# YOLOv8 Object Detection and Sheet Counting Application Report

## 1. Overall Approach

The objective of this project was to develop a Streamlit-based web application capable of detecting objects in uploaded images using the YOLOv8 model. Additionally, the application calculates the number of sheets in a stack, based on the detected height of the stack and an average sheet thickness.

The process begins with the user uploading an image, which is then processed by the YOLOv8 model. The model identifies objects and draws bounding boxes around them. For calculating the number of sheets, the height of the detected stack is used along with a known average thickness per sheet. The results, including the annotated image and the estimated number of sheets, are displayed to the user.

## 2. Frameworks/Libraries/Tools

- **Streamlit**: A web framework for creating interactive applications. It was used to build the user interface and handle user inputs such as image uploads.
- **Ultralytics YOLOv8**: A state-of-the-art object detection model that was employed to detect objects in images and extract bounding box information.
- **OpenCV**: A powerful library for image processing. It was used for reading, resizing, and manipulating images, as well as drawing bounding boxes and text annotations.
- **NumPy**: A fundamental package for numerical computations in Python. It was used to handle image data and perform array operations.
- **Pillow**: An image manipulation library in Python. It was used for handling image file formats and conversions.

## 3. Challenges and Solutions

### Challenge 1: Model Integration and Inference

- **Issue**: Integrating the YOLOv8 model into a Streamlit app and ensuring compatibility with the framework.
- **Solution**: The model was successfully integrated by carefully handling image input and output formats. The `Ultralytics` library was utilized for model loading and prediction, while OpenCV and NumPy handled image preprocessing and postprocessing.

### Challenge 2: Handling Image Uploads and Processing

- **Issue**: Ensuring the app could handle various image formats and maintain consistent image quality.
- **Solution**: The `Streamlit` file uploader was used to accept image uploads. OpenCV was then used to decode the images, resize them to the required dimensions, and prepare them for model input.

### Challenge 3: Accurate Sheet Counting

- **Issue**: Estimating the number of sheets accurately based on the detected bounding box height.
- **Solution**: A known average thickness per sheet was used to calculate the number of sheets from the detected stack height. This approach provided a reasonable estimate, considering the limitations of visual detection.

# 4. Future Scope

## 4.1. Enhanced Object Detection and Classification

- **Improvement**: Incorporate additional classes or refine the current detection model to distinguish between different objects more accurately. Training the model on a more extensive dataset could improve detection accuracy.

## 4.2. Sheet Thickness Variation

- **Feature**: Implement a mechanism to adjust the average thickness of sheets based on user input or additional contextual information. This could account for variations in sheet thicknesses in different scenarios.

## 4.3. User Interface Enhancements

- **Improvement**: Enhance the UI/UX by providing more interactive features, such as zooming into detected objects, adjusting detection sensitivity, or selecting specific objects of interest for further analysis.

## 4.4. Model Training and Optimization

- **Feature**: Allow users to fine-tune the model with their datasets directly through the application. This feature could involve providing an interface for labeling and uploading training images.

## 4.5. Deployment and Scalability

- **Improvement**: Deploy the application on cloud platforms to improve accessibility and scalability. This would involve setting up robust backend services to handle large volumes of user requests and data.

# 5. Conclusion

This project successfully demonstrates a practical application of the YOLOv8 model for object detection and specific counting tasks, like estimating the number of sheets in a stack. The use of Streamlit allows for an intuitive and accessible user experience. While the current implementation achieves its core objectives, there is significant scope for improvement and expansion, particularly in enhancing detection accuracy, user interface, and overall functionality. The future work section outlines several promising directions for continued development and optimization.