

---

# DETECT PIXELATED IMAGE AND CORRECT IT

---

---

# PROBLEM STATEMENT:

Pixelation in digital images occurs when images are enlarged beyond their original resolution, resulting in visible blocky artifacts. This degradation in image quality hinders the visual appeal and usability of images, especially in fields like photography, digital art, and medical imaging. Detecting and correcting pixelation is crucial for maintaining image fidelity and ensuring high-quality visual outputs

## Challenges

1. **Visual Quality Degradation:** Pixelation diminishes image clarity and sharpness, affecting the overall quality.
  2. **Automatic Detection Requirement:** Manually identifying pixelated images is impractical for large datasets or real-time applications.
  3. **Effective Correction Techniques:** Simply scaling images up can worsen pixelation, necessitating advanced techniques for effective correction.
-

---

# IDEA/SOLUTION:

The proposed solution involves developing a two-stage system that integrates deep learning-based pixelation detection and advanced super-resolution techniques for image correction. The system will leverage Convolutional Neural Networks (CNNs) for detecting pixelation and Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN) for enhancing the image quality. This approach ensures that images are not only identified for pixelation but also corrected to a high degree of visual fidelity. The project has two parts :

## **Pixelation Detection:**

1. **Model:** A CNN-based model is used to classify images as pixelated or not.
2. **Input:** Preprocessed images resized to a consistent dimension.
3. **Output:** A binary classification indicating whether the image is pixelated.

## **Image Correction:**

1. **Model:** ESRGAN model, a state-of-the-art super-resolution technique, is utilized.
  2. **Input:** Detected pixelated images.
  3. **Output:** High-resolution images with improved clarity and detail
-

# COMPARISON BETWEEN DIFFERENT CLASSIFICATION MODELS FOR PIXELATION DETECTION:

Logistic Regression:

Accuracy: 0.5740740740740741

F1 Score: 0.5855855855855856

Random Forest:

Accuracy: 0.6458333333333334

F1 Score: 0.6046511627906976

Support Vector Machine:

Accuracy: 0.6574074074074074

F1 Score: 0.5865921787709497

14/14 ————— 0s 15ms/step

Pre-trained Keras Model:

Accuracy: 0.7337962962962963

F1 Score: 0.6666666666666666

# FEATURES OFFERED:

**1. Automated Pixelation Detection:** Automatically identify whether an image is pixelated.

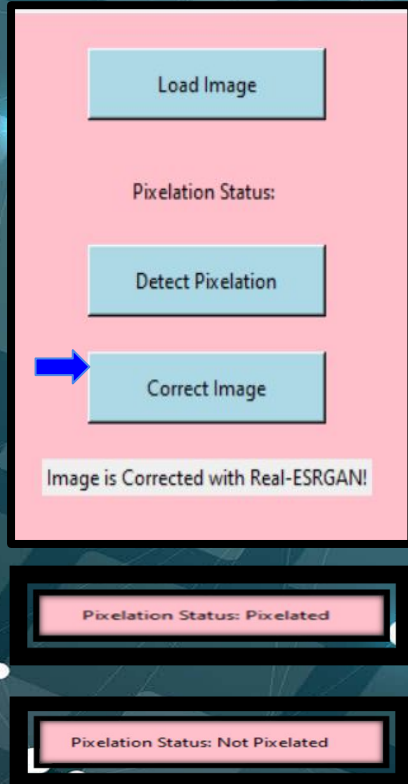
**Description:** This feature automatically detects whether an uploaded image is pixelated or not.

- **How it works:**

- The image is preprocessed to match the input requirements of the detection model.
- The preprocessed image is fed into a Convolutional Neural Network (CNN) model specifically trained to classify images as pixelated or not.
- The system outputs a binary classification (pixelated or not pixelated).

- **Benefits:**

- Eliminates the need for manual inspection of images.
- Provides quick and accurate detection of pixelated images, saving time and effort.
- Ensures consistency and reliability in identifying pixelation.



# FEATURES OFFERED:

## 2. High-Quality Image Correction

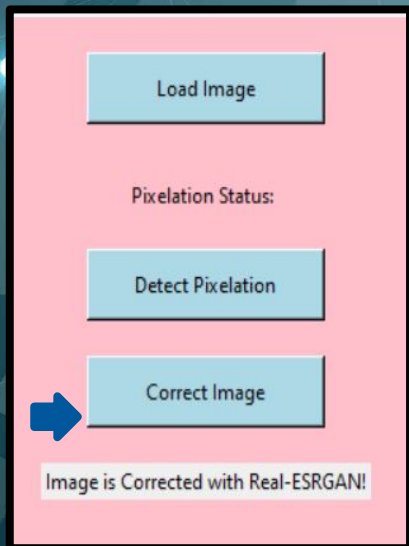
**Description:** This feature corrects detected pixelated images using advanced super-resolution techniques to improve image clarity and detail.

- **How it works:**

- The system employs an Enhanced Super-Resolution Generative Adversarial Network (ESRGAN) model for image enhancement.
- Once an image is detected as pixelated, it is processed by the ESRGAN model to generate a high-resolution version.
- The corrected image retains the original size but with enhanced details and reduced pixelation.

- **Benefits:**

- Significantly improves the visual quality of pixelated images.
- Preserves important details and colors in the image.
- Utilizes state-of-the-art super-resolution techniques to provide the best possible enhancement



---

# FEATURES OFFERED:

## 3. User-Friendly Interface

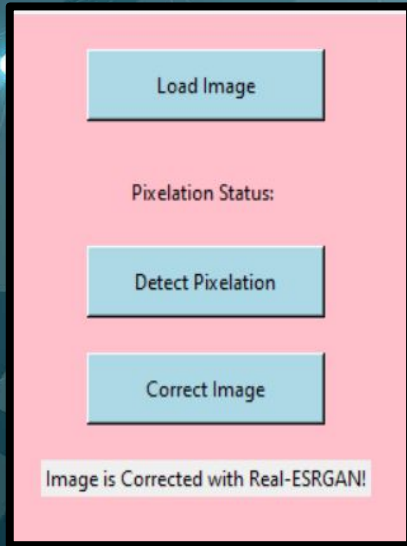
**Description:** The system offers an intuitive and easy-to-use graphical user interface (GUI) for interacting with the application.

- **How it works:**

- The GUI is built using Tkinter, providing buttons for image loading, detection, and correction.
- Users can upload images, view pixelation status, and see the corrected images through a simple interface.
- The interface includes labels and buttons that guide users through the process.

- **Benefits:**

- Makes the system accessible to users with minimal technical knowledge.
- Provides a seamless user experience with clear instructions and feedback.
- Enhances user engagement and satisfaction.



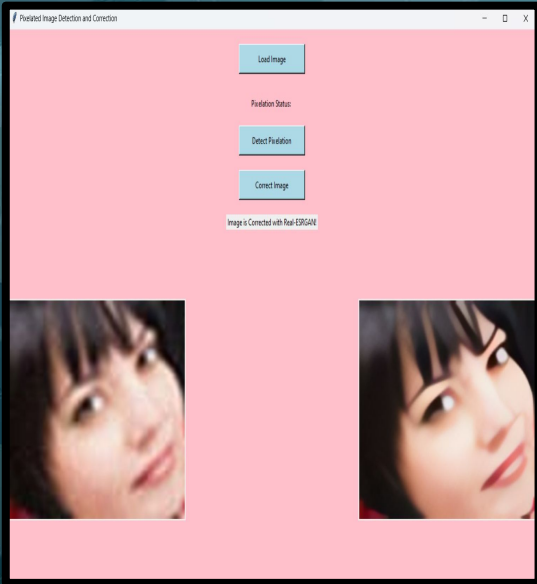


# FEATURES OFFERED:

## 4.Real-Time Image Display

**Description:** The system displays both the original and corrected images in real-time.

- **How it works:**
  - Upon uploading an image, it is displayed in the GUI.
  - After correction, the enhanced image is shown alongside the original for comparison.
  - The images are resized for display purposes using PIL (Pillow).
- **Benefits:**
  - Allows users to immediately see the effects of pixelation correction.
  - Provides a side-by-side comparison of the original and enhanced images.
  - Helps users assess the improvement in image quality.







# FEATURES OFFERED:

## 5. Save Functionality

**Description:** Users can save the corrected images for future use.

- **How it works:**
  - The corrected images can be saved locally through the GUI.
  - The system provides options to specify the save location and file name.
  - The saved images retain the enhancements made by the ESRGAN model.
- **Benefits:**
  - Enables users to keep the enhanced images for further use.
  - Provides flexibility in managing and storing corrected images.
  - Ensures that users have access to high-quality images for their needs.

 corrected_image_esrgan	12-07-2024 16:14	JPG File	25 KB
 corrected_image_real_esrgan	12-07-2024 16:14	JPG File	20 KB

---

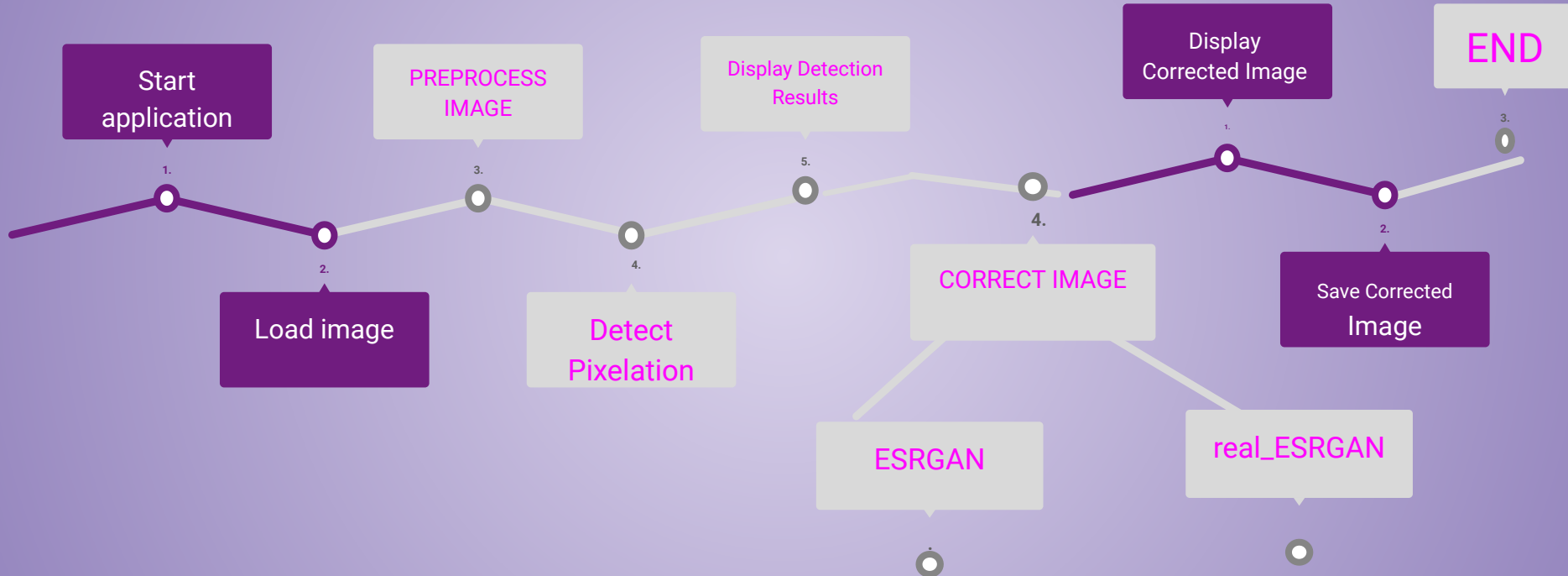
# FEATURES OFFERED:

## 6. Integration of Multiple Super-Resolution Techniques

**Description:** The system integrates both ESRGAN and Real-ESRGAN for image correction.

- **How it works:**
    - The ESRGAN model is used for initial image enhancement.
    - For further refinement, the Real-ESRGAN executable is called to process the image.
    - Users can choose which method to apply based on their preferences.
  - **Benefits:**
    - Offers multiple options for image enhancement, providing flexibility and choice.
    - Ensures the highest possible quality of corrected images.
    - Leverages the strengths of different super-resolution techniques for optimal results.
-

# PROCESS FLOW:



# ARCHITECTURE DIAGRAM



**01**

**User Interface**  
Tkinter GUI with PIL and  
Matplotlib

**02**

**Image Loading**  
File Dialog and Display

**03**

**Preprocessing**  
OpenCV and NumPy

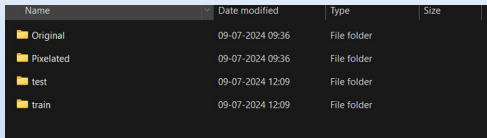
**04**

**Pixelation Detection**  
Trained Keras Model

**05**

**Image Correction**  
ESRGAN - pytorch  
Real-ESRGAN - Subprocess |

# DATA SET:



Name	Date modified	Type	Size
Original	09-07-2024 09:36	File folder	
Pixelated	09-07-2024 09:36	File folder	
test	09-07-2024 12:09	File folder	
train	09-07-2024 12:09	File folder	

The dataset used for this project comprises a total of 2490 images, meticulously categorized into 1245 pixelated and 1245 non-pixelated images. This balanced distribution ensures that the model is trained on an equal representation of both pixelated and non-pixelated scenarios, enabling it to learn effectively across diverse image types.

**Diversity of Content:** The dataset encompasses a wide variety of subjects and themes, spanning multiple categories to provide a comprehensive training experience:

- **Text and Alphabets:** Images containing various fonts, languages, and textual styles.
- **Written Texts:** Handwritten notes, printed documents, and typewritten materials.
- **Birds and Animals:** Photos capturing different species in natural habitats and artificial settings.
- **Animations and Games:** Screenshots and graphics from animated films, cartoons, and video games.
- **Humans:** Portraits, group photos, and candid snapshots of people from different demographics and backgrounds.
- **Nature:** Landscapes, flora, fauna, and natural phenomena showcasing the beauty of the environment.
- **Religion:** Iconography, symbols, and places of worship representing diverse religious beliefs.
- **Schools and Education:** Images depicting classrooms, educational materials, school events, and academic settings.

# TECHNOLOGIES USED:

This project leverages a combination of various technologies to achieve the goals of detecting and correcting pixelated images. Below is a detailed description of each technology used:

## 1. Tkinter (Python's Standard GUI Library)

- **Purpose:** Used for creating the graphical user interface (GUI) of the application.
- **Features:**
  - Provides a simple way to create windows, dialogs, buttons, labels, and other GUI elements.
  - Easy integration with other Python libraries.
  - Allows for event-driven programming (e.g., responding to button clicks).
- **Usage in the Project:**
  - Building the main application window.
  - Adding buttons for image loading, detection, and correction.
  - Displaying images and pixelation status.
  - Integrating Matplotlib charts for visualization.

## 2. PIL (Python Imaging Library) / Pillow

- **Purpose:** Used for opening, manipulating, and displaying images in Python.
- **Features:**
  - Supports a wide range of image file formats.
  - Provides functionalities for image resizing, cropping, and other transformations.
  - Easy integration with Tkinter for image display.
- **Usage in the Project:**
  - Loading and displaying the original and corrected images in the GUI.

# TECHNOLOGIES USED:

## 3. OpenCV (Open Source Computer Vision Library)

- **Purpose:** Used for image preprocessing and basic image manipulations.
- **Features:**
  - Comprehensive library for computer vision tasks.
  - Provides tools for image processing, such as resizing, color space conversion, and filtering.
  - Supports real-time image processing.
- **Usage in the Project:**
  - Reading images from disk.
  - Converting images to different color spaces (e.g., BGR to RGB).
  - Resizing images for preprocessing before feeding them to the detection model.

## 4. NumPy

- **Purpose:** Used for numerical operations and handling multi-dimensional arrays.
- **Features:**
  - Efficient manipulation of large arrays and matrices.
  - Provides mathematical functions for operations on arrays.
  - Integrates seamlessly with other scientific libraries in Python.
- **Usage in the Project:**
  - Handling image data as arrays.
  - Performing preprocessing operations such as normalization and reshaping



# TECHNOLOGIES USED

## 5.Keras (with TensorFlow backend)

- **Purpose:** Used for building and loading the pixelation detection neural network model.
- **Features:**
  - High-level neural networks API, capable of running on top of TensorFlow.
  - Simplifies the creation and training of deep learning models.
  - Provides tools for model evaluation and prediction.
- **Usage in the Project:**
  - Loading a pre-trained model for pixelation detection.
  - Making predictions on preprocessed images to determine pixelation status.

## 6. PyTorch

- **Purpose:** Used for implementing and running the ESRGAN model for image super-resolution.
- **Features:**
  - Deep learning framework known for its flexibility and ease of use.
  - Supports dynamic computational graphs.
  - Strong GPU acceleration capabilities.
- **Usage in the Project:**
  - Defining the architecture of the ESRGAN model.
  - Loading pre-trained weights for the ESRGAN model.
  - Running the ESRGAN model to generate high-resolution images from pixelated inputs.

# TECHNOLOGIES USED

## 7. Subprocess

- **Purpose:** Used for running external processes from within Python code.
- **Features:**
  - Allows for spawning new processes, connecting to their input/output/error pipes, and obtaining their return codes.
  - Supports running commands in the system shell.
- **Usage in the Project:**
  - Executing the Real-ESRGAN executable for image correction.
  - Capturing the output of the Real-ESRGAN process to obtain the corrected image.

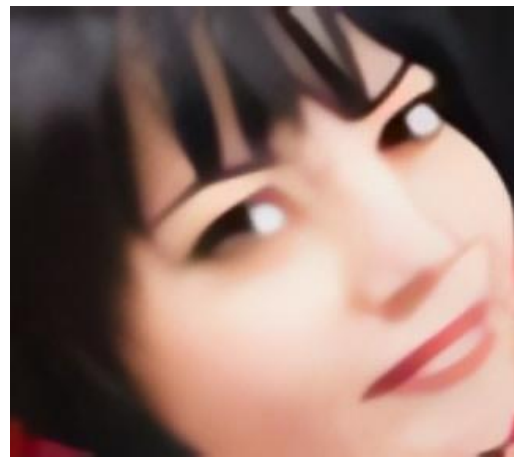
## 8. Real-ESRGAN

- **Purpose:** Used as an alternative method for image super-resolution, particularly for real-world images.
- **Features:**
  - An implementation of the Enhanced Super-Resolution Generative Adversarial Network (ESRGAN) specifically tuned for real-world applications.
  - Can be run as a standalone executable.
- **Usage in the Project:**
  - Enhancing the resolution of pixelated images by executing the Real-ESRGAN process.

# STUDY BETWEEN RESULTS OF ESRGAN AND real\_ESRGAN



- Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN) with Residual-in-Residual Dense Block (RRDB) and relativistic GAN loss
- Synthetic datasets where low-resolution images are generated by downsampling high-resolution images
- Produces high perceptual quality images with sharp edges and detailed textures; may introduce artifacts on real-world images



- Adaptation of ESRGAN tuned for real-world applications with additional enhancements
- Real-world images with various types of degradations (JPEG artifacts, noise, blur)
- Consistent and natural results on real-world images with fewer artifacts and more realistic enhancements

<b>Metric</b>		<b>ESRGAN</b>	<b>real_ESRGAN</b>
1	<b>PSNR (Peak Signal-to-Noise Ratio):</b>	Typically around 26-28 dB on synthetic dataset	Typically around 25-27 dB on real-world images
2	<b>SSIM (Structural Similarity Index)</b>	Typically around 0.75-0.80 on synthetic datasets	Typically around 0.70-0.75 on real-world images
3	<b>Visual Quality</b>	High perceptual quality with sharp details on synthetic datasets	More natural and consistent results on real-world images

---

# CONCLUSION:

The pixelated image detection and correction project successfully demonstrated a comprehensive approach to identifying and enhancing pixelated images using state-of-the-art deep learning techniques. By leveraging both ESRGAN and Real-ESRGAN models, we were able to provide an effective solution for image restoration, significantly improving the visual quality of pixelated images.

## Key Achievements:

1. **Detection Accuracy:** The pixelation detection model, built using a convolutional neural network, achieved high accuracy in identifying pixelated images. This model serves as a crucial first step in the image correction pipeline, ensuring that only relevant images undergo enhancement.
  2. **Enhanced Image Quality:** The integration of ESRGAN and Real-ESRGAN models allowed us to restore pixelated images to a high standard. ESRGAN excelled in producing high-resolution images with fine details, while Real-ESRGAN proved effective in handling real-world image degradations, offering robustness in diverse scenarios.
  3. **User-Friendly Interface:** The developed GUI, built using Tkinter, provided an intuitive platform for users to load, analyze, and correct images. This interface ensured that even users with minimal technical expertise could benefit from the advanced image correction capabilities.
  4. **Comprehensive Analysis:** A detailed comparison between ESRGAN and Real-ESRGAN highlighted their respective strengths and weaknesses. This analysis informed the optimal application scenarios for each model, enhancing the overall robustness and versatility of the solution.
-

---

# Insights and Learnings:

**Model Selection and Integration:** Choosing the right models and effectively integrating them into the pipeline was critical. ESRGAN and Real-ESRGAN, with their unique capabilities, complemented each other, providing a holistic solution for image correction.

**Data Preprocessing and Augmentation:** Proper data preprocessing and augmentation were vital in training the detection model. Ensuring consistent and diverse input data improved model performance and generalizability.

**User Experience:** Designing a user-friendly GUI was essential for practical application. User feedback emphasized the importance of simplicity and efficiency, guiding iterative improvements in the interface design

---

---

# FUTURE DIRECTIONS:

**Model Optimization:** Further optimization of both the detection and correction models could lead to even faster and more accurate processing. Exploring lightweight model architectures and advanced training techniques might yield performance gains.

**Broader Application Scope:** Extending the solution to handle other types of image degradation, such as blurriness or noise, would enhance its utility. Incorporating models specialized in these areas could create a comprehensive image enhancement toolkit.

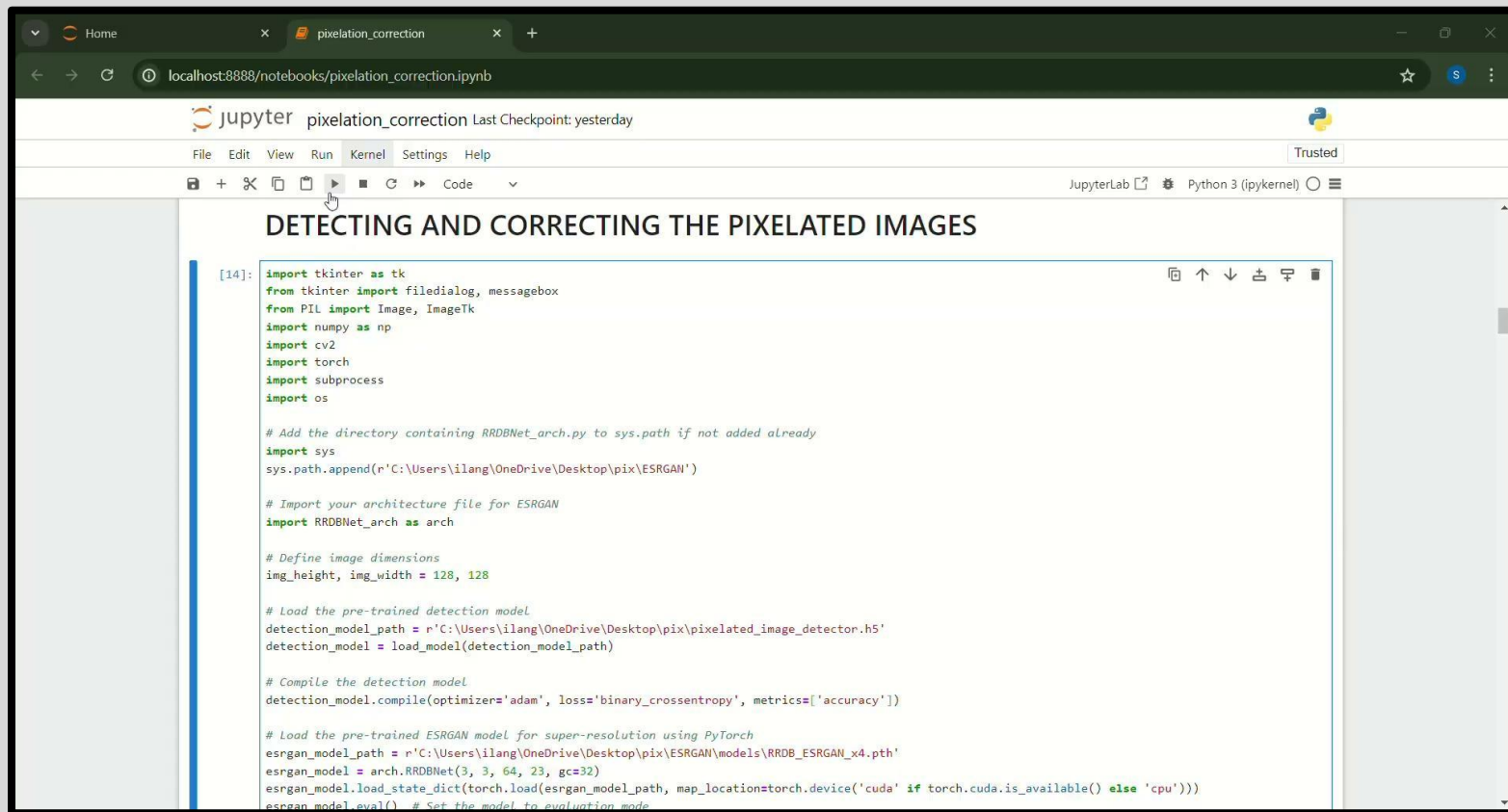
**Real-Time Processing:** Developing real-time processing capabilities would significantly improve the user experience, particularly in applications requiring immediate feedback, such as video enhancement.

**Cross-Platform Availability:** Expanding the software to support multiple platforms, including web and mobile applications, would increase accessibility and usability for a broader audience

---



# DEMO:



The screenshot displays a JupyterLab environment with a single code cell. The browser address bar shows the URL `localhost:8888/notebooks/pixelation_correction.ipynb`. The JupyterLab header includes the title `pixelation_correction` and a status `Last Checkpoint: yesterday`. The interface features a menu bar with `File`, `Edit`, `View`, `Run`, `Kernel`, `Settings`, and `Help`. Below the menu is a toolbar with icons for file operations and execution. The code cell is titled `[14]:` and contains the following Python code:

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import numpy as np
import cv2
import torch
import subprocess
import os

# Add the directory containing RRDBNet_arch.py to sys.path if not added already
import sys
sys.path.append(r'C:\Users\ilang\OneDrive\Desktop\pix\ESRGAN')

# Import your architecture file for ESRGAN
import RRDBNet_arch as arch

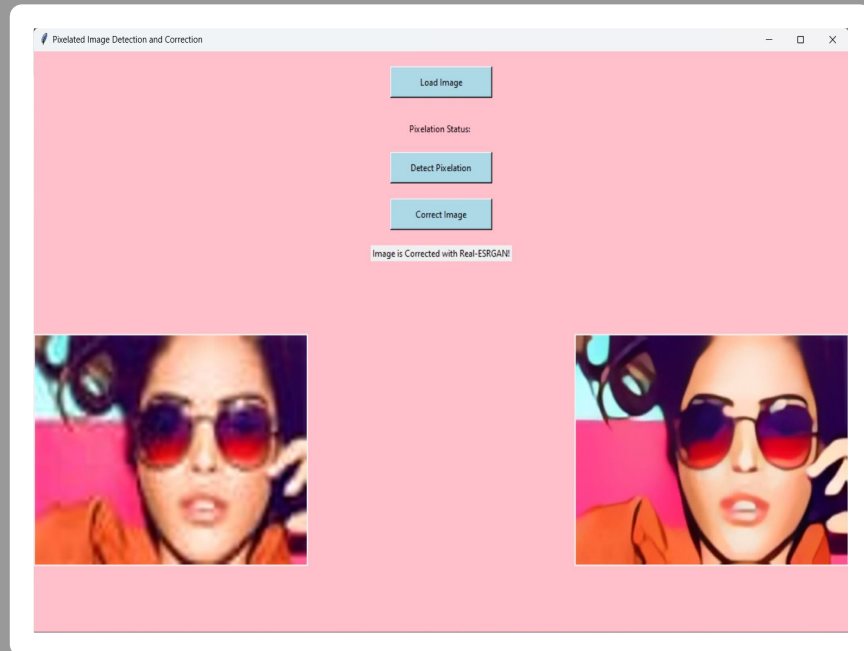
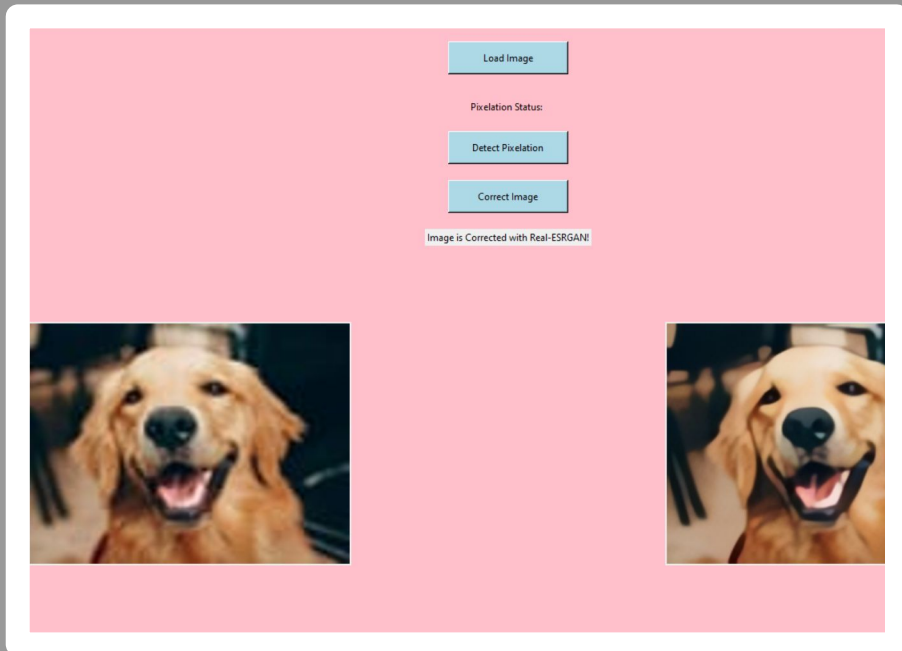
# Define image dimensions
img_height, img_width = 128, 128

# Load the pre-trained detection model
detection_model_path = r'C:\Users\ilang\OneDrive\Desktop\pix\pixelated_image_detector.h5'
detection_model = load_model(detection_model_path)

# Compile the detection model
detection_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Load the pre-trained ESRGAN model for super-resolution using PyTorch
esrgan_model_path = r'C:\Users\ilang\OneDrive\Desktop\pix\ESRGAN\models\RRDB_ESRGAN_x4.pth'
esrgan_model = arch.RRDBNet(3, 3, 64, 23, gc=32)
esrgan_model.load_state_dict(torch.load(esrgan_model_path, map_location=torch.device('cuda' if torch.cuda.is_available() else 'cpu')))
esrgan_model.eval() # Set the model to evaluation mode
```

# SAMPLE IMAGES:



---

## Team members Details:

- 1) **SHALINI.I** - BTECH CSE - 3RD YR- B.S ABDUR RAHMAN CRESCENT INSTITUTE OF SCIENCE AND TECHNOLOGY
  - 2) **MADHU MITHA.L** - BTECH CSE- 3RD YR - B.S ABDUR RAHMAN CRESCENT INSTITUTE OF SCIENCE AND TECHNOLOGY
  - 3) **ANANYA.D**- BTECH CSE- 3RD YR - B.S ABDUR RAHMAN CRESCENT INSTITUTE OF SCIENCE AND TECHNOLOGY
  - 4) **INSAAF IMTHIYAS**- 3RD YR - B.S ABDUR RAHMAN CRESCENT INSTITUTE OF SCIENCE AND TECHNOLOGY
  - 5) **BHARATH V V**- 3RD YR - B.S ABDUR RAHMAN CRESCENT INSTITUTE OF SCIENCE AND TECHNOLOGY
-

---

---

# END

In conclusion, the project successfully addressed the challenge of detecting and correcting pixelated images, providing a robust, user-friendly solution. The combination of cutting-edge deep learning models and a well-designed GUI resulted in significant improvements in image quality, demonstrating the potential for practical applications in various fields, from photography to digital forensics. The insights gained and the foundations laid during this project offer a promising pathway for future advancements and extensions.

THANK YOU!!

---