2024S-T3 BDM 3035 - Big Data Capstone Project 01
(DSMM Group 1 & Group 3)

Milestone Report 5

AI Image Remastering

Guide
Meysam Effati

Group E

Kuncheria Tom	C0900973
Prince Thomas	C0894907
Remya Kannan Kandari	C0895293
Shanmuga Priyan Jeevanandam	C0889053
Sravya Somala	C0907007

Development and Deployment of Web User Interface using Flask

1. Introduction

In our ongoing effort to enhance and streamline the image restoration project utilizing the Real-ESRGAN model, we have developed a web user interface (UI) using Flask. This milestone report outlines the technical steps, challenges faced, and solutions implemented in creating a locally deployable Flask-based web application. The UI facilitates users to upload images, select scaling factors, and compare the original and restored images using a slider for a seamless and interactive experience. Additionally, we experimented with public deployment options, including Docker and Hugging Face, and documented the technical hurdles encountered.

2. Technical Implementation

UI Development with Flask

The web user interface was designed to be user-friendly and visually appealing. Below are the core components and technical details of the implementation:

1. **HTML and CSS for Frontend Design:**
    - The UI was crafted using HTML for the structure and CSS for styling.
    - A fixed header was added to provide navigation options like "Contact Developer," "Privacy Policy," and "Help" via a hamburger menu.
    - The main container holds the form for uploading images, selecting scaling factors, and buttons for initiating the enhancement process.
    - Buttons were styled with gradients and hover effects for better visual feedback. The icons were added to buttons for enhanced usability.
    - A comparison slider was implemented using the img-comparison-slider library to allow users to compare the original and restored images.
2. **JavaScript for Dynamic Interactions:**
    - JavaScript was used to handle file uploads, button interactions, and form submissions.
    - The file input dynamically updates the displayed file name and resets the results when a new image is uploaded.
    - The anime toggle button dynamically updates its state and affects the scaling options available.

- The selected scaling factor is highlighted, and buttons are enabled or disabled based on the anime toggle state.
- AJAX requests handle the form submission, ensuring the UI remains responsive during the image processing.

3. **Flask for Backend Processing:**
   - Flask was chosen for its flexibility and ability to handle complex backend processing while serving a customized frontend.
   - The main route (/) renders the index.html page.
   - The /upscale route handles image processing. It accepts POST requests containing the image file and selected scaling factor, then processes the image using the Real-ESRGAN model.
   - The load_model function dynamically loads the appropriate model weights based on the selected scale and anime toggle state. This function ensures the correct model is used for different upscaling factors.
   - The processed image is saved and the original and restored images are returned as JSON to the frontend for display.

3. Challenges Faced

1. **Customization Limitations of Streamlit:**
   - Initial attempts to use Streamlit for the UI were abandoned due to its limited customization options. Streamlit's simplicity is advantageous for quick prototypes, but it lacked the flexibility needed for our project's specific requirements.
   - The inability to implement a comparison slider and other advanced UI elements necessitated a shift to Flask, where HTML, CSS, and JavaScript provided the needed control.
2. **Integration of Comparison Slider:**
   - Implementing the comparison slider required careful handling of image loading and ensuring the slider functioned smoothly across different devices and screen sizes.
   - Ensuring the images were loaded before initializing the slider to avoid errors and display issues was a technical hurdle successfully overcome with JavaScript.
3. **Handling Large Model Files in Deployment:**
   - Public deployment options were explored, with Hugging Face providing a suitable environment with 16 GB RAM in the free tier. However, Flask is not natively supported on Hugging Face.
   - A Docker container was created to encapsulate the application, but the large size of the model files (~10 GB) caused runtime errors during deployment.
   - Efforts are ongoing to resolve these deployment issues, with the possibility of reverting to a less customized UI on Streamlit if necessary.

4. Performance Considerations

The performance of the application varies significantly based on the hardware used. On CPU-only systems, the image processing time is considerably longer compared to systems with dedicated GPUs. The model's computational demands highlight the importance of hardware acceleration for real-time or near-real-time applications.

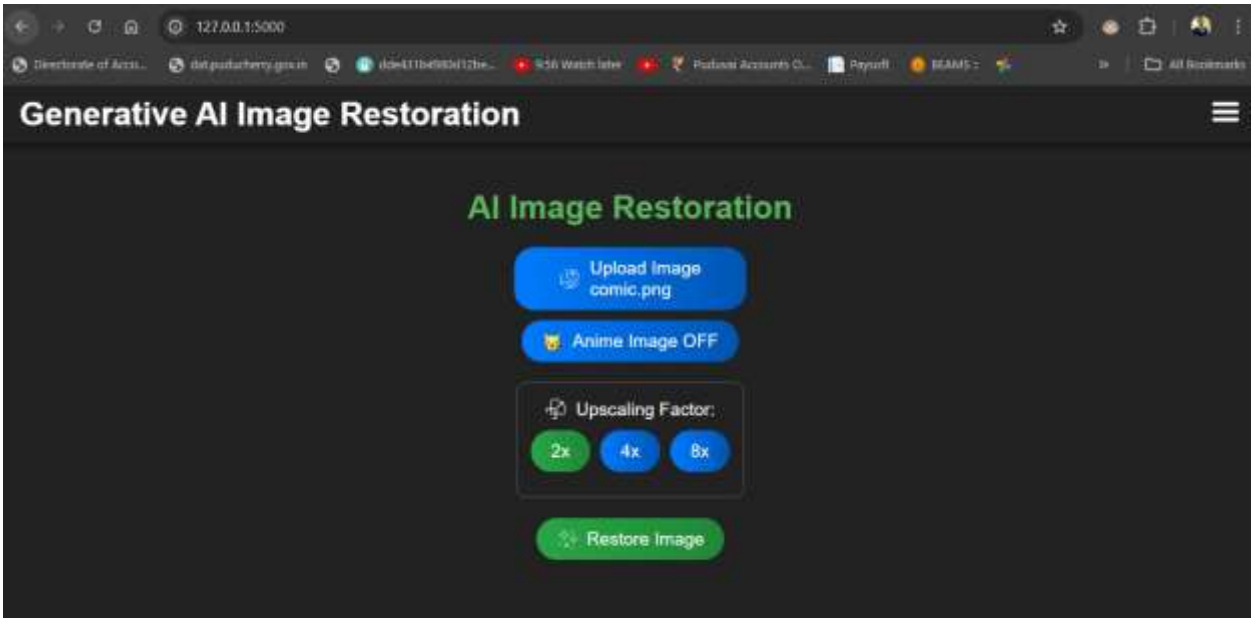
5. Additional Features

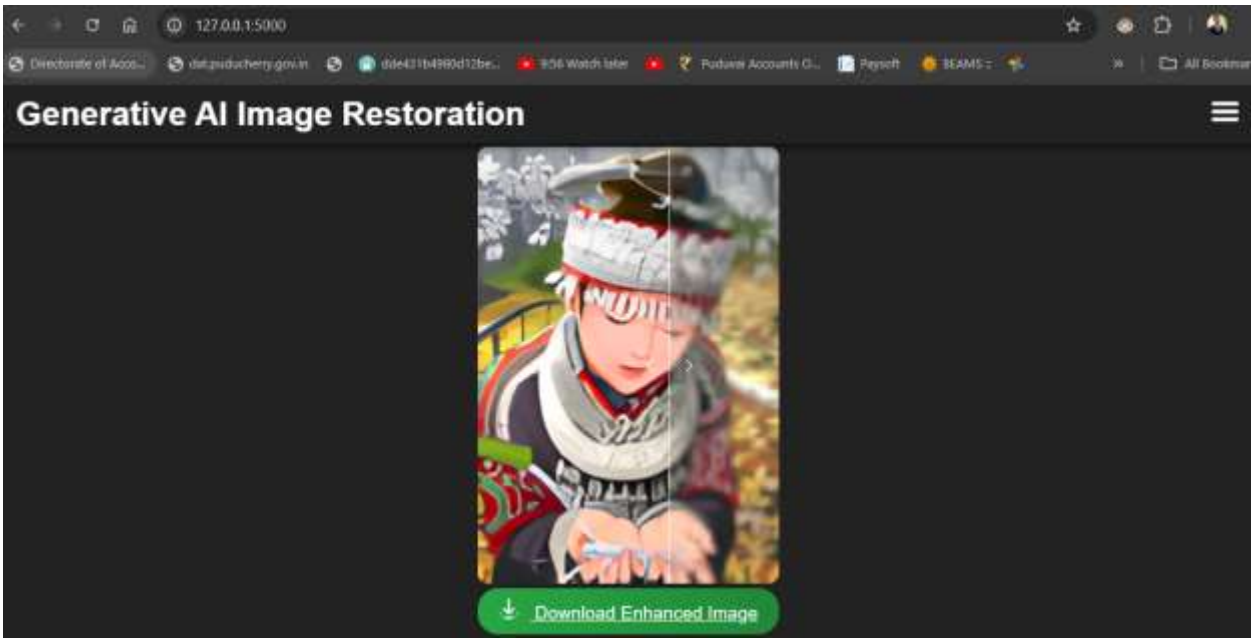
1. **Dynamic Button States:**
    - The anime toggle button and scaling factor buttons dynamically update based on user interactions, providing visual feedback and ensuring the correct processing path is taken.
2. **Responsive Design:**
    - The UI is designed to be responsive, ensuring usability across different devices and screen sizes. The use of flexbox for layout management and media queries for responsiveness ensures a consistent user experience.
3. **Enhanced User Experience:**
    - Visual effects such as button gradients, hover states, and icon usage enhance the overall user experience. The dark-themed UI with light text ensures readability and reduces eye strain.

6. Future Work

1. **Public Deployment:**
    - Resolving the issues with Docker deployment on Hugging Face remains a priority. If successful, this will provide a robust public-facing version of the application.
    - Alternatively, exploring other cloud platforms or reverting to Streamlit with basic UI for public deployment may be necessary.
2. **Optimization:**
    - Further optimization of the image processing pipeline to reduce processing times on CPU-only systems.
    - Exploring potential GPU-accelerated deployment options for faster processing.
3. **User Feedback Integration:**
    - Gathering user feedback to further refine the UI and functionality.
    - Implementing additional features based on user requests, such as batch processing or additional scaling options.

7. Working Sample Webpage





8. Conclusion

The development of a Flask-based web user interface marks a significant milestone in our image restoration project. Overcoming the challenges of UI customization and deployment has paved the way for a more robust and user-friendly application. The ability to dynamically interact with the application, select scaling factors, and compare results in real-time enhances the user experience and provides valuable insights into the model's performance. Moving forward, resolving deployment issues and optimizing performance will be key focus areas to ensure the application meets the needs of a broader user base.

9. Reference

https://flask.palletsprojects.com/en/3.0.x/

https://developer.mozilla.org/en-US/docs/Web/JavaScript

https://www.tutorialspoint.com/ajax/index.htm

https://huggingface.co/docs

GitHub Link: https://github.com/shanmugapriyan357/Big-Data-Capstone-Project-AI-Image-Remastering-/tree/Dev