

Single Image Reflection Removal (SIRR)

CV Final Project

7113056013

游宸睿

2024/12

Outline

1. Introduction
2. Methodology
3. Results
4. Conclusion
5. References
6. Code Repository

1. Introduction

1. Introduction

- **What is SIRR?**

在日常攝影中，不必要的光學反射問題十分常見，例如拍攝玻璃反射或鏡頭耀光等情況。Single Image Reflection Removal (SIRR) 是一種影像處理方法，致力於從單一圖像中去除這些不必要的光學反射，以解決這一問題。

此 Project 的目標主要在於評估和比較不同的 SIRR 模型，並測試它們在實際應用中的效果。評估內容涵蓋了對多個 Dataset 進行訓練和測試，以及透過 PSNR（Peak Signal-to-Noise Ratio）、SSIM（Structural Similarity Index）、LPIPS（Learned Perceptual Image Patch Similarity）等指標進行比較和分析。

1. Introduction

- 我總共實作了以下三篇論文，並進行效果分析

[1] Single Image Reflection Separation with Perceptual Losses (CVPR 2018)

- Zhang, Xuaner, Ren Ng, and Qifeng Chen. "Single image reflection separation with perceptual losses." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4786-4794. 2018.

[2] Single Image Reflection Removal Exploiting Misaligned Training Data and Network Enhancements (CVPR 2019)

- Wei, Kaixuan, Jiaolong Yang, Ying Fu, David Wipf, and Hua Huang. "Single image reflection removal exploiting misaligned training data and network enhancements." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8178-8187. 2019.

[3] Single Image Reflection Removal through Cascaded Refinement (CVPR 2020)

- Wei, Kaixuan, Jiaolong Yang, Ying Fu, David Wipf, and Hua Huang. "Single image reflection removal exploiting misaligned training data and network enhancements." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8178-8187. 2019.

由於論文名字都很長，後續的簡報中，我會直接用 [1]、[2]、[3] 代表這三篇論文。

[1] Single Image Reflection Separation with Perceptual Losses (CVPR 2018)

這篇論文提出一種模型，透過 single image 將 Transmission layer 與 Reflection layer 分離。不同於過去依賴運動軌跡的方法，改用 predefined priors 來定位圖片中的 minimum edges and corners，進而進行兩層 layers 的分離。

模型訓練需要包含透射層與反射層的資料集，採用了兩種不同的 perceptual loss 評估效能，強調分離後兩層的獨立性。此外，該模型也應用於光斑去除與霧霾消除等影像增強任務。

本論文參考了《*A Generic Deep Architecture for Single Image Reflection Removal and Image Smoothing*》，但使用了四種不同的 loss function：

1. **Feature Loss**：基於預測層與 VGG-19 預訓練網路的五個卷積層計算。
2. **Adversarial Loss**：使用 Conditional GAN 確保分離後的層自然無異常色彩或殘留反射。
3. **Exclusion Loss**：利用邊緣不重疊特性，更完整地分離兩層。
4. **Low-level Loss**：針對低階特徵進行優化。

總結來說，該模型結合四種 loss function 與 gradient descent，實現單張圖片中 Transmission layer 和 Reflection layer 的有效分離。

[2] Single Image Reflection Removal Exploiting Misaligned Training Data and Network Enhancements (CVPR 2019)

這篇論文在《*A Generic Deep Architecture for Single Image Reflection Removal and Image Smoothing*》的基礎上進行改進，主要包括三點：

1. 簡化了 **basic residual block**：移除 Batch Normalization (BN) 層，發現其反而會降低性能，並提出不含 BN 層的 **BaseNet** 架構。
2. 擴大 **network** 容量：將 **network** 寬度從 64 擴展到 256 個 feature maps 以提升性能。
3. 將 **augmented network** 的 **features** 進行串聯：利用增強網路的特徵進行串聯處理。

該研究解決了兩大挑戰：

1. 從 Single Image 中進行 Reflection Removal 不穩定。
2. 現實生活中獲取 Transmission layer 的 label 相當困難，導致 Training data 稀少。

為此，論文提出一種強調 **Sensitive to Contextual Information** 的神經網路架構，增強處理不確定性的能力，並使用 **Multi-scale Spatial Context** 的方式強化影像的 features，判斷應保留與丟棄的部分。此外，透過 Generated Synthetic Dataset 增加訓練數據，有效克服資料不足的問題。

總體而言，這篇論文提出了 BaseNet 並加入了 **Contextual Information**、**Pyramid Pooling module** 等方法，全面優化了模型效能。

[3] Single Image Reflection Removal through Cascaded Refinement (CVPR 2020)

這篇論文受 《*Hidden community detection in social networks*》 和 《*Revealing Multiple Layers of Hidden Community Structure in Networks*》 的概念啟發，將 Transmission layer 和 Reflection layer 對應到主要群集 (dominant communities) 與次要群集 (weak communities / hidden communities)，並提出了新的網路架構 **ICBLN architecture (Cascaded Network)**。

ICBLN 包含兩個子網路：

1. **Transmission Generative Sub-network (G_T)**：學習主要群集。
2. **Reflection Generative Sub-network (G_R)**：專注於次要群集。

這兩個網路共享資訊，但學習不同參數，逐步優化輸出結果。ICBLN 結合 **LSTM** 和多層卷積結構，並在 encoder 和 decoder 中加入 skip layers，防止輸出模糊。此外，**Residual Reconstruction Loss** 在每個階段提供監督，縮小預測與真實輸出間的誤差。

另外，convolutional LSTM unit 與其他單元不同之處在於擁有四個 gate，分別為 input gate、forget gate、output gate、cell state。他們發現，隨著 iteration 推移， G_R 會逐步改善，間接對 G_T 也有明顯的幫助。

1. Introduction

- 我使用了以下幾個資料集
 - Training dataset:
 - 7,643 images from [Pascal VOC dataset](http://host.robots.ox.ac.uk/pascal/VOC/) (<http://host.robots.ox.ac.uk/pascal/VOC/>)
 - 90 real-world training images from [Berkeley real dataset](https://github.com/ceciliavision/perceptual-reflection-removal) (<https://github.com/ceciliavision/perceptual-reflection-removal>)
 - Testing dataset
 - 100 synthetic testing images from [CEILNet dataset](https://github.com/fqnchina/CEILNet/tree/master/testdata_reflection_synthetic_table2) (https://github.com/fqnchina/CEILNet/tree/master/testdata_reflection_synthetic_table2)
 - 20 real testing images from [Berkeley real dataset](https://github.com/ceciliavision/perceptual-reflection-removal) (<https://github.com/ceciliavision/perceptual-reflection-removal>)
 - 454 real testing pairs from [SIR² dataset](https://sir2data.github.io) (<https://sir2data.github.io>)
 - containing three subsets (i.e., Objects (200), Postcard (199), Wild (55))
 - 20 testing images from [Nature](https://github.com/JHL-HUST/IBCLN) (<https://github.com/JHL-HUST/IBCLN>)

2. Methodology

使用設備&版本

- **CPU:** AMD R5-3600
- **RAM:** 32GB ddr4
- **GPU:** GeForce RTX3070 (8GB gddr6)
- **CUDA 版本:** 11.8 (我用CUDA 12.x TensorFlow 會抓不到 GPU 不知道為啥)
- **cuDNN 版本:** 8.9.6 for CUDA 11.x

2.1 Dataset

2.1.1 Dataset 架構

SIRR的實作中，Dataset主要劃分為三個layer：

- **Transmission layer:** 這一層保留了原始圖片中期望保留的內容。
- **Reflection layer:** 此層涵蓋了由光線反射所造成的影像，例如由玻璃或水引起的反射，是SIRR中希望去除的部分。
- **Blended:** 綜合了 Transmission layer 和 Reflection layer 的所有元素，包括需要保留的內容和希望去除的反射內容。

2.1.1 Dataset 架構

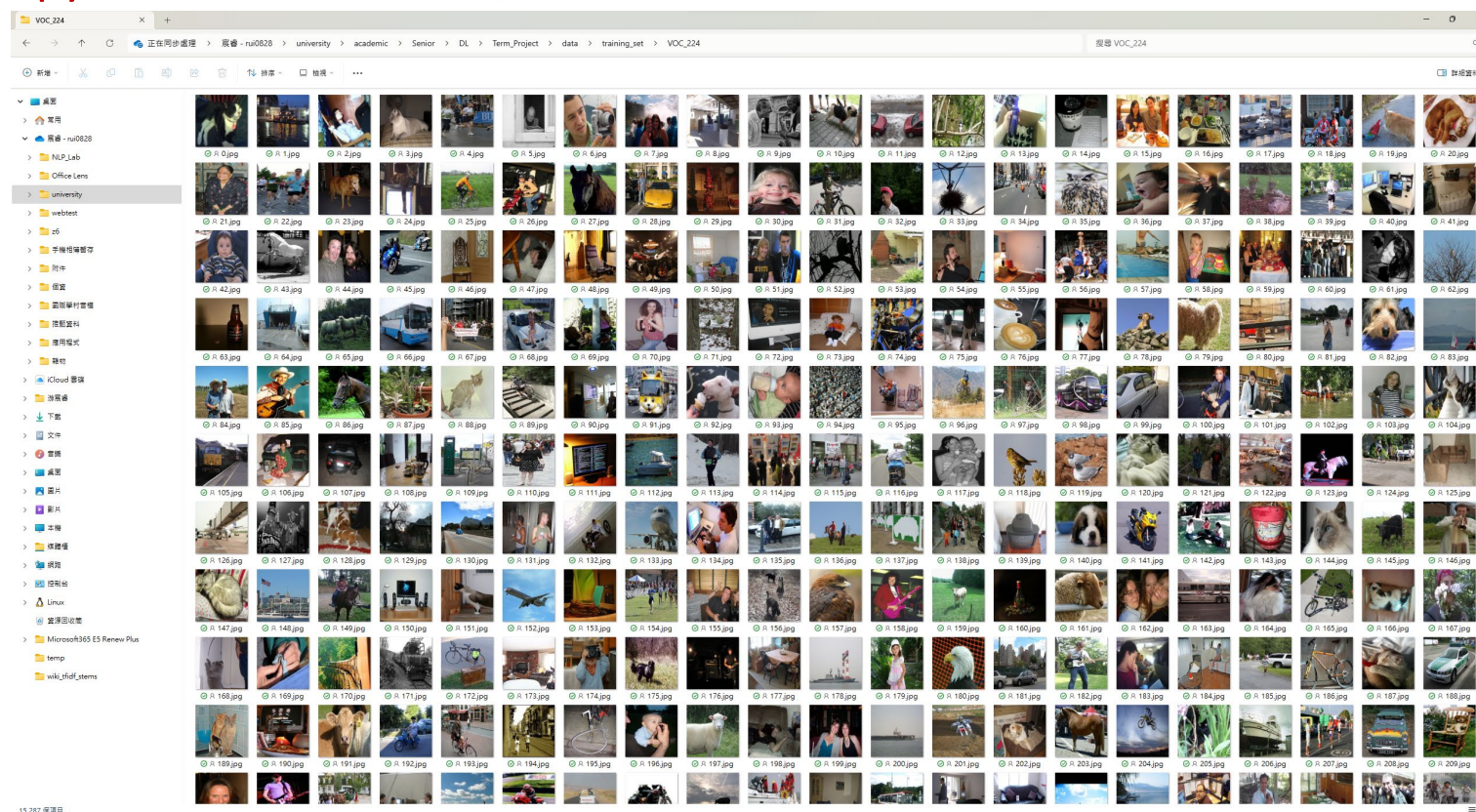
訓練模型時，Dataset包含兩種類型的資料：

- **Real Data:** 從實際拍攝的照片中獲得的資料，包含 Blended 和 Transmission layer 的資料（缺少 Reflection layer）。
- **Synthetic Data:** 透過合成圖片模擬反射效果的資料，需要 Transmission layer 和 Reflection layer 的資料來做合成。

而模型測試階段，無論是 Real Data 還是 Synthetic Data，均必須具備 Blended 和 Transmission layer，以便計算 PSNR、SSIM、LPIPS 等指標，用於評估模型效能。

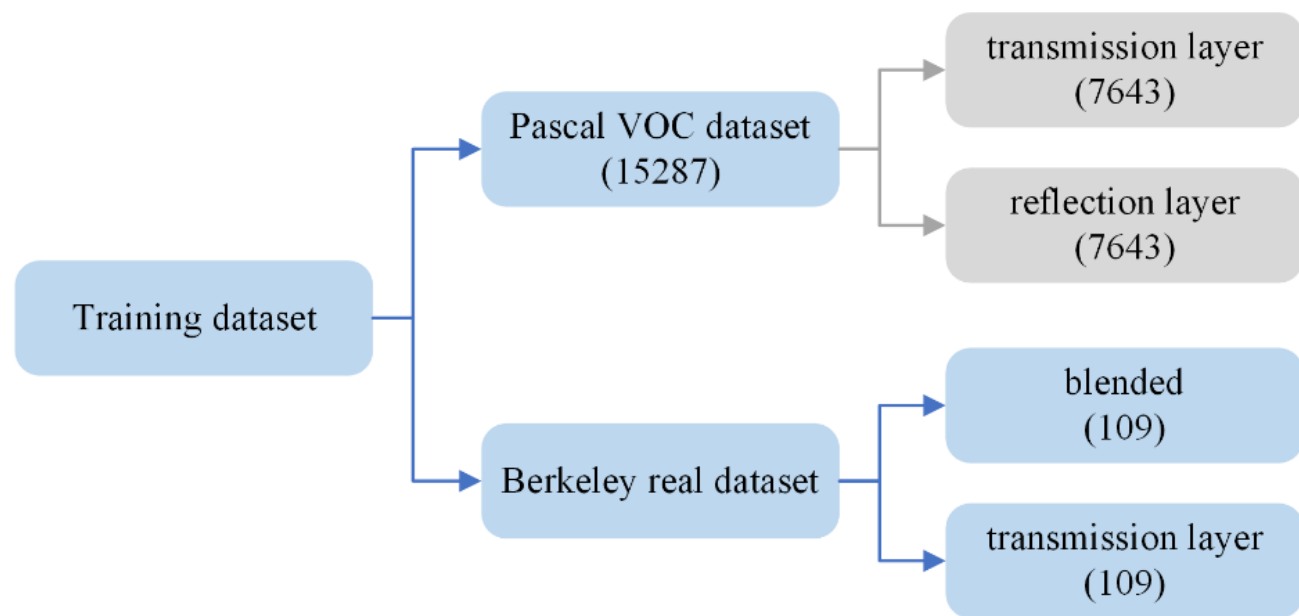
2.1.2 整理 Training dataset

根據論文 Source Code 的要求，需要抓出 Pascal VOC dataset 中 provided ids 的圖片並裁切至 224x224 的大小，我使用 Python 的 Pillow 套件來實作批量的裁切，程式碼呈現於附件的 [other code/crop VOC224.py](#) 中，結果如下圖所示。



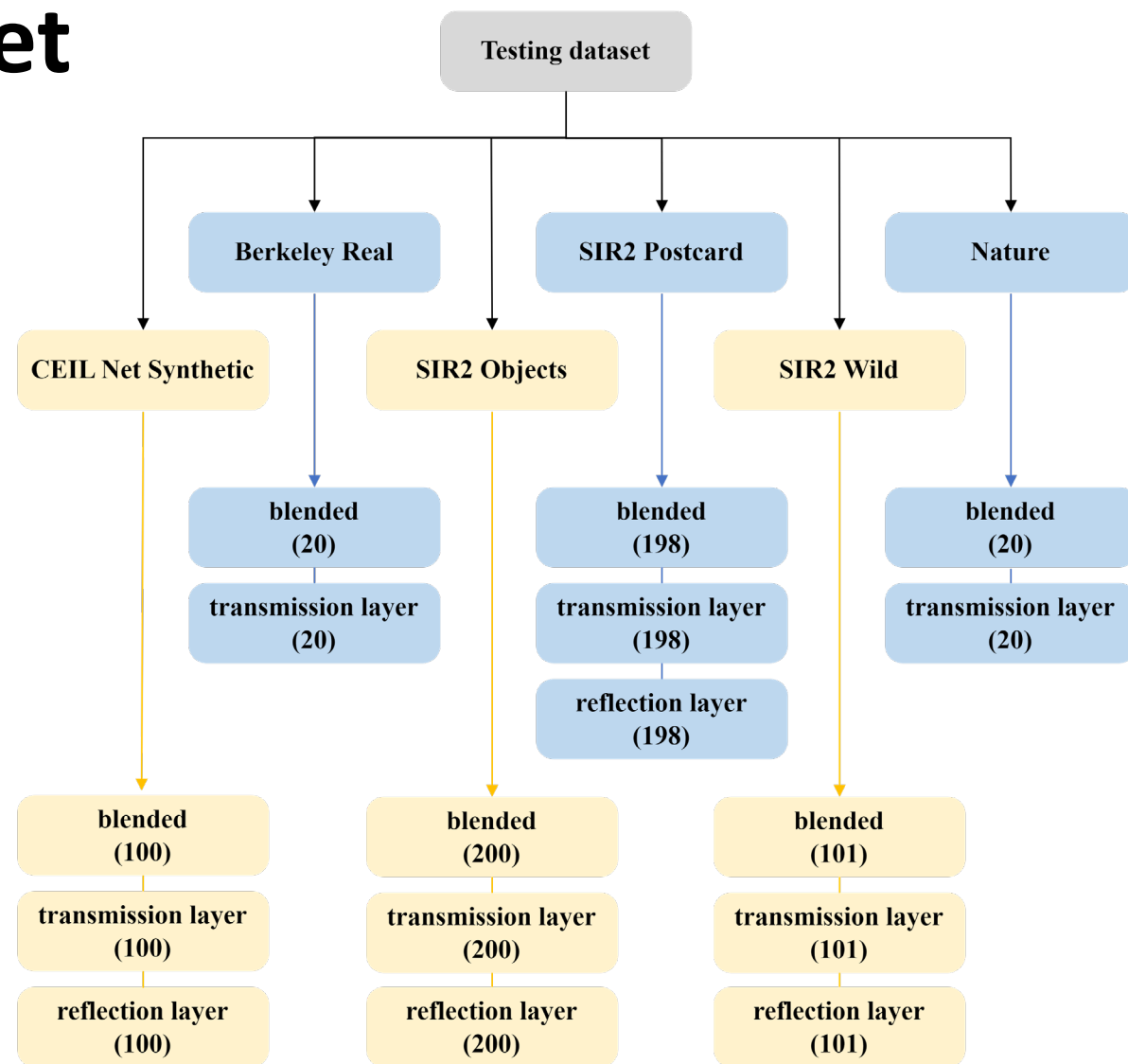
2.1.2 整理 Training dataset

整理 Pascal VOC dataset 時，我沒有直接將其分為 Transmission layer 和 Reflection layer。這是因為某些模型（例如 Paper [2]）的 Code 需要將整份資料放在同一個資料夾進行輸入，訓練過程才會分層。相對地，也有些模型（例如 Paper [3]）需要將這兩個 layer 分開，因此需要分為兩個資料夾進行單獨輸入。我在實作 Paper [2] 時，便會進一步將 Pascal VOC dataset 分層。至於 Berkeley real dataset 本身沒有太大問題，因此無需特別整理。下圖呈現了所有訓練資料集的內容架構。



2.1.3 整理 Testing dataset

整理 Testing dataset 的過程中，主要挑戰來自於資料儲存格式的差異。為了便於後續實作的資料使用，我將資料各自分類並存放至 Blended、Transmission layer、Reflection layer 三個資料夾中，並統一了檔案命名方式。Testing dataset 的整體結構如右圖所示。



2.2 第 [1] 篇 Paper 實作

[1] Single Image Reflection Separation with Perceptual Losses (CVPR 2018)

Source Code link : <https://github.com/ceciliavision/perceptual-reflection-removal>

使用環境&套件版本：Python: 3.10.9、TensorFlow: 2.10.1 (此 Model 指定)

2.2.1 建立 Training 資料集

此 model 的 training synthetic data 需要將 reflection layer 跟 transmission layer 分開並放入指定資料夾，因此我將已裁切過的 Pascal VOC dataset 直接平均分配至兩個資料夾中，以便 Training 使用，實作的程式在附件的 `model_1_code/run.ipynb` 中的 `# Create synthetic_data` 部分中展示。

2.1.2 Training

- **Code:** `model_1_code/run.ipynb` 中的 `# Training` 部分

```
!python main.py --data_syn_dir data/synthetic_data_100/ --data_real_dir  
data/real_data/ > training_output.txt
```

遇到問題: GPU Out of Memory

資料集中總共有 7643 筆 cropped VOC dataset 和 109 筆 Berkely real dataset，故總共 7752 筆 training images。跑約 25 分鐘後於第 1 個 epoch 的第 1384 個 cnt 跳 Out of Memory。經評估後，即使沒有 Out of Memory，1 個 epoch 大約需要 140 分鐘，計畫跑 100 個 epoch 大約需要連續跑 10 天。由於硬體上的限制，只好刪減 Training Data。

```
loaded pre-trained\model.ckpt
```

```
[i] Total 7752 training images, first path of real image is data/real_data/blended/1.jpg.
```

```
1488  iter: 1 1382 || D: 0.76 || G: 1.28 0.76 || all: 23.63 || loss: 43.41 0.00 || mean: 117.75 11.24 || time: 0.44
1489  iter: 1 1383 || D: 0.68 || G: 0.58 0.76 || all: 23.62 || loss: 76.80 9.94 || mean: 117.72 11.24 || time: 0.31
1490  iter: 1 1384 || D: 1.03 || G: 0.39 0.76 || all: 23.63 || loss: 119.71 11.80 || mean: 117.73 11.24 || time: 0.31
```

```
Profiling failure on CUDNN engine 1#TC: RESOURCE_EXHAUSTED: Out of memory while trying to allocate 16978192 bytes.
Profiling failure on CUDNN engine 1: RESOURCE_EXHAUSTED: Out of memory while trying to allocate 16808192 bytes.
Profiling failure on CUDNN engine 0#TC: RESOURCE_EXHAUSTED: Out of memory while trying to allocate 16777216 bytes.
Profiling failure on CUDNN engine 0: RESOURCE_EXHAUSTED: Out of memory while trying to allocate 16777216 bytes.
```

遇到問題: GPU Out of Memory

經評估後，將 cropped VOC dataset 刪減至100筆，Berkely real dataset 維持原本的109筆，重新跑一次Training，仍然在第19個 epoch 時 Out of Memory。

```
1953   iter: 19 74 || D: 0.73 || G: 0.85 0.81 || all: 20.89 || loss: 85.33 3.04 || mean: 104.13 10.90 || time: 0.24
1954   iter: 19 75 || D: 0.72 || G: 0.45 0.80 || all: 20.91 || loss: 123.85 10.68 || mean: 104.23 10.91 || time: 0.21
1955   iter: 19 76 || D: 0.70 || G: 0.71 0.80 || all: 21.02 || loss: 131.95 9.73 || mean: 104.79 10.90 || time: 0.25

tensorflow:From C:\Users\asd47\AppData\Roaming\Python\Python310\site-packages\tensorflow\python\training\saver.py:1064: remove_checkpoint (from tensorflow.python.checkpoint.checkpoint_managers)
ions for updating:
dard file APIs to delete files with this prefix.
03 02:22:08.023682: W tensorflow/core/common_runtime/bfc_allocator.cc:479] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.98GiB (rounded to 2126513408)requested by op concat_4
ause is memory fragmentation maybe the environment variable 'TF_GPU_ALLOCATOR=cuda_malloc_async' will improve the situation.
allocation summary follows.
allocation summary follows.
```

但因為每個 epoch 都有 checkpoint，因此可以從中斷的 epoch 開始重跑就好，後續大約每跑 2~3 個 epoch 就會 Out of Memory 一次，最後跑到40個 epoch 停止，無法跑到預計的100層。

2.2.3 Testing

- **Code:** `model_1_code/run.ipynb` 中的 `# Testing` 部分

```
!python main.py --task "pre-trained\0040" --is_training 0
```

遇到問題: GPU Out of Memory (Berkeley Real)

使用 Berkeley real dataset (20 real images) 做為測試資料集時，跑下去一張都還沒完成就會跳 Out of Memory，其餘資料集皆有順利跑完。

```
6
) ran out of memory trying to allocate 1.38GiB with freed_by_count=0. The caller indicates that
) ran out of memory trying to allocate 2.74GiB with freed_by_count=0. The caller indicates that
) ran out of memory trying to allocate 1.38GiB with freed_by_count=0. The caller indicates that
) ran out of memory trying to allocate 1.38GiB with freed_by_count=0. The caller indicates that
) ran out of memory trying to allocate 365.56MiB with freed_by_count=0. The caller indicates tha
volution redzone checking; skipping this check. This is benign and only means that we won't che
) ran out of memory trying to allocate 541.46MiB with freed_by_count=0. The caller indicates tha
) ran out of memory trying to allocate 4.15GiB (rounded to 4460274176) requested by op concat_1
```


遇到問題: GPU Out of Memory (Berkeley Real)

- 嘗試解決 (修改Source code 的 main.py) :

1. 跑之前先清理GPU內存([ref1](#))，Code如下:

```
13  from numba import cuda
14  cuda.get_current_device().reset()
```

2. Enable Dynamic Memory Allocation ([ref2](#)) 並提高預設的Memory allocate量([ref3](#))。

```
349 ##### Session #####
350 gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=1)
351 config=tf.ConfigProto(gpu_options=gpu_options)
352 config.gpu_options.allow_growth = True
353 sess = tf.Session(config=config)
354 # sess = tf.Session()
```

References:

ref1: [深度学习训练时GPU相关配置和问题 tf gpu allocator=cuda malloc async-CSDN博客](#)

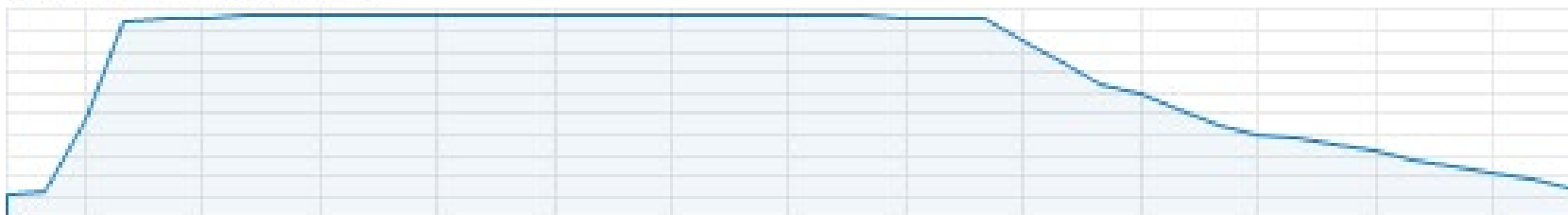
Ref2: [python - Tensorflow doesn't allocate full GPU memory - Stack Overflow](#)

Ref3: [Solving Out Of Memory \(OOM\) Errors on Keras and Tensorflow Running on the GPU \(linkedin.com\)](#)

遇到問題: GPU Out of Memory (Berkeley Real)







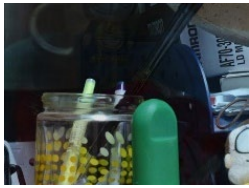

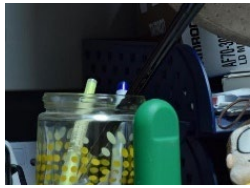

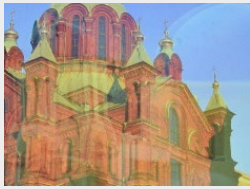


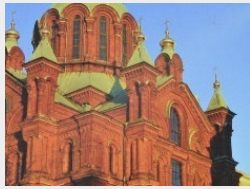






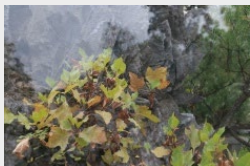
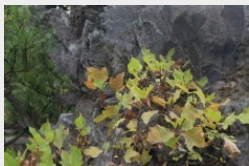
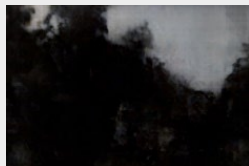
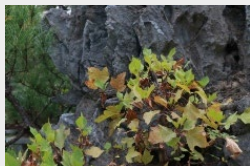
- **結果：**仍然無法解決那份資料集會 Out of Memory 的問題(如下圖，滿載後程式停止)，其原因推測是該資料集之圖片畫質較高、檔案較大。執行後 GPU 顯存滿載，滿載一小段時間後跳 Out of Memory。Source Code 中目前沒有找到其他能夠調整優化的地方，近期也沒有升級硬體設備的打算，因此這個 Model 的實作只好止於此，無法 Berkeley Real Dataset 的 Testing 成果，其餘資料集皆有順利跑完。

專屬 GPU 記憶體使用量



2.2.4 Image Results

右圖展示了各個資料集其中一張圖片的效果，詳細的評估將於 **3. Results** 章節中呈現。

	Input (blended)	Output (Transmission layer)	Output (Reflection layer)	Ground Truth (Transmission layer)	Ground Truth (Reflection layer)
CEIL Net Synthetic					
SIR ² Objects					
SIR ² Postcard					
SIR ² Wild					
Nature					Not Provided

2.2.5 計算 PSNR、SSIM、LPIPS

- **Note:** 這三個評估指標會在 Result 章節詳細介紹，這裡僅說明實作過程。

此 model 之 Testing 輸出為 Transmission layer 之圖片檔，故需要另外寫一段程式來計算 PSNR、SSIM、LPIPS 指標，才能於**3. Result** 章節中與其他模型互相比較，我使用 Python 實作，PSNR 透過 Numpy 套件搭配其公式計；SSIM 則透過 OpenCV 套件參考網路上的資料 [\(ref\)](#) 實作；LPIPS 則直接套用 Lpips 套件使用，程式碼呈現於 [other_code/eval_paper_1.py](#) 中。

2.3 第 [2] 篇 Paper 實作

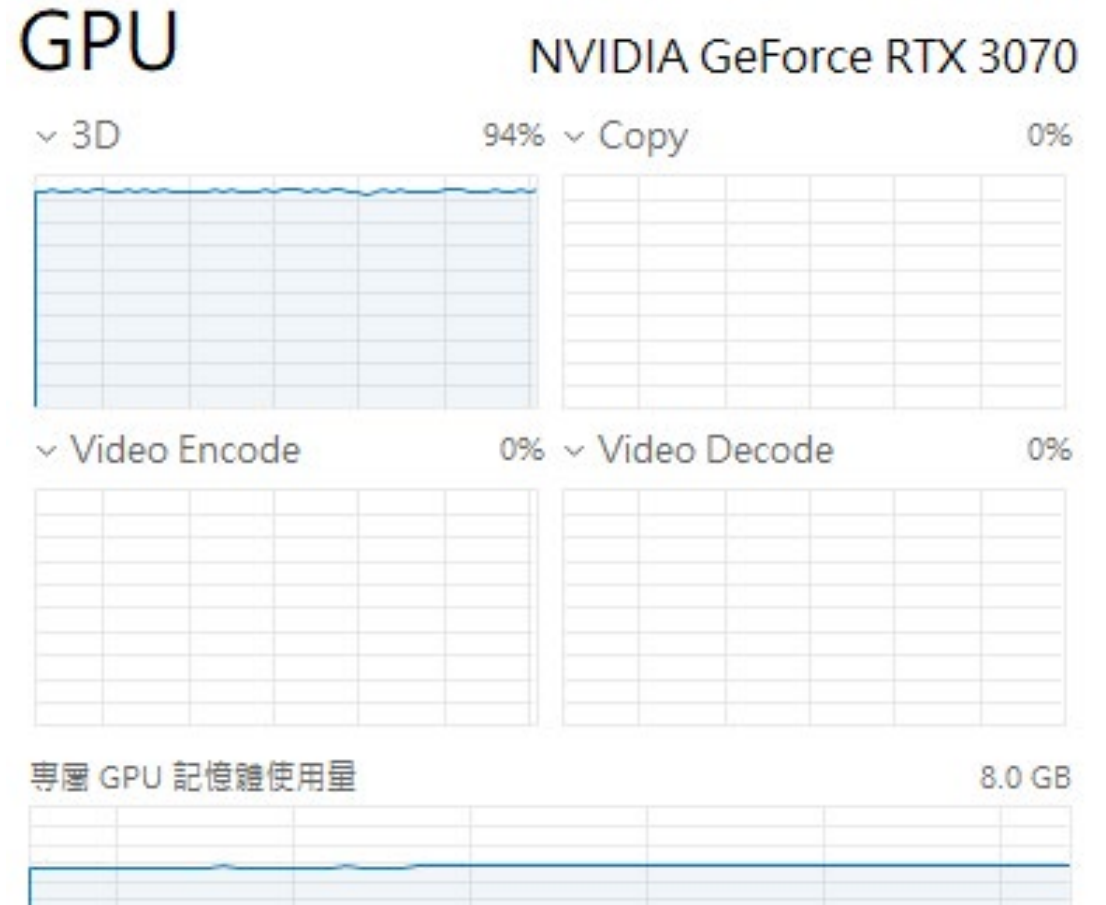
[2] Single Image Reflection Removal Exploiting Misaligned Training Data and Network Enhancements (CVPR 2019)

Source Code link: <https://github.com/Vandermode/ERRNet>

使用環境&套件版本: Python 3.8.18

2.3.1 Training

- **Code:** `model_2_code/run.ipynb` 中的 `# Training` 部分
Train 此 Model 時，我發現此 Model 是使用 3D 運算再跑 Training，而未使用大量使用顯存 (如右圖)，因此便嘗試了將全部的 7643 筆 cropped VOC dataset 和 109 筆 Berkely real dataset 匯入進行 Training，希望能跑起來。結果發現其也並非完全不佔用顯存，仍然在第 6 個 epoch 時 Out of Memory。



2.3.1 Training

最終僅完成了 5 個 epoch (在第 6 個 epoch 中 OOM)，仍然是硬體上的限制無法優化。

因此我也只能使用第 5 個 epoch 的 checkpoint 來跑 Testing 了。

```
torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 2.73 GiB. GPU 0 has a total capacity of 8.00 GiB of which 0 bytes is free. Of the allocated memory 4.55 GiB is allocated by PyTorch, and 907.32 MiB is reserved by PyTorch but unallocated. If reserved but unallocated memory is large try setting max_split_size_mb to avoid fragmentation. See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF
```


2.3.2 Testing

此 Model 的 Testing Code 已包含了 PSNR 和 SSIM 指標，因此我僅需要加上 LPIPS 並套入指定 Testing set，便能完成評估。

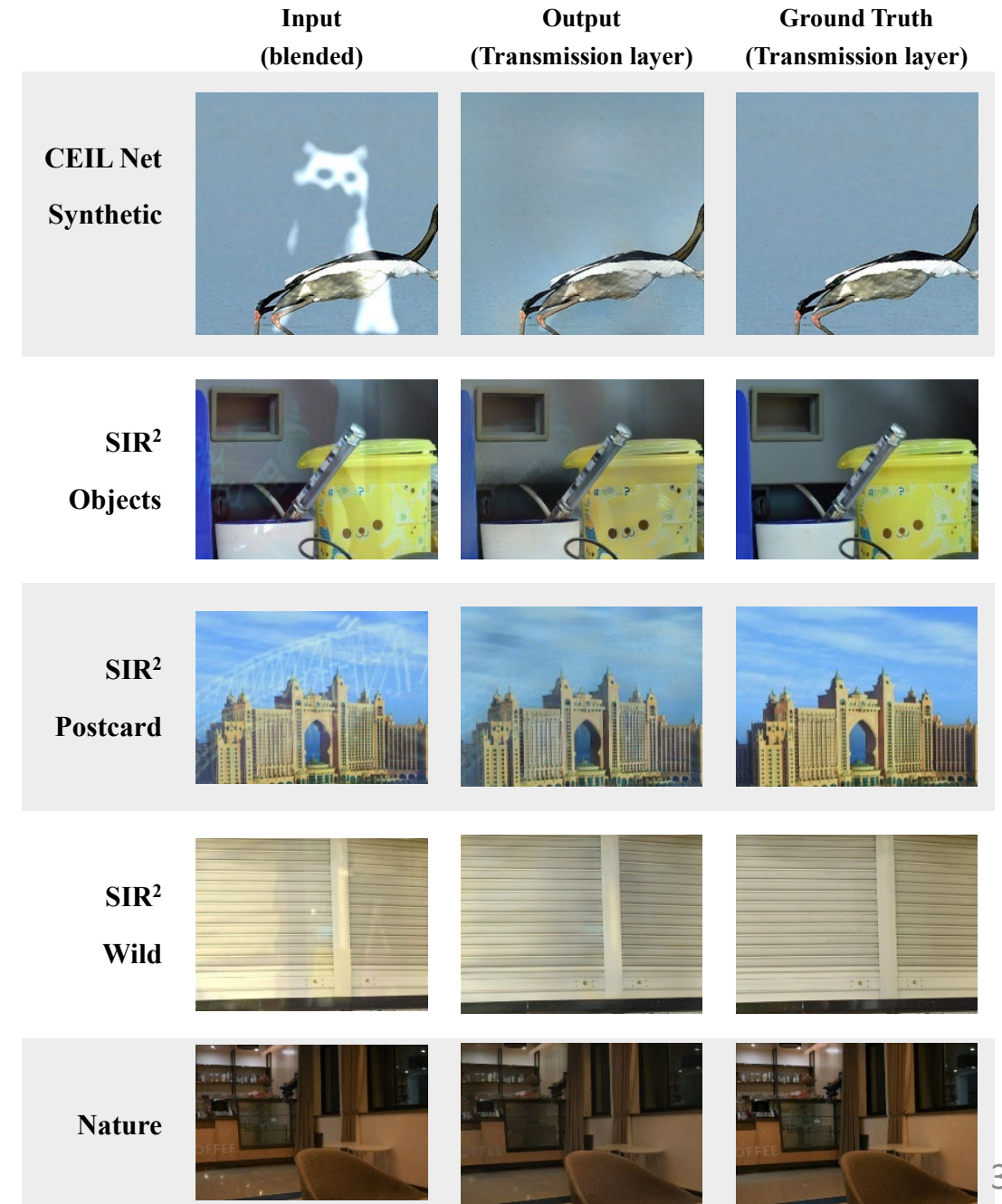
而 LPIPS 的實作程式碼已在 Paper [1] 的實作過程介紹過，這裡就不多加贅述。

測試 Berkeley Real Dataset 時，同樣有 Out of Memory 的情況發生，因此無法呈現該 Dataset 的 Testing result。

2.3.3 Image Results

右圖展示了 Model output 與 Ground Truth 的視覺比較圖，但是此 Model 只有輸出 Transmission layer，因此沒有 Reflection layer 的比較。

詳細的評估與模型比較將於 **3. Results** 章節中呈現。



2.4 第 [3] 篇 Paper 實作

[3] Single Image Reflection Removal through Cascaded Refinement (CVPR 2020)

Source Code link: <https://github.com/JHL-HUST/IBCLN>

使用環境&套件版本: Python 3.11.2

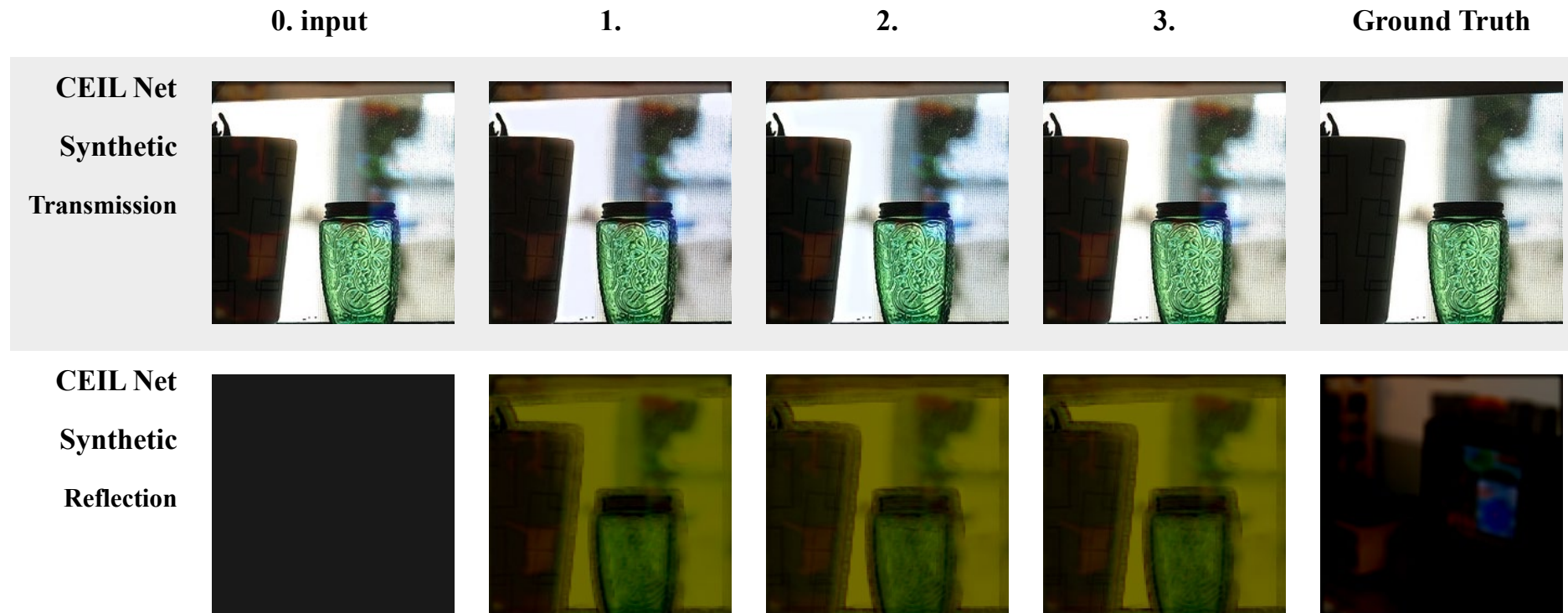
2.4.1 Training

在實作 Paper [3] 時，一樣使用了整份的 7643 筆 cropped VOC dataset 和 109 筆 Berkely real dataset 匯入進行 Training，於第 14 個 epoch 訓練時 Out of Memory，因此我會用第 13 個 epoch 的 model 進行 Testing。

```
...  
File "c:\Users\asd47\AppData\Local\Programs\Python\Python311\Lib\site-packages\torch\nn  
    return F.pad(input, self.padding, 'reflect')  
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  
torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 700.00 MiB. GPU 0 has
```

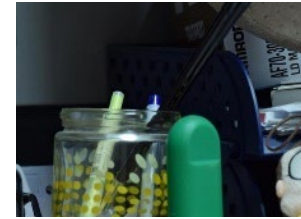
2.4.2 Testing

此 Model 的 Testing Code 只輸出圖片檔，因此也是需要另外寫程式來計算三個評估指標，在 Paper [1] 已經詳細說明了實作過程便不再加以說明。比較特別的是，這個 model 的 Transmission layer 和 Reflection layer 會互相分離三次，並輸出三份結果，因此可以看到他在每一組測試資料的三層結果，同樣的 Berkeley Real 仍然在測試中 Out of Memory，其餘資料將呈現於下表，由於表格較大，分多頁呈現。



2.4.2 Testing

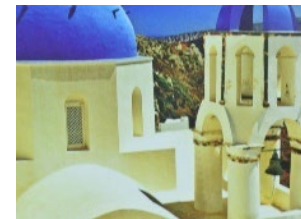
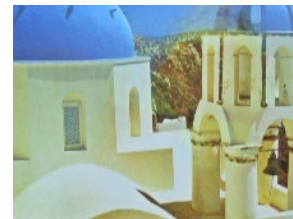
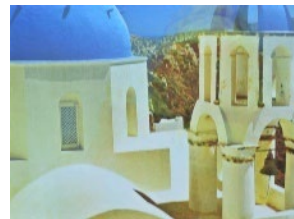
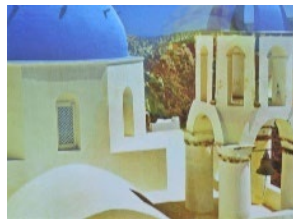
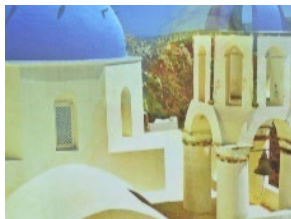
SIR²
Objects
Transmission



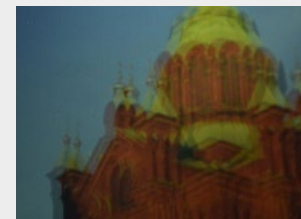
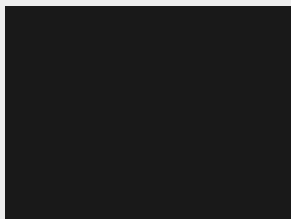
SIR²
Objects
Reflection



SIR²
Postcard
Transmission



SIR²
Postcard
Reflection



2.4.2 Testing

**SIR² Wild
Transmission**



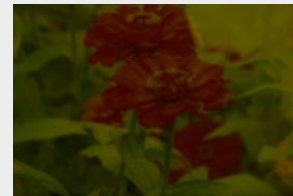
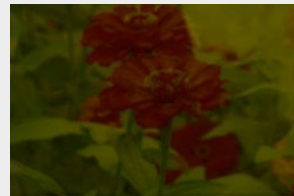
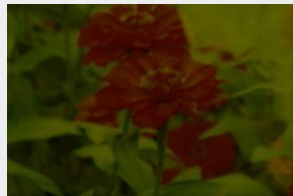
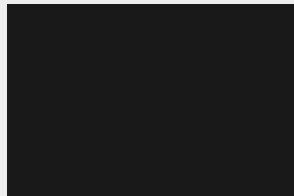
**SIR² Wild
Reflection**



**Nature
Transmission**



**Nature
Reflection**



Not
Provided

3. Results

3.1 評估指標

- **PNSR (Peak Signal-to-Noise Ratio)**

- 透過計算兩張圖片像素值的均方差、色彩通量和原始圖像中像素值的最大可能範圍來評估圖像品質。較高的 PNSR 值表示原始圖像 (Ground Truth) 和處理後圖像 (Model Output) 之間的差異較小，表示較好的圖像品質。

- **SSIM (Structural Similarity Index)**

- 考慮了亮度、對比度和結構這三個因素。SSIM 的數值範圍從 -1 到 1，兩個完全相同的影像的 SSIM 值為 1，指指標更好地反映了人類視覺感受。

- **LPIPS (Learned Perceptual Image Patch Similarity)**

- 是唯一使用深度模型進行評估的指標，它使用了 SRGAN 計算 perceptual loss，評估兩張圖片之間特徵的差異。較低的 LPIPS 值代表兩張圖片越相似。

3.2 Models 數據比較

本 Project 總共實作了 3 篇論文，由於GPU硬體限制，沒有辦法用所有 Dataset 去 Train 完期望的 epoch 數量，以下是我實作各 model 訓練的參數：

- **Paper 1:** cropped VOC dataset 100筆、Berkely real dataset 109筆、Epoch = 40
- **Paper 2:** cropped VOC dataset 7643筆、Berkely real dataset 109筆、Epoch = 5
- **Paper 3:** cropped VOC dataset 7643筆、Berkely real dataset 109筆、Epoch = 13

3.2 Models 數據比較

Dataset	Index	Paper [1]	Paper [2]	Paper [3]
CEIL Net Synthetic (100)	PSNR↑	29.32	20.66	28.71
	SSIM↑	0.89	0.84	0.85
	LPIPS↓	0.17	0.15	0.27
Berkeley Real (20)	PSNR↑	OOM	OOM	OOM
	SSIM↑	OOM	OOM	OOM
	LPIPS↓	OOM	OOM	OOM
SIR ² Objects (200)	PSNR↑	29.86	20.30	29.84
	SSIM↑	0.94	0.86	0.94
	LPIPS↓	0.11	0.14	0.11
SIR ² Postcard (199)	PSNR↑	28.35	17.90	29.44
	SSIM↑	0.95	0.81	0.96
	LPIPS↓	0.17	0.23	0.17
SIR ² Wild (55)	PSNR↑	30.18	22.03	30.19
	SSIM↑	0.95	0.87	0.95
	LPIPS↓	0.12	0.14	0.12
Nature (20)	PSNR↑	29.34	19.50	28.99
	SSIM↑	0.87	0.72	0.86
	LPIPS↓	0.18	0.25	0.17

Note that “OOM” means GPU out of memory when testing.

3.3 分析

數據層面來說，Paper [1] 普遍的 PSNR 和 SSIM 分數較佳，但在某些資料集 (例如 Nature) 上的 SSIM 略低、LPIPS 得分略高，可能會與現實生活中肉眼視覺上有感受差異。Paper [2] 在大多數資料集上的 PSNR、SSIM 和 LPIPS 得分都低於其他模型，顯示其整體性能較差，在 SIR² Postcard 和 Nature 更為明顯。Paper [3] 在大部分資料集上表現穩定，尤其在 SIR² Postcard 和 SIR² Wild 資料集上的 SSIM 得分相當高，在 CEIL Net Synthetic 資料集上的 LPIPS 特別高。

但若考量到 Training 資料量和 Epoch 數，Paper [1] 使用較少的資料量進行訓練，而訓練 Epoch 數較多，Paper [2] 和 Paper [3] 使用較多的資料量進行訓練，但訓練 Epoch 數較少。因此，Paper [2] 的表現相對較差，可能是由於 Epoch 數不足等因素導致，並不能因此便斷定 Paper [2] 的方法效能較差。

總結來說，三個 Model 在我的實作過程中，都各自有不同的缺陷 (Training set 較少或 Epoch 數不足)，因此我不認為我的數據結果有很大的可信度。由於時間關係，我也沒有進一步去測試同樣以較少的資料集和較小的 Epoch 數之效能比較。但就我的已完成的實驗數據而言，Paper [1] 表現較為優異；Paper [3] 表現中等；Paper [2] 則整體表現較差，不過三篇都沒有訓練到預期的 Epoch 數，這可能需要更好的設備資源以及更長的時間才能完成。

4. Conclusion

這次的Project中，我深入了解了 SIRR 的原理以及各種不同的方法，以及 PSNR、SSIM 和 LPIPS 這些評估指標在幹麻。實作的過程中，我花了相當多的時間來架環境和 Debug，中途也解決了不少問題。雖然三個模型都有成功動起來，但最終由於硬體設備的限制，未能順利完整的訓練這些模型，有些遺憾。我相信在資源允許的情況下，必能夠更完整地進行所有的訓練過程，並且做出更全面詳盡的分析報告。

總之，這次 SIRR Project 讓我更深入地學習到許多相關知識，也讓我意識到硬體資源對於項目的成果的重要性，這部分有點可惜。

5. References

- [1] Zhang, Xuaner, Ren Ng, and Qifeng Chen. "Single image reflection separation with perceptual losses." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4786-4794. 2018.
- [2] Wei, Kaixuan, Jiaolong Yang, Ying Fu, David Wipf, and Hua Huang. "Single image reflection removal exploiting misaligned training data and network enhancements." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8178-8187. 2019.
- [3] Li, Chao, Yixiao Yang, Kun He, Stephen Lin, and John E. Hopcroft. "Single image reflection removal through cascaded refinement." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3565-3574. 2020.
- [4] [深度学习训练时GPU相关配置和问题 tf_gpu_allocator=cuda_malloc_async-CSDN博客](#)
- [5] [python - Tensorflow doesn't allocate full GPU memory - Stack Overflow](#)
- [6] [Solving Out Of Memory \(OOM\) Errors on Keras and Tensorflow Running on the GPU \(linkedin.com\)](#)
- [7] [OpenCV这么简单为啥不学——1.12、使用ssim函数对两张照片进行相似度分析-腾讯云开发者社区-腾讯云 \(tencent.com\)](#)

6. Code Repository

程式碼實作細節請參考附件或以下 Github 連結：

- <https://github.com/Rui0828/Single-Image-Reflection-Removal-2024-CV-Final-Project>