Image Quality Enhancement: Experimenting with Neural Network Architectures

Aimilia Palaska 10453 - aimiliapm@ece.auth.gr
Academic Supervisor: Anastasios Tefas

January 10, 2025

1 Introduction

This report is created as part of the Neural Networks & Deep Learning course in Electrical and Computer Engineering department of Aristotle University of Thessaloniki. The developed code in Python, along with detailed comments, can be found in the following GitHub Repository [1].

Generative Adversarial Networks (GANs) are a powerful machine learning approach designed to generate new data samples that mimic a given dataset. In the context of image enhancement, GANs aim to bridge the gap between low-quality inputs and their high-quality counterparts, addressing challenges such as noise reduction, resolution improvement, and visual detail recovery. This problem is critical for applications in fields like medical imaging, remote sensing, and photography, where image quality directly impacts decision-making and user experience. Machine learning structures, particularly neural networks, are well-suited for this task because of its ability to learn complex patterns and features from data, enabling GANs to generate realistic outputs that were previously unattainable with traditional methods.

2 Dataset Overview

For this project, the CelebA [2] dataset was selected, a widely recognized benchmark for facial image analysis, comprising over 200,000 high-resolution images of celebrities annotated with attributes. Its versatility, extensive size, and diversity in facial features make it an ideal choice for tasks such as image enhancement. The original images, sized at 218×178 pixels, provided high-quality targets for the enhancement model.

Preprocessing was central to transforming this dataset for the needs of the project. The core of the preprocessing pipeline involved min-max normalization as well as creation of degraded versions of the images to serve as inputs. This was achieved by down-scaling each image to 1/5 of its original size and then resizing it back to the original dimensions, simulating loss of detail and resolution.

2.1 Visualization and Examples

To gain insights into the dataset, random samples of degraded inputs and their corresponding high-quality targets were visualized. This helped in qualitatively assessing the degradation process and verifying its consistency across the dataset. Examples, presented in Figure 1, show visibly blurred and pixelated inputs compared to the clear, detailed targets, highlighting the complexity of the enhancement task.

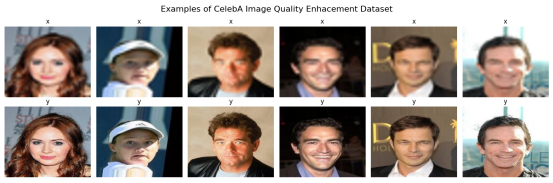


Figure 1: Downgraded and original versions of some CelebA dataset samples.

2.2 Dataset Management and Challenges

Handling the dataset's size required efficient data loading and processing strategies. A parameterized loader was implemented to allow for varying fractions of the dataset, ensuring scalability and adaptability to computational constraints. The dataset was divided into training and testing sets with a 60-40 split to evaluate the model's generalization ability. Preprocessed data was stored in-memory during runtime, minimizing I/O overhead.

In summary, the CelebA dataset's diversity and realism, coupled with a robust preprocessing pipeline, provided a strong foundation for the GAN-based image enhancement model. By creating realistic degraded inputs and managing the dataset effectively, the approach remains scalable and aligned with practical applications.

3 Evaluation Metrics

Evaluating image quality enhancement is inherently more complex than standard classification

tasks. Instead of determining whether a label is correct, it must be assessed whether an enhanced image closely resembles its high-quality counterpart while maintaining structural integrity. This requires metrics that go beyond simple pixel-wise comparisons to capture perceptual quality and overall similarity. For this project, three key metrics for performance evaluation were chosen, with the last two being employed from the Scikit-Image [3] library as ready methods.

- *Mean Squared Error* (MSE) quantifies the average squared difference between the predicted image and the ground truth, providing a basic measure of reconstruction accuracy. While easy to compute, MSE does not correlate well with human perception, making it an insufficient standalone metric.

- *Peak Signal-to-Noise Ratio* (PSNR), derived from MSE, is expressed in decibels and indicates how much signal has been preserved relative to noise. Higher PSNR values suggest better reconstruction quality, but like MSE, it is limited in capturing perceptual nuances.

- *Structural Similarity Index Measure* (SSIM) is particularly valuable for image enhancement as it evaluates structural similarity between the enhanced and ground-truth images. SSIM considers luminance, contrast, and structure, offering a perceptually relevant measure of quality. This aims to test whether enhanced images retain essential features and natural textures.

3.1 Evaluation Process

During evaluation, the model was tested on the separate test set to ensure unbiased performance measurement. For each test image, the degraded input, the ground truth, and the enhanced output were compared using the three metrics. MSE provided a baseline measure of pixel-wise reconstruction, while PSNR and SSIM captured perceptual fidelity and structural similarity.

To illustrate model performance, random examples of degraded inputs, ground-truth images, and enhanced outputs were plotted. Visual comparisons allowed us to identify cases where the model performed well or struggled, providing insights beyond numerical metrics. Each enhanced image was scaled appropriately for visualization to ensure interpretability.

3.2 Challenges and Considerations

A significant challenge in evaluating image enhancement models is the subjective nature of image quality. While quantitative metrics like PSNR and SSIM are valuable, they may not always align perfectly with human perception. For instance, an image with a slightly lower PSNR might appear more visually pleasing due to better texture restoration. To address this, qualitative assessments through visualizations complemented the numerical evaluations.

In summary, the combination of MSE, PSNR, and SSIM provided the framework for evaluating image enhancement performance. The goal of these metrics alongside visual comparisons, is to ensure that the models not only improve pixel accuracy but also preserve structural and perceptual fidelity, aligning its outputs with human expectations of quality.

4 Experimental Setup

The repository is structured as follows; the `source` module encapsulates the whole functionality, with the `source/trainer.py` file providing an abstraction class to facilitate experiments. Execution scenarios are configured with the `main.py` script, while extensive results and plots can be found in the corresponding directories in the top level.

4.1 Baselines

To establish reference points for evaluating the performance of the GANs, two baseline models were implemented.

The first model was a **Convolutional Neural Network (CNN)**, implemented in the script `conv_net.py`. This simple encoder-decoder architecture was designed using convolutional and transposed convolutional layers, serving as a benchmark to assess the improvement offered by more complex approaches. The structure consisted of an encoder-decoder framework with three convolutional layers for feature extraction, followed by transposed convolutional layers for up-sampling. Rectified Linear Unit (ReLU) activations were applied to the intermediate layers, while a sigmoid activation in the output layer constrained pixel values to the range [0,1]. The loss function used was Mean Squared Error (MSE), which minimized pixel-wise differences between the enhanced and target images.

The second model was a **Radial Basis Function (RBF) Network**, implemented in the script `rbf_net.py`. This network leveraged learnable RBF centers and widths to map degraded images to high-quality outputs, providing a direct comparison against more sophisticated techniques. The architecture featured a single hidden layer containing RBF neurons, with the outputs from the RBF layer mapped back to the image space using a fully connected layer. During training, parameters for the RBF centers and

widths were learned, optimizing the network for image restoration. Similar to the CNN, MSE was employed as the loss function to minimize reconstruction error.

4.2 GAN model

The GAN model was structured with a deep convolutional architecture to handle image generation tasks efficiently. The generator could leverage different backbones — ResNet18 [4], VGG16 [5], or a custom convolutional backbone — allowing for versatile feature extraction and transformation from degraded inputs to high-fidelity outputs. This flexibility enabled the generator to adapt and refine image quality, focusing on producing realistic and enhanced visual content.

The discriminator served as a binary classifier, tasked with distinguishing between real and generated images. It utilized either a backbone-based feature extractor, drawing on pre-trained architectures like ResNet18 or VGG16, or a standalone convolutional network designed for discriminative purposes. Both configurations were optimized to extract and analyze features at multiple scales, facilitating accurate classification.

The convolutional implementation consisted of several key layers.

- *Convolutional Layers*: Used for feature extraction at different scales, processing the input image through filters to extract spatial features. They reach up to 256 or 512 channels with the first configuration being prioritized in the experiments.
- *Batch Normalization*: Applied to stabilize and speed up training by normalizing the outputs of previous layers.
- *ReLU Activations*: Introduced non-linearity, allowing the model to learn complex representations from extracted features.
- *Transpose Convolutional Layers*: Utilized for up-sampling the feature maps to produce high-resolution outputs from lower-dimensional feature representations.

The loss functions were pivotal in guiding the training process. The generator's loss function combined Binary Cross-Entropy (BCE) to deceive the discriminator with Mean Squared Error (MSE) for pixel-wise accuracy, ensuring the generated images not only appeared realistic but also matched the desired output at the individual pixel level. Conversely, the discriminator's BCE loss refined its ability to differentiate between real and synthetic images, enhancing its robustness throughout training. The training proceeded with alternating updates to the generator and discriminator.

	CONV	RBF
Avg. Duration	45.16 sec	49.94 sec
Avg. MSE	18138.50	21462.97
Avg. PSNR	5.94	4.8139
Avg. SSIM	0.0032	1.4004

Table 1: Baseline results for the convolutional and radial basis function networks with configuration: 100 epochs, batch size 32 and learning rate 0.001. The average duration refers to one epoch.

4.3 Training Configuration

The experiments involved varying key hyperparameters to evaluate their impact on the models' performance. The learning rate was adjusted independently for the convolutional and radial basis function (RBF) networks while keeping the Adam optimizer as the default optimization algorithm. For the batch size, a value of 64 was selected, which represented the maximum allowable size within the constraints of the HPC infrastructure.

Other parameters that were systematically varied included the fraction of the dataset used, the learning rates for the generator and discriminator, the GAN backbone architecture, and the number of training epochs. Hyperparameter tuning was conducted iteratively based on the observed progress during training, allowing for dynamic adjustments to optimize performance. This approach ensured the training was both adaptable and responsive to the dataset and computational resources.

5 Results

5.1 Baselines

The baseline models performed below expectations, as anticipated, due to their inadequacy for image generation tasks. This is evident from the almost flat loss function plots throughout the epochs, presented in Figure 2. The convolutional baseline showed slightly better performance, as reflected in the metrics summarized in Table 1. Despite this improvement, neither method was efficient enough to produce visually meaningful results, which prevented the inclusion of example images. These outcomes highlight the limitations of traditional architectures in handling image enhancement tasks, underscoring the necessity of advancing towards more capable architectures like the GAN framework.

5.2 GAN

For the default backbone, three experiments were conducted with varying configurations, as shown in Table 2.

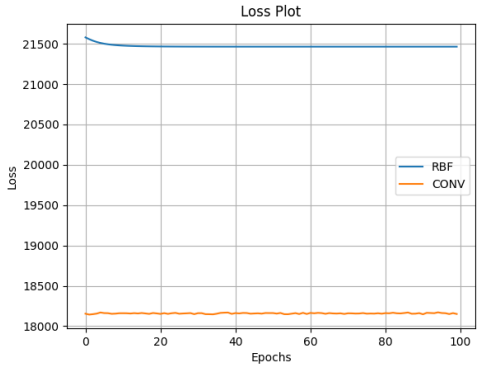
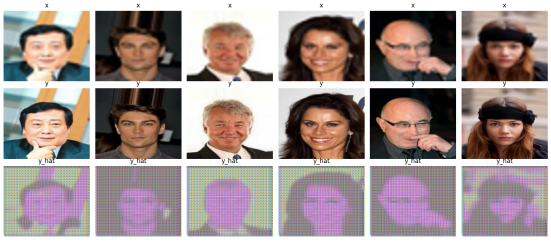


Figure 2: Loss for the baseline models. The curves' stability points to the inefficiency of the baseline models at handling the image enhancement task.



(a) Experiment 1

(b) Experiment 2

(c) Experiment 3

Figure 3: Output of test examples for the three default backbone GAN experiments.

	Exp. 1	Exp. 2	Exp. 3
Samples	10,000	50,000	70,000
Epochs	50	50	100
Batch Size	32	64	64
Gen. LR	2e-04	1e-04	1e-04
Disc. LR	5e-05	1e-04	5e-05

Table 2: Configuration of the three default backbone experiments.

	Exp.1	Exp 2.	Exp. 3
Avg. Duration	67 sec	292 sec	433 sec
Avg. MSE	18291	18193	18254
Avg. PSNR	5.91	5.95	5.93
Avg. SSIM	8.12	-0.003	0.003

Table 3: GAN results for the default backbone.

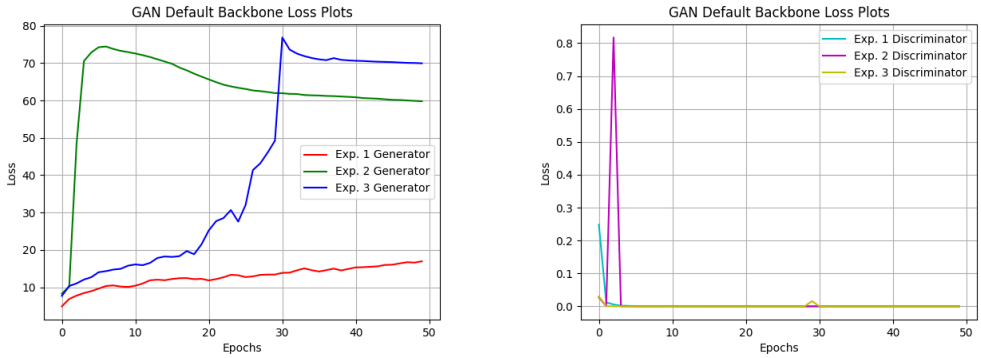
The resulting images from these experiments are depicted in Figure 3, showcasing five examples for each scenario: the original image (y), the degraded input (x), and the enhanced output ($\hat{y}$). These examples are evaluated during testing, meaning the model has not been trained specifically on these clear images.

Experiment 1 performed relatively poorly, retaining only basic structural elements of the images but struggling with color accuracy and overall quality. Experiments 2 and 3 displayed more robust performance, with Experiment 3 being the most effective. These two experiments produced outputs that closely resemble the color and structural qualities of the input images, but with notably enhanced clarity.

Table 3 presents the metrics calculated for each experiment in the default backbone setup. Notably, while some metrics have slightly degraded or remained similar to the baseline results, the examples in Figure 3 indicate a great improvement. This suggests that although mathematically supported, these metrics may not fully capture the effectiveness of the enhanced images for this specific task.

As far as the loss plots, showed in Figure 4, the discriminator losses remain near zero while the generator losses increase, suggesting an imbalance in training where the discriminator is too strong or the generator is under-performing. This could be due to the discriminator's high capacity, poor generator initialization or architecture, vanishing gradients from overly confident discriminator predictions, mode collapse in the generator, or unbalanced update frequencies or learning rates.

Unfortunately, the ResNet18 and VGG16 backbones were not deemed ideal architectures for this task. The results showed simple color gradients. In hindsight, this should be expected since both models output 512-channel 7x7 feature maps before the classification layer. Attempting to revert back to 3 channels while retaining 180 x 220 spa-



(a) Generators (b) Discriminators

Figure 4: Loss plots for the three GAN experiments.

tial dimensions could only be done with transpose convolutional layers, which seems redundant given that it was implemented in the default backbone. Nevertheless, one of the ResNet18 examples is shown in Figure 5.

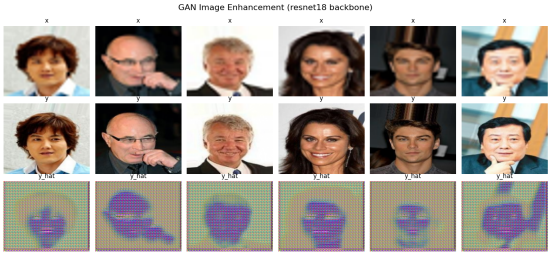


Figure 5: GAN test results with the ResNet18 backbone.

6 Discussion

This report's results hint at potential improvements in image enhancement quality, particularly as the size of the dataset increases. If enough computational resources were accessible, it would be of interest to test against the entire CelebA dataset, which has approximately 200,000 pictures, instead of less than 50% of it. Another important observation could be derived from a larger batch size, for example 128.

Lastly, addressing the mismatch of generator and discriminator losses would require balancing the training dynamics. Future experiments could explore limiting discriminator updates, regularizing the discriminator, improving generator design, or using alternative loss functions.

Acknowledgments

Results presented in this work have been produced using the Aristotle University of Thessaloniki (AUTh) High Performance Computing Infrastructure and Resources. Additionally, the open-source language model ChatGPT [6] was utilized in parts of these experiments. It generated portions of the code which were then tested and analyzed, as well as enhanced the overall readability and clarity of this report.

References

[1] E. Palaska, "celeba-gan." https://github.com/emily-palaska/celeba-gan, 2025.

[2] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild." http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html, 2015.

[3] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: Image processing in python." https://doi.org/10.7717/peerj.453, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[6] OpenAI, "ChatGPT." https://chat.openai.com/.