Image Processing Final Project Report

110511010 楊育陞 110511067 葉哲伍

1 Introduction

The goal of this project is to enhance image clarity through image super resolution. We employ two different methods to address various scenarios. For general cases, we iteratively apply the Hybrid Attention Transformer (HAT) [1] and histogram specification to enhance the image quality. For face images, we blend the results of applying three different models: HAT, CodeFormer [2], and GFPGAN [3] to achieve better performance.

2 Fundamentals

2.1 Hybrid Attention Transformer (HAT)

Comparing to other previous transformer-based models, HAT introduces a hybrid attention mechanism, which combines the channel attention and window-based self-attention. These mechanisms can enhance the spatial range of input information when processing the image.

2.2 HAT-GAN

The HAT-GAN has the same architecture as the HAT model. The difference is that the HAT-GAN model is trained in the method of ESRGAN, which is a GAN-based super resolution method. The HAT-GAN model have better performance for real-world images.

2.3 Codebook Lookup Transformer (CodeFormer)

CodeFormer is a transformer-based model for blind face restoration. This approach casts the face restoration problem as a code prediction task and use a learned codebook to guide the restoration process.

2.4 Generative Facial Prior GAN (GFPGAN)

GFPGAN uses a generative facial prior to guide the face restoration process of a GAN-based model. The generative facial prior is learned from a large-scale face dataset and can provide a strong prior for face restoration. This approach can achieve a good balance of realness and fidelity.

3 Method and Improvement

3.1 Case 1: For General Images

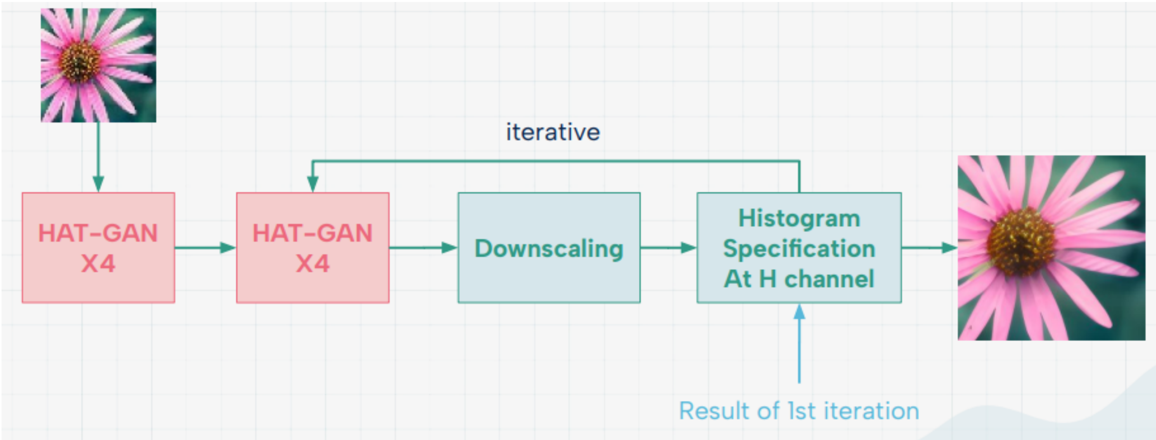


Figure 1: Schematic diagram for case 1

(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Table 1: Experiment Results for case 1

1. Initial Approach - Iterative HAT-GAN and Downscaling

First, we apply the HAT-GAN model to the input image directly. The result is as shown in Table 1(a), which is satisfactory already. To further improve the result, we try to apply the HAT-GAN model iteratively. The idea is to apply the HAT-GAN model again to upscale the image to 16x, and then downscale it to 4x. The results are shown in Table 1(b) to 1(f). We can see that the image becomes smoother after each iteration. However, there are significant color changes in the image.

2. Improvement - Histogram Specification

To address the color changes, we apply histogram specification on the hue channel of the image. We match the hue histogram of the result image to the hue histogram of the first iteration result. The results are shown in Table 1(g) to 1(k). We can see that the color changes are mitigated.

3. Blend the Results

Although the result of the last iteration is smooth, it is less similar to the original image in some features. To address this issue, we blend the result of the last iteration with the result of the first iteration. After experimenting with different blending ratios, we find that the best blending ratio is 0.8 for the last iteration result and 0.2 for the first iteration result. The final result is shown in Table 1(l).

3.2 Case 2: For Human Face Images

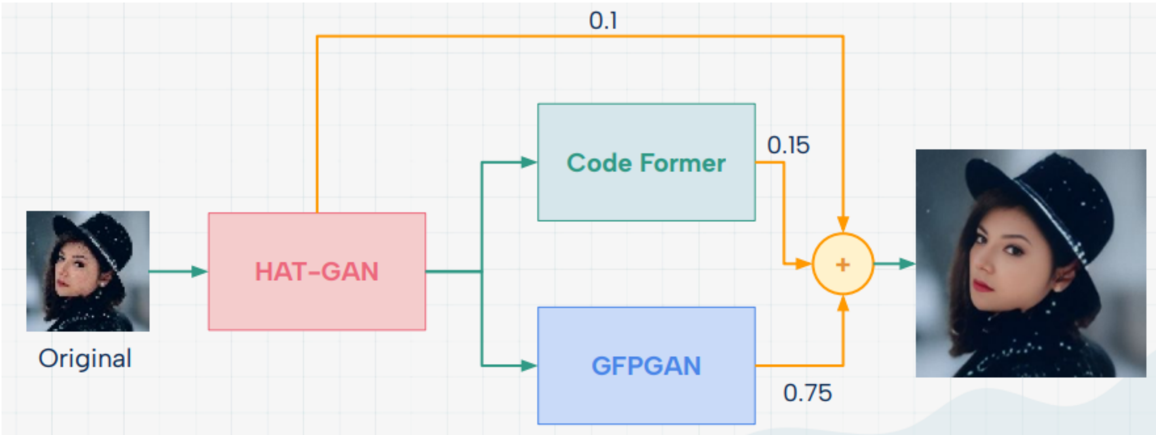


Figure 2: Schematic diagram for case 2

(a) (b) (c) (d)

Table 2: Experiment Results for case 2

1. HAT-GAN Result

The result of applying the HAT-GAN model to the input image is shown in Table 2(a). The result is satisfactory except for the face region.

2. CodeFormer vs GFPGAN

To address the face region, we apply face restoration models. We choose CodeFormer and GFPGAN for this task. The result of applying CodeFormer is shown in Table 2(b), and the result of applying GFPGAN is shown in Table 2(c). We can see that the result of CodeFormer is more realistic and detailed, while the result of GFPGAN keeps some features of the original image.

3. Blend the Results

Since the characteristics of the results of CodeFormer and GFPGAN are different, we blend the results of the two models as Figure 3. Also, we try to blend the result of the HAT-GAN model. After experimenting with different blending ratios, we find that the best blending ratio is 0.75 for CodeFormer, 0.15 for GFPGAN, and 0.1 for HAT-GAN. The final result is shown in Table 2(d).

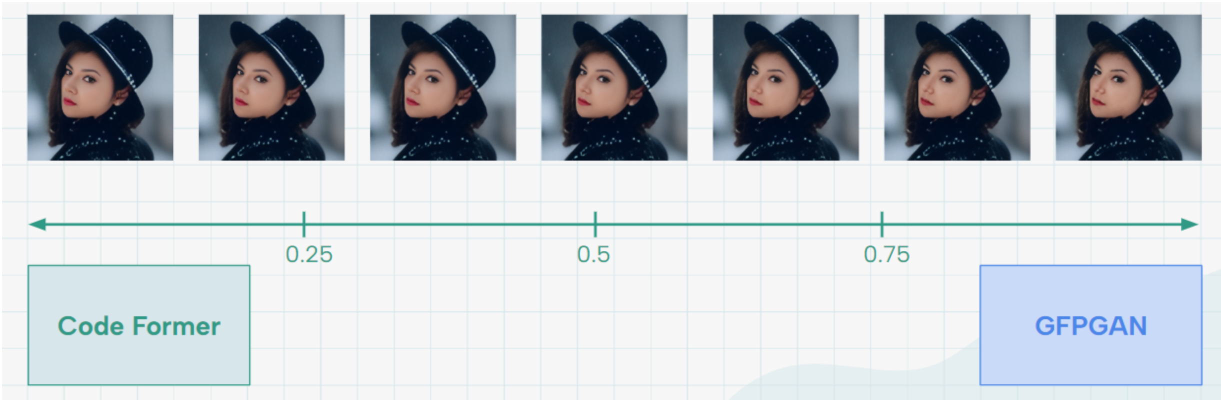


Figure 3: Blending Results of CodeFormer and GFPGAN

4 Implementation

4.1 HAT-GAN

For the HAT-GAN model, we use the pre-trained model provided by the authors. We apply the model to the input image and get the result. In our method, we apply the model on the result of the previous iteration, which needs more computational resources than applying the model on the input image. So we tile the image into overlapping patches and apply the model on each patch. After getting the result of each patch, we stitch them together to get the final result.

4.2 CodeFormer and GFPGAN

For the CodeFormer and GFPGAN models, we also use the pre-trained models provided by the authors. We apply the models to the whole input image and get the results with face regions restored.

4.3 Downscaling and Histogram Specification

For downscaling, we use the `cv2.resize` function in OpenCV with the `INTER_AREA` interpolation method. For histogram specification, we use the `cv2.calcHist` function in OpenCV to calculate the hue histogram of the result image and the first iteration result image. Then we use the `cv2.LUT` function in OpenCV to match the hue histogram of the result image to the hue histogram of the first iteration result image.

4.4 Blending

For blending, we use the `blend` function in PIL. To make experiments easier, we implement a user interface to adjust the blending ratios and see the results in real-time. The user interface is implemented with the `gradio` library.

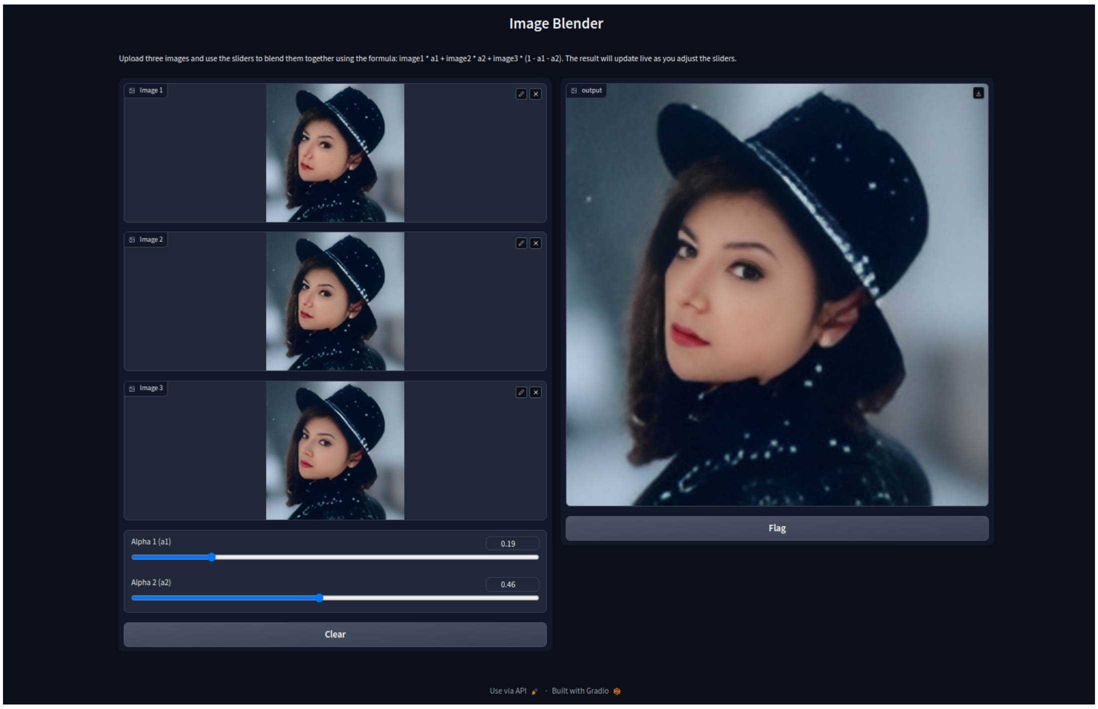
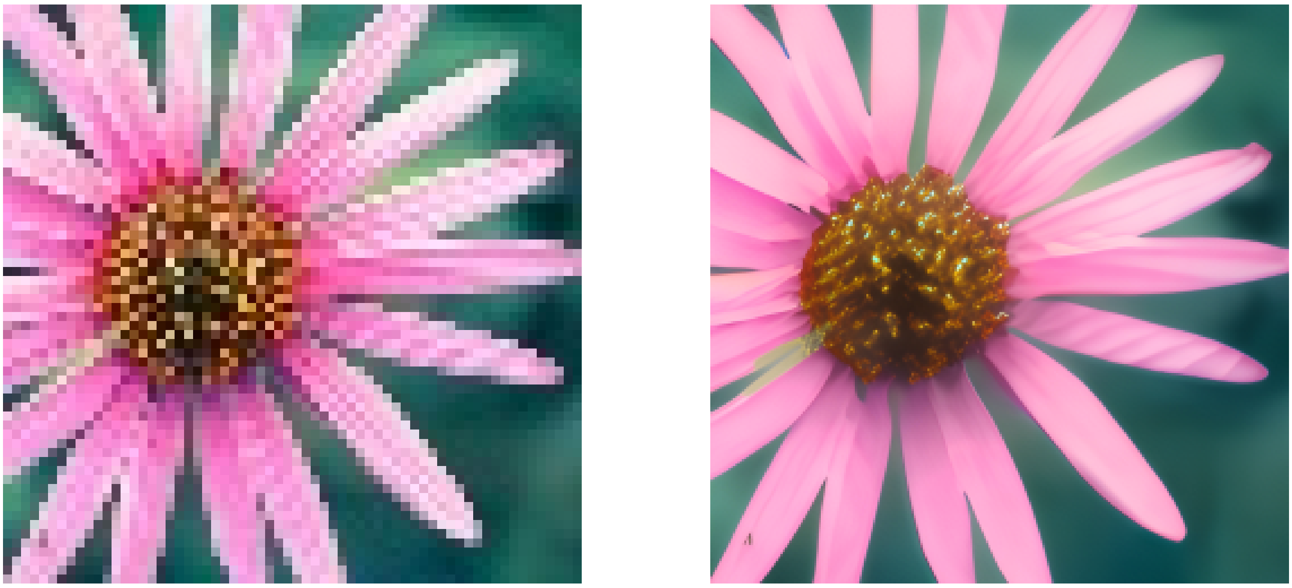


Figure 4: User Interface for Blending Experiment

5 Result



(a) Original Image 1

(b) Result Image 1

Figure 5: Result Images 1

(a) Original Image 2

(b) Result Image 2

Figure 6: Result Images 2

References

[1] HAT: Hybrid Attention Transformer for Image Restoration

[2] Towards Robust Blind Face Restoration with Codebook Lookup Transformer

[3] Towards Real-World Blind Face Restoration with Generative Facial Prior