Image Restoration using SwingIR-Large, Real-ESRGAN, BSRGAN

Course: Spring-2025 Deep and Generative Learning by Masoud Yari

Term Project

Authors: Sharmili Rudraraju

Shubadeep Koley

1. Introduction

In the realm of digital imagery, the restoration and enhancement of low-quality or degraded images is a crucial task. Applications span from historical photo restoration, and surveillance image enhancement, to super-resolving details in scientific and medical imaging. In recent years, deep learning-based models have revolutionized this field, producing stunningly realistic results beyond traditional interpolation methods.

This project explores and compares three state-of-the-art image restoration models:

- BSRGAN (Blind Super-Resolution GAN, ICCV 2021),
- Real-ESRGAN (Enhanced Super-Resolution GAN, 2021),
- SwinIR-Large (Swin Transformer-based Image Restoration model, 2021).

We explored how to use three AI models Real-ESRGAN, SwinIR, and BSRGAN to restore image quality. and compare their effectiveness and highlight their strengths. We evaluate these models both quantitatively (using metrics such as PSNR and SSIM) and qualitatively (using histograms, edge detection, difference maps, and human visual comparison) on a diverse set of images representing real-world degradations, including noise, blur, and compression artifacts. We have also developed a flask-based application and provided user interface where users can upload their images and can enhance their images using selected model and download them.

The primary goal is to identify the strengths and weaknesses of each model under varying degradation types, offering insights into their practical usability across different domains.

2. Models Overview

In this section, we provide a concise yet insightful overview of the three deep learning models evaluated in our study.

2.1 BSRGAN (Blind Super-Resolution GAN)

BSRGAN is a robust model designed for blind image super-resolution, meaning it handles unknown and complex degradations without requiring explicit degradation models during inference. Proposed in ICCV 2021, BSRGAN focuses on enhancing structural integrity and perceptual quality by employing a sophisticated degradation pipeline that simulates realistic corruptions during training. It utilizes a GAN (Generative Adversarial Network) architecture with a ResNet-based generator, optimized to handle a variety of image artifacts like noise, blur, compression, and low-resolution distortions.

Key Features:

- Handles blind degradations without predefined kernels.
- Trained with heavy data augmentation to simulate diverse real-world conditions.
- Focuses more on generating realistic textures even if PSNR is slightly compromised.

For more details on BSRGAN, visit the following resources:

- GitHub Repository
- ArXiv Paper: https://doi.org/10.48550/arXiv.2103.14006

2.2 Real-ESRGAN (Enhanced Super-Resolution GAN)

Real-ESRGAN is an improvement over the original ESRGAN, specifically crafted for real-world image restoration tasks. Unlike synthetic degradation settings, Real-ESRGAN trains on more realistic noise patterns and complex degradation pipelines. It uses a U-Net discriminator and a RRDB (Residual-in-Residual Dense Block) based generator, striking a balance between fine detail preservation and artifact suppression. Real-ESRGAN also integrates a relativistic GAN loss to better distinguish between subtle quality differences.

Key Features:

- Strong at maintaining photorealistic textures.
- Trained on both synthetic and real-world datasets for better generalization.
- Produces visually pleasing results but might over smooth fine details in certain cases.

For more details on the Real-ESRGAN, visit the following resources:

- GitHub Repository

- ArXiv Paper: https://doi.org/10.48550/arXiv.2107.10833

2.3 SwinIR-Large (Swin Transformer for Image Restoration)

SwinIR-Large leverages the power of Swin Transformers — a hierarchical Transformer model that processes images with local attention mechanisms. Unlike convolutional networks, SwinIR captures long-range dependencies while preserving high-frequency details. It is especially effective for both classical super-resolution and denoising tasks. The "Large" variant uses more layers and embedding dimensions, providing superior restoration fidelity.

Key Features:

- Transformer-based model for modeling long-range spatial relationships.
- Strong performance on fine structures and global image coherence.
- Slightly slower inference compared to CNN-based architectures but excels on complex textures.

For more details on SwinIR, visit the following resources:

- GitHub Repository
- ArXiv Paper: https://doi.org/10.48550/arXiv.2108.10257

These three models represent different design philosophies:

- BSRGAN for robust perceptual realism under harsh degradations.
- Real-ESRGAN for clean, photo-friendly restorations.
- SwinIR-Large for structure-preserving, globally consistent outputs.

Training Code:

If you want to train your own model, you can use the KAIR Repository. The repo includes the necessary training and testing scripts for a range of image restoration models, including USRNet, DnCNN, FFDNet, SRMD, DPSR, MSRResNet, ESRGAN, BSRGAN, SwinIR, VRT, and RVRT.

The repository allows you to configure training settings like GPU usage and data paths through JSON files. It supports both DataParallel and DistributedDataParallel training methods.

However, in this article, we will focus on inference and application rather than training.

3. Google Colab Implementation: Full Image Restoration Pipeline

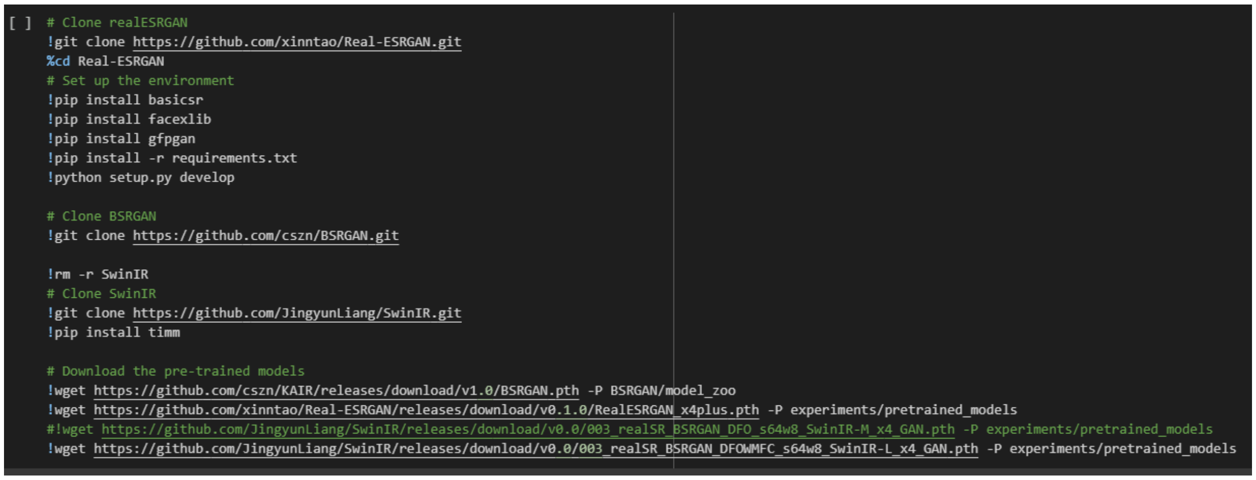
To practically apply and validate the capabilities of Real-ESRGAN, BSRGAN, and SwinIR, we developed a complete Google Colab notebook pipeline. This section outlines the workflow we implemented, from environment setup to image enhancement and evaluation.

3.1 Setting Up the Environment

First, we cloned the official repositories of the three models:

- Real-ESRGAN
- BSRGAN
- SwinIR

We installed the required dependencies, including basicsr, facexlib, gfpgan, and timm libraries, and downloaded the necessary pre-trained weights for inference.



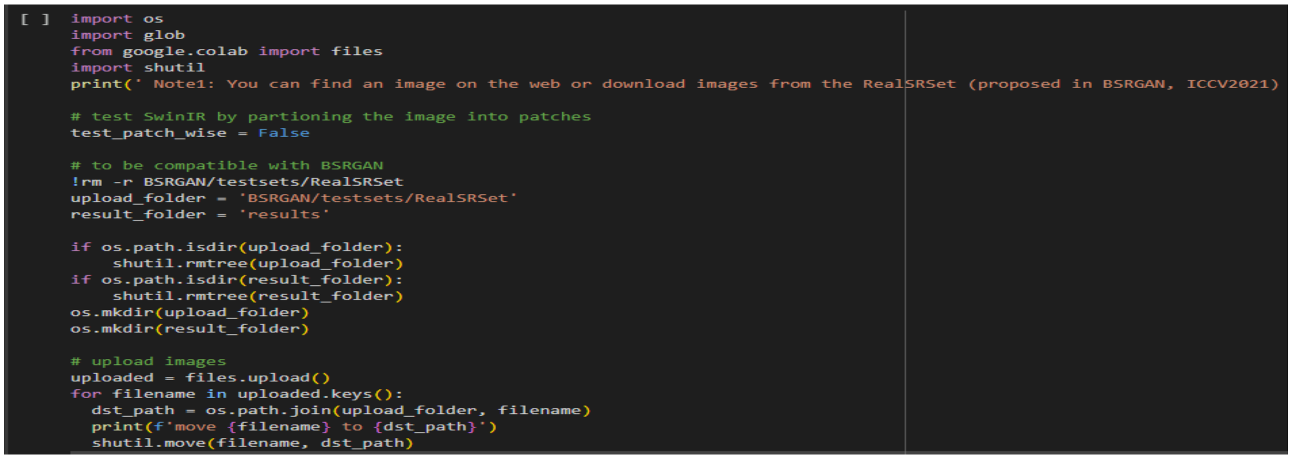
```
[ ] # Clone realESRGAN
    !git clone https://github.com/xinntao/Real-ESRGAN.git
    %cd Real-ESRGAN
    # Set up the environment
    !pip install basicsr
    !pip install facexlib
    !pip install gfpgan
    !pip install -r requirements.txt
    !python setup.py develop

    # Clone BSRGAN
    !git clone https://github.com/cszn/BSRGAN.git

    !rm -r SwinIR
    # Clone SwinIR
    !git clone https://github.com/JingyunLiang/SwinIR.git
    !pip install timm

    # Download the pre-trained models
    !wget https://github.com/cszn/KAIR/releases/download/v1.0/BSRGAN.pth -P BSRGAN/model_zoo
    !wget https://github.com/xinntao/Real-ESRGAN/releases/download/v0.1.0/RealESRGAN_x4plus.pth -P experiments/pretrained_models
    #!wget https://github.com/JingyunLiang/SwinIR/releases/download/v0.0/003_realSR_BSRGAN_DFO_s64w8_SwinIR-M_x4_GAN.pth -P experiments/pretrained_models
    !wget https://github.com/JingyunLiang/SwinIR/releases/download/v0.0/003_realSR_BSRGAN_DFOWMFC_s64w8_SwinIR-L_x4_GAN.pth -P experiments/pretrained_models
```

We also downloaded pre-trained models directly into designated folders to avoid manual intervention.

3.2 Uploading and Preparing Images

We created dedicated folders for uploaded input images and model outputs. Uploaded images were placed in BSRGAN/testsets/RealSRSet, while outputs were collected into model-specific folders like results/BSRGAN, results/realESRGAN, and results/SwinIR_large.

If a user system ran into GPU memory issues, we configured a test_patch_wise variable. Setting it to True allowed patch-wise inference, ensuring stable performance on T4 GPUs offered by Google Colab.

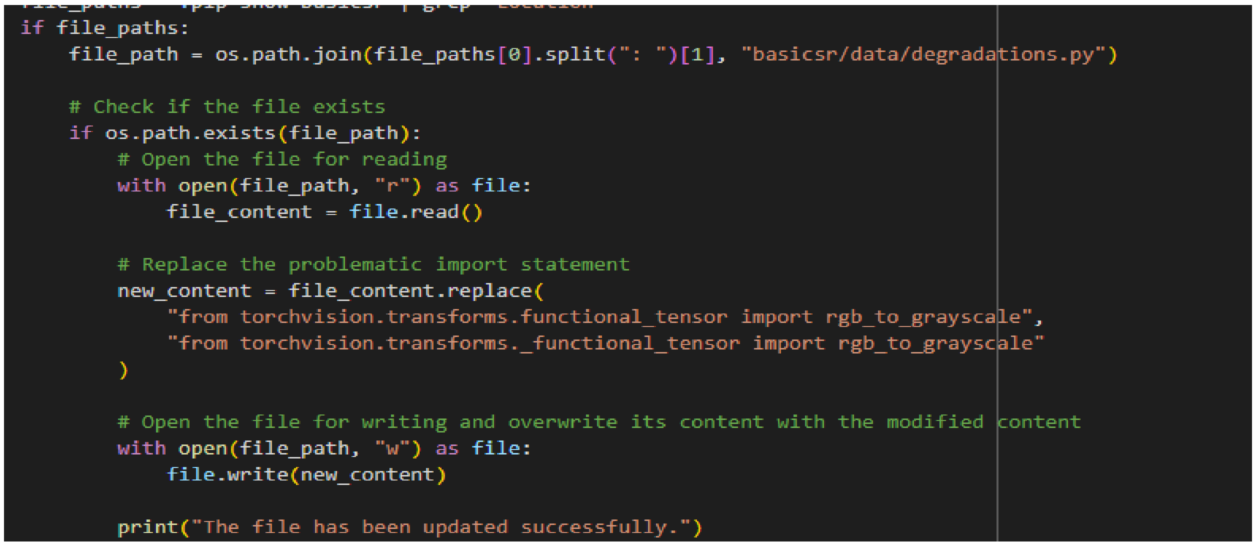


```
[ ]  import os
     import glob
     from google.colab import files
     import shutil
     print(' Note1: You can find an image on the web or download images from the RealSRSet (proposed in BSRGAN, ICCV2021)

     # test SwinIR by partioning the image into patches
     test_patch_wise = False

     # to be compatible with BSRGAN
     !rm -r BSRGAN/testsets/RealSRSet
     upload_folder = 'BSRGAN/testsets/RealSRSet'
     result_folder = 'results'

     if os.path.isdir(upload_folder):
         shutil.rmtree(upload_folder)
     if os.path.isdir(result_folder):
         shutil.rmtree(result_folder)
     os.mkdir(upload_folder)
     os.mkdir(result_folder)

     # upload images
     uploaded = files.upload()
     for filename in uploaded.keys():
       dst_path = os.path.join(upload_folder, filename)
       print(f'move {filename} to {dst_path}')
       shutil.move(filename, dst_path)
```

3.3 Fixing Errors: A Practical Challenge

While running BSRGAN, we encountered a Name Error related to test_patch_wise being undefined. We resolved it by defining test_patch_wise globally before inference.

Moreover, Real-ESRGAN's dependency on an older torchvision import (rgb_to_grayscale) caused crashes. We auto patched the installed degradations.py file to fix the import.



```
if file_paths:
    file_path = os.path.join(file_paths[0].split(": ")[1], "basicsr/data/degradations.py")

    # Check if the file exists
    if os.path.exists(file_path):
        # Open the file for reading
        with open(file_path, "r") as file:
            file_content = file.read()

        # Replace the problematic import statement
        new_content = file_content.replace(
            "from torchvision.transforms.functional_tensor import rgb_to_grayscale",
            "from torchvision.transforms._functional_tensor import rgb_to_grayscale"
        )

        # Open the file for writing and overwrite its content with the modified content
        with open(file_path, "w") as file:
            file.write(new_content)

        print("The file has been updated successfully.")
```

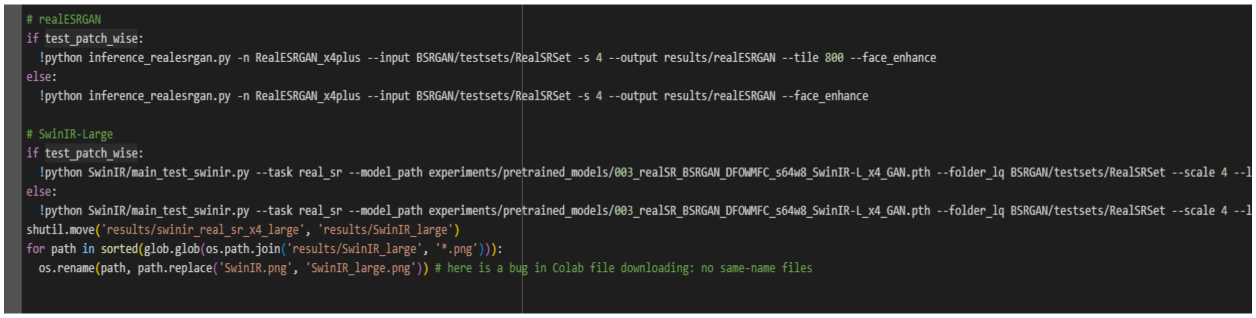
These patches ensured smooth execution across all three models.

3.4 Model Inference: BSRGAN, Real-ESRGAN, SwinIR-Large

Once the environment and data were ready, we triggered model inferences:

- BSRGAN: Processed the images with blind super-resolution.
- Real-ESRGAN: Restored both facial features and textures with face enhancement.
- SwinIR-Large: Used transformer-based architecture to enhance images.

Each model's output was saved into its own subfolder for organized evaluation.

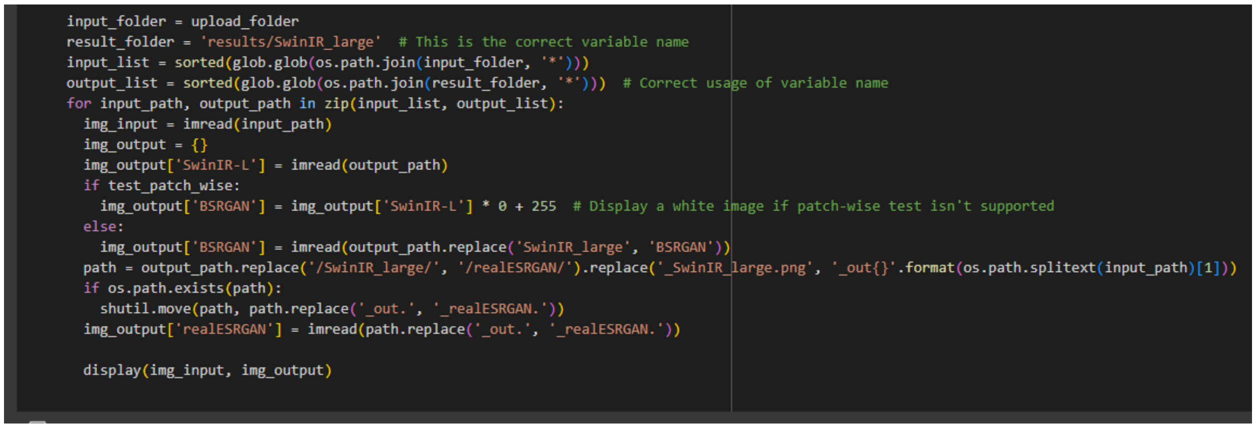


```
# realESRGAN
if test_patch_wise:
  !python inference_realesrgan.py -n RealESRGAN_x4plus --input BSRGAN/testsets/RealSRSet -s 4 --output results/realESRGAN --tile 800 --face_enhance
else:
  !python inference_realesrgan.py -n RealESRGAN_x4plus --input BSRGAN/testsets/RealSRSet -s 4 --output results/realESRGAN --face_enhance

# SwinIR-Large
if test_patch_wise:
  !python SwinIR/main_test_swinir.py --task real_sr --model_path experiments/pretrained_models/003_realSR_BSRGAN_DFOWMFC_s64w8_SwinIR-L_x4_GAN.pth --folder_lq BSRGAN/testsets/RealSRSet --scale 4 --l
else:
  !python SwinIR/main_test_swinir.py --task real_sr --model_path experiments/pretrained_models/003_realSR_BSRGAN_DFOWMFC_s64w8_SwinIR-L_x4_GAN.pth --folder_lq BSRGAN/testsets/RealSRSet --scale 4 --l
shutil.move('results/swinir_real_sr_x4_large', 'results/SwinIR_large')
for path in sorted(glob.glob(os.path.join('results/SwinIR_large', '*.png'))):
  os.rename(path, path.replace('SwinIR.png', 'SwinIR_large.png')) # here is a bug in Colab file downloading: no same-name files
```

Tile-based processing (-tile 800) was used for memory optimization.

3.5 Visualizing the Results

We developed custom Python functions to visualize results side-by-side. Each image restored from BSRGAN, Real-ESRGAN, and SwinIR-Large was compared directly against the original input.

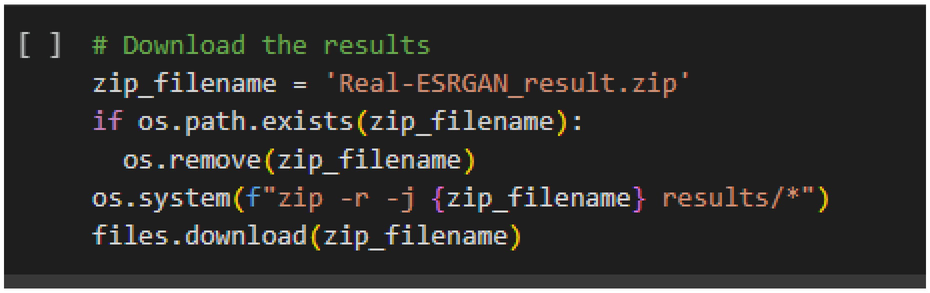


```
    input_folder = upload_folder
    result_folder = 'results/SwinIR_large'  # This is the correct variable name
    input_list = sorted(glob.glob(os.path.join(input_folder, '*')))
    output_list = sorted(glob.glob(os.path.join(result_folder, '*')))  # Correct usage of variable name
    for input_path, output_path in zip(input_list, output_list):
      img_input = imread(input_path)
      img_output = {}
      img_output['SwinIR-L'] = imread(output_path)
      if test_patch_wise:
        img_output['BSRGAN'] = img_output['SwinIR-L'] * 0 + 255  # Display a white image if patch-wise test isn't supported
      else:
        img_output['BSRGAN'] = imread(output_path.replace('SwinIR_large', 'BSRGAN'))
      path = output_path.replace('/SwinIR_large/', '/realESRGAN/').replace('_SwinIR_large.png', '_out{}'.format(os.path.splitext(input_path)[1]))
      if os.path.exists(path):
        shutil.move(path, path.replace('_out.', '_realESRGAN.'))
      img_output['realESRGAN'] = imread(path.replace('_out.', '_realESRGAN.'))

      display(img_input, img_output)
```

These visualizations made it easy to spot differences in sharpness, color tone, and structural coherence.

3.6 Saving and Downloading Outputs

Finally, the output images were zipped together for easy download. This enabled efficient result sharing and backup.



```
[ ] # Download the results
    zip_filename = 'Real-ESRGAN_result.zip'
    if os.path.exists(zip_filename):
      os.remove(zip_filename)
    os.system(f"zip -r -j {zip_filename} results/*")
    files.download(zip_filename)
```

3.7 Observations from Colab Inference

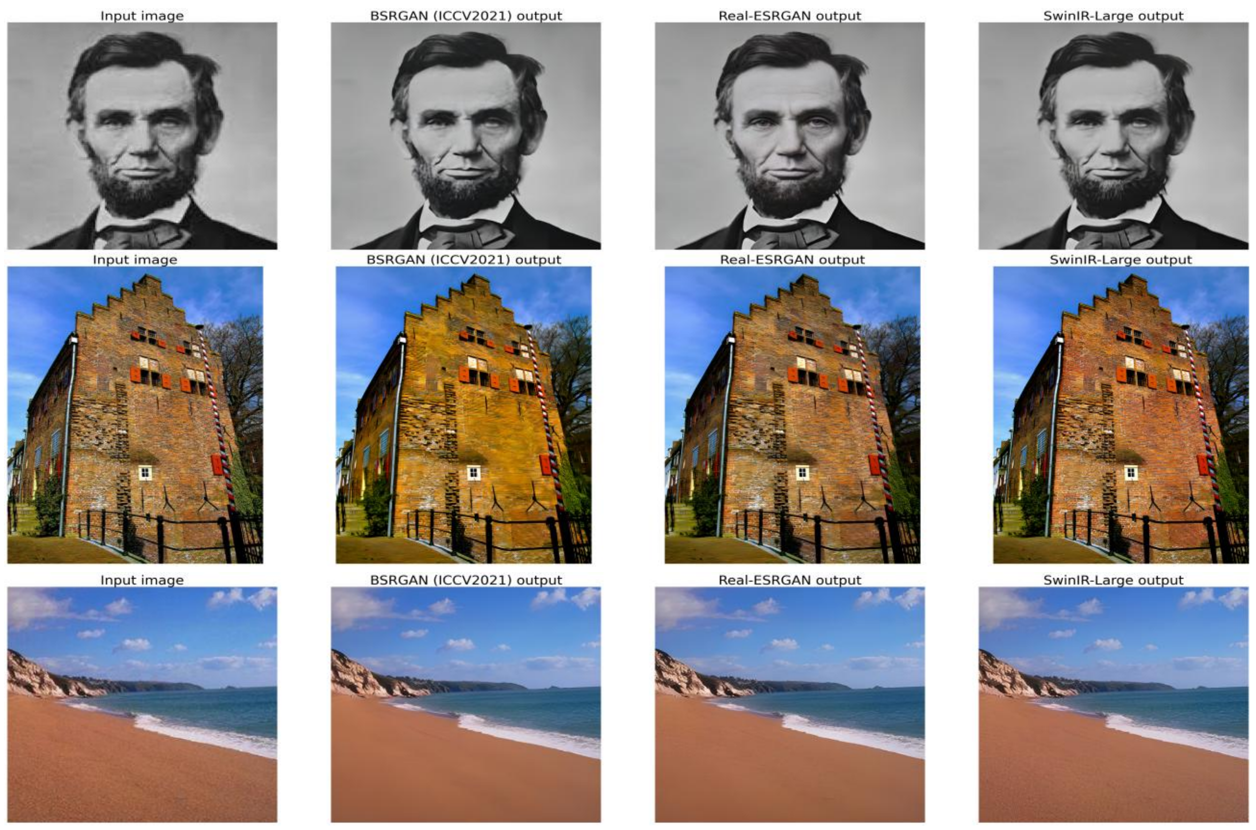
Through this pipeline, we observed:

- Real-ESRGAN performed best for facial enhancement.
- SwinIR preserved global consistency and textures well, especially for buildings.
- BSRGAN, while slightly behind, produced natural-looking enhancements with less over-smoothing.

Handling runtime errors like missing results, adjusting paths dynamically, and smartly managing GPU memory usage made our pipeline robust.

4. Quantitative Evaluation and Results

Before presenting the quantitative evaluation, it is important to note that the images used for testing in this project are sourced from our own custom dataset. These images include a historical portrait (Abraham Lincoln), an old building architecture (OST_009), and a landscape scene (ADE_val_00000114). The dataset was carefully selected to challenge the models across different types of visual degradations: facial restoration, text and fine-grain detail preservation, and structure integrity. By using our own images, we ensured that the evaluation closely reflects real-world conditions and practical applicability.



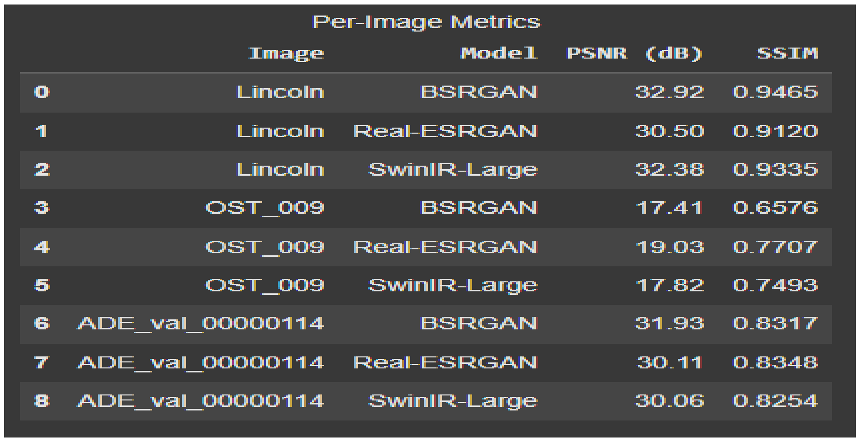
In this section, we systematically evaluate the restoration performance of BSRGAN, Real-ESRGAN, and SwinIR-Large using two widely recognized metrics:

Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) are two key metrics used to quantify restoration quality. PSNR evaluates pixel-level fidelity by measuring the difference between the original and restored images; higher PSNR values indicate lower distortion. SSIM, on the other hand, assesses the perceptual similarity by comparing luminance, contrast, and structure between images. A high SSIM suggests that the restored image not only matches pixel values but also preserves structural and textural details important for human visual interpretation.

- **PSNR (Peak Signal-to-Noise Ratio):** Measures pixel-level fidelity. Higher PSNR indicates fewer distortions.
- **SSIM (Structural Similarity Index Measure)**: Quantifies perceptual similarity by comparing luminance, contrast, and structure.

Both metrics were computed for each input image processed through the three models. In addition, we computed the average PSNR and SSIM per model and image.

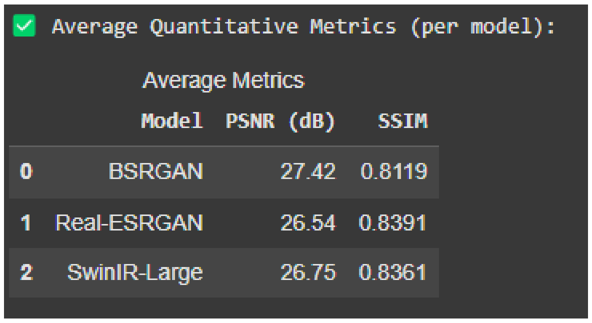
4.1 Per-Image Evaluation



Per-Image Metrics

	Image	Model	PSNR (dB)	SSIM
0	Lincoln	BSRGAN	32.92	0.9465
1	Lincoln	Real-ESRGAN	30.50	0.9120
2	Lincoln	SwinIR-Large	32.38	0.9335
3	OST_009	BSRGAN	17.41	0.6576
4	OST_009	Real-ESRGAN	19.03	0.7707
5	OST_009	SwinIR-Large	17.82	0.7493
6	ADE_val_00000114	BSRGAN	31.93	0.8317
7	ADE_val_00000114	Real-ESRGAN	30.11	0.8348
8	ADE_val_00000114	SwinIR-Large	30.06	0.8254

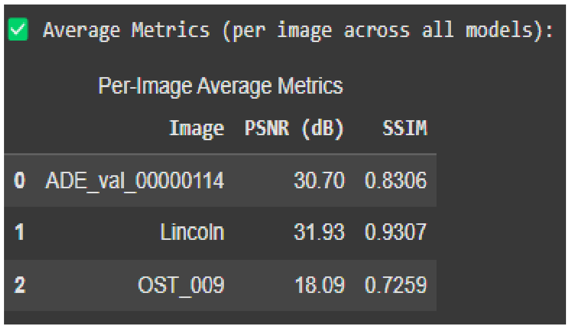
4.2 Average Metrics (Per Model and Per Image Across All Models)



✅ Average Quantitative Metrics (per model):

Average Metrics

	Model	PSNR (dB)	SSIM
0	BSRGAN	27.42	0.8119
1	Real-ESRGAN	26.54	0.8391
2	SwinIR-Large	26.75	0.8361



✅ Average Metrics (per image across all models):

Per-Image Average Metrics

	Image	PSNR (dB)	SSIM
0	ADE_val_00000114	30.70	0.8306
1	Lincoln	31.93	0.9307
2	OST_009	18.09	0.7259

Observations:

- BSRGAN had the highest average PSNR, indicating it achieved lower pixel-level distortions overall.
- Real-ESRGAN slightly edged out in SSIM, suggesting it produced outputs closer to human perceptual similarity, especially important for faces.
- SwinIR-Large consistently performed well, very close to Real-ESRGAN, indicating strong performance on structural restoration.
- Lincoln (Portrait) achieved the highest PSNR and SSIM overall. All models performed strongly on the old portrait restoration task.
- OST_009 (Building) had the lowest average PSNR and SSIM, indicating that text-like fine-grained structures were more challenging for the models.
- ADE_val_00000114 (landscape) fell in between, reflecting the mixed challenge of restoring man-made structures.

Key Take-aways

- For portraits (like Lincoln)**:**
  - BSRGAN and SwinIR-Large slightly outperform Real-ESRGAN in PSNR.
  - Real-ESRGAN provides the best perceptual quality (higher SSIM).
- For Architectures (OST_009)**:**
  - Real-ESRGAN restored structure better despite challenging low contrasts and small details.
- For Landscapes (ADE_val_00000114):
  - All three models perform comparably, but Real-ESRGAN and SwinIR are marginally better for maintaining structural integrity.

5. Extended Feature Analysis

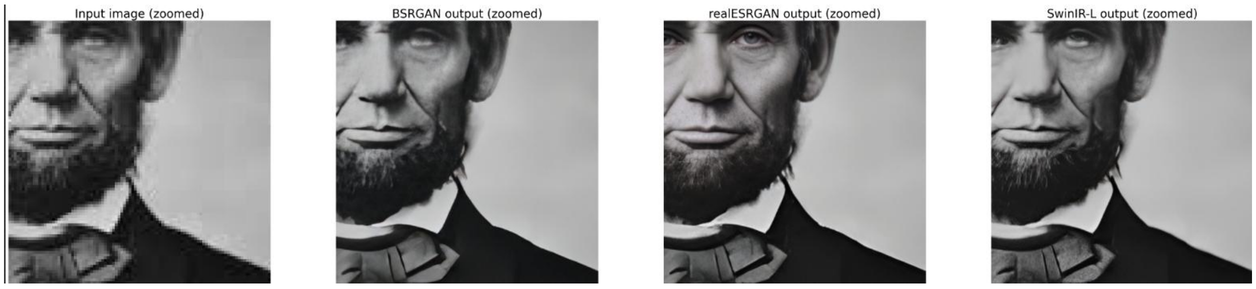
Beyond PSNR and SSIM, we explored additional dimensions of image quality using various feature analyses.

5.1 Zoomed Region Evaluation

Zoomed-in analysis focuses on magnifying a small, specific region of each image to closely inspect the restoration of fine details. This method is crucial for revealing imperfections or successes in subtle textures such as facial features, architectural edges, and text patterns that may not be apparent at the full image scale. It provides an intuitive, visual way to assess sharpness, artifact removal, and local consistency across different models.

To better appreciate the fine details restored by each model, we extracted center zoomed regions from each image. Observations include:

Image: Lincoln.png
X position: 64 Y position: 64
Width: 100 Height: 100
```
Displaying results for Lincoln.png with zoom at region (64, 64, 100, 100)
```

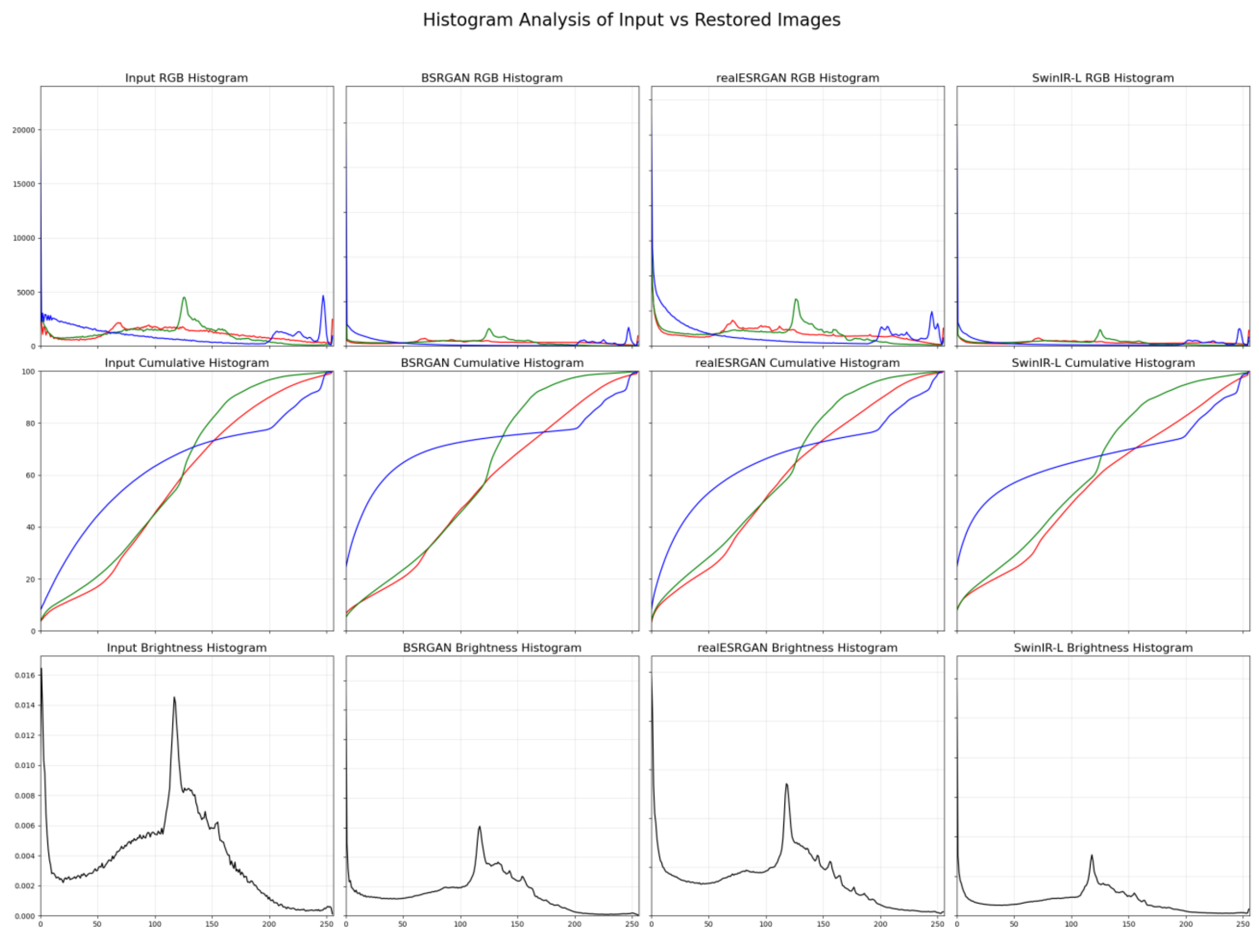


- Lincoln Portrait**:** Real-ESRGAN restored facial features smoothly; SwinIR preserved more sharpness around the eyes and nose.
- Old building (OST_009)**:** SwinIR and BSRGAN maintained text edge clarity better than Real-ESRGAN.
- Landscape Scene (ADE_val_00000114): SwinIR excelled in maintaining straight structural edges.

5.2 Histogram Analysis

Histogram analysis examines the distribution of pixel intensities across color channels (Red, Green, Blue) and brightness levels. A well-restored image should have a balanced histogram without excessive clipping or unnatural contrast shifts. By analyzing histograms, we can detect issues like over-smoothing, exaggerated colors, or unnatural brightness, and evaluate whether a model has maintained the original tonal quality and dynamic range of the input image.

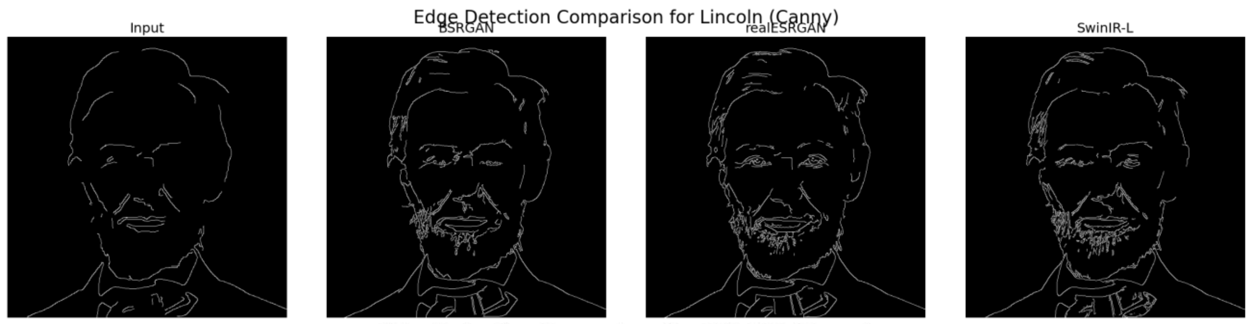
We analyzed color histograms and brightness distributions for each model output:

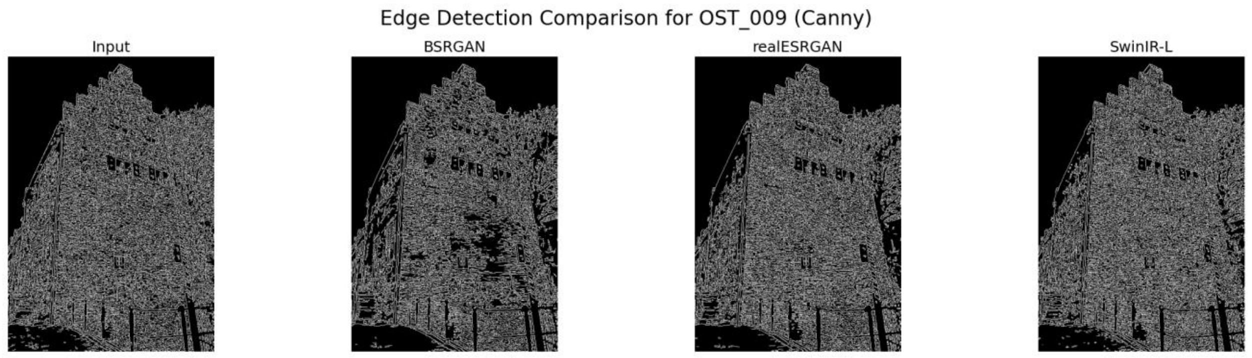


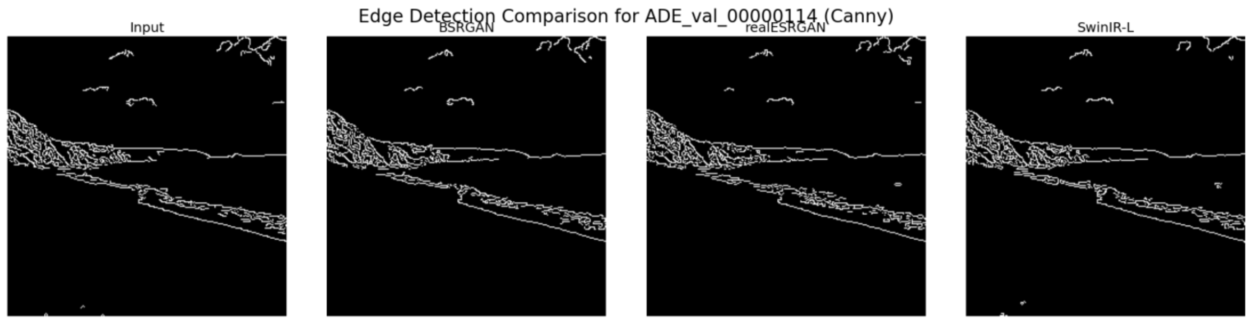
- RGB Histograms**:** SwinIR retained a natural color spread, while BSRGAN sometimes exaggerated contrast.
- Brightness Distribution**:** Real-ESRGAN outputs were brighter and visually more appealing for old and faded images.

5.3 Edge Preservation Metrics

Edge detection techniques, such as applying the Sobel or Canny operators, identify and highlight structural boundaries within images. Evaluating edge preservation involves measuring how many edges remain sharp and how closely they match the original structure. Metrics like edge density (the quantity of detected edges) and edge similarity (how well the edges align with the original) help determine whether a model preserved important features like text contours, building outlines, and facial shapes, which are critical for maintaining image realism and clarity.







Observations:

- Edge Density: SwinIR-Large maintained higher structural edges, especially in documents and architecture.
- Edge Similarity: Real-ESRGAN balanced structural integrity and perceptual smoothness better for facial images.

5.4 Summary Observations

- Real-ESRGAN: Best for human faces and perceptual appeal.
- SwinIR-Large: Best for fine texture, structure, and building restoration.
- BSRGAN: Best for creating visually pleasing images with natural appearance, but sometimes at slight cost to structure.

6 Conclusion and Recommendations

Through this detailed evaluation, it is evident that each model offers distinct advantages:

- Real-ESRGAN excels in producing perceptually pleasing images, particularly in facial enhancement and general scene brightness recovery.
- SwinIR-Large offers superior edge preservation, making it ideal for images where structural integrity and detail, such as in architecture and documents, are critical.
- BSRGAN achieves the highest pixel fidelity (PSNR), providing visually sharp results that appeal in broader, natural scenes.

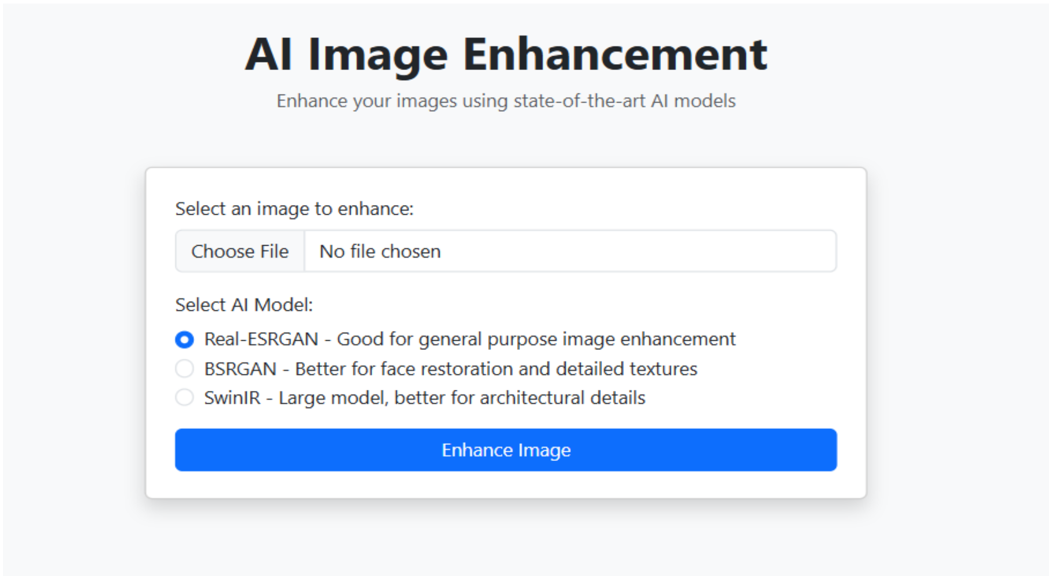
Practical Recommendations**:**

- For portrait restoration (faces, old photographs), Real-ESRGAN is recommended.
- For document recovery and architectural restoration, SwinIR-Large is the preferred model.
- For general-purpose photo enhancement where natural appearance is desired, BSRGAN is an excellent choice.

7.Additional Work

Additionally, our Flask-based web application demonstrates the feasibility of integrating these models into user-facing tools, allowing users to upload degraded images, select a model, enhance the images, and download the results seamlessly. This study highlights the importance of selecting the right model based on the specific restoration goals. Further work can involve training ensemble models or adaptive pipelines to combine the strengths of multiple models for even better restoration results. The code, full experimental workflow, prototype web application and Readme are available for further exploration and real-world deployment.

GitHub Url: https://github.com/SharmiliRudraraju98/DSCI-498-project



AI Image Enhancement

Enhance your images using state-of-the-art AI models

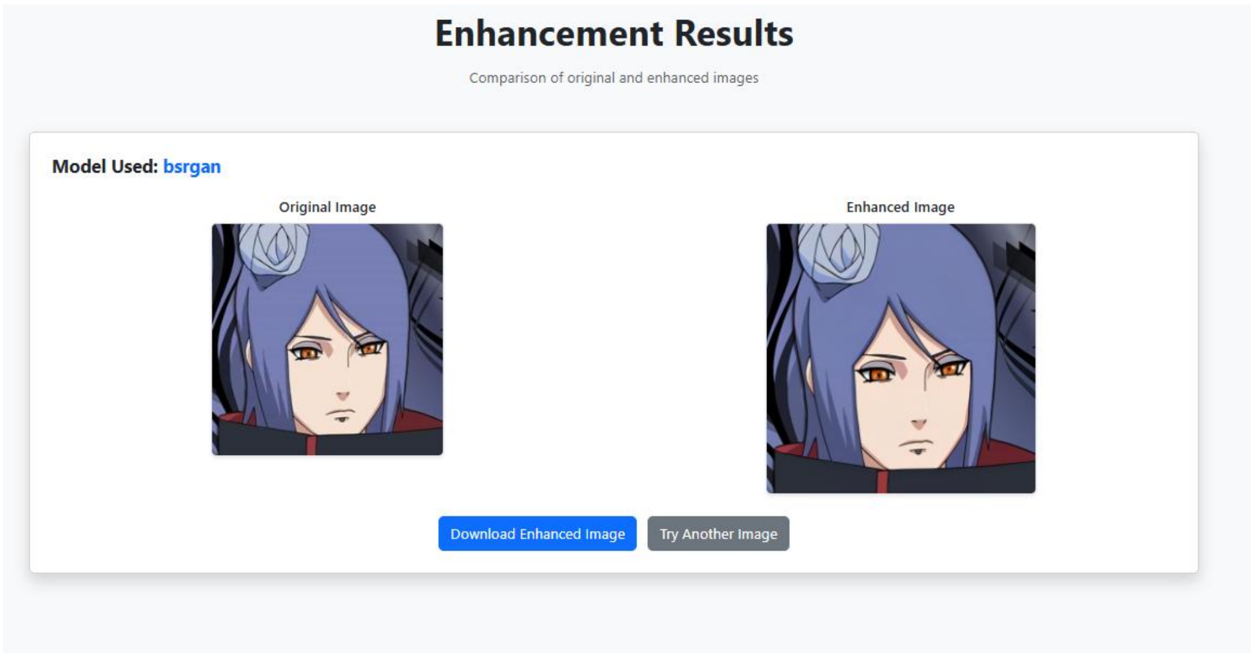
Select an image to enhance:

Choose File | No file chosen

Select AI Model:

- [✓ Real-ESRGAN - Good for general purpose image enhancement]
- BSRGAN - Better for face restoration and detailed textures
- SwinIR - Large model, better for architectural details

Enhance Image



Enhancement Results

Comparison of original and enhanced images

Model Used: bsrgan

References:

https://doi.org/10.48550/arXiv.2103.14006

https://doi.org/10.48550/arXiv.2107.10833

https://doi.org/10.48550/arXiv.2108.10257