Deep Learning Lab2 Report

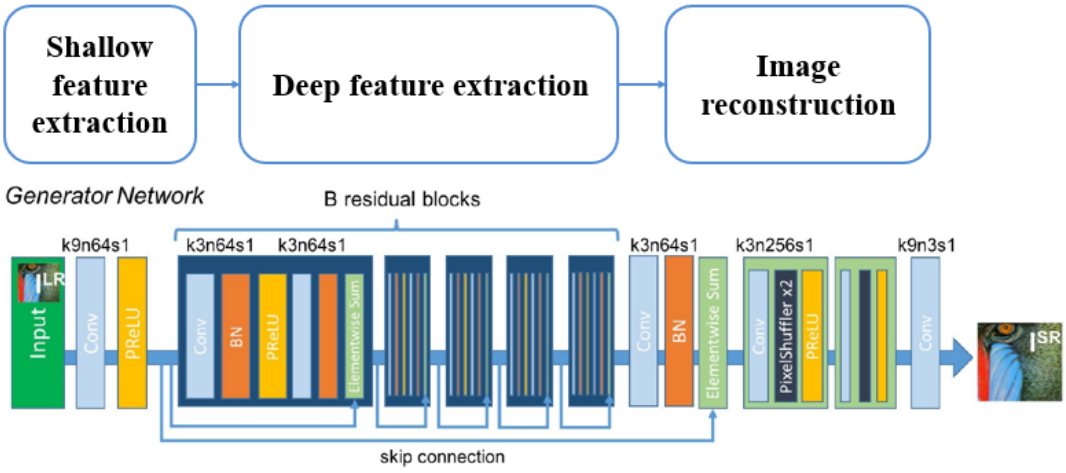
110511118 陳孟頡

Task 1.

I. SRResNet Implementation

1. Overview

Super resolution 為現今機器學習的重要研究領域之一，其應用模型種類齊全，然而，大部分模型有著相似的架構，也就是: 淺層特徵萃取→深層特徵萃取→圖像重建。以下針對 SRResNet 進行各層的實作說明。



2. Shallow feature extraction

此次實作的 SRResNet 中，以單一一層 9x9 kernel 來進行。使用 9x9 而非傳統 3x3 kernel，是為了快速縮圖，減少計算量，而使用單一 kernel，能在不使用大量參數及計算量的狀況下，擷取圖像中淺層的特徵資訊。

```
self.conv1 = nn.Conv2d(in_channels=3, out_channels=hidden_channels, kernel_size=9, stride=1, padding=4)
self.act1 = nn.PReLU()
```

3. Deep feature extraction

深層特徵萃取階段，會從 low resolution 影像中學習、提取高維特徵。這些特徵有助於重建 high resolution 版本的影像的更多細節。

Task 1 中，使用 16 層 residual block 進行深層特徵萃取，每個 block 以 3x3 convolution、batch normalization、PreLU 構成，skip connection 的加入，使整個模型進行 back propagation 更加順利，方便進行訓練。

```
self.residualBlock = nn.Sequential(
    nn.Conv2d(in_channels=hidden_channels, out_channels=hidden_channels, kernel_size=3, stride=1, padding=1),
    nn.BatchNorm2d(hidden_channels),
    nn.PReLU(),
    nn.Conv2d(in_channels=hidden_channels, out_channels=hidden_channels, kernel_size=3, stride=1, padding=1),
    nn.BatchNorm2d(hidden_channels),
)
```

4. Image reconstruction

取得圖片資訊後，SRResNet 使用兩個 up-sampling block 來進行圖像重建，每個 block 會將圖像放大為兩倍，因此經過整個模型後能將原圖片放大為 4 倍。

```
self.upsampleBlock = nn.Sequential(
    nn.Conv2d(in_channels=hidden_channels, out_channels=hidden_channels*(2**2), kernel_size=3, stride=1, padding=1),
    nn.PixelShuffle(2),
    nn.PReLU()
)
```

II. Training Method

1. Overview

Data augmentation	Horizontal flip
	Vertical flip
Batch size	Training: 1
	Testing: 8
Loss function	L1 loss
Optimizer	Adam
Learning rate	1e-4 ~ 1e-6
Learning rate scheduler	CosineAnnealingWarmRestarts

2. Data augmentation

此次作業嘗試的 data augmentation，以及其效果整理如下表:

Data augmentation	Accuracy	Explanation
Horizontal flip、Vertical flip	+	增加模型對於圖像不同方向特徵的處理能力
Random rotation	-	無法解決旋轉造成的黑邊問題，使用 random crop 也會使準確度降低
Color jitter	~	顏色變化對於 SR 任務並無顯著提升，主要著重於圖像的紋理變化
Gaussian blur	-	圖片平緩化可能使圖像中的細節消失，使重建圖片更加困難

最後採用水平翻轉以及垂直翻轉。

3. **Batch size**

在 super resolution 中，單一圖像的輪廓以及材質對於重建高精確度圖片至關重要，此資訊屬於 pixel-level，而非 batch-level，因此若在 super resolution 任務使用常用的大 batch size，可能使模型失去對於單一圖片特徵的萃取能力。因此在訓練時將 batch size 設為 1，使 SRResNet 能最佳提取圖像細節特徵。

4. **Loss function**

MSE loss 取模型預測圖片與高精度圖片的平方差，因此模型對不確定性高區域，傾向產生趨近於平均值數值以減少 loss，如此一來，所生成的 super resolution 圖片會看起來較為平滑、缺乏紋理及細節，呈現模糊狀態。

$$MSE\ loss = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

相對而言，L1 loss 對 pixel error 計算較為平均，所產出的圖像會略顯銳利，而非平滑的模糊圖片，因此適合用於 super resolution task。

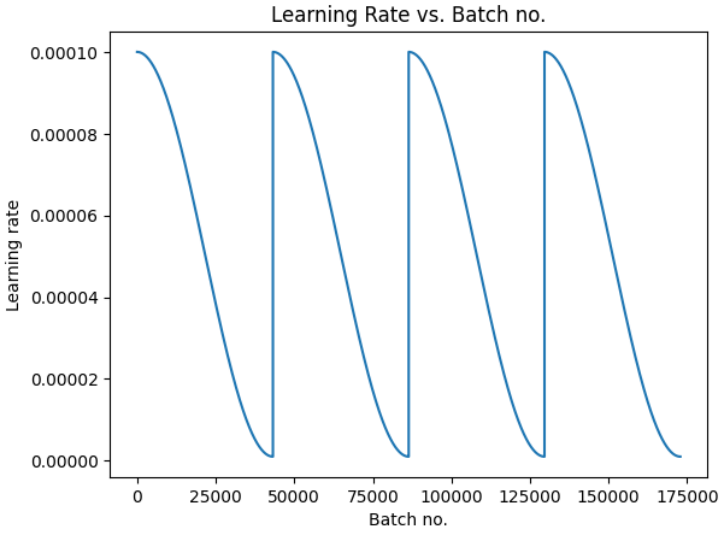
$$L1\ loss = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

5. **Optimizer**

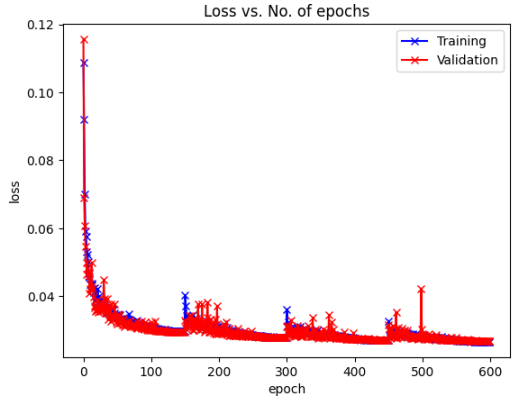
使用包含 momentum 及 adaptive learning rate 的 Adam。

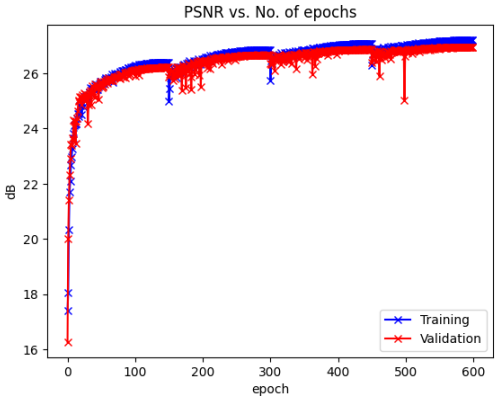
6. **Learning rate scheduler**

使用 CosineAnnealingWarmRestarts，且在 training 過程中 restart 四次以跳脫 local minimum，如下圖:



III. Result





Model size:

```
Total params: 1,549,462
Trainable params: 1,549,462
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 289.50
Params size (MB): 5.91
Estimated Total Size (MB): 295.46
```

Testing PSNR: 25.67 dB

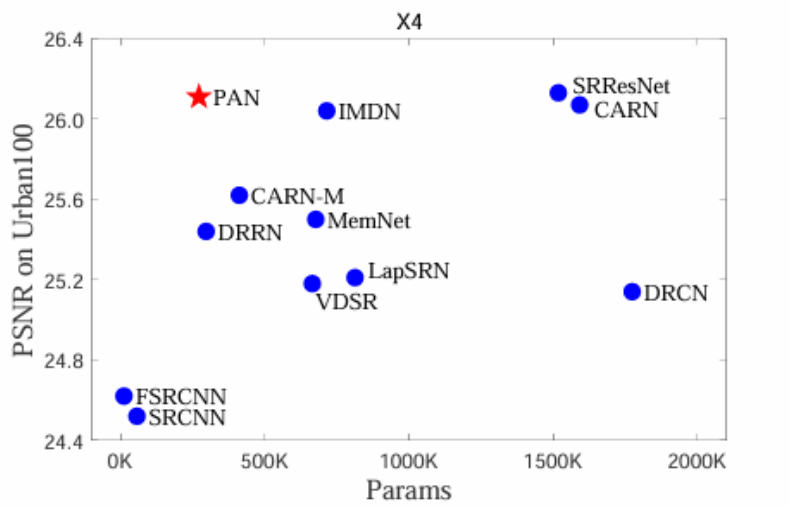
```
PSNR for the testing data: 25.67 dB
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Task 2.

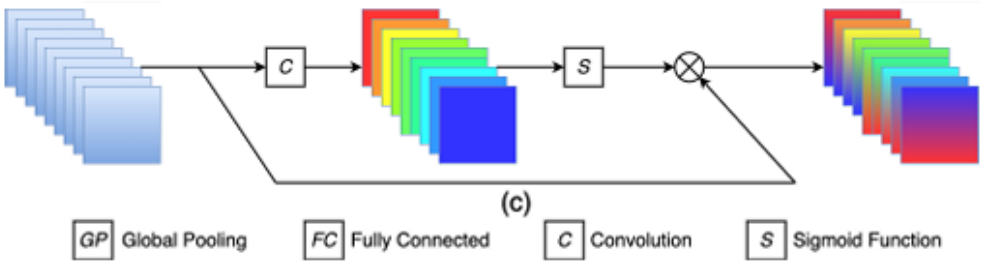
I. Model : PAN

1. Model feature

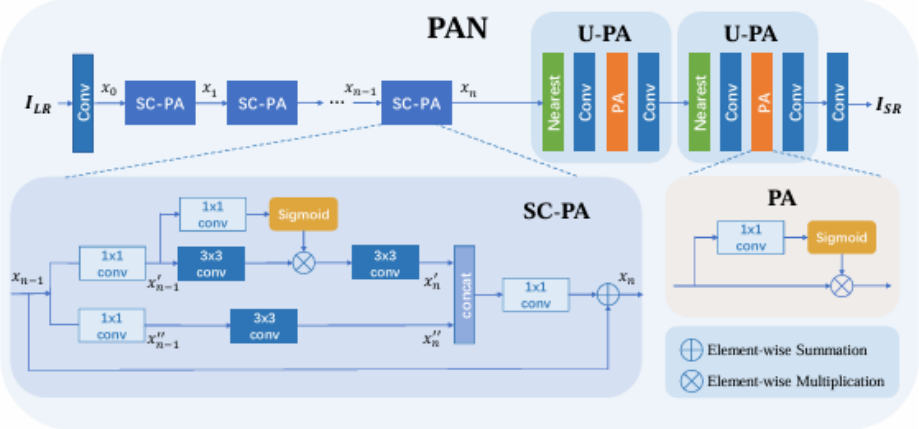
所採用的 PAN 模型，旨在以輕量的模型進行 super resolution，並在其中加入 pixel attention 以增進模型準確度。此模型在 AIM 2020 舉辦的 Efficient Super Resolution challenge 中取得佳績。



進行 super resolution 時，能提取出整張圖片 pixel 資料特徵，對於增進重建圖片能力有很大的幫助，而 pixel attention 即為其中一個增加 pixel 資料處理能力的方法。其主要的做法如下圖，會先經由 1x1 convolution 來取得 attention feature vector，接著進行 sigmoid 轉換為 0~1 之間數值後，便可與原 feature maps 進行相乘。



2. Network architecture



PAN 模型主要也分為 shallow feature extraction、deep feature extraction 以及 image reconstruction 部分，以下逐一介紹:

(1) Shallow feature extraction:

為了減少參數，PAN 使用單一 convolution 層進行淺層特徵萃取。

```
self.shallow_feature = nn.Sequential(
    nn.Conv2d(in_channels=3, out_channels=emb_dim, kernel_size=3, stride=1, padding=1)
)
```

(2) Deep feature extraction:

深度特徵萃取部分，則由 SC-PA(Self-Calibrate) block 組成，主要為不同的 receptive field 與 pixel attention 組成。

```
class SCPA_block(nn.Module):
    def __init__(self, emb_dim):
        super(SCPA_block, self).__init__()
        self.emb_dim = emb_dim

        self.conv_half1_1 =     nn.Conv2d(in_channels=emb_dim//2, out_channels=emb_dim//2, kernel_size=1, stride=1)
        self.conv_half3_1 =     nn.Conv2d(in_channels=emb_dim//2, out_channels=emb_dim//2, kernel_size=3, stride=1, padding=1)
        self.pixel_attention =  pixel_attention(emb_dim//2)
        self.conv_half3_2 =     nn.Conv2d(in_channels=emb_dim//2, out_channels=emb_dim//2, kernel_size=3, stride=1, padding=1)

        self.conv_half1_2 = nn.Conv2d(in_channels=emb_dim//2, out_channels=emb_dim//2, kernel_size=1, stride=1)
        self.conv_half3_3 = nn.Conv2d(in_channels=emb_dim//2, out_channels=emb_dim//2, kernel_size=3, stride=1, padding=1)

        self.conv_full1 = nn.Conv2d(in_channels=emb_dim, out_channels=emb_dim, kernel_size=1, stride=1)

        self.act1 = nn.ReLU()

    def forward(self, x):
        channels = x.shape[1]
        half_channels = channels//2
        path1 = x[:, :half_channels, :, :]
        path2 = x[:, half_channels:, :, :]

        # Path 1: with PA
        x1 = self.act1(self.conv_half1_1(path1))
        x1 = self.pixel_attention(x1) * self.act1(self.conv_half3_1(x1))
        x1 = self.act1(self.conv_half3_2(x1))

        # Path 2: No PA
        x2 = self.act1(self.conv_half1_2(path2))
        x2 = self.act1(self.conv_half3_2(x2))

        y = torch.cat((x1, x2), dim=1)
        y = self.act1(self.conv_full1(y)) + x

        return y
```

(3) Image reconstruction:

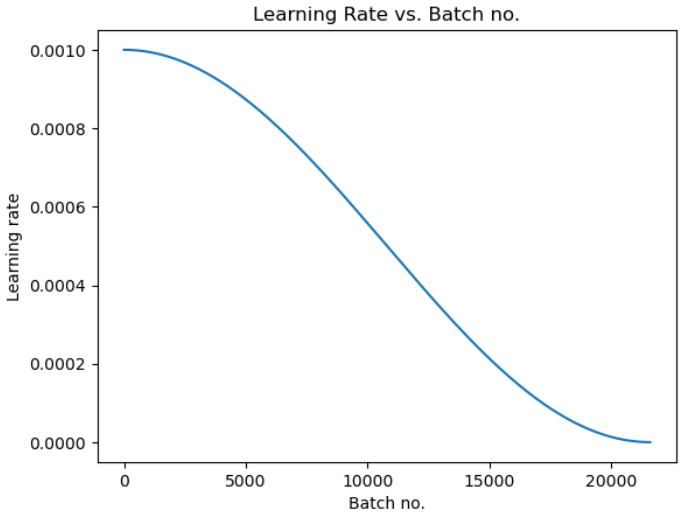
最後的重建影像部分，由 U-PA block 來進行。其主要包含 nearest neighbor up-sample layer、convolution layer 以及 pixel attention。之所以使用 neighbor up-sampling 是為了進一步縮減參數，而 pixel attention 的加入進一步提升模型準確度。

```python
class UPA_block(nn.Module):
    def __init__(self, emb_dim, scale_factor=2):
        super(UPA_block, self).__init__()
        self.emb_dim = emb_dim
        self.scale_factor = scale_factor

        self.upsample = nn.Upsample(scale_factor=scale_factor, mode='nearest')
        self.conv1 = nn.Conv2d(emb_dim, emb_dim, kernel_size=3, stride=1, padding=1)
        self.pixel_attention = pixel_attention(emb_dim)
        self.conv2 = nn.Conv2d(emb_dim, emb_dim, kernel_size=3, stride=1, padding=1)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        x = self.upsample(x)
        x = self.relu(self.conv1(x))
        x = self.pixel_attention(x)
        x = self.relu(self.conv2(x))

        return x
```

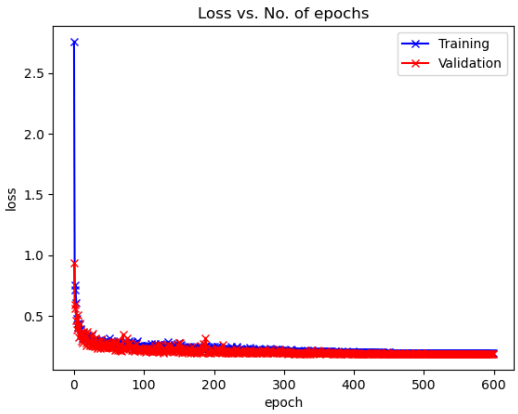
II. Training method

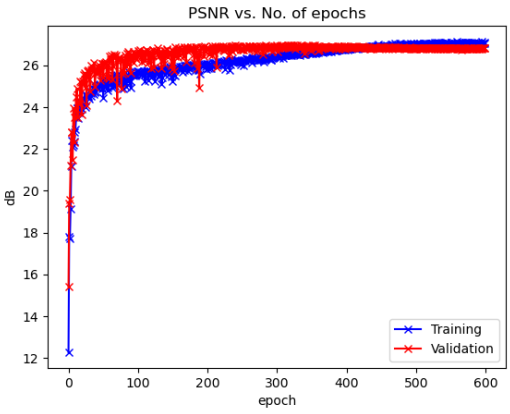
所參考的論文資料，提供了訓練的參數以及方式，本次採用相近的方式進行訓練:

Data augmentation	Horizontal and vertical filp
Batch size	8 (論文中使用 32)
Loss function	L1 loss
Optimizer	Adam
Learning rate	1e-3 ~ 1e-7
Learning rate scheduler	CosineAnnealingLR



III. Result





Testing PSNR: 25.78 dB

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
PSNR for the testing data: 25.78 dB
```

Questions & Answers

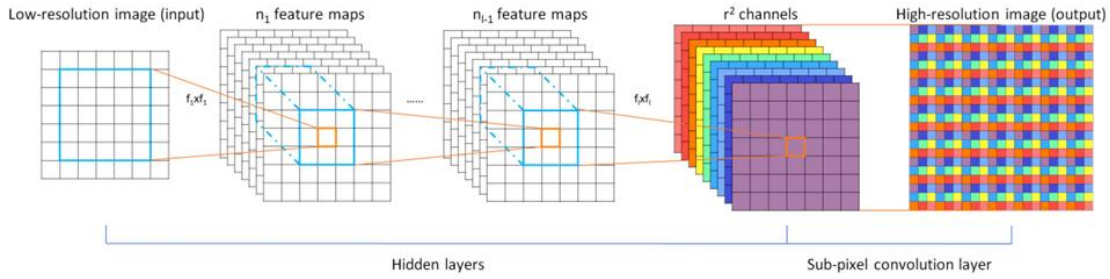
I. What is PixelShuffle?

PixelShuffle 是一種將輸入資料 up-sample 的方法，其運作流程如下:

1. Input shape:
   由模型將輸入圖片，從 RGB 3 個 channel (H, W, 3) 轉為包含更多 hidden dimensions 的 feature maps (H, W, C)

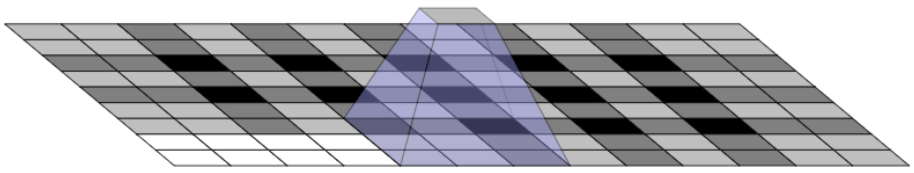
2. Reshape:
   依照 up-scale factor: **r**，將 feature maps 的 channel 數調整為 **C*r^2**，此時 feature maps 的維度為(H, W, C*r^2)

3. Shuffle:
   以 shuffle 的方式將(H, W, C*r^r)轉為(H*r, W*r, C)，來將圖片放大 r 倍



使用 PixelShuffle 的有點有以下幾點:

1. Efficient up-sampling:
   PixelShuffle 沒有一般 deconvolution(transpose convolution)會有的不均勻重疊(uneven overlap)問題，因此較不會產生 checkerboard artifacts。



2. Lower computational cost:
   因為不需要學習 filter，使 PixelShuffle 的參數和計算量，都少於 deconvolution，因此對於減少模型參數有幫助。

3. Better image quality:
   PixelShuffle 直接將 feature maps 重建為高解析度影像，因此可以產生更自然的放大影像，因此非常適合 super resolution 任務。

II. What is PSNR and why it is not the only metric used for evaluating super-resolution?

PSNR 常用來量測重建圖片的品質，透過與原圖片進行 pixel-wise 的比較，計算兩者的 MSE loss。其公式如下:

$$PSNR = 10 * \log_{10}\left(\frac{{MAX_i}^2}{MSE}\right) = 20 * \log_{10}\left(\frac{MAX_i}{MSE}\right)$$

其中 MAX_i 為圖像點顏色的最大數值，MSE 為 super resolution 重建圖片與 high resolution 的 MSE loss。

PSNR 被普遍使用的原因，在於其使用容易以及可取得 pixel-level 的差異性數值。然而，PSNR 也存在幾個問題:

1. 僅計算數值品質:
   PSNR 僅對圖片進行 pixel-wise 差異計算，並未將人眼主觀對皮片的感受納入考量範圍。

2. 偏好平緩圖片:
   由於 PSNR 計算包含 MSE loss，因此也會產生偏好平緩、模糊圖片，此對於一般人眼偏好的銳利圖片大相逕亭。

3. 未考慮結構相似度:
   PSNR 並未計算圖片的結構差異，若重建的圖片中含有輕微的結構扭曲，可能仍有高的 PSNR 值

為解決以上問題，現今有許多其他用以計算圖片品質的方式:

1. SSIM (Structural Similarity Index)
   主要是比較圖片之間的亮度、對比度、結構。亮度部分，會以高解析度圖片灰階平均作為指標；對比度部分，由圖像的標準差作為參考依據；而結構則是由計算兩張圖片的斜方差來獲得。

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha}[c(\mathbf{x},\mathbf{y})]^{\beta}[s(\mathbf{x},\mathbf{y})]^{\gamma}，$$

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}，c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}，s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

   因 SSIM 比較整體圖片結構，適合用來判別圖片的相似程度。

2. IS (Inception Score)
   此方法使用 Inception 網路來計算生成圖片的分數，主要判斷其清晰程度，以及生成的多樣性(SR task 中不常使用)。Inception 網路為已訓練好的 classify model。它使用 ImageNet 資料集來進行訓練，總計有超過百萬張照片，總計有 1000 的分類。

3. FID (Fréchet Inception Distance)
   FID 會比較生成圖片與原圖的特徵分佈，計算方法為 IS 的延伸，也是以 pre-trained model 進行。此方法常用於 GAN-based super resolution 模型。與 PSNR 不同之處在於，FID 分數越低，表示生成圖片越接近原圖，圖片品質越高。