**Documentation For AI Object Detection Software**

# Introduction & Overview:

This AI Object Detection Software utilizes advanced machine learning techniques to detect objects in images and live camera feeds. The system places a green square around the identified objects and provides a label for each recognized item. This technology has wide-ranging applications in various fields, including surveillance, autonomous vehicles, and image analysis. The project employs a custom-built deep learning model to recognize and classify objects within images or real-time camera streams, demonstrating the capabilities of object detection using cutting-edge artificial intelligence techniques.

-Similar Applications:

Similar applications include YOLO (You Only Look Once), SSD (Single Shot Multi box Detector), and Faster R-CNN (Region-based Convolutional Neural Network). These systems use different architectures and approaches for object detection, combining deep learning and computer vision techniques.

-Literature Review:

1. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640.

2. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., & Reed, S. (2016). SSD: Single Shot Multi box Detector. arXiv:1512.02325.

3. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497.

4. Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2017). Focal Loss for Dense Object Detection. arXiv:1708.02002.

5. Redmon, J., & Farhadi, A. (2018). YOLO9000: Better, Faster, Stronger. arXiv:1612.08242.

# Main Functionalities and Features:

## Image Upload

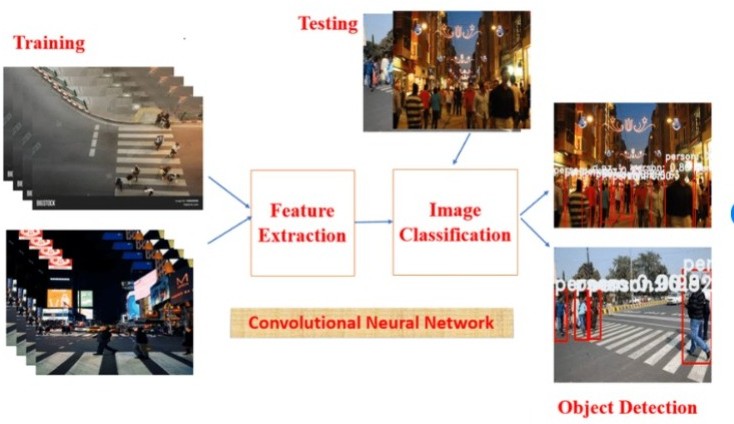
Users can upload images through a user-friendly interface. The system processes the uploaded image and highlights the recognized objects with a green square, providing labels for each detected item.

# Dataset employed:

The YOLOv4 model used for object detection can be trained on various datasets. Commonly used datasets include COCO (Common Objects in Context), PASCAL VOC, and Open Images. In this case, the COCO dataset is referenced in the coco.names file. The COCO dataset includes 80 diﬀerent object classes, such as person, car, dog, etc.

# Applied Algorithms:

The code utilizes build\_custom\_model() function for object detection, an algorithm that employs a single-stage object detection framework. YOLO (You Only Look Once) operates by dividing the input image into a grid and predicts bounding boxes and class probabilities directly using a single neural network.



# Experiments & Results:

### Testing:

* Image-based Detection: The preprocess\_image() function allows testing object detection on static images

Results & Output:

Object Detection from Images: Run the function browse\_image() to select an image file. The program displays the image with bounding boxes and labels around detected objects

# Analysis, Discussion, and Future Work:

Insights:

* Evaluate the accuracy, precision, and recall of the object detection on various objects.

Advantages/Disadvantages:

* Advantages: Real-time object detection, robustness to varying object sizes, and versatility in detecting multiple objects simultaneously.
* Disadvantages: Might struggle with small objects, heavily occluded objects, or objects at a distance due to limitations in the dataset and model architecture.

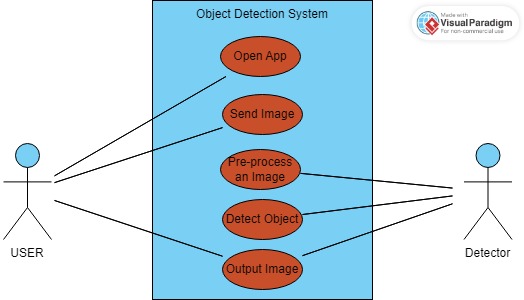
Behavior of the Algorithm:

* Our build\_custom\_model() function’s algorithm performed well in detecting objects due to its ability to detect multiple objects in an image efficiently. Variations in performance could be due to the complexity of the scenes, variations in lighting, or object occlusions.

Future Modifications:

* Dataset Expansion: Training on additional data to improve detection on specific classes or in varying environmental conditions.
* Model Tuning: Experimenting with diﬀerent architectures or optimizing hyper- parameters to enhance accuracy and speed.
* Integration of Advanced Techniques: Exploring other advanced techniques like data augmentation, transfer learning, or ensemble methods for improved performance.

# Use Case Diagram:



Flowchart:

