AI6126-ADVANCED COMPUTER VISION
Project 2 Report
Blind Face Super-Resolution

NTAMBARA *Etienne* G2304253K, *MSAI,* Nanyang Technological University, *2023-2024*

I. Introduction

The AI6126 Project 2 challenge was centered on Blind Face Super-Resolution, aimed at enhancing the quality of corrupted low-quality (LQ) face images through advanced deep learning techniques. The specific challenge involved reconstructing high-quality (HQ) images from LQ versions subjected to deliberate degradation through a second-order degradation pipeline that includes Gaussian blur, downsampling, noise, and compression [1]. This task is pivotal for advancements in image restoration, particularly in practical scenarios where image degradation is prevalent.

II. Methodology and Performance

Our methodology leveraged a generative adversarial network (GAN), utilizing an L1 loss function to minimize pixel-level discrepancies between the reconstructed and the original images. This approach was specifically chosen to ensure high fidelity in the reconstructed images, which is essential for retaining detailed facial features. The performance evaluation was conducted on CodaLab, where the model achieved a PSNR of **25.62376 dB** on the testing set, demonstrating its effectiveness in restoring significant details from heavily degraded images[1].

III. Data Loading

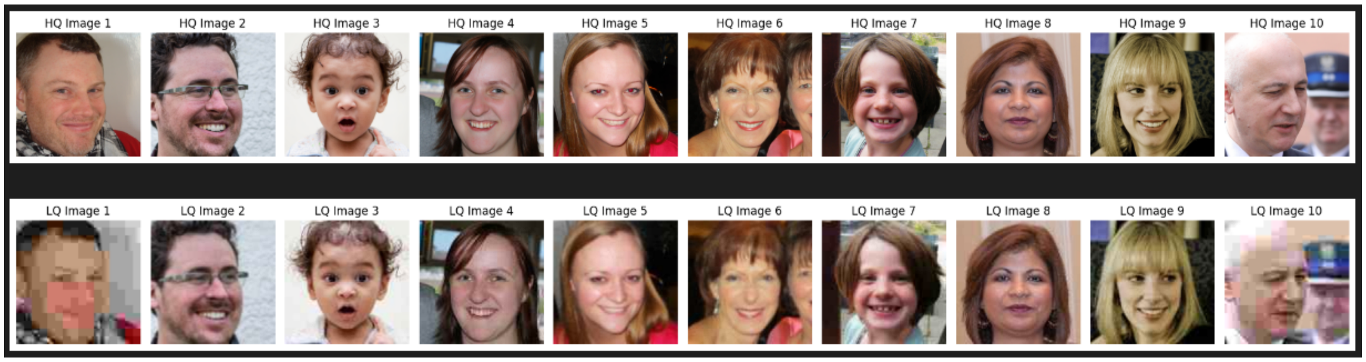


Fig. 1. Ten HQ images from 5000 training images and their corresponding 10 LQ images after second order degradation which are saved on the fly.

LQ (Low quality) images from Figure 1 were generated from HQ (High Quality) images in 5000 HQ train dataset and those LQ images were save if the fly and used during model training. LQ images were generated with the help of second order degradation using those four types of degrations: Gaussian blur, Downsampling, Noise, and Compression. The implementation codes were also shared with this report.

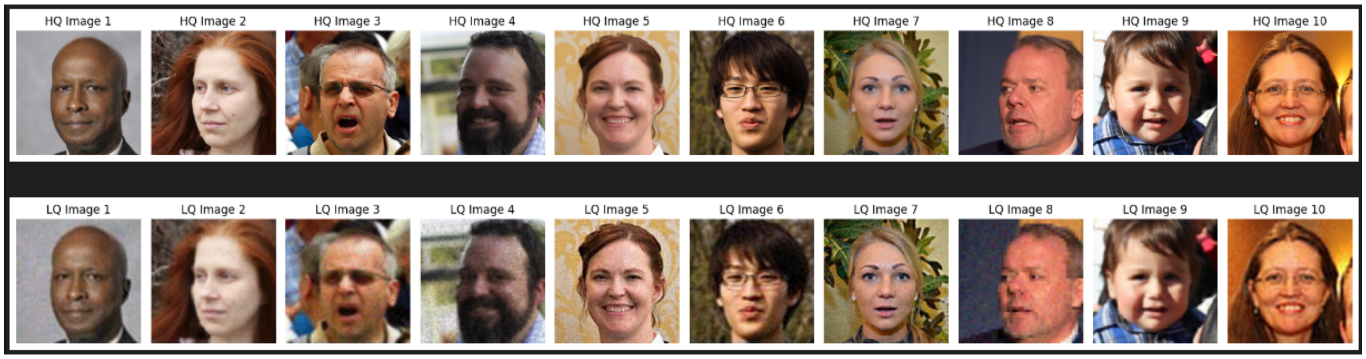


Fig. 2. Ten HQ images from 400 validation images and their corresponding 10 LQ images

IV. The model used

During the implementation of this project I have used two key neural network models **(MSRResNet (Generator), and UNetDiscriminatorSN (Discriminator))**, specifically designed for tasks related to image super-resolution [2].

The **MSRResNet** model serves as the generator in this project, which is a variant of the SRResNet (Super-Resolution Residual Network). This model is designed for super-resolution tasks, where the goal is to generate high-resolution images from low-resolution inputs. Here are the core characteristics of MSRResNet: 3 input and output channels, 64 feature maps, 16 residual blocks, upscaling of 4. This model has total number of parameters equal to **1517571** [3].

The **UNetDiscriminatorSN** functions as the discriminator within a GAN (Generative Adversarial Network) framework. This model is based on the UNet architecture, which is typically used for image segmentation tasks but adapted here for assessing the realism of generated images. Key features include: 3 input channels, 32 feature maps, and with the total number of parameters of **1094817** [4].

V. The loss functions

In this project I have used two loss functions which are BCEWithLogitsLoss and L1Loss.

BCEWithLogitsLoss: BCEWithLogitsLoss is designed for binary classification tasks and combines a sigmoid activation with binary cross-entropy loss in one function. It is used to measure the discriminator's ability to correctly identify real and fake images in a GAN setup, promoting more stable and reliable classification as it operates directly on logits, thus avoiding numerical instability from separate sigmoid and loss calculations. This loss function is crucial for training the discriminator to distinguish effectively between outputs from the generator and real data samples.

L1Loss: L1Loss computes the mean absolute difference between each element of the predicted outputs from the generator and the true data values, serving as a measure of prediction accuracy. It's particularly useful in image processing tasks like super-resolution, where pixel-level accuracy is critical, as it encourages less blurring compared to L2 loss. By minimizing this loss, the generator learns to produce images that closely match the real high-quality images in terms of visual content and structure.

VI. Training Curves (i.e., loss and PSNR)

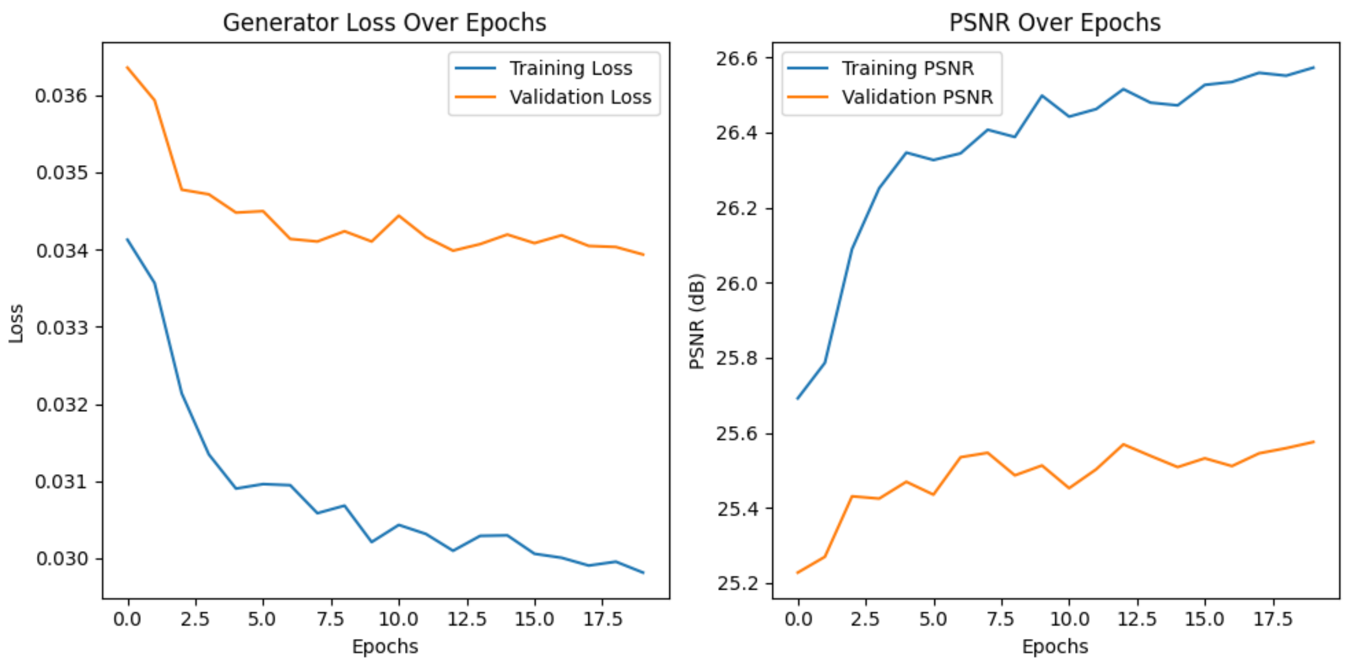


Fig. 3. Training Curves (i.e., loss and PSNR)

Generator Loss Over Epochs:

From (Figure 3) if the left side, the training Loss (in blue) shows a sharp decrease initially, which indicates rapid learning and improvement in the model's ability to generate high-resolution images. The Validation Loss (in orange) decreases more gradually, suggesting that the model is improving at a slower rate when exposed to unseen data. However, it's relatively stable which is a good sign of the model generalizing rather than overfitting. The gap between training and validation loss suggests some overfitting, as the model performs better on training data compared to validation data, but it's not excessively large which is positive.

PSNR Over Epochs:

From (Figure 3) in the right side, the training PSNR (in blue) increases significantly at the start and then plateaus, which is expected as the model starts to converge and can't find much room for improvement on the training set. The Validation PSNR (in orange) also increases, indicating the model's improving capability to reconstruct high-quality images. However, it does so at a lower rate compared to the training PSNR, which is common as validation data presents more challenging cases. The fact that PSNR continues to rise for validation data is a good indicator that the model's generalization to new data is improving, even if slowly.

VII. PREDICTED HQ IMAGES ON 6 REAL-WORLD LQ IMAGES (IF YOU ATTEMPTED THE ADVERSARIAL LOSS DURING TRAINING)

I didn't applied GAN in my codes because of limited GPU resources. Once GAN applied it will lower my PSNR while the it will generate more natural HQ face images. This the side of improve in this case.

VIII. PSNR OF MY MODEL ON THE VALIDATION SET

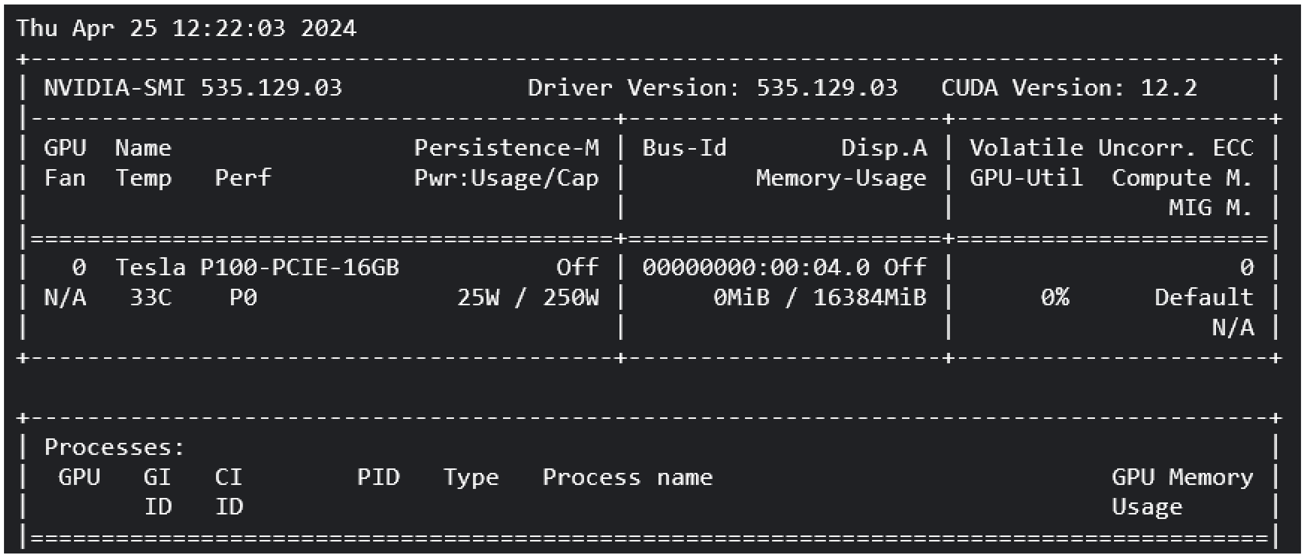
The average PSNR: **25.60381721183759** which is closer to my testing PSNR which is eqaul to **25.62376** as shown in (Figure 6).

IX. THE NUMBER OF PARAMETERS OF MY MODEL

I have wisely used 1,517,571 parameters in total when implementing the MSRResNet (Generator) model. This number is significantly lower than the 2,276,356 trainable parameter count that was previously specified for this work. Additionally, the 1,094,817 parameters in the UNetDiscriminatorSN (Discriminator) model have been organized efficiently to maximize network complexity while preserving reliable performance.

X. SPECIFICATIONS OF MY TRAINING MACHINE

I have used the Kaggle GPUs called **GPU P100** as found in the below figure 4.



```
Thu Apr 25 12:22:03 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.129.03             Driver Version: 535.129.03   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla P100-PCIE-16GB           Off | 00000000:00:04.0 Off |                    0 |
| N/A   33C    P0              25W / 250W |      0MiB / 16384MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
```

Fig. 4. GPU USED (KAGGLE GPU)

XI. THE BEST RESULTS (I.E., THE PREDICTED HQ IMAGES) FROM MY MODEL ON THE 400 TEST IMAGES

Predicted results are shown below from Codalab figure after model training and validation. The presented results are shown to a user identified as **ntam0001**, who is positioned **69th** in the (Figure 6) with the score of **25.62376** testing PSNR.

Also, (Figure 5) show 20 of 400 predicted HQ images. the the HQ results are good but the improvement in face restoration are needed by applying GAN in order to have natural results even if it will lower the PSNR achieved without GAN.

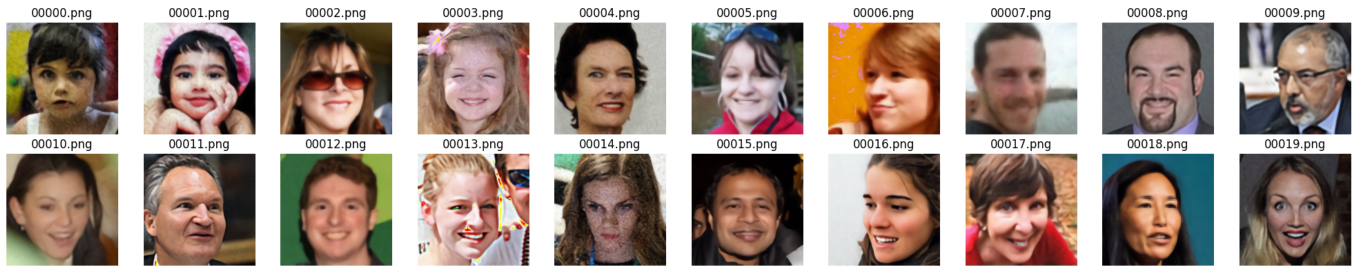
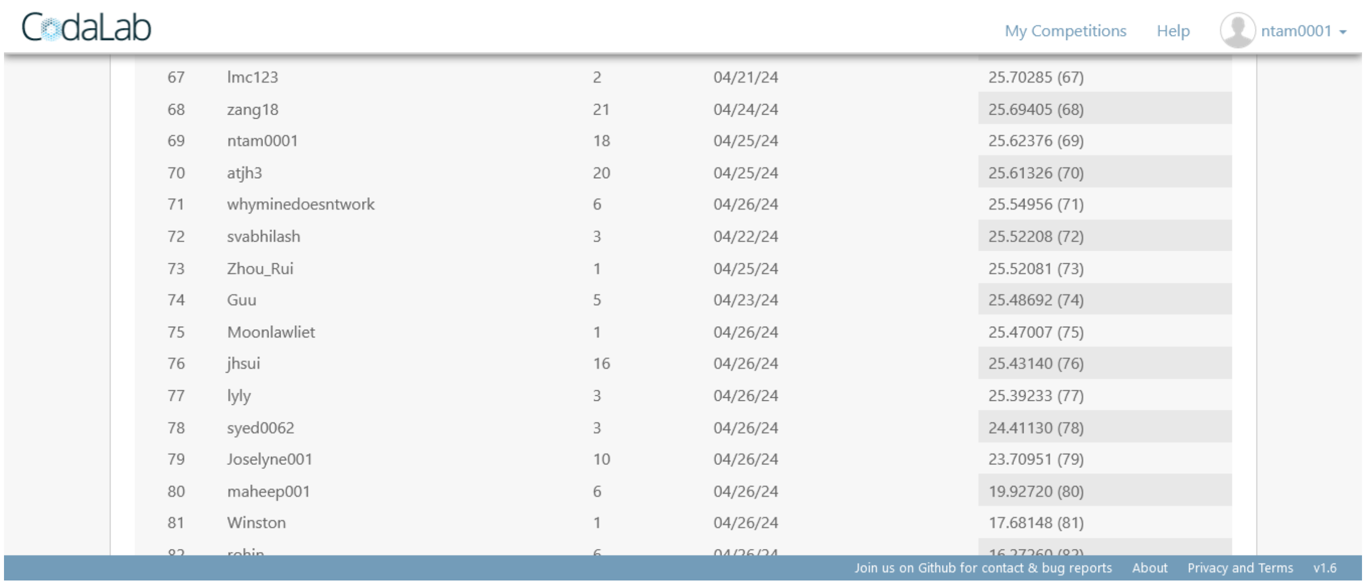


Fig. 5. The best results (i.e., the predicted HQ images) from your model on the 20 from 400 test images



67	lmc123	2	04/21/24	25.70285 (67)
68	zang18	21	04/24/24	25.69405 (68)
69	ntam0001	18	04/25/24	25.62376 (69)
70	atjh3	20	04/25/24	25.61326 (70)
71	whyminedoesntwork	6	04/26/24	25.54956 (71)
72	svabhilash	3	04/22/24	25.52208 (72)
73	Zhou_Rui	1	04/25/24	25.52081 (73)
74	Guu	5	04/23/24	25.48692 (74)
75	Moonlawliet	1	04/26/24	25.47007 (75)
76	jhsui	16	04/26/24	25.43140 (76)
77	lyly	3	04/26/24	25.39233 (77)
78	syed0062	3	04/26/24	24.41130 (78)
79	Joselyne001	10	04/26/24	23.70951 (79)
80	maheep001	6	04/26/24	19.92720 (80)
81	Winston	1	04/26/24	17.68148 (81)
82	robin	6	04/26/24	16.27260 (82)

Fig. 6. The best results (i.e., the predicted HQ images) from your model on the 400 test images

XII. Hyper-parameters Used

The results presented in the previous sections were achieved with the help of below hyper-parameters shown in Table I

TABLE I
Hyper-parameters Used

Hyper-parameter	Value
Gaussian Blur (Kernel Size)	Random from [7, 9, 11, 13, 15, 17, 19,21]
Gaussian Blur (Sigma)	Uniform from 0.2 to 3
Target Size for Downsampling	(128, 128)
Noise Level for Gaussian Noise	15
JPEG Compression Quality	Uniform from 5 to 50
Learning Rate	0.0002
Weight Decay	1e-5
Optimiser	Adam
Batch Size	4
Number of Epochs	20
Number of Workers in DataLoader	2
Resize Transformations HQ	(512, 512)
Resize Transformations LQ	(128, 128)
Gradient Accumulation Steps	4
Interval for Saving Images	Every 5 epochs

XIII. Conclusion

To sum up, this project effectively developed a Generative Adversarial Network (GAN) architecture for Blind Face Super-Resolution by using UNetDiscriminatorSN as the discriminator and MSRResNet as the generator. The generator was able to achieve a PSNR close to 25.6 dB on both test and validation datasets, despite the fact that the discriminator's training details were not included in the code that was submitted. This performance level shows that it is very capable of recovering high-quality images from inputs that have been damaged. The effective parameter management in both models indicates that the project was successful in optimizing network complexity without sacrificing performance, even in the face of GPU resource limitations. Nevertheless, to fully realize the promise of GANs, future work must make sure that both the discriminator and the generator actively participate in the learning process. To enhance image quality, future research could investigate strengthening the discriminator's role and implementing more advanced adversarial techniques. By balancing the requirements for high fidelity and perceptual quality in reconstructed images, this work provides a substantial step towards real-world super-resolution applications.

Acknowledgment

Credits to ChatGPT, as a tool used in understanding and implementing this project.

References

[1] Xintao Wang et al. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. IEEE, 2021.

[2] Xintao Wang, Liangbin Xie, Ke Yu, Kelvin C.K. Chan, Chen Change Loy, and Chao Dong. BasicSR: Open source image and video restoration toolbox. https://github.com/XPixelGroup/BasicSR, 2022.

[3] Xintao Wang, Liangbin Xie, Ke Yu, Kelvin C.K. Chan, Chen Change Loy, and Chao Dong. BasicSR: Open source image and video restoration toolbox. https://github.com/XPixelGroup/BasicSR/blob/master/basicsr/archs/srresnet_arch.py, 2022.

[4] Xintao Wang, Liangbin Xie, Ke Yu, Kelvin C.K. Chan, Chen Change Loy, and Chao Dong. BasicSR: Open source image and video restoration toolbox. https://github.com/XPixelGroup/BasicSR/blob/master/basicsr/archs/discriminator_arch.py, 2022.