2021

智慧化企業整合

SR-NET與ESRGAN生成圖像之比較

第一組
110034532 白晢睿
110034535 黃于瑞
110034561 洪聖博
110034564 楊笠笙

Outline

01 背景介紹

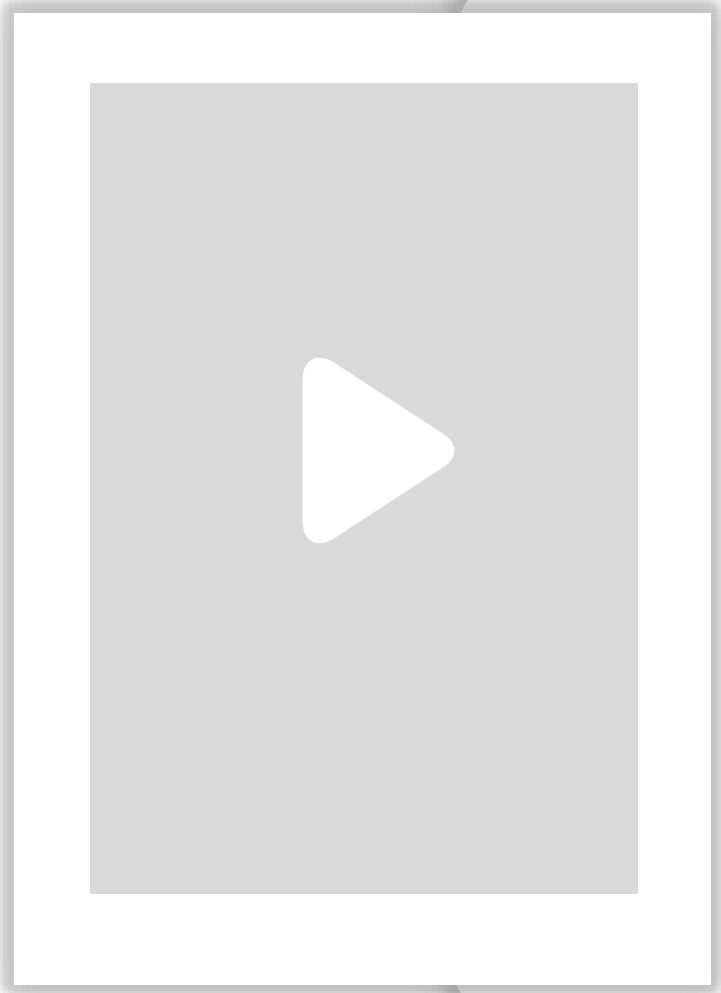
02 方法介紹

03 個案研究

04 結論

Part 1

背景介紹



背景說明

5W1H

WHY

一些較舊的照片或是受限於手機硬體鏡頭的因素，導致照片無法呈現更清晰的細節。

WHAT

透過機器學習，不論是CNN 或是 ESRGAN 對照片進行去模糊化。

WHERE

課程歷史教材需要採用具有年代的照片或是手機相片優化程式。

WHEN

拍出照片的當下又或是需要觀察照片細節的時候。

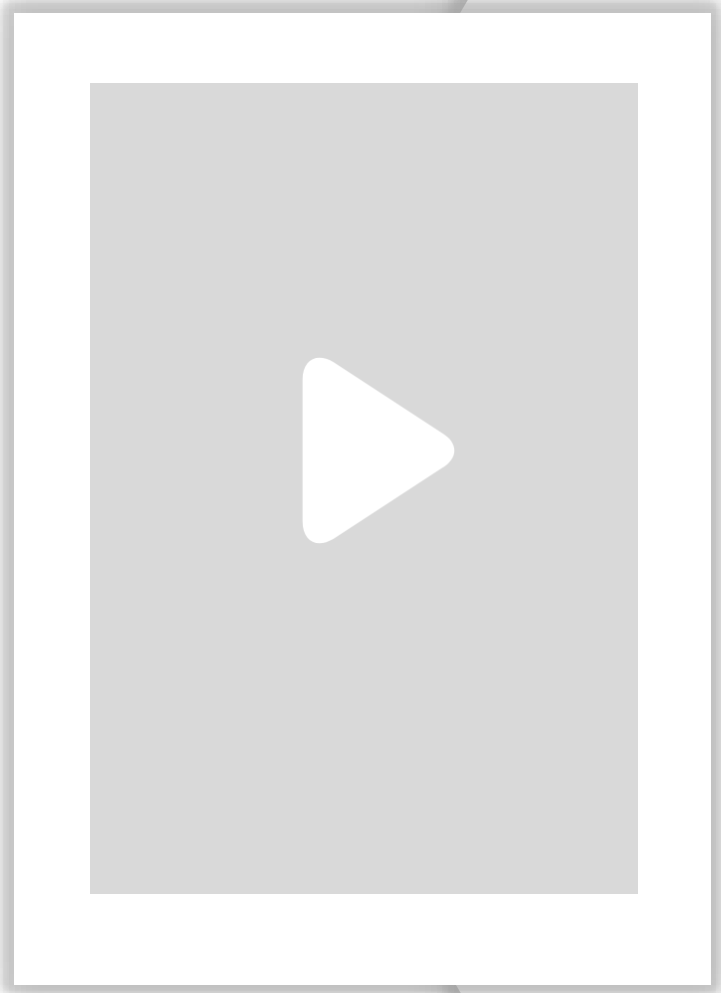
WHO

一般手機用戶又或是歷史學者。

HOW

利用機器學習 CNN 或是 ESRGAN 進行相片去模糊清晰化。

Part 2
方法介紹



```python
def get_D(input_shape):
    w_init = tf.random_normal_initializer(stddev=0.02)
    gamma_init = tf.random_normal_initializer(1., 0.02)
    df_dim = 64
    lrelu = lambda x: tl.act.lrelu(x, 0.2)

    nin = Input(input_shape)
    n = Conv2d(df_dim, (4, 4), (2, 2), act=lrelu, padding='SAME', W_init=w_init)(nin)

    n = Conv2d(df_dim * 2, (4, 4), (2, 2), padding='SAME', W_init=w_init, b_init=None)(n)
    n = BatchNorm2d(act=lrelu, gamma_init=gamma_init)(n)
    n = Conv2d(df_dim * 4, (4, 4), (2, 2), padding='SAME', W_init=w_init, b_init=None)(n)
    n = BatchNorm2d(act=lrelu, gamma_init=gamma_init)(n)
    n = Conv2d(df_dim * 8, (4, 4), (2, 2), padding='SAME', W_init=w_init, b_init=None)(n)
    n = BatchNorm2d(act=lrelu, gamma_init=gamma_init)(n)
    n = Conv2d(df_dim * 16, (4, 4), (2, 2), padding='SAME', W_init=w_init, b_init=None)(n)
    n = BatchNorm2d(act=lrelu, gamma_init=gamma_init)(n)
    n = Conv2d(df_dim * 32, (4, 4), (2, 2), padding='SAME', W_init=w_init, b_init=None)(n)
    n = BatchNorm2d(act=lrelu, gamma_init=gamma_init)(n)
    n = Conv2d(df_dim * 16, (1, 1), (1, 1), padding='SAME', W_init=w_init, b_init=None)(n)
    n = BatchNorm2d(act=lrelu, gamma_init=gamma_init)(n)
    n = Conv2d(df_dim * 8, (1, 1), (1, 1), padding='SAME', W_init=w_init, b_init=None)(n)
    nn = BatchNorm2d(gamma_init=gamma_init)(n)

    n = Conv2d(df_dim * 2, (1, 1), (1, 1), padding='SAME', W_init=w_init, b_init=None)(nn)
    n = BatchNorm2d(act=lrelu, gamma_init=gamma_init)(n)
    n = Conv2d(df_dim * 2, (3, 3), (1, 1), padding='SAME', W_init=w_init, b_init=None)(n)
    n = BatchNorm2d(act=lrelu, gamma_init=gamma_init)(n)
    n = Conv2d(df_dim * 8, (3, 3), (1, 1), padding='SAME', W_init=w_init, b_init=None)(n)
    n = BatchNorm2d(gamma_init=gamma_init)(n)
    n = Elementwise(combine_fn=tf.add, act=lrelu)([n, nn])

    n = Flatten()(n)
    no = Dense(n_units=1, W_init=w_init)(n)
    D = Model(inputs=nin, outputs=no, name="discriminator")
    return D
```

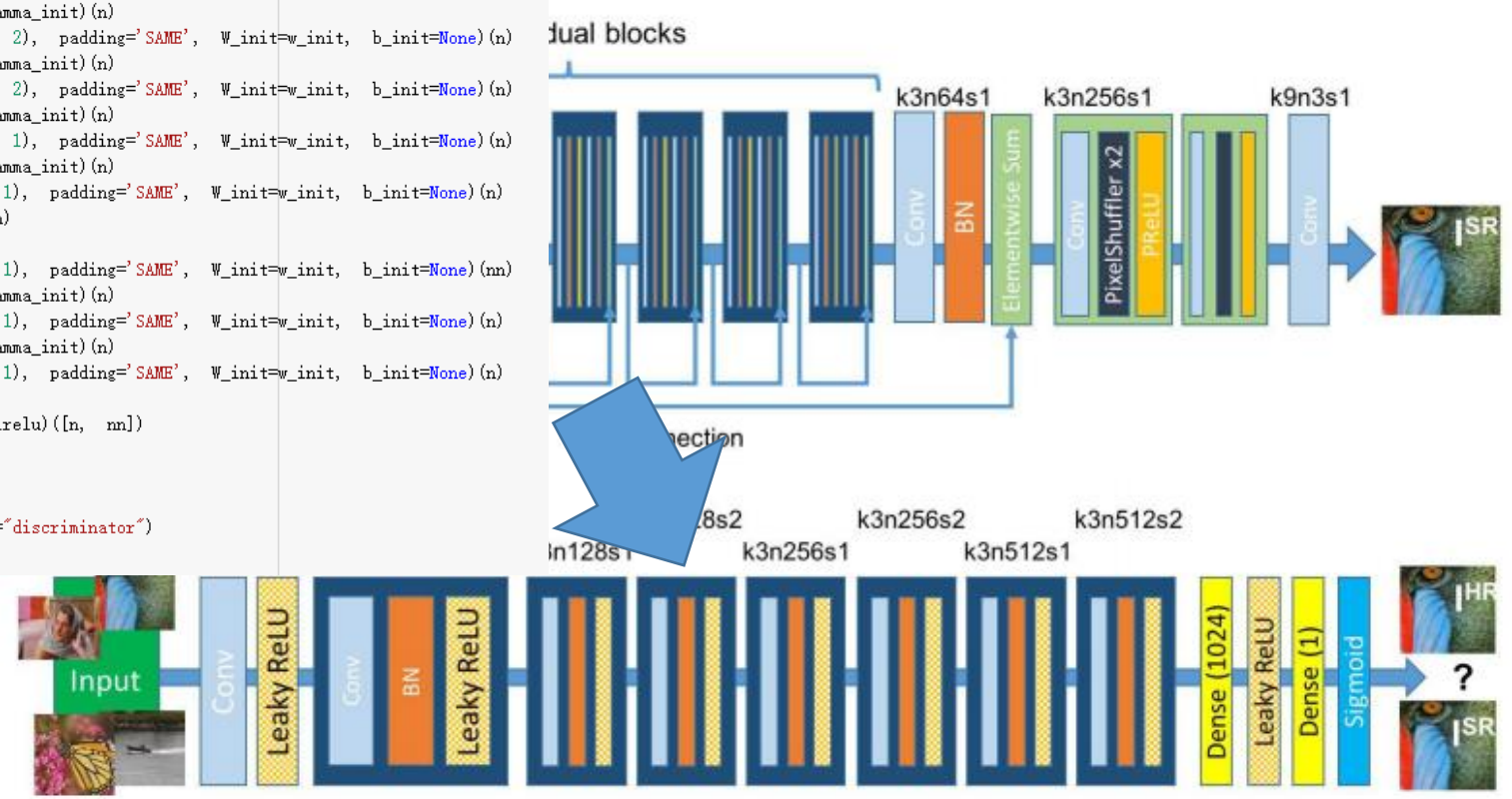
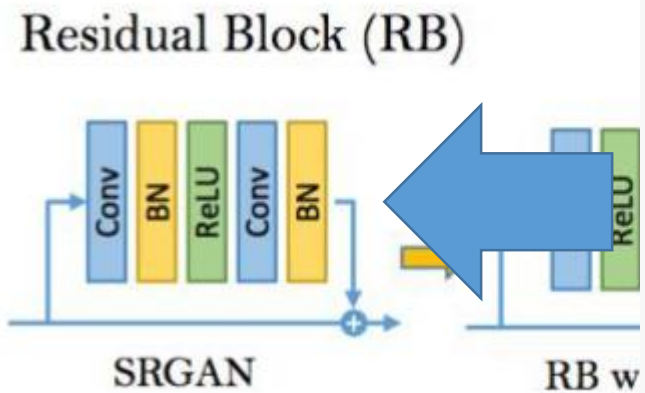


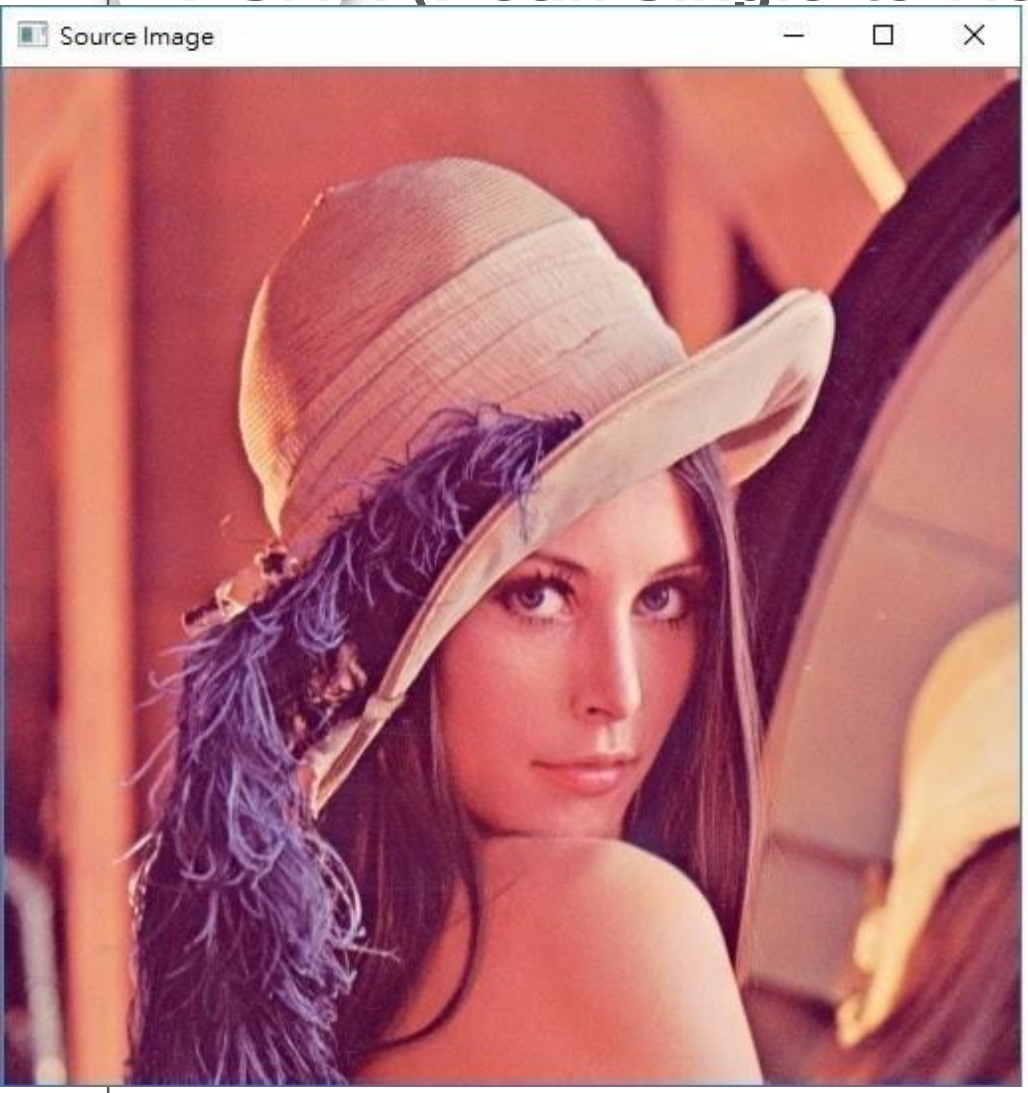
Figure 4: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

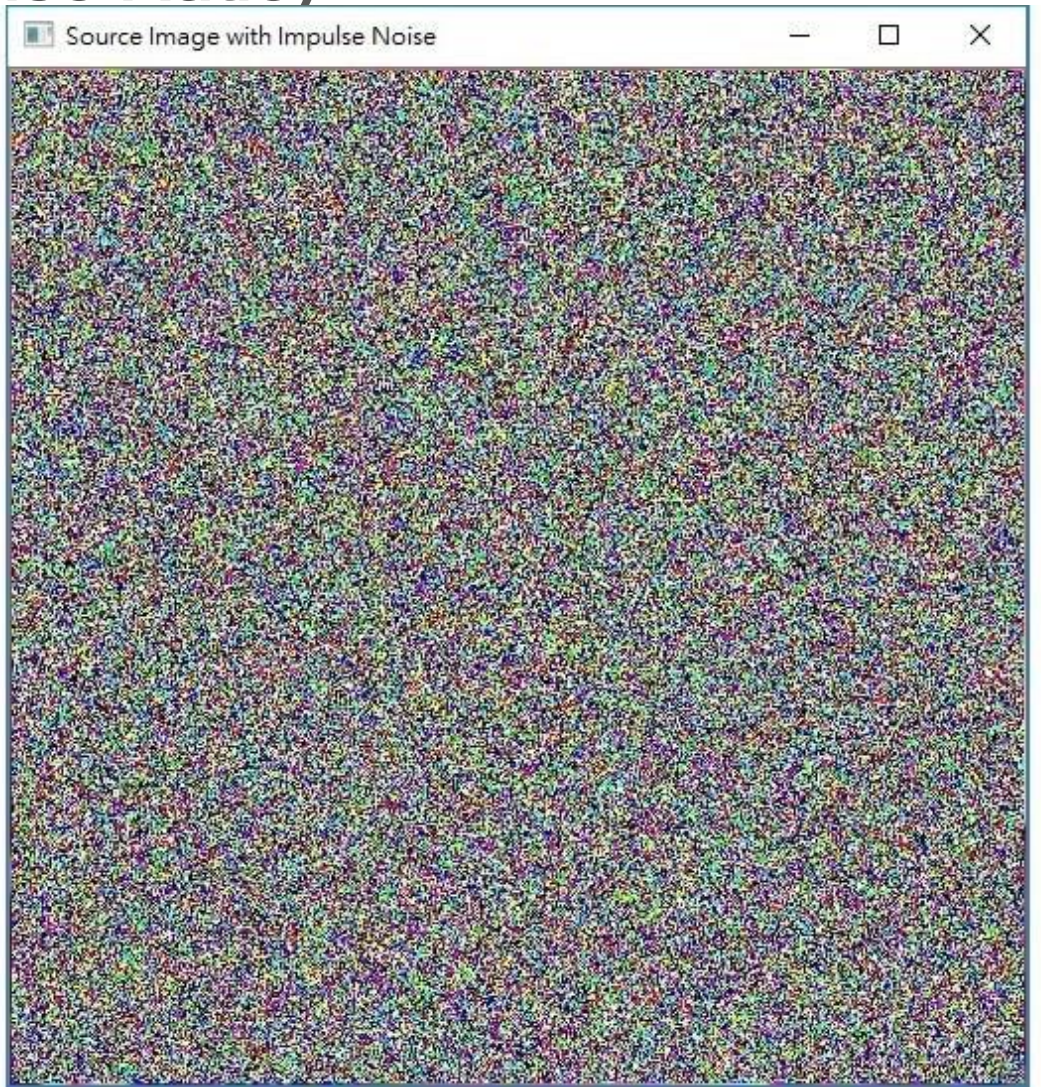
ESRGAN




```
class ResidualDenseBlock_5C(nn.Module):
    def __init__(self, nf=64, gc=32, bias=True):
        super(ResidualDenseBlock_5C, self).__init__()
        # gc: growth channel, i.e. intermediate channels
        self.conv1 = nn.Conv2d(nf, gc, 3, 1, 1, bias=bias)
        self.conv2 = nn.Conv2d(nf + gc, gc, 3, 1, 1, bias=bias)
        self.conv3 = nn.Conv2d(nf + 2 * gc, gc, 3, 1, 1, bias=bias)
        self.conv4 = nn.Conv2d(nf + 3 * gc, gc, 3, 1, 1, bias=bias)
        self.conv5 = nn.Conv2d(nf + 4 * gc, nf, 3, 1, 1, bias=bias)
        self.lrelu = nn.LeakyReLU(negative_slope=0.2, inplace=True)

        # initialization
        # mutil.initialize_weights([self.conv1, self.conv2, self.conv3, self.conv4, self.conv5], 0.1)

    def forward(self, x):
        x1 = self.lrelu(self.conv1(x))
        x2 = self.lrelu(self.conv2(torch.cat((x, x1), 1)))
        x3 = self.lrelu(self.conv3(torch.cat((x, x1, x2), 1)))
        x4 = self.lrelu(self.conv4(torch.cat((x, x1, x2, x3), 1)))
        x5 = self.conv5(torch.cat((x, x1, x2, x3, x4), 1))
        return x5 * 0.2 + x
```

PSNR (Peak Single-to-Noise Ratio)





Impulse Noise
Pa: 0.5
Pb: 0.5

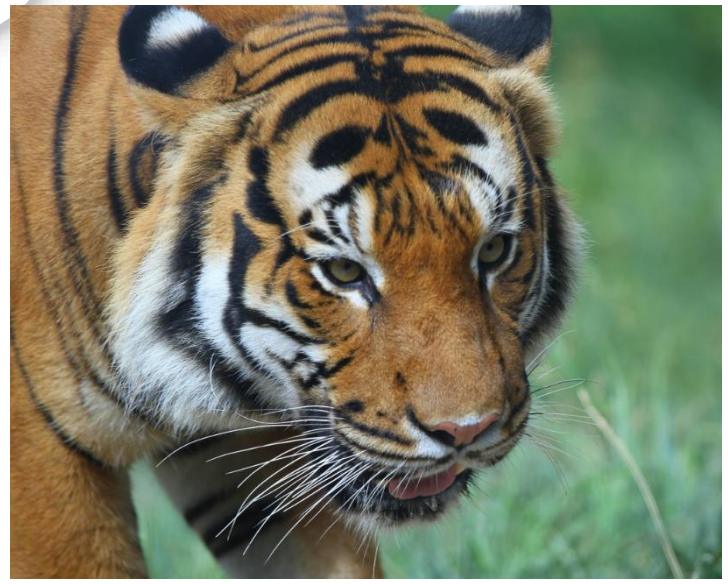
PSNR: 5.18667

Jason Chen's Blog

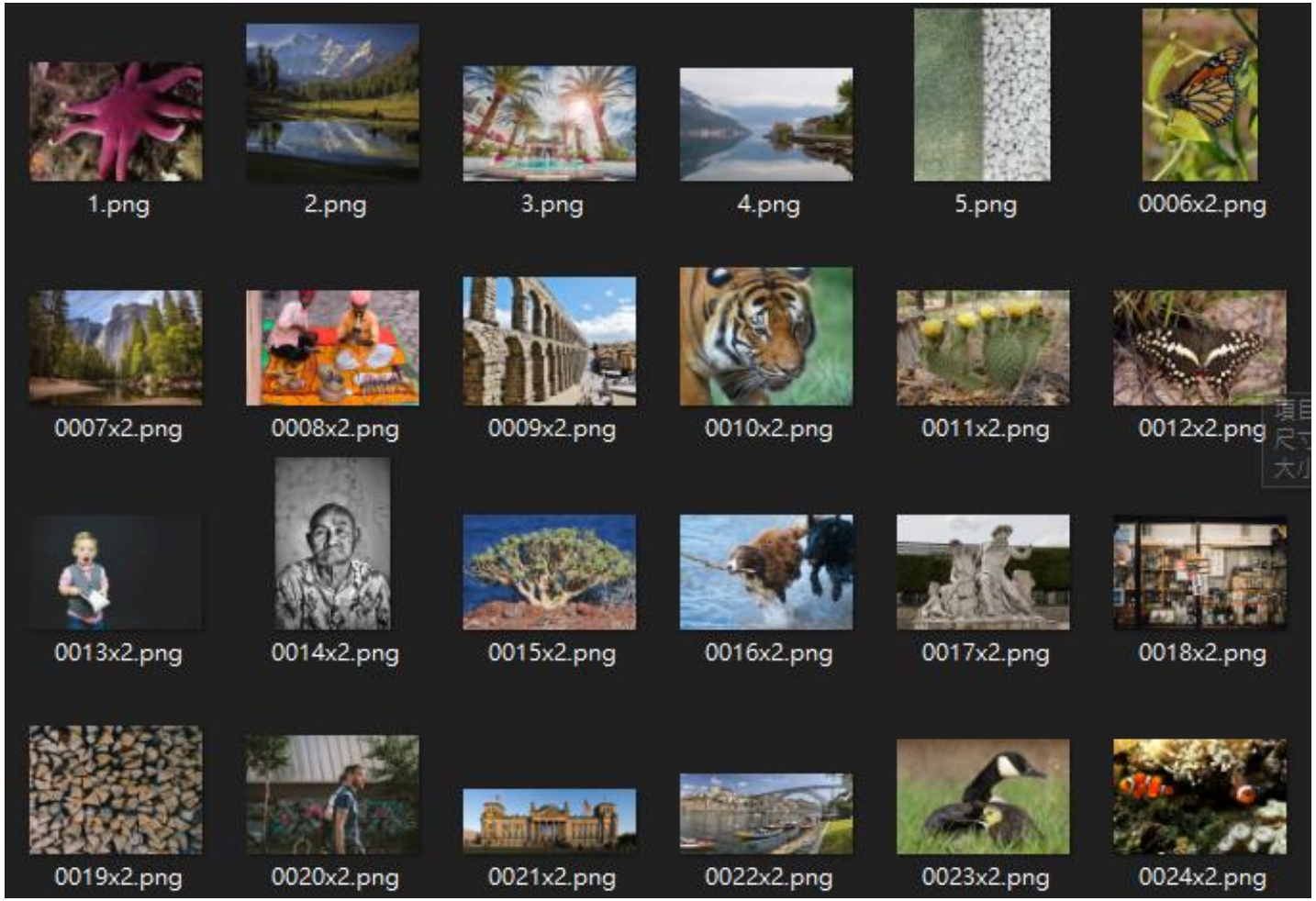
Part 3

個案研究

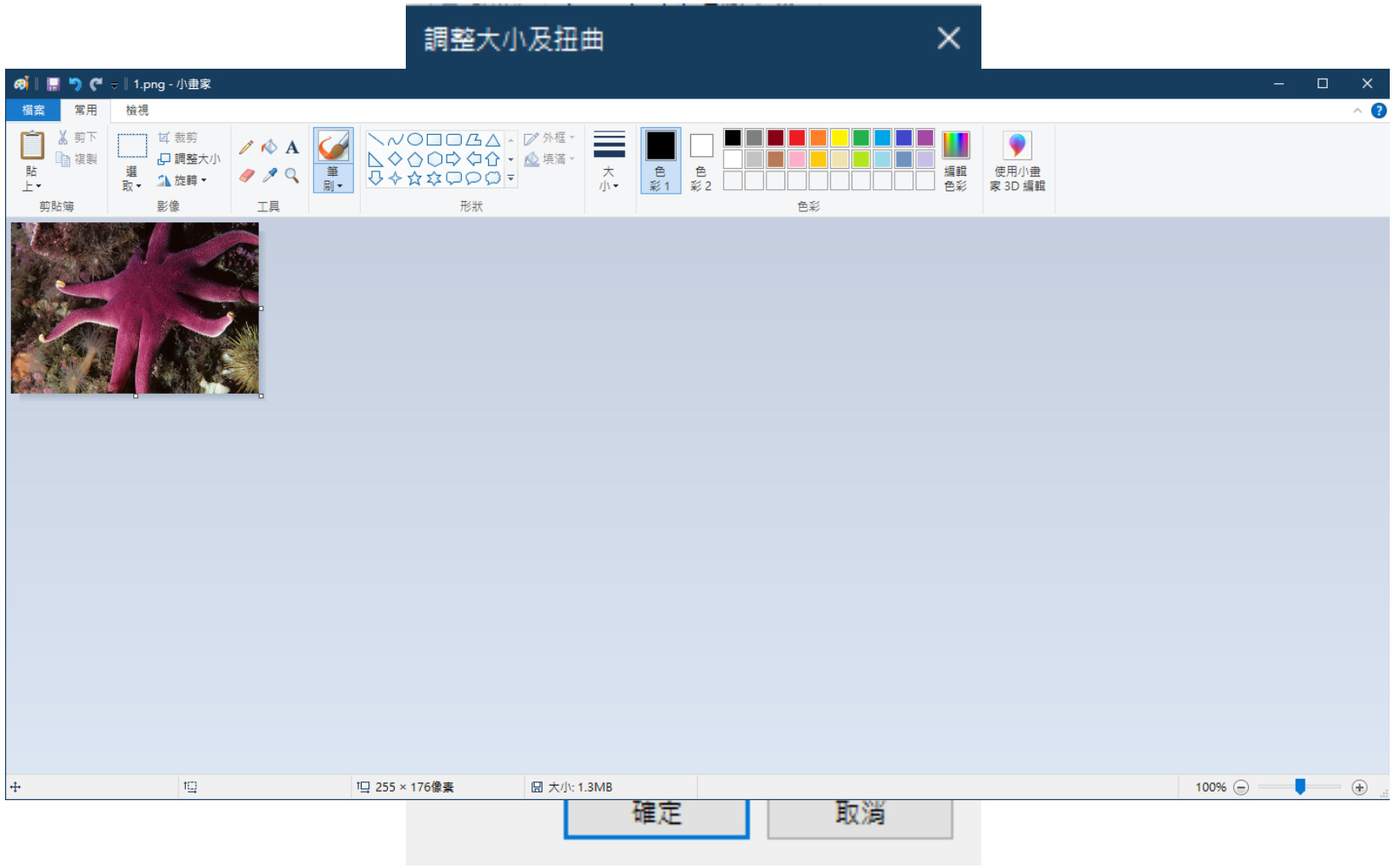
div2k數據集



資料前處理(1/3)


1.png
2.png
3.png
4.png
5.png
0006x2.png
0007x2.png
0008x2.png
0009x2.png
0010x2.png
0011x2.png
0012x2.png
0013x2.png
0014x2.png
0015x2.png
0016x2.png
0017x2.png
0018x2.png
0019x2.png
0020x2.png
0021x2.png
0022x2.png
0023x2.png
0024x2.png

資料前處理(2/3)



資料前處理(3/3)


HAND-MADE
ENGELS
SCHMIEDE
CUSTOM MOTORCYCLES
SERVICE


ENGELS
SCHMIEDE

SR-NET模型建立(1/2)

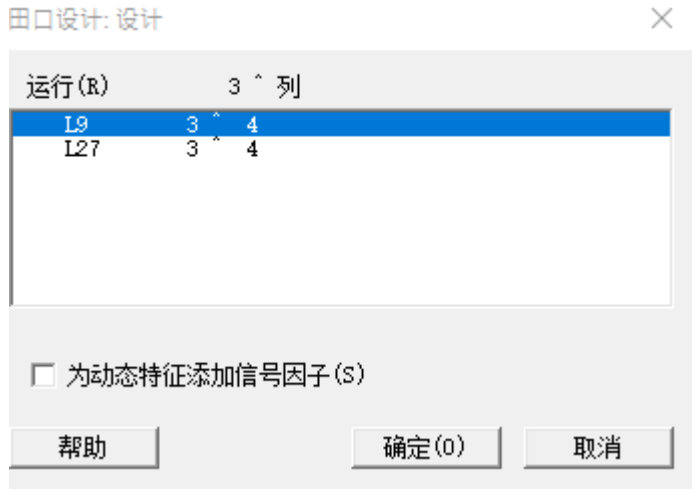
```
def down(filters , kernel_size, apply_batch_normalization = True):
        downsample = tf.keras.models.Sequential() # Sequential() 定義模型的開始
        downsample.add(layers.Conv2D(filters,kernel_size,padding = 'same', strides = 2)) # 2D 卷積層
        if apply_batch_normalization:
                downsample.add(layers.BatchNormalization())
        downsample.add(keras.layers.LeakyReLU())
        return downsample


def up(filters, kernel_size, dropout = False): # dropout 對抗過擬合的正則化方法
        upsample = tf.keras.models.Sequential()
        upsample.add(layers.Conv2DTranspose(filters, kernel_size,padding = 'same', strides = 2)) # Con2DTranspose 反卷積, 特徵還原成圖片
        if dropout:
                upsample.dropout(0.2)
        upsample.add(keras.layers.LeakyReLU())
        return upsample
```

SR-NET模型建立(2/2)

```
def model():
    inputs = layers.Input(shape= [SIZE,SIZE,3])
    d1 = down(128,(3,3),False)(inputs)
    d2 = down(128,(3,3),False)(d1)
    d3 = down(256,(3,3),True)(d2)
    d4 = down(512,(3,3),True)(d3)
    d5 = down(512,(3,3),True)(d4)
    #upsampling
    u1 = up(512,(3,3),False)(d5)
    u1 = layers.concatenate([u1,d4])
    u2 = up(256,(3,3),False)(u1)
    u2 = layers.concatenate([u2,d3])
    u3 = up(128,(3,3),False)(u2)
    u3 = layers.concatenate([u3,d2])
    u4 = up(128,(3,3),False)(u3)
    u4 = layers.concatenate([u4,d1])
    u5 = up(3,(3,3),False)(u4)
    u5 = layers.concatenate([u5,inputs])
    output = layers.Conv2D(3,(2,2),strides = 1, padding = 'same')(u5)
    return tf.keras.Model(inputs=inputs, outputs=output)
```

Model 1 DOE factor:

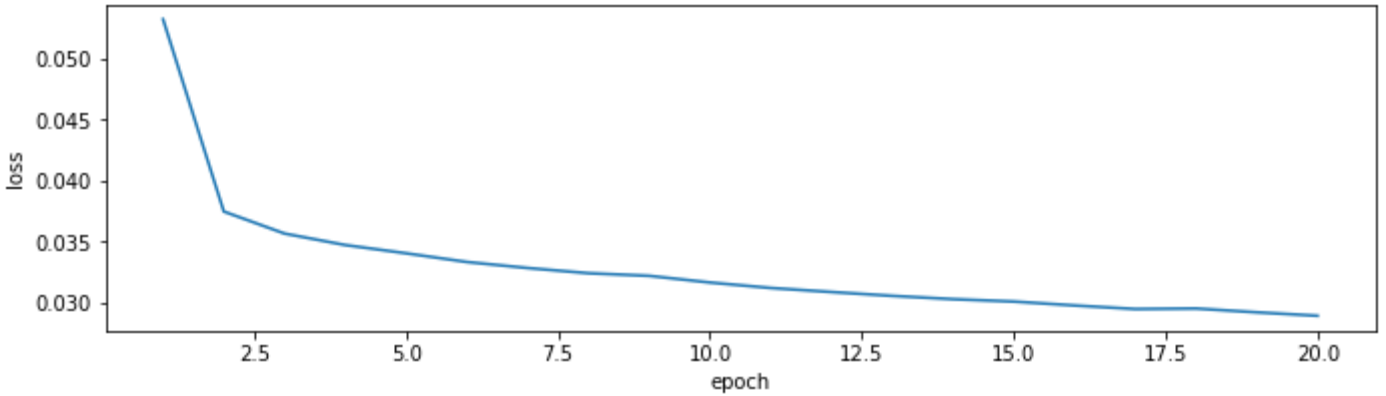
	Level 1	Level 2	Level 3
Learning rate	0.01	0.001	0.0001
Optimizer	AdaDelta	Adam	Adagrad
Activate function	Tanh	ELU	ReLu
Filters	32	64	128

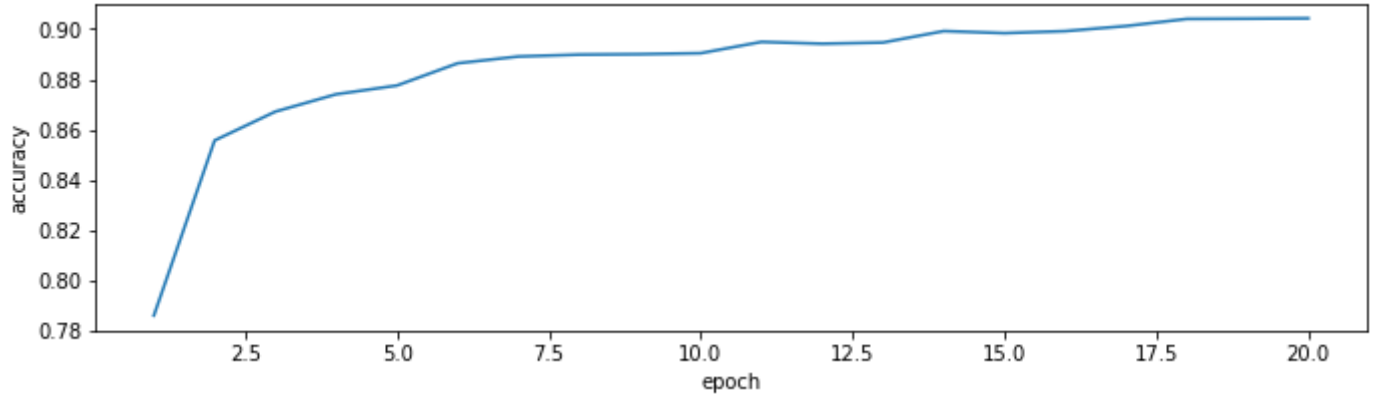


OA L9(34)

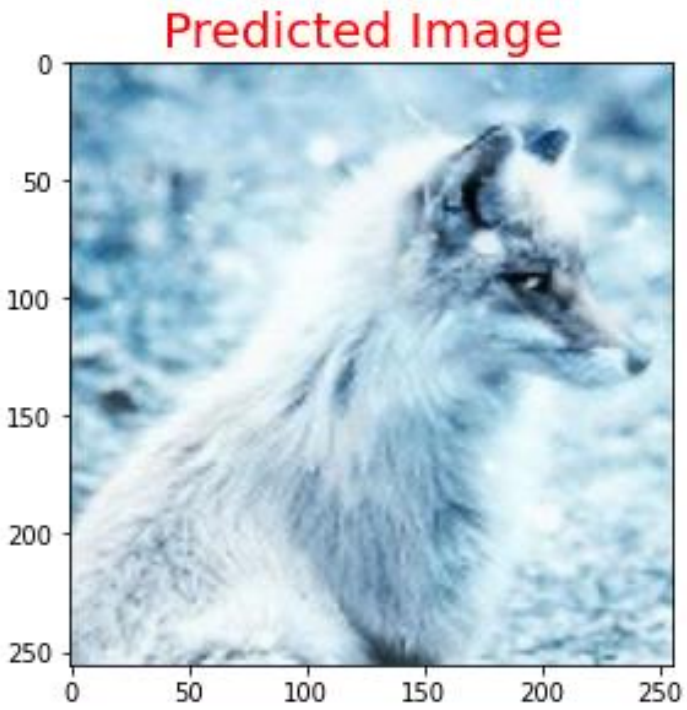
		Learning rate	Optimizer	Activate function	Filters	Accuracy	Loss
	1	0.01	RMSprop	Tanh	32	0.4665	0.8670
	2	0.01	Adam	Sigmoid	64	0.3716	0.7261
	3	0.01	Adagrad	ReLu	128	0.8505	0.0385
✔	4	0.001	RMSprop	Sigmoid	128	0.9037	0.0282
	5	0.001	Adam	ReLu	32	0.8930	0.0297
	6	0.001	Adagrad	Tanh	64	0.8984	0.0296
	7	0.0001	RMSprop	ReLu	64	0.8666	0.0353
	8	0.0001	Adam	Tanh	128	0.8601	0.0363
	9	0.0001	Adagrad	Sigmoid	32	0.8239	0.0414

SR-NET參數調整


loss
0.050
0.045
0.040
0.035
0.030
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
epoch

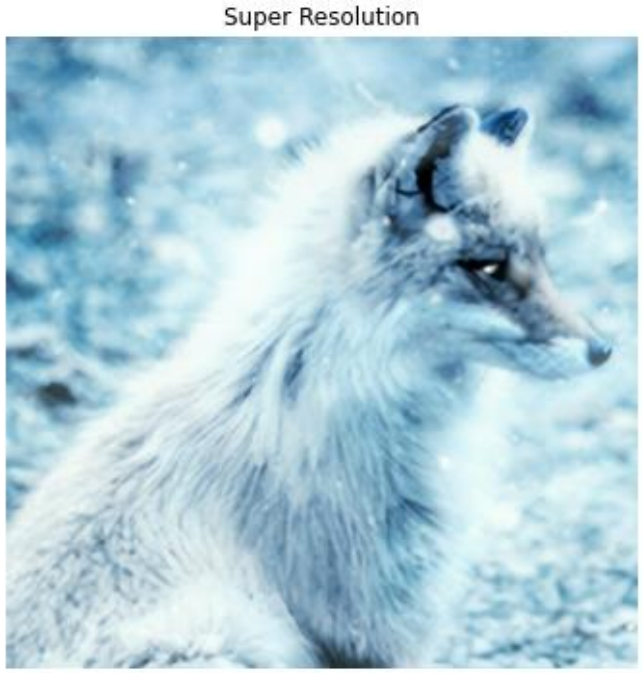

accuracy
0.90
0.88
0.86
0.84
0.82
0.80
0.78
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
epoch

生成圖像結果



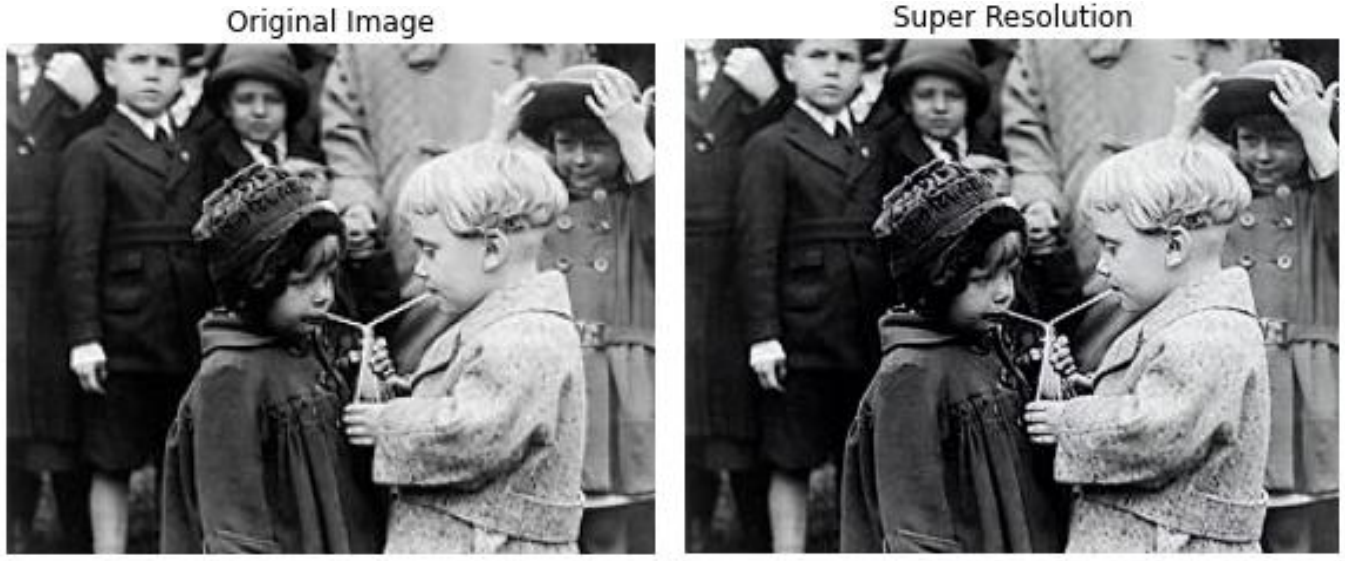
CNN生成圖像
PSNR：19.285225

生成圖像結果



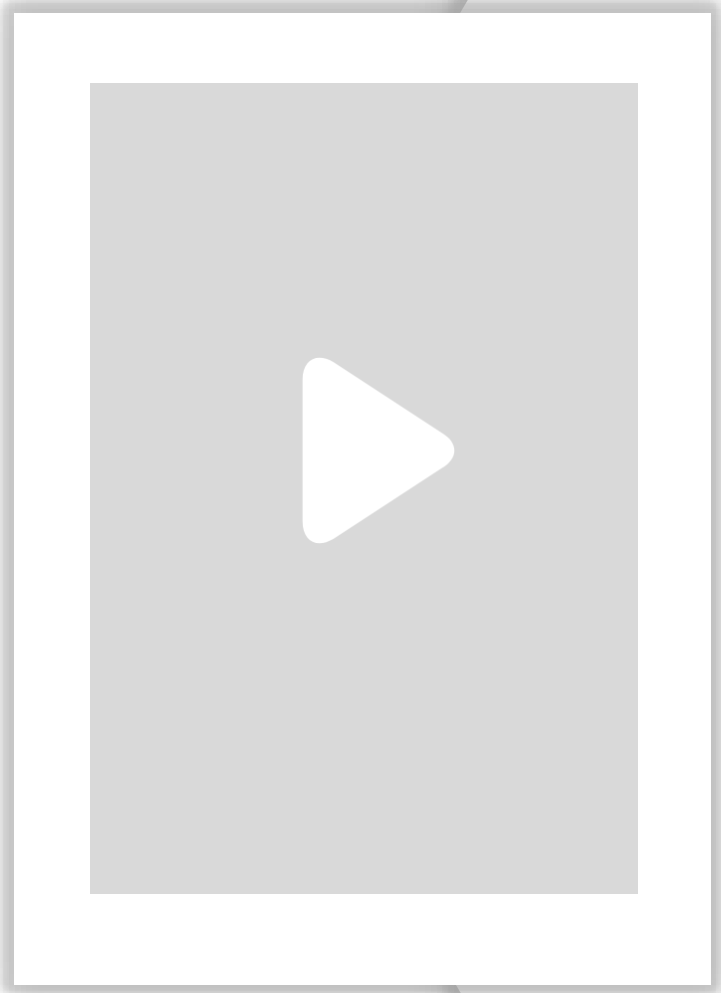
ESRGAN生成圖像
PSNR：28.029171

ESRGAN 優化原始圖像



Part 4

結論



結論

貢獻

ESRGAN在經過div2k數據及訓練後，也可對於一般的照片進行清晰化。
利用去除批量正規化層使照片去除原本SRGAN所產生的偽影瑕疵。

侷限性

這次因為時間的關係，所以只對於CNN進行參數調整，因為GAN模型的參數較難調整，因為要同時調整判別器與生成器之間的參數，如果其中一方設定的太高或太低，會導致另一方的模型崩潰，有時間上與技術上的困難。

適用性

可以應用於手機相機模組。
藉由優化原始圖像細節，對於物件偵測有更好的運用。

未來改善

GAN模型在現今機器學習中十分熱門，也有好幾種不同的模型，可以透過更大量的資料輸入，與事實的在每個階段存檔，在模型崩潰前找出準確率與損失函數最滿意的參數調整。

參考文獻

- Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., ... & Change Loy, C. (2018). Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops* (pp. 0-0).
- Kang, X., Liu, L., & Ma, H. (2020). ESR-GAN: Environmental Signal Reconstruction Learning With Generative Adversarial Network. *IEEE Internet of Things Journal*, *8*(1), 636-646.
- Takano, N., & Alaghband, G. (2019). Srgan: Training dataset matters. *arXiv preprint arXiv:1903.09922*.
- Nagano, Y., & Kikuta, Y. (2018, July). SRGAN for super-resolving low-resolution food images. In *Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management* (pp. 33-37).

group1

Project 2

THANK. YOU