Problem Statement

Detect Pixelated Image and Correct It

Category: Artificial Intelligence, Machine Learning, Deep Learning, Autonomous Driving
Participants: 1st-4th Semester Students
Prerequisites: Concepts in Machine Learning Programming Skills (Python) Deep Learning - Train/Validate/Test with Data

Description: The problem has two parts – one is image classification and the other is image generation. 1. Given an Image, check if it is blocky or pixelated. Pixilation means blurry, blocky squares you see when an image is zoomed too much. Design an algorithm that can detect whether the image is pixelated or not. The challenge is to design an algorithm or ML/AI model, which is extremely lightweight or computationally efficient such that we can run this algorithm at 60 Hz or 60 frames per sec (FPS) and must be minimum 90% accurate. The pixelated images are rare and hence the algorithm should not create too many false positives. Algorithm / model quality will be measured by F1-score, precision recall curve etc. Also, the algorithm should be able to work on 1080p resolution input with the same performance. It is ok to downscale the large image to any desired input size to work on large images.

a. Input Image Size: 1920x1080 (it's ok to downscale and feed into the algorithm/model).

b. Inference Speed Target: Min 30Hz, better to have 60 Hz.

c. Accuracy Metric: F1 Score/Precision Recall.

d. Should work in rare class scenarios: if only one in 1000 images are pixelated then the algorithm must predict at most 10% False Positives.

2. Given a pixelated image, design an algorithm to improve the quality of the image i.e., restore the lost information. This problem is also known as jpeg restoration. Here also the challenge is to design a highly efficient algorithm that can run at least at 20 FPS. The quality of restoration can be examined by comparing with ground truth images using any metric like LPIPS, PSNR etc. If a non pixelated image is given then the algorithm should not enhance it and leave it intact. Algorithm should work on 1080p resolution images.

Unique Idea Brief

Transform the image pixelation detection and fixing application into an interactive studio that offers real-time feedback and customizable enhancement filters.This studio allows users to only detect and fix pixelation but also apply various enhancement filters, preview changes instantly, and fine-tune settings to achieve the desired image quality.

Features Offered

Real-Time Feedback: As users adjust settings (e.g., blur intensity, sharpening level), the application displays real-time previews of the changes on the uploaded image

Undo/Redo Functionality: Allow users to revert changes or reapply previous enhancements, making it easy to experiment with different settings.

Before/After Comparison: Enable a side-by-side comparison of the original image and the enhanced version, helping users visualize the improvements.

Pixelation Detection Insights: Provides a detailed analysis of pixelation, offering recommendations for improvement.

Pixel Correction functionality: Provides a pixel correction functions to deblur the input image

Process Flow

Upload Image: User uploads an image file (JPEG, PNG, BMP).

Delete Image: User deletes the uploaded image.

Check Pixelation: Uses OpenCV and laplacian variance to detect pixelation in the uploaded image.

Image Analysis: Converts the uploaded image to grayscale, analyzes intensity values, and plots graphs
.
Fix Pixelation: Applies image processing techniques (Gaussian blur, unsharp masking,edge detection filter) to reduce pixelation.

Download Image: Allows users to download the processed image.

Architecture Diagram

The architecture diagram shows the main components and their interactions in the Image Pixelator application.

Main Application: The core tkinter application where all components are placed.

Header (Label): Displays the title of the application
.
Button Frame (Frame): Contains all action buttons for uploading, deleting, checking pixelation, analyzing image, fixing pixelation, and downloading.

Upload Image, Delete Image, Check Pixelation, Image Analysis, Fix Pixelation, Download Image: Buttons that trigger specific actions when clicked.

Image Display (Label): Displays the uploaded image or the processed image.

Uploading/Deleting Label: Shows messages related to image upload and deletion operations.

Processing Label: Shows messages related to processing actions such as checking pixelation, analyzing image, fixing pixelation, and downloading.

```
+--------------------------------+
|        Main Application       |
|  +--------------------------+  |
| |      Header (Label)      | |
|  +--------------------------+  |
|  +--------------------------+  |
| |    Button Frame (Frame)  | |
| |  +----------------------+|  |
| | | Upload Image (Button)||  |
| |  +----------------------+|  |
| |  +----------------------+|  |
| | | Delete Image (Button)||  |
| |  +----------------------+|  |
| |  +----------------------+|  |
| | | Check Pixelation     ||  |
| | | (Button)             ||  |
| |  +----------------------+|  |
| |  +----------------------+|  |
| | | Image Analysis       ||  |
| | | (Button)             ||  |
| |  +----------------------+|  |
| |  +----------------------+|  |
| | | Fix Pixelation       ||  |
| | | (Button)             ||  |
| |  +----------------------+|  |
| |  +----------------------+|  |
| | | Download Image       ||  |
| | | (Button)             ||  |
| |  +----------------------+|  |
|  +--------------------------+  |
|  +--------------------------+  |
| |    Image Display (Label) | |
|  +--------------------------+  |
|  +--------------------------+  |
| |  Uploading/Deleting Label|  |
|  +--------------------------+  |
|  +--------------------------+  |
| |   Processing Label       | |
|  +--------------------------+  |
+--------------------------------+
```

Technologies used

The Image Pixelator application utilizes Python with the following technologies:

- **Tkinter**: for the graphical user interface (GUI)
- **Matplotlib**: for plotting graphs
- **NumPy**: for numerical operations
- **PIL / Pillow**: for image processing tasks
- **OpenCV**: for advanced image processing and computer vision
- **time**: for simulating delays and animations

Team members and contribution:

Team Leader: Dhruv Manish Dharod
Contribution: Whole Project building, Improvising, Report making

Conclusion

In conclusion, the Image Pixelator application showcases the effective integration of Python's powerful libraries and tools. By leveraging Tkinter for GUI development, Matplotlib and NumPy for image analysis and visualization, PIL/Pillow for image processing, and OpenCV for advanced computer vision tasks, the application provides a robust platform for uploading, analyzing, manipulating, and downloading images. The use of these technologies not only enhances user interaction and functionality but also demonstrates the versatility and capability of Python in handling complex image processing tasks within a user-friendly interface.

✨Thank you✨