## **REPORT**

## **Related Work**

The field of object detection has seen remarkable advancements, particularly with models like YOLO (You Only Look Once). YOLO has revolutionized real-time object detection by combining high speed with accuracy, making it a popular choice for tasks such as vehicle detection, pedestrian tracking, and even satellite image analysis. Recent versions like YOLOv8 and YOLOv11 have further optimized detection accuracy and introduced advanced features like segmentation.

In building detection, studies often focus on satellite or aerial imagery to identify structures for urban planning, disaster management, or real estate assessments. However, most existing solutions emphasize detection without classifying buildings based on size or functionality. Additionally, deploying such systems often lacks accessibility for non-technical users, as many implementations require programming knowledge.

My work builds on these advancements by integrating YOLO's object detection capabilities with an interactive interface using Gradio, allowing seamless interaction with the model. Unlike traditional applications, this project adds a practical layer of size-based classification and real-world unit scaling for better insights.

## **My Contribution and Approach**

### **Contribution**

1. Developed an end-to-end pipeline for building detection and classification based on real-world measurements.
2. Integrated a user-friendly interface using Gradio, enabling users to upload images and instantly view predictions.
3. Introduced a custom classification scheme for buildings (small, medium, large) based on configurable area thresholds.
4. Leveraged Python libraries to streamline detection, scaling, and output generation in accessible formats (e.g., CSV and summary text).

### **Approach**

1. **Model Selection**:
   * Chose YOLO for its speed and accuracy in object detection. Fine-tuned a pre-trained YOLO11 model for the task.
   * Set thresholds for confidence (99%) and IoU (0.3) to optimize performance for high-quality predictions.
2. **Data Processing**:
   * Images are analyzed to detect building boundaries, which are then converted into real-world dimensions using a predefined scaling factor.
   * Bounding box areas are computed and classified into size categories based on the thresholds.
3. **Output Integration**:
   * Results include:
     + Annotated images showing detected buildings.
     + CSV files containing detailed detection data (coordinates, size categories, etc.).
     + Text summaries for quick insights.
4. **Interface Development**:
   * Used Gradio to create a web-based application for interactive usage.
   * Included visualization of original input, predictions, and log images to enhance usability.

## **Details of Work and ML Libraries**

### **Key ML Libraries and Tools Used**

1. **Ultralytics YOLO**:
   * Loaded the YOLOv8 model for object detection.
   * Configured model settings (confidence threshold, IoU, etc.) for optimal performance.
2. **Gradio**:
   * Built an intuitive user interface for seamless image uploads and result visualization.
3. **Pandas**:
   * Processed detection data to generate detailed CSV files for further analysis.
4. **Pillow (PIL)**:
   * Handled image processing tasks such as opening and displaying annotated images.
5. **OS**:
   * Managed file paths dynamically for saving predictions and loading logs.

## **Performance**

The model achieved:

1. **Detection Accuracy**:
   * Successfully detected buildings with minimal false positives due to high confidence settings.
   * Bounding boxes were accurately drawn, even in crowded scenes with overlapping structures.
2. **Classification Performance**:
   * Buildings were categorized correctly based on the specified size thresholds.
   * The scaling factor was validated with real-world measurements to ensure consistency.
3. **Efficiency**:
   * Real-time prediction with minimal latency (~1-2 seconds per image on a GPU-enabled system).
   * Generated output files (images, CSVs) in under 5 seconds, ensuring smooth application performance.
4. **User Experience**:
   * The Gradio interface provided a responsive and visually engaging platform for predictions.

## **Challenges and Lessons Learned**

### **Challenges**

1. **Model Tuning**:
   * Achieving a balance between precision and recall with high confidence thresholds was time-consuming.
   * Fine-tuning the model for diverse datasets to ensure robustness in detecting buildings of varying shapes and sizes.
   * Making it able to detect buildings in various directions was challenging.
2. **Scaling Factor Validation**:
   * Translating pixel dimensions into real-world measurements required domain knowledge and iterative testing.
3. **Interface Issues**:
   * Integrating dynamic logs into the Gradio interface while maintaining responsiveness was challenging.

### **Lessons Learned**

1. Early validation of data processing pipelines prevents errors in later stages.
2. User interface design must align with both technical and non-technical user needs.

## **Future Work**

1. **Expand Dataset**:
   * Incorporate satellite imagery, drone footage, or diverse geographic regions to improve model generalizability.
2. **Enhanced Scalability**:
   * Deploy the application on cloud platforms for larger-scale usage and integrate GPU acceleration for faster processing.

## **Few Screenshots from App.**







