機器學習工程應用　　期末專題競賽報告

姓名:楊憲閔　　學號:F74096124　　系所:資訊系四乙

1. 資料前處理

```
image = imread_uint(images[i], n_channels=3)
label = imread_uint(labels[i], n_channels=3)

image = uint2single(image)
label = uint2single(label)

np.random.seed(seed=0)  # for reproducibility
image += np.random.normal(0, 15/255., image.shape)

image = single2tensor4(image)
label = single2tensor4(label)

image = image.to(device)
label = label.to(device)
```

```
def imread_uint(path, n_channels=3):
    # input: path
    # output: HxWx3(RGB or GGG), or HxWx1 (G)
  if n_channels == 1:
    img = cv2.imread(path, 0)  # cv2.IMREAD_GRAYSCALE
    img = np.expand_dims(img, axis=2)  # HxWx1
  elif n_channels == 3:
    img = cv2.imread(path, cv2.IMREAD_UNCHANGED)  # BGR or G
    if img.ndim == 2:
      img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)  # GGG
    else:
      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # RGB
  return img
```

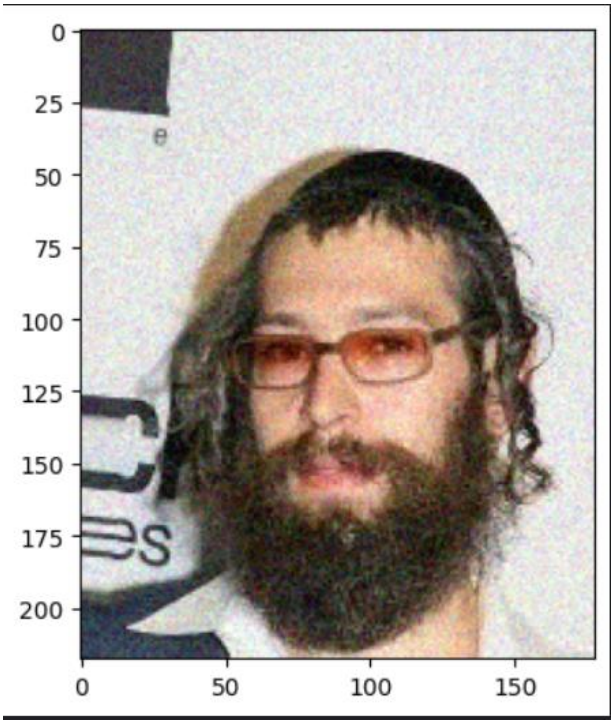
首先先透過 opencv 的 cv2.imread 讀進來圖片，又因讀進來的是 BGR 格式，所以再利用 cvtColor 對其進行轉換，轉換成最常用的 RGB 格式

```python
def uint2single(img):
    return np.float32(img/255.)
```

先轉成 float 形式，再同除以 255 讓 data 呈現 0~1 的分布

```python
np.random.seed(seed=0)  # for reproducibility
image += np.random.normal(0, 15/255., image.shape)
```

對其進行添加高斯噪聲，讓他變成一個更有噪音的圖像，就能讓模型學習全面性的除噪

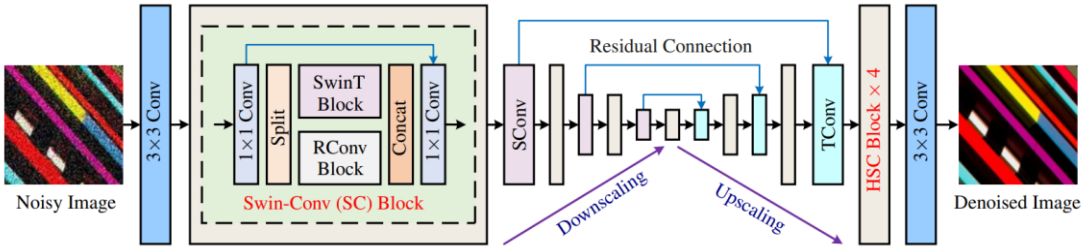



```python
def single2tensor4(img):
    return torch.from_numpy(np.ascontiguousarray(img)).permute(2, 0, 1).float().unsqueeze(0)
```

再來從 float 轉 tensor，先利用 from_numpy()將 ndarray 轉成 tensor，再將其從(H,W,C)轉成(C,H,W)的 tensor 格式，最後在最前面新增一個維度，讓 training 可以成為 batch 格式來跑

2. 訓練模型建立

首先先定義模型，使用的是與論文作者提出的 SCUnet 架構



這個架構主要是結合了兩個近期的新架構:DRUNet 和 SwinIR 的基礎架構，提出了新的概念:swin-conv(SC)，其中的概念為先經過一個 1*1 的 convolutions，並切割為兩個，來分別送進 SwinT(swin transformer block)與 RConv(residual convolutional block)，這兩個分別又進行了 upscaling 與 downscaling，之後再進行 concate，因此這個 SC 具備了提取局部特徵(Convolution block)與捕捉全局特徵關係(SwinT&RConv)，進而達到去噪的能力
後半部則是透過 encoder 與 decoder 來進行重建，又含有 residual connection 來讓 encode 過程時的特徵圖連接到 decode upsampling 的過程，來保有些許的圖像訊息，避免在 upsampling 時導致特徵丟失

```python
def test_onesplit(model, L, refield=32, min_size=256, sf=1, modulo=1):
    '''
    model:
    L: input Low-quality image
    refield: effective receptive filed of the network, 32 is enough
    min_size: min_sizeXmin_size image, e.g., 256X256 image
    sf: scale factor for super-resolution, otherwise 1
    modulo: 1 if split
    '''
    h, w = L.size()[-2:]

    top = slice(0, (h//2//refield+1)*refield)
    bottom = slice(h - (h//2//refield+1)*refield, h)
    left = slice(0, (w//2//refield+1)*refield)
    right = slice(w - (w//2//refield+1)*refield, w)
    Ls = [L[..., top, left], L[..., top, right], L[..., bottom, left], L[..., bottom, right]]
    Es = [model(Ls[i]) for i in range(4)]
    b, c = Es[0].size()[:2]
    E = torch.zeros(b, c, sf * h, sf * w).type_as(L)
    E[..., :h//2*sf, :w//2*sf] = Es[0][..., :h//2*sf, :w//2*sf]
    E[..., :h//2*sf, w//2*sf:w*sf] = Es[1][..., :h//2*sf, (-w + w//2)*sf:]
    E[..., h//2*sf:h*sf, :w//2*sf] = Es[2][..., (-h + h//2)*sf:, :w//2*sf]
    E[..., h//2*sf:h*sf, w//2*sf:w*sf] = Es[3][..., (-h + h//2)*sf:, (-w + w//2)*sf:]
    return E
```

再透過模型輸出時需切割圖片為四個小部分:左上、右上、左下、右下，因為在這個 dataset 中圖片較為大張，需要先做切割才能較為有效地學習除噪或是修復圖片，同時也是在預測時能對小面積進行處理。而切割後還需還原成原本大小的圖片，因此就定義好一個原始 tensor，再對其進行拼接以輸出預測
輸出後再透過 L1loss function 來計算輸出與 label 之間的差距，以進行更新權重

3. 參數調整

用跟論文作者使用之參數相似，差別在於 Adam 的 initial learning rate 使用的是 0.0005，而作者則是從 0.0001 開始

4. 預測結果

(左圖為 output，右圖為 label)

(最後分數)

參考資料:

https://arxiv.org/pdf/2203.13278