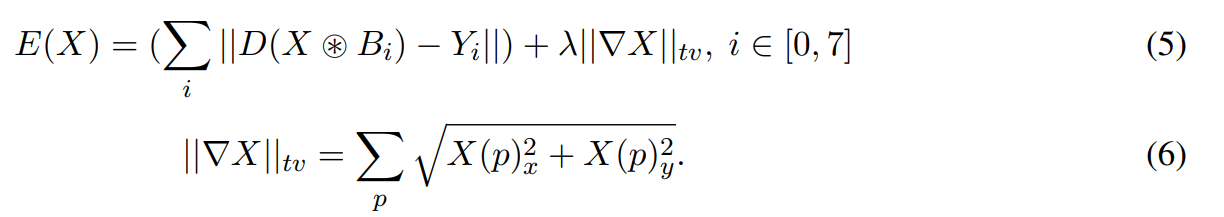
* Experiment a.
  + Assumption

Multi-image optimization-based approch是透過多張不同拍攝角度的低解析度照片(拍攝角度要非常接近但不一致)，並採用以下公式來求得一張高解析度照片:



我認為餵入越多張照片，可以為像素鄰近點帶來更多資訊，對高解析度照片的生成是有幫助的，會讓此高解析度的照片品質更好，但如上式(5)所示，越多張照片意味著目標函數內的，故生成一張高解析度照片所需時間也會變得較長。

* + Justification

為了驗證我的假設，我分別用2張、4張、6張照片做運算，為了讓實驗更嚴謹，我挑選的4張照片有2張是剛剛前面挑過的2張；挑選的6張照片有4張是剛剛前面挑過的4張。並記錄運算時間，輸出的照片再分別與original HR image ('./reference/HR\_zebra\_test.png')計算PSNR值。結果如下表:

|  |  |  |  |
| --- | --- | --- | --- |
|  | result | PSNR | 執行時間 |
| 2 images |  | 20.8333 dB | 669 sec |
| 4 images |  | 23.6862 dB | 1107 sec |
| 6 images |  | 25.0554 dB | 1536 sec |

由上表可看出使用照片張數越多，PSNR值越高，代表越接近真正的高解析度照片。且我的實驗方式是原本的照片量再多挑兩張的方式做多照片疊加運算，並不是隨機挑選2張4張6張的方式，因為隨機挑選照片可能因為某張低解析度照片對高解析度的資訊提供特別有幫助，那這樣就有失實驗嚴謹性。不過張數越多造成計算量增加，所以運算時間也隨之增加。驗證我的Assumption。

* Experiment b.
  + - Assumption

因為此兩個model(net\_F16B2\_epoch\_15.pth & net\_F64B8\_epoch\_100.pth)所用的訓練資料類別太單調，model學到的多半是斑馬的紋路與草坪的紋理，而image\_hidden/內的圖片的某些物體的紋理是model沒學到的，所以推測餵入image\_hidden/內的圖片給model會產出類似斑馬紋理或是草坪紋理的樣子出現，並且因為訓練次數的不同，越多訓練次數的model理論上學的越紮實，故推測net\_F64B8\_epoch\_100.pth所產出的HR image會比net\_F64B8\_epoch\_100.pth所產出的HR image看起來更銳利。

* + - Justification 1

由下表可看出model所產出的圖片有些許奇怪，像是斑馬線上的行人已經被融入成斑馬條紋的感覺，所以會有點扭曲，另外像是模糊的小狗圖片經過model產出的HR image相當奇怪，也有一點斑馬紋路的感覺，而熊貓圖與小狗圖的草坪紋理以及馬的紋理在高解析度下的處理都不錯，因為training dataset內就有這些物件的紋路可以學習。而在不考慮圖片的奇怪性，net\_F64B8\_epoch\_100.pth所產出的HR image的確比net\_F16B2\_epoch\_15.pth所產出的HR image更佳清晰，上述驗證了Assumption。



* + - Justification 2

由Justification 1看出小狗圖與馬路圖的HR image表現極差，所以我找了3張小狗的圖片與3張斑馬線的圖片當作training & validation set來重新訓練model，而training & validation set的圖片之高解析度圖片就是原圖，低解析度圖片採用opencv.resize將長寬各除以4來得到。我訓練了兩個model，分別是小狗的SR model與斑馬線的SR model。不單靠一個model訓練的原因是這種dense to dense problem有別於一般的classification problem，如果把所有的類別的data全部丟入同一model做訓練，對super resolution task來說會有過多的雜訊，產出的HR image效果會不佳，這種dense to dense task傾向先採用classifier辨識場景或類別，再透過該單一類別的model來處理。而我設置的參數與Implementation 1.4相同，{n\_Fear, nResBlock, nEpoch} = {64, 8, 100}。結果如下：

|  |  |  |
| --- | --- | --- |
| Original LR image | 原本Implementation的model產出(長寬各放大四倍)  {n\_Fear, nResBlock, nEpoch} = {64, 8, 100} | 新訓練資料集的model(長寬各放大四倍)  {n\_Fear, nResBlock, nEpoch} = {64, 8, 100} |
|  |  |  |
|  |  |  |

可看出新訓練資料集的model在超參數不變的情況下，效果都比原本的model好，所以驗證了資料集對model performance有很大影響。新model產出的HR image雖可看出是小狗群與斑馬線，但還是稍顯模糊，我推斷是我model餵入的訓練圖片過少(模仿Implementation只餵入兩張)以及訓練圖片本身解析度不高所導致。

* Experiment c. (該專案程式碼位置位於Experiment c/資料夾)
  + Assumption

課堂中老師有提到super resolution task中最難做到的就是一些整齊的線條紋理或是文字，因為這種類型的圖片只要解析度不高，很容易被人眼察覺。所以我打算挑戰看看處理文字方面的照片，看效果如何。下表分別是我採用的training data & validation data & testing data。(做法都是將手機拍攝之照片做opencv.resize，HR image是將原始長寬各除以2，LR image是將原始長寬各除以8，故跟implementation part一樣是實作4倍長寬縮放的super resolution task)

|  |  |  |
| --- | --- | --- |
|  | HR image | LR image |
| Training data 1 |  |  |
| Training data 2 |  |  |
| Validation data 1 |  |  |
| Testing data 1 |  |  |

上面所有照片都從同一本書所拍攝的，故性質雷同，預期testing data效果會不錯，並且推測若learning rate設置妥當，每次更新的model其validation set之avg. psnr會越來越大。理論上最後一次更新的model是最後要用來做testing的模型。

* Justification

我的超參數設置與implementation 1.4相同，{n\_Fear, nResBlock, nEpoch} = {64, 8, 100}。而我用這100個model來檢驗看看哪個model所產出的HR image與reference data計算PSNR值最高，結果竟然是第5次update的model表現最好，我知道這樣的方法是作弊的方式，不符合model training rule，正確的方式應該是在update model的過程中，看哪次對validation data的PSNR值最佳就選用該model來做testing，但是我在訓練過程中忘記紀錄每次update model對validation data的PSNR值，這是我這次實驗的瑕疵。不過這邊我就將錯就錯，將第五次更新的model(最佳的模型)其餵入testing data之output來做比較：

|  |  |  |
| --- | --- | --- |
| LR image | Model output | Reference(HR image) |
|  |  |  |

Model output與HR image計算psnr值為24.0884，可以發現一些簡單筆畫的國字與數字表現得還不錯，但是較複雜筆畫的國字其表現就不太好，另外像是原始圖片上的紅框處，經過model處理過後變成灰框，原因是因為training dataset中沒有紅色的部分，都只有黑色白色居多，所以model沒有學到紅色方面的information。