Sheet Detector with YOLOv3

Problem Statement

The objective is to develop a web application that detects and counts sheets in an image using the YOLOv3 object detection model. The application should provide a user-friendly interface for uploading images, process these images using YOLOv3, and display the number of detected sheets along with the annotated image.

Approach

1. **Model Selection**: We chose YOLOv3 for its balance between speed and accuracy. YOLOv3 is a well-known object detection model capable of detecting multiple objects in real-time.
2. **Web Framework**: Flask was selected as the web framework due to its simplicity and ease of integration with machine learning models.
3. **User Interface**: The UI was designed to be intuitive, allowing users to easily upload images and view detection results.

Implementation Steps

1. **Environment Setup**: Created a virtual environment and installed necessary dependencies.
2. **Model Integration**: Loaded the pre-trained YOLOv3 model with its configuration and weight files.
3. **Web Interface**: Developed HTML templates for the upload form and results page, and styled them with CSS for a professional look.
4. **Image Processing**: Implemented functions to process uploaded images, run the YOLOv3 model, and draw bounding boxes around detected sheets.
5. **Deployment**: Set up routes in Flask to handle image uploads and display results.
6.

Frameworks/Libraries/Tools

Python Libraries

- **OpenCV**: Used for image processing and loading the YOLOv3 model.
- **NumPy**: Utilized for numerical operations on image data.
- **Flask**: Chosen as the web framework to handle routing and template rendering.

Tools

- **YOLOv3**: Pre-trained model used for object detection.
- **Virtualenv**: Used to create an isolated Python environment.

Challenges and Solutions

Challenge 1: Model Integration

Issue: Integrating the YOLOv3 model with OpenCV and ensuring it works correctly with the provided configuration and weights files. **Solution**: Referred to the OpenCV documentation and various online resources to correctly set up the YOLOv3 model. Ensured that the paths to the configuration and weights files were correctly specified.

Challenge 2: Accurate Detection

Issue: The initial model did not accurately detect sheets since YOLOv3 is pre-trained on the COCO dataset, which does not include sheets. **Solution**: Adjusted the confidence threshold and Non-Maximum Suppression (NMS) parameters to improve detection results. Noted the need for a custom-trained YOLOv3 model for better accuracy, which is part of the future scope.

Challenge 3: User Interface Design

Issue: Creating a visually appealing and user-friendly interface. **Solution**: Used HTML and CSS to design a clean and centered layout for the upload form and results page. Ensured that the UI elements are responsive and easy to interact with.

Future Scope

Custom Model Training

- **Objective**: Train a custom YOLOv3 model specifically for sheet detection using a dataset of sheet images.
- **Benefit**: Improved accuracy and reliability of detection results.

Additional Features

- **Real-time Detection**: Implement a feature to use a webcam for real-time sheet detection.
- **Image Preprocessing**: Add preprocessing steps such as image resizing and normalization to enhance detection performance.
- **User Feedback**: Collect user feedback on detection accuracy to further refine the model and improve the application.

Deployment

- **Cloud Hosting**: Deploy the application on a cloud platform such as AWS or Heroku for broader accessibility.
- **API Integration**: Develop an API for the detection functionality to enable integration with other applications.

By implementing these improvements and additional features, the application can become more robust, accurate, and user-friendly, providing a valuable tool for sheet detection in various contexts.