, they called this issue Single Image SR (SISR).

There are many methods proposed for the problem to improve the LR to HR image such as Kim et al. [3] introduced a very effective architecture with their deeply-recursive convolutional network (DRCN), which allows for long-range pixel dependencies while minimising the number of model parameters. The handcrafted bicubic filter in the SR pipeline is essentially replaced by the CNN model, which is also one of the most widely used models in SR at Wenzhe Shi [4]. In addition, the CNN model reduces the computing complexity of the entire SR operation. More specifically and closest to our paper is the project that uses the Generative Adversarial Networks (GAN) model and uses this platform to upgrade our model.

2.2 The contribution:

A powerful family of neural networks called general adversarial networks (GANs) is used in unsupervised learning. Ian J. Goodfellow created and published it in 2014 [5]. GANs are essentially a system of two competing neural network models that can assess, capture, and duplicate the changes within a dataset.
Based on the theoretical foundation and upgraded from the GAN model,

Super Resolution GAN(SRGAN) has two components, the Generator and Discriminator, which are similar to GAN designs. The Generator generates data based on a probability distribution, and the Discriminator tries to predict whether the data came from the Generator or the Input Dataset. The generator then tries to refine the output so as to deceive the discriminator.

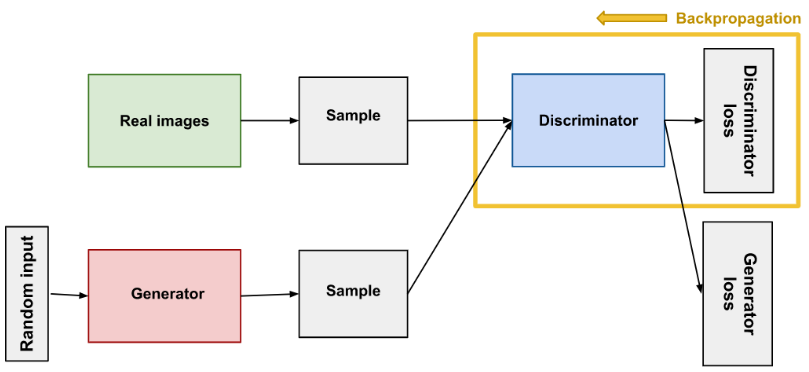


Figure 2: The generator and discriminator architectural details

3. Data preparation

3.1 Training dataset

For the training set, our team uses DIV2K [11] which is a popular single-image super-resolution dataset containing 1000 2K resolution high-quality images and splitted into 800 images for training, 100 images for validation, and 100 images for testing. It was collected for NTIRE2017 and NTIRE2018 Super-Resolution Challenges [11] with a large diversity of contents and sizes.

Figure 3: Examples of DIV2K training set

3.2 Test dataset

Our team uses 3 test sets for testing the performance of SRGAN model:

3.2.1. Set14 [10] dataset - 14 images

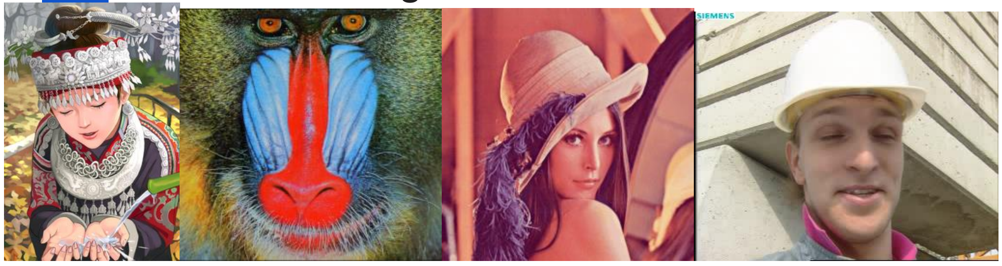


Figure 4: Examples of Set14 [10] Dataset

3.2.2. Set5 [7]

5 images ("baby", "bird", "woman", "butterfly", and "head"):

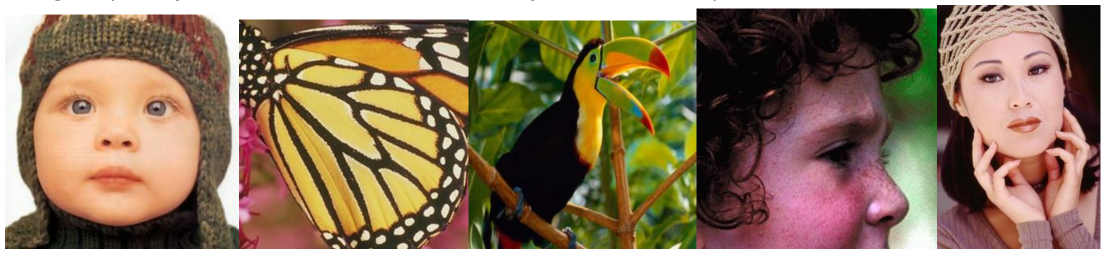


Figure 5: Set5 [7] Dataset

3.2.3. Urban100 [10] - 100 urban images

Figure 6: Urban100 [10] Dataset

4. Methods

First of all, we take data from our resources. The data includes many high-resolution images of different sizes, so to minimise the calculation, we cropped them into a set of same-size images (96x96). From the high-resolution images, we downscaled them by 4x factors into low-resolution images (24x24). The division into two types in the training set (I) aims to generate a generative model (G) that can estimate the super-resolution (SR) images from LR corresponding to its HR one. Based on the original paper [8] we need to solve underneath formula to find the coefficient for model:

$$\widehat{\theta_G} = argmin_{\theta_G} \frac{1}{N} \sum_{1}^{N} \quad l^{SR}(G_{\theta_G}(I_n^{LR}, I_n^{HR}))$$

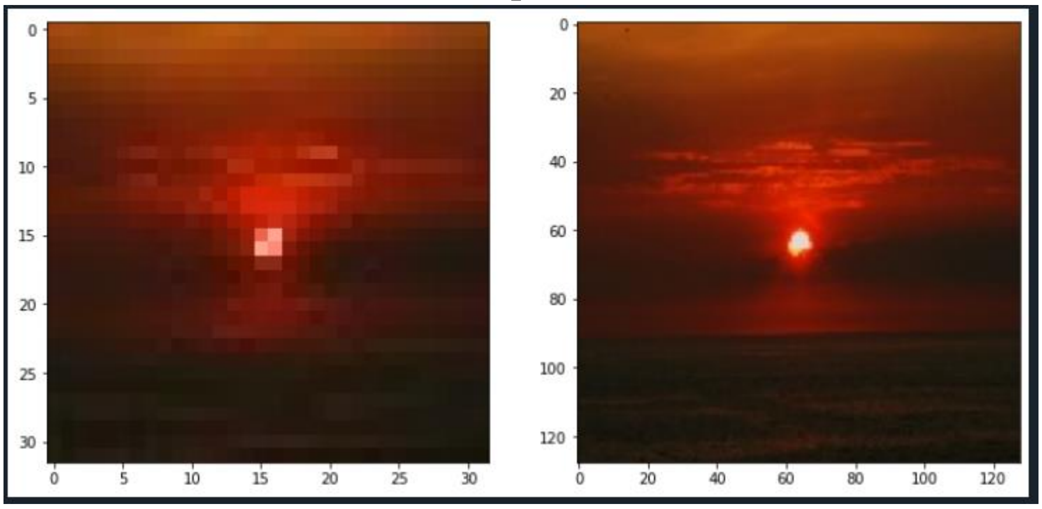


Figure 7: Example of crop data of LR (left) & HR (right)

4.1. Adversarial network architecture:

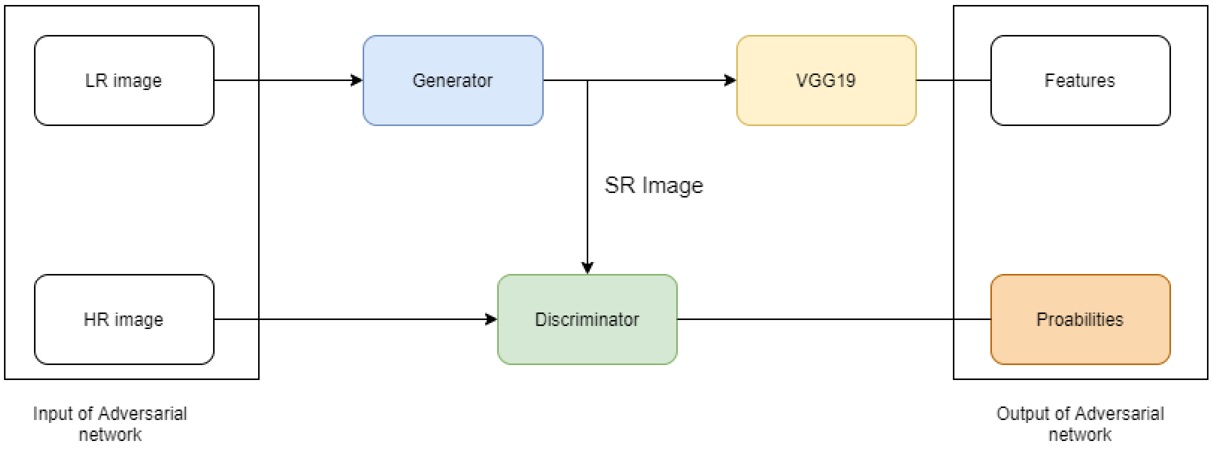
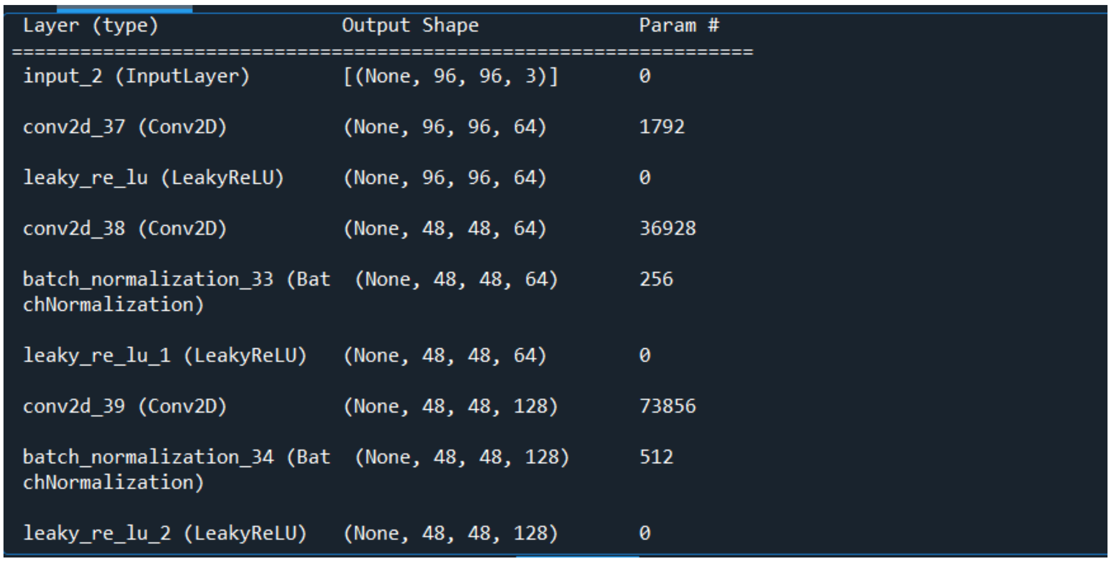


Figure 8: Adversarial network architecture

4.1.1. Discriminator Network:



```
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 96, 96, 3)]       0

 conv2d_37 (Conv2D)          (None, 96, 96, 64)        1792

 leaky_re_lu (LeakyReLU)     (None, 96, 96, 64)        0

 conv2d_38 (Conv2D)          (None, 48, 48, 64)        36928

 batch_normalization_33 (Bat  (None, 48, 48, 64)       256
 chNormalization)

 leaky_re_lu_1 (LeakyReLU)   (None, 48, 48, 64)        0

 conv2d_39 (Conv2D)          (None, 48, 48, 128)       73856

 batch_normalization_34 (Bat  (None, 48, 48, 128)      512
 chNormalization)

 leaky_re_lu_2 (LeakyReLU)   (None, 48, 48, 128)       0
```

Figure 9: Example of layers in Discriminator Network

In figure 5, it is clear that the Discriminator Network includes three layers: Conv2D, Batch Normalisation, and Leaky ReLU activation. About Conv2D, the convolutional net replaces the pooling functions with strides and allows the network to learn its own space. The second is Batch Normalisation, which helps deal with some problems that occurred in poor initialization and reduces the slope for the gradient. Leaky ReLU activation ($\alpha = 0.2$) allows the models to learn faster when handled with colour space (recommended to use in discriminator [12]).

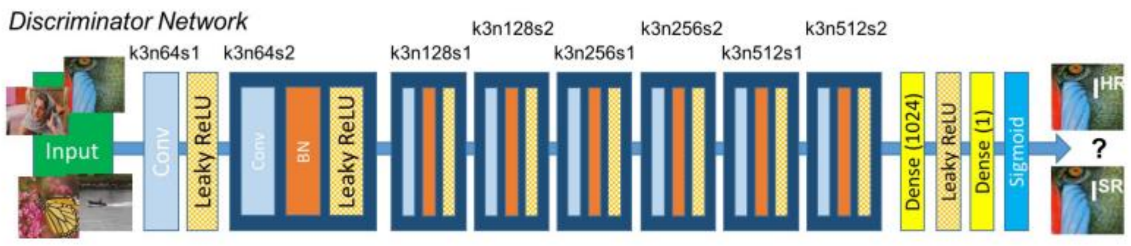


Figure 10: Architecture of Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

The network contains 8 blocks of 3x3 filter kernel, increased from 64 to 512 kernels followed by the VGG network. At the end of block 8, we have a layer with 512 feature maps and 2 strides of convolutional layers. Next, a layer of Dense(1024) to flatten the result and a Leaky ReLU layer to create a slope. Finally, a dense function with 1 with a sigmoid function that classifies real or fake images. Conclusion, the Discriminator Network distinguishes the origin image from its generated SR image.

4.1.2. Generator Network:

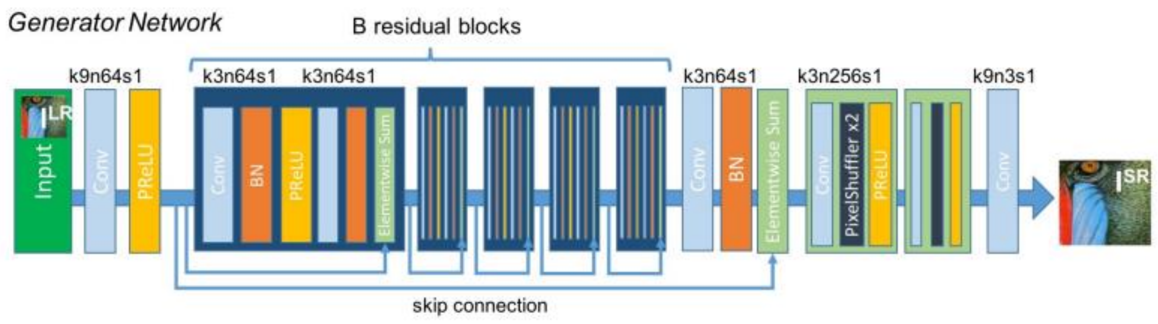
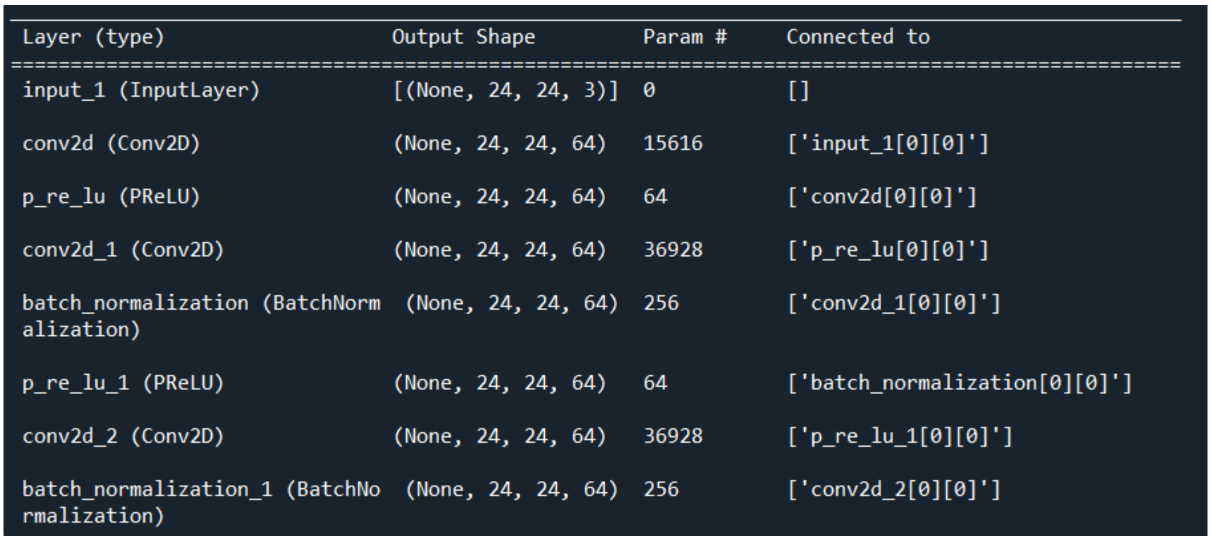


Figure 11: The architecture of the Generator Network

The architect shows that we have several B residual blocks that repeat a couple of times. For every block of ResNet, we start with a Conv2D layer, Batch Normalisation, and ParametricReLU (PReLU not LeakyReLU) to handle the exception using the Tanh function. At the end of each block, we add the output for every input feed to the network.



```
 Layer (type)                    Output Shape          Param #     Connected to
==================================================================================================
 input_1 (InputLayer)            [(None, 24, 24, 3)]   0           []

 conv2d (Conv2D)                 (None, 24, 24, 64)    15616       ['input_1[0][0]']

 p_re_lu (PReLU)                 (None, 24, 24, 64)    64          ['conv2d[0][0]']

 conv2d_1 (Conv2D)               (None, 24, 24, 64)    36928       ['p_re_lu[0][0]']

 batch_normalization (BatchNorm  (None, 24, 24, 64)    256         ['conv2d_1[0][0]']
 alization)

 p_re_lu_1 (PReLU)               (None, 24, 24, 64)    64          ['batch_normalization[0][0]']

 conv2d_2 (Conv2D)               (None, 24, 24, 64)    36928       ['p_re_lu_1[0][0]']

 batch_normalization_1 (BatchNo  (None, 24, 24, 64)    256         ['conv2d_2[0][0]']
 rmalization)
```

Figure 12: Example of layers in Generator Network

Then, we have some upscaling parts, an upscale block consists of a Conv2D, an UpSampling with the size of 2, and a PRELU activation. Finally, we combine B residual blocks with upscale blocks and get the result of an RGB image. The ResNet optimised for MSE allows us to train a generative model that can increase the similarity between HR image and its SR image. That decreases the percentage of discriminator when distinguishing a super-resolved image from its real image.

4.2. Perceptual loss function:

We define the definition of the perceptual loss function to compare the difference between HR and SR images. We can use many humans to evaluate images, but a practical way would be to use pre-trained network features that have been trained on millions on images. This loss function is the combination of two things: content loss and adversarial loss:

$$l^{SR} = l_X^{SR}\,(content\ loss) + 10^{-3} l_{Gen}^{SR}(adversarial\ loss)$$

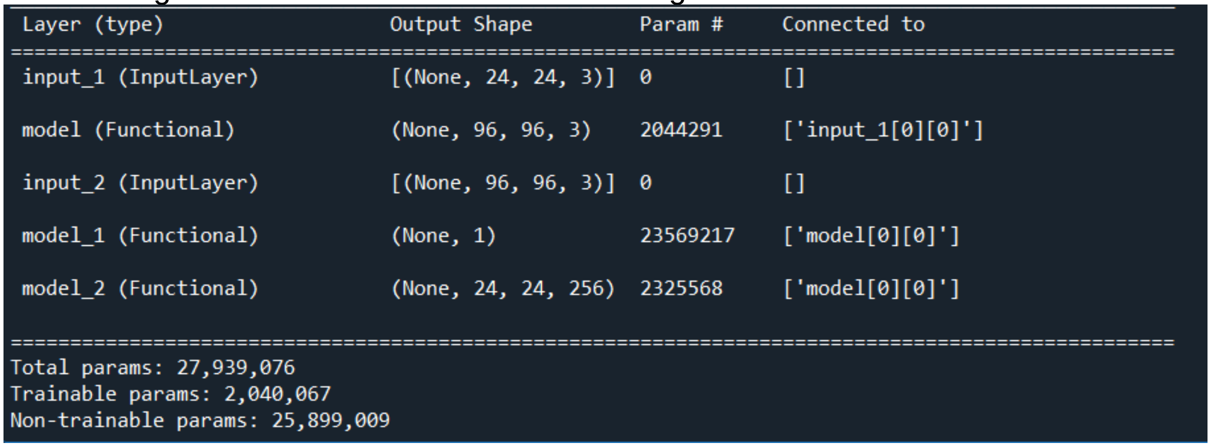
From the formula, for every one part of content loss which corresponds to 1/1000 of adversarial loss.

4.2.1 Adversarial loss:

Adversarial loss function is defined by the probabilities of the discriminator over all training samples:

$$l_{Gen}^{SR} = \sum_{n=1}^{N} \quad - logD_{\theta_D}(G_{\theta_G}(I^{LR}))$$

That is an equation of binary cross entropy which can be either 0 or 1, determining how close or how far from real images.



```
 Layer (type)                   Output Shape          Param #     Connected to
==========================================================================================
 input_1 (InputLayer)           [(None, 24, 24, 3)]  0           []

 model (Functional)             (None, 96, 96, 3)    2044291     ['input_1[0][0]']

 input_2 (InputLayer)           [(None, 96, 96, 3)]  0           []

 model_1 (Functional)           (None, 1)            23569217    ['model[0][0]']

 model_2 (Functional)           (None, 24, 24, 256)  2325568     ['model[0][0]']

==========================================================================================
Total params: 27,939,076
Trainable params: 2,040,067
Non-trainable params: 25,899,009
```

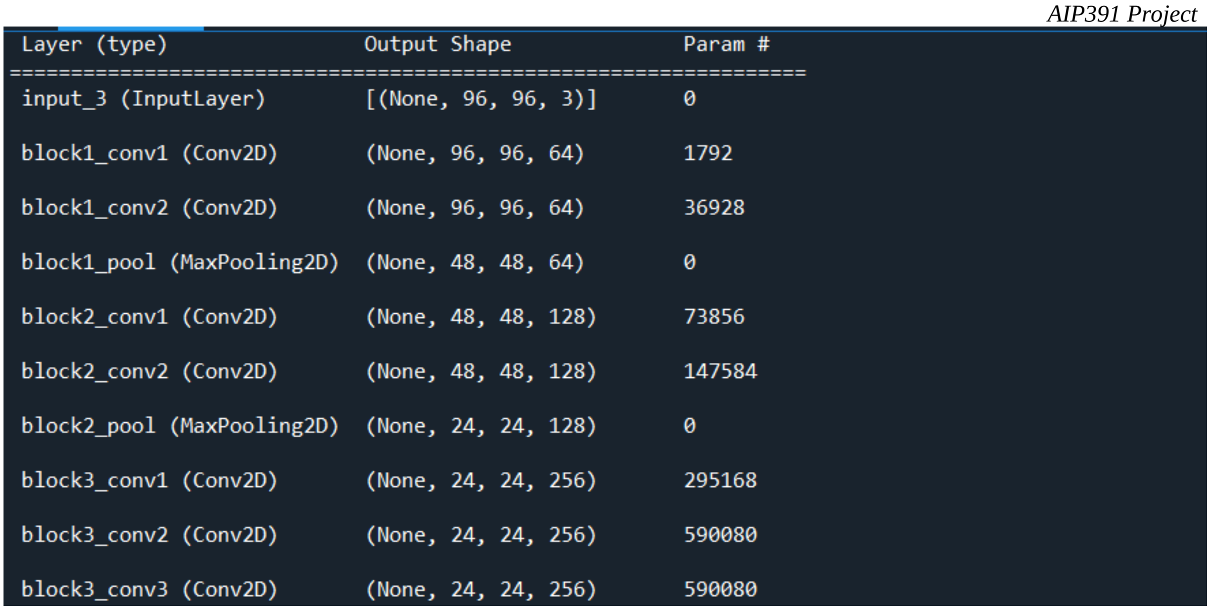
Figure 13: Example of training model

4.2.2. Content loss (using VGG19):

Firstly, we focus on MSE loss:

$$l_{MSE}^{SR} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH} \quad (I_{x,y}^{HR} - G_{\theta_G}(I^{LR})_{x,y})^2$$

Before, MSE was used for optimization of SR image algorithms by calculating the pixel losses. However, its results did not satisfy people about the smooth and plane. Therefore, in order to get close to the perceptual of humans, in the original paper [8], they use the content loss for optimising.



```
Layer (type)                Output Shape              Param #
=================================================================
input_3 (InputLayer)        [(None, 96, 96, 3)]       0

block1_conv1 (Conv2D)       (None, 96, 96, 64)        1792

block1_conv2 (Conv2D)       (None, 96, 96, 64)        36928

block1_pool (MaxPooling2D)  (None, 48, 48, 64)        0

block2_conv1 (Conv2D)       (None, 48, 48, 128)       73856

block2_conv2 (Conv2D)       (None, 48, 48, 128)       147584

block2_pool (MaxPooling2D)  (None, 24, 24, 128)       0

block3_conv1 (Conv2D)       (None, 24, 24, 256)       295168

block3_conv2 (Conv2D)       (None, 24, 24, 256)       590080

block3_conv3 (Conv2D)       (None, 24, 24, 256)       590080
```

Figure 14: VGG Network example

The content loss function is defined by using VGG19 based on the ReLU activation. To describe the VGG19 network, we said it is the alternative of a feature map. We can extract the feature map from anywhere in VGG19 with i-th convolution layers but NOT including the i-th maxpooling (1st convolution layer follows 1st maxpooling, 2nd convolution layer follows 1st maxpooling,etc...). Like the MSE, the VGG loss is formulated as:

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

$W_{i,j}$ and $H_{i,j}$describe the dimensions of the respective feature maps within the VGG network

5. Results and Discussion

To evaluate the performance of the image super-resolution model, our team measures the similarity between SR image and HR image based on **Peak signal-to-noise ratio** (**PSNR**) and **structural similarity index measure** (**SSIM**).

5.1 PSNR

PSNR[dB] is calculated via Mean Squared Error (MSE):

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [HR(i,j) - SR(i,j)]^2$$

Where HR and SR are high-resolution image and super-resolution image having the same size $m\ x\ n$.
And PSNR formula:

$$PSNR = 20log_{10}(\frac{MAX_{HR}^2}{\sqrt{MSE}})$$

Where MAX_{HR} is the maximum pixel value of an HR image, an image with RGB has a maximum pixel value is 255. The larger PSNR, the better image quality.

5.2 SSIM

SSIM [6] compares the similarity between two images. The SSIM [6] formula is based on three comparison components: luminance (l), structure(s), and contrast(c):

$$SSIM(x, y) \ = l(x, y)^{\alpha} * c(x, y)^{\beta} * s(x, y)^{\gamma}$$

Where x, y are two input images, and $\alpha, \beta, and\ \gamma$ are used to adjust the importance of components. SSIM [6] has range [-1, 1]. When SSIM index closes to 1 means that two input images are similar.
Because SSIM [6] only calculates on one colour component, to apply SSIM to the RGB colour space, we will compute SSIM of each colour component, then average the result [9]:

$$SSIM(x, y) \ = [SSIM(x, y)_R + SSIM(x, y)_G + SSIM(x, y)_B] \ / 3$$

Our SRGAN model was trained with 77 epochs and it took approximately 3 hours on Google Colaboratory with support of GPU. After training 77 epochs, we compare the PSNR and SSIM at some epochs which are considered at Figure 15.

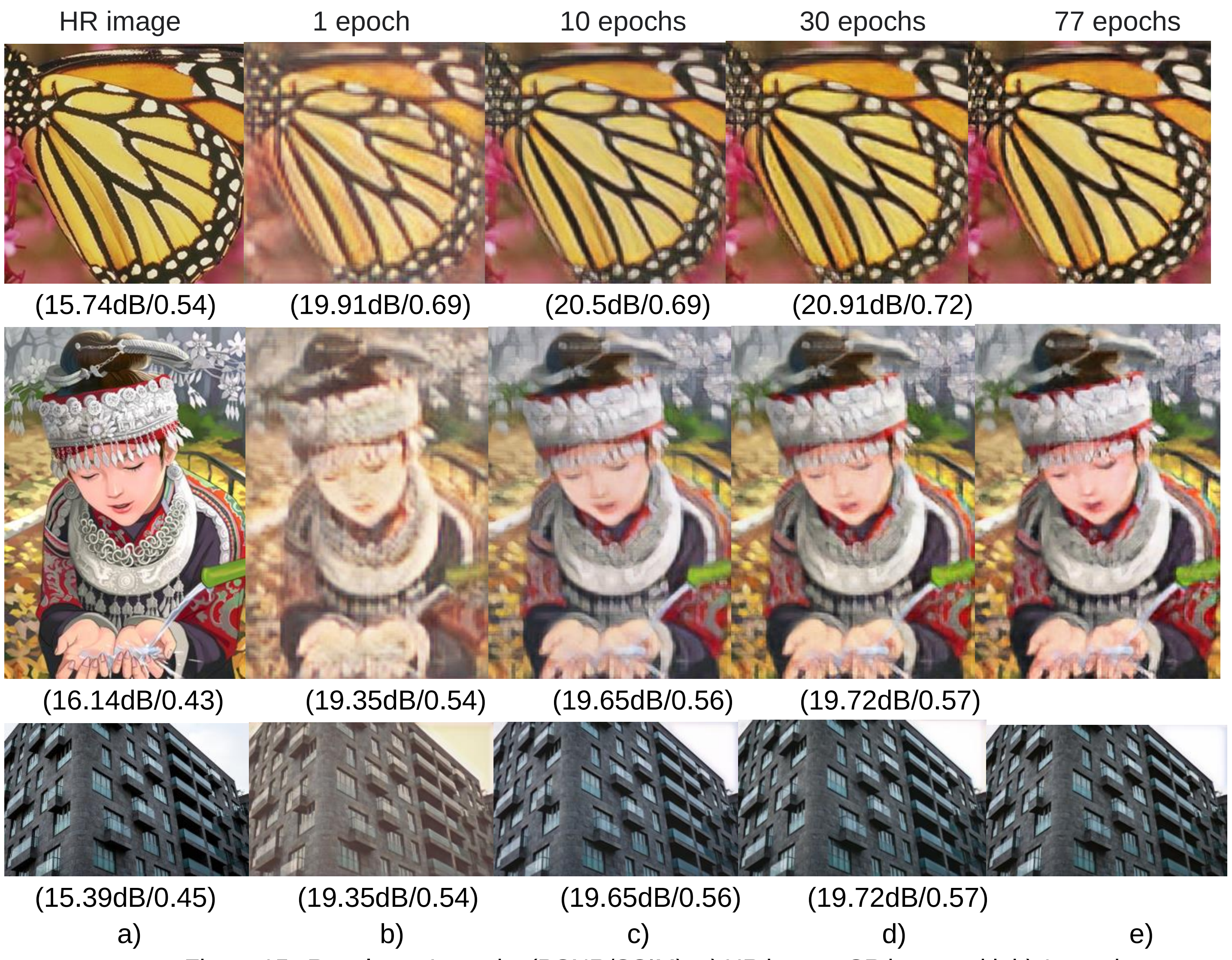
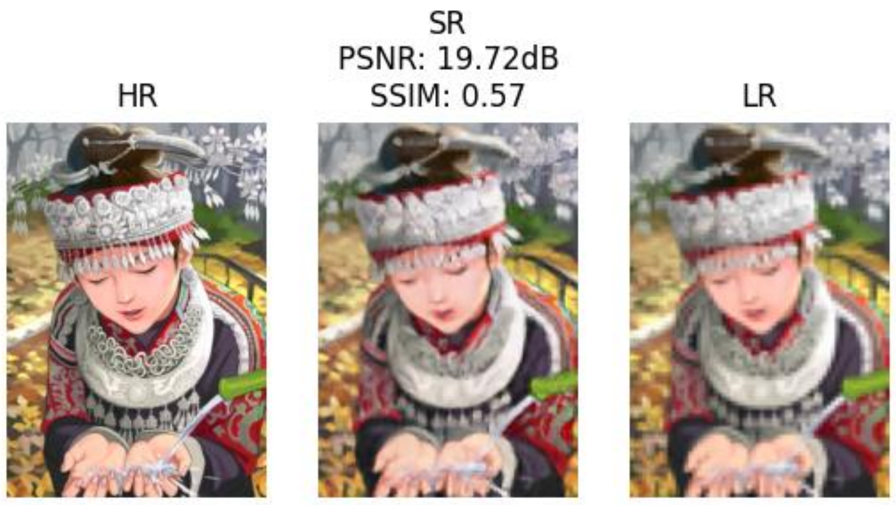
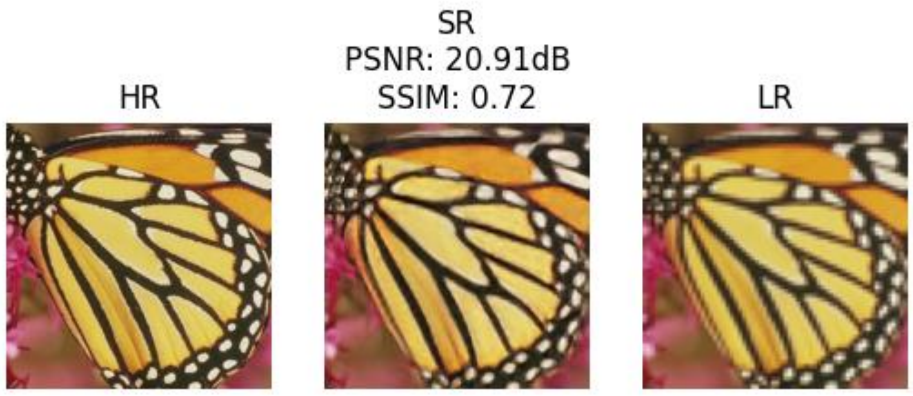


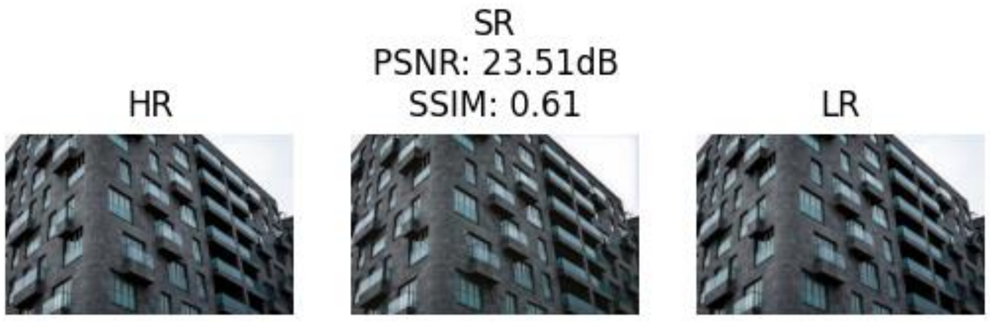
Figure 15 : Results at 4 epochs (PSNR/SSIM). a) HR image, SR image with b) 1 epoch, c) 10 epoch, d) 30 epoch, e) 77 epochs (PSNR/SSIM are shown in brackets)



a)



b)



c)

Figure 16: Comparison of SR image with HR image and LR image at epoch 77 with PSNR/SSIM index, a) Set14, b) Set5, c) Urban100

6. Conclusion and Perspectives

In our report, we have implemented the SRGAN model for image-super resolution and measure the image quality after enhancing the resolution from the LR image with the results of our training epochs. By researching and experiencing, we gain a lot about not only using AI to analyse images but also achieving knowledge for starting and doing an AI project.

References

[1] Krizhevsky, By Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. “ImageNet Classification with Deep Convolutional Neural Networks.” *Communications of the ACM* 60(6): 84–90.

[2] Yang, Chih-Yuan, Chao Ma, and Ming-Hsuan Yang. “Single-Image Super-Resolution: A Benchmark.” https://eng.ucmerced.edu/people/cyang35 (July 16, 2022).

[3] Kim, Jiwon, Jung Kwon Lee, and Kyoung Mu Lee. 2016. “Deeply-Recursive Convolutional Network for Image Super-Resolution.” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-December:1637–45. doi: 10.1109/CVPR.2016.181.

[4] Shi, Wenzhe, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. “Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network.” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-December:1874–83. doi: 10.1109/CVPR.2016.207.

[5] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. “Generative Adversarial Networks.” *Communications of the ACM* 63(11):139–44. doi: 10.1145/3422622.

[6] Wang, Zhou, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. 2004. “Image Quality Assessment: From Error Visibility to Structural Similarity.” *IEEE Transactions on Image Processing* 13(4):600–612. doi: 10.1109/TIP.2003.819861.

[7] Bevilacqua, Marco, Aline Roumy, Christine Guillemot, and Marie Line Alberi Morel. 2012. “Low-Complexity Single-Image Super-Resolution Based on Nonnegative Neighbour Embedding.” *BMVC 2012 - Electronic Proceedings of the British Machine Vision Conference 2012* (September). doi: 10.5244/C.26.135.

[8] Ledig, Christian, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-January:105–14. doi: 10.1109/CVPR.2017.19.

[9] Åström, Freddie, Michael Felsberg, and Reiner Lenz. 2011. "Color Persistent Anisotropic Diffusion of Images." Pp. 262–72 in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 6688 LNCS.

[10] Huang, Jia Bin, Abhishek Singh, and Narendra Ahuja. 2015. "Single Image Super-Resolution from Transformed Self-Exemplars." *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 07-12-June-2015:5197–5206. Doi: 10.1109/CVPR.2015.7299156.

[11] Agustsson, Eirikur, and Radu Timofte. 2017. "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study." *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 2017-July:1122–31. doi: 10.1109/CVPRW.2017.150.

[12] Radford, Alec, Luke Metz, and Soumith Chintala. 2016. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*: 1–16.

Source code Github: SRGAN model

Teamwork Plan Management: LINK

Contribution rate

Member	Research (%)	Attention (%)	Ideas (%)	Implement (%)	Attitude (good/bad)
Tran Nguyen Phuc Vinh	100%	80%	90%	70%	Good
Dinh Hoang Lam	100%	100%	100%	85%	Good
Tran Duy Ngoc Bao	100%	100%	100%	80%	Good
Comment	Overall, our teamwork is good. Instead of some arguments that happened while doing the project and poor equipment, however, we overcame them and completed our jobs clearly. End.				