

Comparing The Performance Of Different Image Inpainting Models

杜承祐

1. Introduction

Image Inpainting 是一項電腦視覺中的重要任務，目的是在圖像中被遮擋或損壞的區域生成合理且連貫的內容。這項技術在影像處理以及醫學影像等領域具有廣泛的應用價值。近年來，卷積神經網路、Transformer 和其他深度學習方法的發展，為圖像修復提供了強大的工具。然而，不同模型的設計在性能、適用場合和計算資源需求上存在顯著差異。

在研究中，不同的圖像修復模型對於遮擋區域的形狀與大小的處理能力存在差異。在本次作業中，我們選擇嘗試不同作者提出的 Image Inpainting 模型，觀察這些模型在處理不同程度的遮擋區域時的性能表現。此外，訓練與測試過程的計算資源限制也是一項挑戰，特別是在僅能使用單一 GPU 的條件下，如何有效執行模型訓練成為需要解決的現實問題。

本作業旨在通過實驗比較不同作者提出的圖像修復模型，理解其核心結構和設計差異，並透過實際訓練，比較它們在遮擋圖像修復中的表現。同時，針對硬體資源有限的條件，在小型資料集上進行嘗試，調整模型訓練的參數與流程，確保能夠在可行的時間內完成模型訓練與測試。

為了完成上述目標，我們選擇了三種圖像修復模型進行實驗。每種模型使用原作者提供的程式，並針對特定情形進行調整，以符合訓練時的使用環境，三種模型皆使用相同的數據集進行訓練與測試。我們還根據硬體資源的限制，針對每個模型調整合適的超參數（如批量大小、訓練週期等）進行了調整，盡量確保模型訓練的時長、使用的資料等變因相近，以此比較模型之間的差異。評估標準使用了多種常見的評估指標（如 PSNR、SSIM 及 LPIPS）來量化模型的性能差異。

本作業的宗旨在於進行一個基於有限硬體資源及有限資料的圖像修復實驗框架，對比了不同模型在小型數據集上的性能表現，並分析了其在不同比例的遮擋圖形下的適用性。通過對比實驗，我們深入理解了圖像修復技術的基本執行架構、關鍵挑戰和未來改進的方向。

2. Method Description

本次研究的實驗流程主要分為三個階段：模型訓練、測試資料預測及效能評估。在模型訓練階段，通過觀察與分析作者提供的程式碼，深入理解所選模型的架構與訓練流程，並針對實驗過程中出現的問題進程式碼修改。在不使用預訓練模型的前提下，以完全隨機初始化的參數進行模型訓練。在測試資料預測階段，利用資料集提供的多組來自不同場景的圖片及遮罩進行測試，對比不同類型圖片的修復結果。在效能評估階段，針對不同模型與資料類型的預測結果進行差異分析，並使用多種評估指標（如 PSNR 與 SSIM）對生成結果進行量化比較。以下將針對所使用的資料集內容及三個實驗階段進行詳細說明。

2.1. Dataset Formet

所使用的資料集由 data 與 test 兩個資料夾構成，所有圖片的尺寸均為 256x256 像素。其中，data 資料夾用於模型的訓練與驗證，其內包含 place2_train、place2_val、celeba_train 及 celeba_val，分別作為兩種來源資料進行兩種獨立的訓練與驗證。此外，train_mask 與 val_mask 資料夾中存放了遮罩（以 .png 格式儲存），遮罩覆蓋圖片的比例從 1% 至 60% 不等。訓練集中，每種來源各包含 1000 張圖片及對應的 1000 張遮罩；驗證集中，每種來源各包含 500 張圖片及對應的 500 張遮罩。

test 資料夾則用於測試模型的預測性能，內含來自不同來源的 6 種圖片類型：celeba_test、FFHQ_test、paris_test、place2_church_indoor_test、place2_church_outdoor_test 與 Shunghaitech_test，每種類型均包含 300 張圖片。test_mask 資料夾中包含 300 張遮罩，這些遮罩分別覆蓋於每種類型的圖片，作為測試過程中的輸入，用以評估模型的修復效果。

2.2. Training and Testing

三個模型的訓練過程分別在獨立的程式檔案中執行，以確保程式結構的清晰性和易讀性。然而，訓練的整體流程相似，均採用 Jupyter Notebook 編寫，並在 Google Colab 環境中運行，以便統一運行環境並減少維護成本。主要流程分為以下幾個階段：初始化設定、在 Place2 上訓練與測試、在 CelebA 上訓練與測試。

在初始化設定階段，首先透過 git clone 指令下載作者提供的原始程式專案，隨後利用 shutil 模組將所需資料集載入執行環境，最後設置訓練與測試的共通參數，為後續的訓練過程做好準備。一旦完成初始化設定，即可依序進行訓練與測試。

以下將詳細說明三個挑選的 Image Inpainting 模型的訓練過程，並說明在實驗過程中進行的程式改動及調整。

2.2.1. MADF model

該模型的架構首先定義多個子結構拆分多種功能，包含預訓練的 VGG16 模型、FilterGen 動態卷積過濾器、ConvWithFilter、AttConv 注意力機制合成特徵、DecActiv 上採樣特徵等，接著在主要的 MADFNet 結構中使用這些功能的組合，屬於較為龐大的模型架構。

在(圖一)中顯示的為訓練時使用的參數，其中在 dataset.py 中作者使用運行環境的絕對路徑與傳入參數的資料夾路徑串接作為索引圖片的路徑，因此必須將資料集放置在運行路徑下並設置傳入參數為相對路徑，防止錯誤發生。關鍵的 batch_size 設置為 4，確保 GPU ram 不會因記憶體溢出而意外中止訓練、max_iter 設置為 5000 次，約為 20 epochs，最終訓練時長約為 2 個小時。

```

#設定固定參數
IMAGE_SIZE=256
N_THREADS=2
BATCH_SIZE=4
MAX_ITER=5000 #epochs=20
SAVE_MODEL_INTERVAL=1000

#設定train相關參數
TRAIN_ROOT_PLACE2="data/place2_train"
VALID_ROOT_PLACE2="data/place2_valid"
TRAIN_MASK="data/train_mask"
TRAIN_PY="train.py"

```

(圖一) MADF training on Place2 之參數設置

測試流程中，作者設定傳入資料集的方法為圖片與遮罩組合的路徑所組成的文字檔，因此首先定義額外的函數生成符合需求的文字檔案，接著針對 6 種不同的測試資料分別以 shell 指令執行 test.py 進行測試，使用參數如下(圖二)，最後將模型參數與預測結果儲存便完成，使用 CelebA 作為測試資料集的運作方法基本相同，僅需更改資料來源的路徑即可運行。

```

#設定固定參數
IMAGE_SIZE=256
N_THREADS=2
BATCH_SIZE=4

#設定test相關參數
SNAPSHOT=' /content/drive/MyDrive/中興_深度學習/HW4/2-MADF/place2/ckpt/5000.pth'
TEST_PY="test.py"

```

(圖二) MADF testing on Place2 之參數設置

2.2.2. AOT_GAN

該模型中首先定義 AOTBlock (自適應上下文傳遞塊)、UpConv (上採樣卷積) 等方法，並在 InpaintGenerator 主架構中以 encoder、多層 AOTBlock、decoder、UpConv 的模式組成網路架構。

作者提供的程式中在 src/trainer/trainer.py 的實現有部分區塊在我們的執行情況下會出現錯誤，因此進行部分修改，修改結果如(圖二)顯示。首先是第 147 行的模型儲存時機的判斷邏輯，我們新增了額外的條件判斷當前執行的 iteration 輪次，確保在最後一輪訓練結束後能夠正常儲存模型參數。第二個為 save() 函數內部的行為，由於作者提供的程式為了實現分散式訓練，在多處與單一 GPU 訓練的程式進行了分支處理，然而此處無相關實現，因此新增分支判斷以符合我們使用的訓練模式。

```

# 請以下方程式替換src.trainer.trainer.py的第147行，確保訓練結束時至少保存一次
if self.args.global_rank == 0 and ((self.iteration % self.args.save_every) == 0 or (self.iteration == self.args.iterations)):

#請以下方程式替換src.trainer.trainer.py的第73行整個函數，確保儲存邏輯無誤
def save(self,):
    if self.args.global_rank == 0:
        print(f"\nsaving {self.iteration} model to {self.args.save_dir} ...")
        if (self.args.distributed == True):
            model_to_save_G = self.netG.module
            model_to_save_D = self.netD.module
        else:
            model_to_save_G = self.netG
            model_to_save_D = self.netD

        torch.save(
            model_to_save_G.state_dict(), os.path.join(self.args.save_dir, f"G{str(self.iteration).zfill(7)}.pt")
        )
        torch.save(
            model_to_save_D.state_dict(), os.path.join(self.args.save_dir, f"D{str(self.iteration).zfill(7)}.pt")
        )
        torch.save(
            {"optimG": self.optimG.state_dict(), "optimD": self.optimD.state_dict()},
            os.path.join(self.args.save_dir, f"O{str(self.iteration).zfill(7)}.pt"),
        )

```

(圖三) train.py 修改部分與說明

下方(圖三)顯示為訓練時使用的參數，作者預設的資料集情形是允許指定不同圖片與多種遮罩作為資料，因此設置 data_train 與 mask_type 參數作為類型名稱符合程式的處理方式。根據此參數使用 shell 指令執行 train.py 即可開始訓練，其中程式的實現在開始執行時會嘗試從預設的模型儲存路徑載入最近一次儲存的結果，因此可以透過分次執行簡單進行更多輪次的訓練，此處為了統一標準，共進行約兩個小時的訓練。

```

DIR_IMG = '/content/AOT-GAN-for-Inpainting/mydata'
DIR_MASK = '/content/AOT-GAN-for-Inpainting/mydata'
DATA_TRAIN = 'place2_train'
MASK_TYPE = 'train_mask'

MAX_ITER = 5000 # --iterations
nthread = 2 # --num_workers
SAVE_FREQ = 500 # --save_every
Batch_SIZE = 4 # batch_size
IMG_SIZE = 256 # --image_size

```

(圖四) AOT-GAN training on Place2 之參數設置

測試部分，作者的實現中允許傳入資料集的絕對路徑，因此只需依序對 6 個不同的測試資料設置 dir_image 與 dir_mask 與其他參數即可(圖五)，最後儲存模型參數與預測結果完成測試，使用 CelebA 作為訓練集的流程同上。

```

test_data_names = ["FFHQ_test", "Shunghaitech_test", "celeba_test", "paris_test", "place2_church_indoor_test", "place2_church_outdoor_test"]
test_mask_name="test_mask"

dir_root = "/content/AOT-GAN-for-Inpainting/mytest"
dir_output = "/content/output"

PRE_TRAIN = '/content/drive/MyDrive/中興_深度學習/HW4/4-AOT-GAN/place2/experiments/aotgan_place2_train_train_mask256/G0005000.pt' # --pre_train

nthread = 2 # --num_workers
Batch_SIZE = 4 # batch_size
IMG_SIZE = 256 # --image_size

```

(圖五) AOT-GAN testing on Place2 之參數設置

2.2.3. TFill

該模型的架構首先分別定義 discriminator、decoder、encoder 與 transformer 的基本結構，接著詳細定義 NLayerDiscriminator、StyleDiscriminator 和 RefinedGenerator 作為主要運行架構。

作者提供的程式中對於 dominate、ninja 與 visdom 模組有依賴，必須先行安裝。在 dataloader/data_loader.py 的 CreateDataset. _load_mask 出現與我們使用的遮罩資料不相容的程式，因此做出以下修正(圖四)，當 mask_type 設為 3 時遮罩產生模式為使用傳入資料集進行處理，然而作者使用的情境為遮罩大小比圖片大，並且希望使用隨機轉置與裁減等操作增加資料的多元性，然而本次實驗使用的是一對一合成，因此去除處理遮罩的部分步驟。

```
#請用以下程式替換 dataloader.data_loader.py 91行到118行的elif內邏輯
elif mask_type == 3:
    # external mask from "Image Inpainting for Irregular Holes Using Partial Convolutions (ECCV18)"
    if self.opt.isTrain:
        mask_index = random.randint(0, self.mask_size-1)
    else:
        mask_index = item

    mask_transform = transforms.Compose(
        [
            transforms.Resize([h, w])
        ]
    )
    mask_pil = Image.open(self.mask_paths[mask_index]).convert('L')
    mask = mask_transform(mask_pil)

    if self.opt.isTrain:
        mask = self._mask_dilation(mask)
    else:
        mask = np.array(mask) < 128
    mask = torch.tensor(mask).view(1, h, w).float()

    mask_pil.close()
    return mask, mask_type
```

(圖四) data_loader.py 修改部分與說明

訓練使用的所有圖片以儲存成文字檔的路徑讀取，因此先定義額外函數進行處理，接著根據作者提供的 train.sh 範例修改部分參數，符合我們的硬體資源與資料集，使用參數如下(圖五)，皆著開始訓練，其中透過 checkpoints_dir 與 continue_train 參數的設置可實現分次執行，最終訓練 40000 輪約耗時 2 小時。

```
python train.py \
--name place2 \
--img_file /content/TFill/mydata/place2_train.txt \
--mask_file /content/TFill/mydata/train_mask.txt \
--model tc \
--coarse_or_refine coarse \
--netI original \
--n_encoders 12 \
--n_decoders 0 \
--netD style \
--gpu_ids 0 \
--load_size 256 \
--fine_size 256 \
--batch_size 8 \
--nThreads 2 \
--display_port 8093 \
--attn_G \
--add_noise \
--display_ncols 0 \
--iter_count 0 \
--n_iter 40000 \
--n_iter_decay 5000 \
--print_freq 50 \
--save_iters_freq 5000 \
--save_latest_freq 5000 \
--checkpoints_dir /content/checkpoints \
--no_html \
--preprocess none \
--mask_type 3 \
--continue_train
```

(圖五) TFill training on Place2 之參數設置

測試部分傳入參數為資料集路徑，因此只需依序對 6 個不同的測試資料設置 img_file 與 mask_file 參數即可(圖六)，最後儲存模型參數與預測結果完成測試，使用 CelebA 作為訓練集的流程同上。

```
test_data_names = ["FFHQ_test", "Shunghaitech_test", "celeba_test", "paris_test", "place2_church_indoor_test", "place2_church_outdoor_test"]
test_mask_name="test_mask"

dir_root = "/content/TFill/mytest"
dir_output = "/content/results"
pretrain_root = "/content/checkpoints" # --checkpoints_dir
NAME = "place2"
```

(圖六) TFill testing on Place2 之參數設置

2.3. Loss Calculation

在先前的訓練與測試中，我們針對三個模型分別使用 test 資料集進行模型預測並儲存生成圖片的結果，在此階段中我們使用另一個獨立的程式專案統一對三個模型的生成結果進行效能評估，我們選擇以常見的三種評估指標作為標準，分別是 PSNR、SSIM、LPIPS，可對評結果進行分析與比較。

PSNR (Peak Signal-to-Noise Ratio) 計算方式基於像素之間的均方差做為基礎，表示重建圖像與原始圖像之間的差異。PSNR 值越高，表示重建圖像的質量越好，然而其容易受到易受像素值範圍的影響，無法準確反映生成品質。計算公式如下：

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) = 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right)$$

SSIM (Structural Similarity Index) 是衡量兩幅圖像結構相似度的指標，通過亮度、對比度和結構三個方面來量化生成圖像的品質，透過模仿人類視覺系統，重點關注結構信息。SSIM 值在 [0, 1] 之間，越接近 1，表示圖像越相似，此評估標準相比於 PSNR 更符合人眼觀察的結果，計算公式如下：

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

LPIPS (Learned Perceptual Image Patch Similarity) 是基於深度學習的相似性評量方法。它利用深度卷積神經網絡（如 VGG 或 AlexNet）提取特徵，衡量兩幅圖像在特徵空間中的距離。其計算方法首先使用預訓練模型提取輸入圖像的多層特徵，並計算每層特徵之間的歐幾里得距離，最後透過加權融合各層距離，得到最終的 LPIPS 值。LPIPS 值越小，兩幅圖像越相似，該方法能夠捕捉高層次的語義信息，比 PSNR 和 SSIM 更符合人類感知，屬於影像生成的重要評估指標。

在實作中我們首先透過 skimage 模組載入 psnr 與 ssim 方法，這兩種皆接收兩個圖片實例作為參數，接著安裝並匯入 lpips 模組，該方法接收兩個 tensor 作為輸入參數進行神經網

路的相似計算，因此需要額外處理。我們首先定義 `process_and_calculate_losses(image_path1, image_path2)` 函數，此函數接收原始圖片與生成圖片的路徑，透過 PIL 載入圖片並經過適當處理後，呼叫三種評估指標的計算方法，反回一個元組，表示為三種指標的計算結果。

在對三個模型分別計算評估結果的部分，分別針對模型個別的生成結果儲存格式設置對應的批次處理函數，為了加速計算過程，我們使用多線程拆分處理工作，透過紀錄每一個原始圖片、生成圖片組合的評估結果，最終計算得到整體的平均指標，以一組測試資料類型為單位，對於一個模型共紀錄 6 組資料集*3 種評估指標的計算結果。

3. Results Comparison

在此部分中，我們將針對評估指標進行量化分析與討論，並透過視覺化呈現比較模型之間預測結果的差異。

3.1. Quantitative Comparison


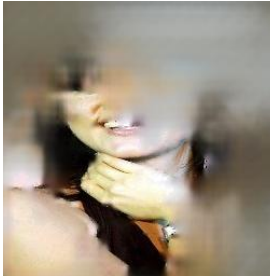

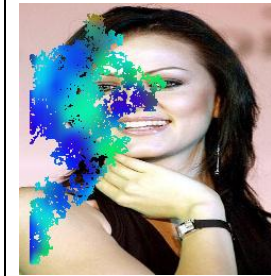


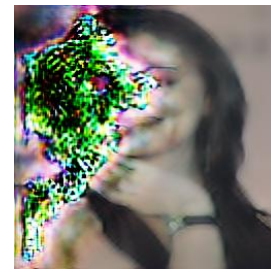

(表 1) Place2 訓練集下各模型的指標表現

Dataset	Index	Model 2	Model 4	Model 5
CelebA (300)	PSNR	15.5949	7.9051	13.1856
	SSIM	0.6225	0.2024	0.6376
	LPIPS	0.4362	0.6577	0.3836
FFHQ (300)	PSNR	16.2676	8.0582	12.8075
	SSIM	0.6152	0.1866	0.6288
	LPIPS	0.4608	0.6846	0.4387
Paris (300)	PSNR	18.2173	9.0108	15.1313
	SSIM	0.5906	0.1558	0.6543
	LPIPS	0.4661	0.6804	0.3958
Place2_church_indoor (300)	PSNR	17.0585	8.9498	12.0854
	SSIM	0.5482	0.1530	0.6332
	LPIPS	0.4663	0.6693	0.3779
Place2_church_outdoor (300)	PSNR	16.2179	7.9198	12.3795
	SSIM	0.5893	0.1720	0.6409
	LPIPS	0.4594	0.6707	0.3806
Shunghaitech (300)	PSNR	17.2293	7.6069	13.9891
	SSIM	0.6463	0.1865	0.6521
	LPIPS	0.4266	0.6703	0.3821

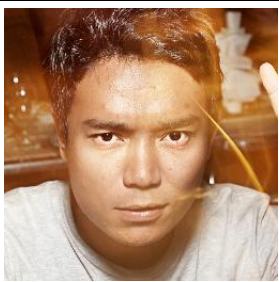

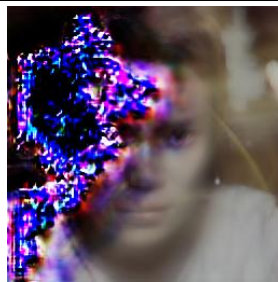

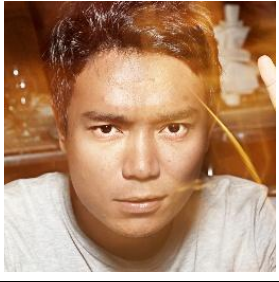
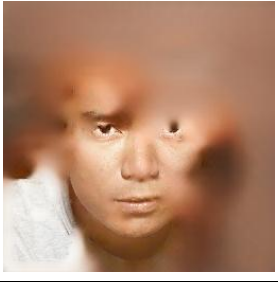
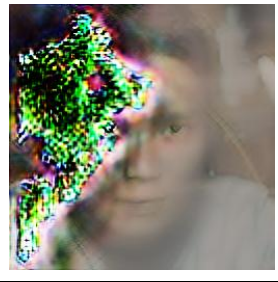

(表 2) CelebA 訓練集下各模型的指標表現

Dataset	Index	Model 2	Model 4	Model 5
CelebA (300)	PSNR	15.4477	9.1209	13.9048
	SSIM	0.6186	0.2230	0.6477
	LPIPS	0.4387	0.6557	0.3686
FFHQ (300)	PSNR	16.1671	9.4935	13.1677
	SSIM	0.6163	0.2123	0.6361
	LPIPS	0.4540	0.6838	0.4141
Paris (300)	PSNR	17.3382	10.0699	14.4061
	SSIM	0.5747	0.1889	0.6525
	LPIPS	0.4930	0.6677	0.3934
Place2_chrch_indoor (300)	PSNR	16.6357	9.7375	13.5730
	SSIM	0.5402	0.1717	0.6428
	LPIPS	0.4829	0.6549	0.3666
Place2_chrch_outdoor (300)	PSNR	15.4119	9.1287	13.5541
	SSIM	0.5758	0.1996	0.6544
	LPIPS	0.4816	0.6568	0.4264
Shunghaitech (300)	PSNR	16.6142	9.2613	15.0014
	SSIM	0.6353	0.2224	0.6642
	LPIPS	0.4474	0.6544	0.3515



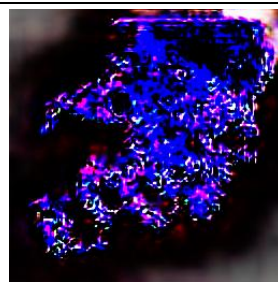
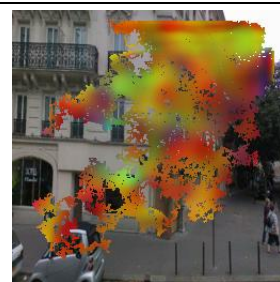

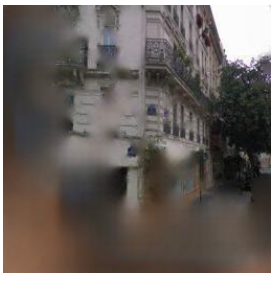
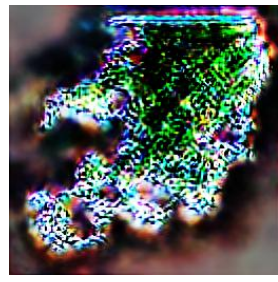
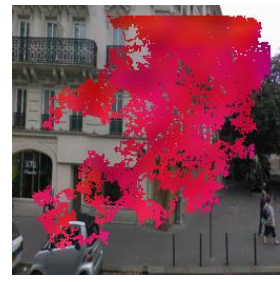
3.2. Qualitative Comparison

Place2				
CelebA				
Train Data	Input	MADF	AOT-GAN	TFill







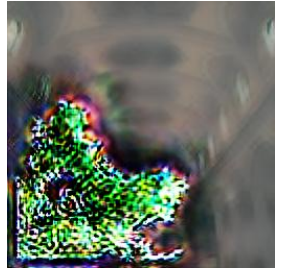

(圖 七) Qualitative comparison of celeba_test

Place2				
Celeb A				
Train Data	Input	MADF	AOT-GAN	TFill



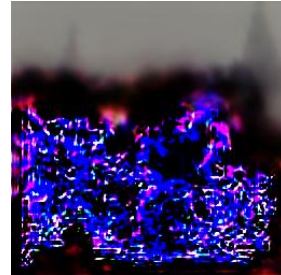
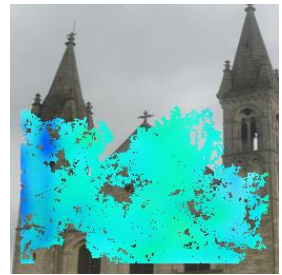


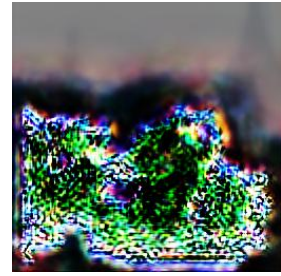

(圖八) Qualitative comparison of FFHQ_test

Place2				
Celeb A				
Train Data	Input	MADF	AOT-GAN	TFill



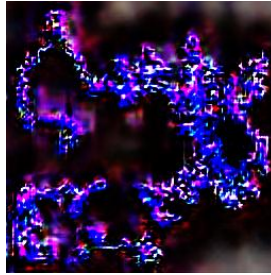


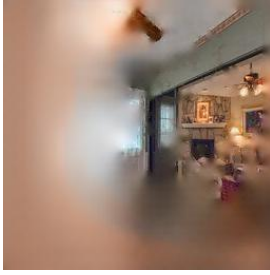
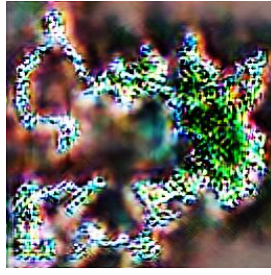

(圖九) Qualitative comparison of paris_test

Place2				
Celeb A				
Train Data	Input	MADF	AOT-GAN	TFill

(圖十) Qualitative comparison of place2_chrch_inoor_test

Place2				
Celeb A				
Train Data	Input	MADF	AOT-GAN	TFill

(圖十一) Qualitative comparison of place2_chrch_ourdoor_test

Place2				
CelebA				
Train Data	Input	MADF	AOT-GAN	TFill

(圖十二) Qualitative comparison of Shunghaitech_test

3.3. Complexity Comparison

(表三) 各模型推理部分之參數數量與 FLOPs

	MADF	AOT-GAN	TFill
# of parameters (M),	85.14	15.20	22.48
FLOPs (GMac)	46.33	72.97	0.016

4. Discussion

從量化分析的角度觀察，可以發現對於同一個模型架構，使用 Place2 或 CelebA 作為訓練資料得出的生成相似度，不論哪種指標其結果皆為接近的，並沒有出現以人臉訓練則在人臉的測試資料集中表現更好的情形，可以推測在目前的訓練量級下還未捕捉到高層次的複雜特徵。

比較不同模型架構之間的指標結果，可以觀察到對於 PSNR 指標，MADF 整體略高於 TFill、明顯高於 AOT-GAN，而 SSIM 與 LPIPS 的部分 MADF 與 TFill 數值是相近的，AOT-GAN 的表現則明顯較差。

由於 SSIM 的數值落在固定範圍 $[0, 1]$ 之間，我們可以用來推測模型預測結果與真實資料的差異性，MADF 與 TFill 整體皆落在 0.6 附近，由於遮罩佔比在 1%到 60%之間不等，對於遮罩小的測資本身相似度會更高，因此平均來說此數值並不太理想，而 AOT-GAN 的數值落在 0.2 附近，很可能其生成結果破壞了原始圖片的區域。

根據視覺呈現的比較結果，可以發現 MADF 模型呈現出較明顯的修復行為，其生成結果偏向於依據周圍顏色進行填充，並具有些微依據周圍輪廓延伸填充的行為，觀察使用 Place2 與 CelebA 訓練的 MADF 模型的區別，可以發現 CelebA 的模型對於邊緣區塊傾向於填充膚色色塊，而 Place2 則是灰褐色，這很可能與使用的訓練資料有關，表示模型雖沒有捕捉到複雜特徵，修補的整體色調卻受到資料的影響。

觀察 AOT-GAN 模型的結果，可以發現除了遮罩覆蓋的區域以外，其生成的圖片還導致周圍未覆蓋區域的損壞，整體呈現為遮罩區域形成雜訊、未覆蓋區域模糊或顏色加深無法辨識，可以看出模型尚不足以學習到任何圖形特徵。

接著分析 TFill 模型，可以發現生成的顏色填充與周圍特徵沒有太大的關連性，然而我們可以觀察使用 CelebA 訓練的模型在 FFHQ_test 與 place2_church_outdoor_test 測試資料生成的圖片，能夠發現其在周圍輪廓較明顯的區域有嘗試生成顏色較深的延伸區域，其他組生成結果也有少部分顯現此特性，可能說明了該模型雖未能學習到顏色之間的關聯性，然而其已開始捕捉到簡單的輪廓特徵。

最後對於模型的複雜度進行分析，可以發現 AOT-GAN 的 FLOPs 是遠大於其他兩者的，因此訓練時相同時間內執行的輪次更少，而 TFill 則相反，儘管參數數量比 AOT-GAN 略大，然而 FLOPs 卻小其他兩者一個量級，因此其能夠在短時間進行上萬次訓練，快速獲得訓練成效，而 MADF 則因為使用 VGG16 預訓練模型，在先天擁有部分特徵提取能力下也有稍多的參數量。

5. Conclusion

整體而言，Image Inpainting 模型在小規模資料與有限的訓練輪次下很難呈現出理想的表現，根據量化指標表明，生成圖片與原始圖片有較大的差距。然而部分模型在視覺上呈現出些微的特徵捕捉表現，MADF 整體而言表現最佳，並且對於顏色填充有較明顯的呈現，很可能與模型架構使用預訓練的 VGG16 特徵提取有關，因此儘管架構龐大仍能實現一定程度的補全。TFill 雖然對於顏色填充表現不佳，然而其顯現出對於輪廓特徵補全的行為，其模型單一輪次訓練快速，同樣耗時內能夠經過更多次迭代，可能在此過程中學習到小部分圖像特徵。AOT-GAN 整體而言表現最差，其生成結果甚至破壞了未覆蓋遮罩的區域，其模型架構複雜，在設定的少量迭代次數下很可能沒有學習到有用的特徵。

6. References

- [1] Image Inpainting by End-to-End Cascaded Refinement with Mask Awareness (IEEE TIP 2021)
<https://github.com/MADF-inpainting/Pytorch-MADF>

[2] AOT-GAN: Aggregated Contextual Transformations for High-Resolution Image Inpainting

(IEEE TVCG 2022)

<https://github.com/researchmm/AOT-GAN-for-Inpainting>

[3] Bridging Global Context Interactions for High-Fidelity Image Completion

(CVPR 2022)

<https://github.com/lyndonzheng/TFill>