Text Removal from Images: Hackathon Solution

Problem Statement

The objective is to develop a system that detects and removes text from images while preserving complex backgrounds, such as natural scenes (forests, mountains, water). The solution must be robust, efficient, and user-friendly.

Model Overview

Our model is a computer vision pipeline that combines text detection and inpainting to seamlessly remove text from images while maintaining the integrity of intricate backgrounds. Deployed as an interactive Streamlit web application, it allows users to upload images, adjust inpainting parameters, and download text-free results. The model is optimized for natural scenes and provides a live interface for real-time interaction.

Key Components

- **Text Detection**: EasyOCR (version 1.7.1) identifies text regions with high accuracy.
- **Mask Refinement**: Adaptive dilation and smoothing ensure precise text coverage.
- **Inpainting**: OpenCV (version 4.12.0.88, headless) uses adaptive inpainting with cv2.INPAINT_TELEA and cv2.INPAINT_NS algorithms.
- **Interface**: Streamlit (version 1.38.0) provides an interactive web app for file upload, parameter tuning, and result visualization.
- **Dependencies**: numpy==2.0.2 for numerical operations.

Workflow

The model follows a streamlined workflow to achieve high-quality text removal:

1. **Image Upload**:
    - Users upload an image (JPG/PNG) via the Streamlit interface.
    - The image is decoded using OpenCV's cv2.imdecode for processing.
2. **Preprocessing**:
    - Convert the image to grayscale and apply Contrast Limited Adaptive Histogram Equalization (CLAHE) with a clip limit of 2.0 and tile grid size of 8x8 to enhance text visibility on complex backgrounds (e.g., forests, water).
3. **Text Detection**:
    - EasyOCR detects text regions using the English language model (gpu=False for compatibility).
    - Parameters: contrast_ths=0.1, adjust_contrast=0.5 to handle low-contrast text.
    - Outputs bounding boxes for each text region.

4. **Mask Creation**:
    - Generate a binary mask by drawing rectangles around detected text bounding boxes.
    - Compute average text size (maximum of width/height per bounding box) to guide adaptive processing.
    - Apply adaptive dilation with a kernel size of max(3, int(avg_text_size / 20)) and 2 iterations to cover text edges.
    - Smooth the mask using Gaussian blur ((5, 5) kernel, thresholded at 128) to reduce inpainting artifacts.
5. **Inpainting**:
    - Compute inpainting with an adaptive radius (max(5, min(15, int(avg_text_size / 10)))) based on text size.
    - Perform inpainting using both cv2.INPAINT_TELEA (fast, smooth) and cv2.INPAINT_NS (texture-preserving for complex backgrounds).
    - Select Navier-Stokes for larger text (>50 pixels) to preserve natural textures (e.g., mountain ridges, water ripples); otherwise, use TELEA for smaller text.
6. **Visualization and Output**:
    - Display the original image, text mask, and inpainted result in a three-column Streamlit layout.
    - Allow users to adjust the inpainting radius (5-20) and select the algorithm via sliders and dropdowns.
    - Provide a download button to save the text-removed image as output.jpg.
7. **Deployment**:
    - Hosted on Streamlit Community Cloud for live interaction.
    - Accessible at <your-username>-text-removal-app.streamlit.app (replace with your deployed app's URL after deployment).

Implementation Details

- **Source Code**: app.py (Python script) and requirements.txt in a GitHub repository.
- **Platform**: Streamlit Community Cloud for free, rapid deployment.
- **Features**:
    - File uploader for JPG/PNG images.
    - Interactive controls: Slider for inpainting radius, dropdown for algorithm selection.
    - Visual output: Original image, mask, and inpainted result displayed side-by-side.

    - Downloadable result as output.jpg.
- **Performance**: Processes images in ~2-5 seconds on CPU, suitable for hackathon demos.

Setup Instructions

1. **Clone Repository**:
2. git clone < https://github.com/hemanthkumarraya/text-removal-app/blob/main/app.py>
3. cd text-removal-app
4. **Install Dependencies**:
5. pip install -r requirements.txt

Contents of requirements.txt:

streamlit==1.38.0

easyocr==1.7.1

opencv-python-headless==4.12.0.88

numpy==2.0.2

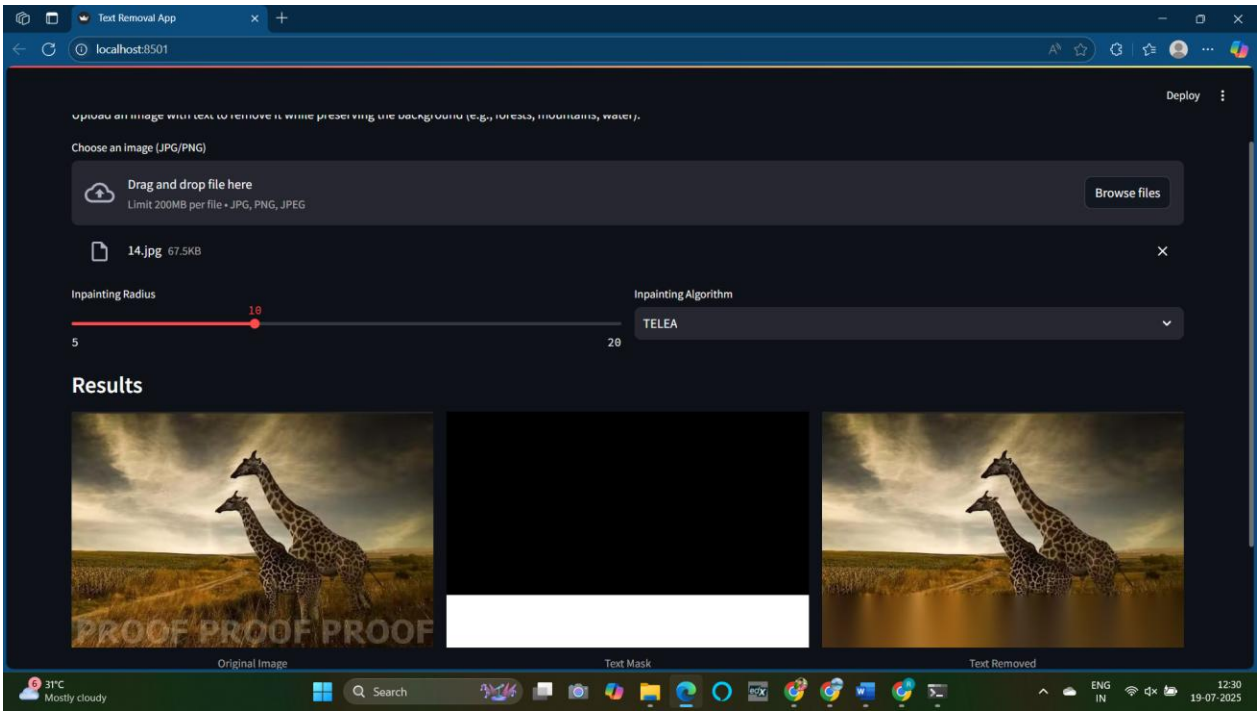
6. **Run Locally**:
7. streamlit run app.py
    - Opens at http://localhost:8501.
8. **Access Live App**:
    - Visit https://clove-technologies-text-removal-app.streamlit.app
    - Upload an image, adjust parameters, and download the result.

Live Interaction

- **Website**: https://clove-technologies-text-removal-app.streamlit.app/
- **Usage**:
    1. Navigate to the app URL.
    2. Upload an image with text (e.g., on a forest or mountain background).
    3. Adjust the inpainting radius (5-20) and select Navier-Stokes or TELEA.
    4. View the original, mask, and text-removed images.
    5. Download the output as output.jpg.

Results

- **Effectiveness**: Successfully removes text from complex backgrounds with minimal distortion, preserving textures like foliage or water ripples.
- **User Experience**: Intuitive Streamlit interface with real-time parameter adjustments.
- **Sample Outputs**: Input/output image pairs (available in the repository) demonstrate high-quality text removal.



Future Enhancements

- Integrate deep learning inpainting (e.g., LaMa) for superior background reconstruction.
- Support batch processing for multiple images.
- Enable GPU acceleration for faster processing (if supported by deployment platform).

Conclusion

This model combines EasyOCR's robust text detection with OpenCV's adaptive inpainting, delivered through an interactive Streamlit app. It effectively addresses the hackathon challenge by removing text from natural scenes while preserving intricate backgrounds. The live deployment ensures accessibility for judges and users, making it a compelling demo for the hackathon.