**Wagon Open door Detection Readme**

**Perquisites:** Python, Anaconda, Webcam, Telegram (Bot, ID)

Also follow supported requirements file.  
Also follow detect.py for kind of detection , export.py to export the model in folder .

Install Python and Anaconda from Chrome.

**Flow :** Place the supporting files in folder that supports streamlit to run the code

**Files to be placed :  
 - Python file  
 - Customized Model(.pt)  
 - .yaml file**

**# For Streamlit**

**Step – 1** Create Virtual Environment

Open Anaconda Prompt and type following commands.

conda update conda

conda create -n envname python=3.11 anaconda

conda activate envname

**Step -2** Install Required Packages

Open Anaconda prompt in the folder of the python file is located and run the following commands.

pip install -r requirements.txt

**Step -3** Run the application.

Open Anaconda prompt in the folder of the python file is located and run the following commands.

streamlit run app.py

**Procedure:**

Download the zip file shared with you unzip it and keep in current directory.

Run

streamlit run app.py

**Custom Dataset Workflow:**

Data preprocessing > Labeling > Load data > Neural Network Algorithm > Batches & Activation , Weights > Training > Testing > Confusion Matrix > Validation > Model > Application

**Step – 1:** Data Preprocessing

Collect custom dataset, and perform Data Augmentation to improve model generalization for accuracy.

Split the dataset into training, validation, and testing sets to ensure unbiased evaluation of the model.

Finally, data normalization is performed during training.

**Step – 2:** Labeling

We do labeling by annotating images with bounding boxes. This process involves specifying the object’s class and its location in the image (This can be done by using makesense.ai, labelimg,…).

We store these annotations in a format compatible with algorithm such as yaml format or coco format.

This labeled dataset serves as input for training the algorithm model accurately.

**Step – 3:** Load Data

Loading custom dataset involves defining a data loader that reads images and corresponding annotations from the dataset directory.

This loader preprocesses the data, converting it into a format compatible with the YOLOv5 model, facilitating efficient training on custom datasets.

Implement transformations such as normalization, resizing, and data augmentation.

**Step – 4:** Neural Network Algorithm

In model, a neural network algorithm is employed to detect objects within images efficiently.

This algorithm utilizes a deep convolutional neural network (CNN), specifically designed for real-time object detection tasks.

Trained on a custom dataset, the algorithm learns to identify and localize objects of interest with high accuracy, enabling robust and precise detection in various real-world scenarios.

**Step – 5:** Batches & Activation, Weights

Split your dataset into training, validation, and testing sets.

Create data loaders to load batches of images and annotations.

Decide on activation functions and initialization strategies for model weights (commonly ReLU and Xavier/Glorot initialization).

**Step – 6:** Training

Building model on a custom dataset involves feeding annotated images and corresponding labels into the model, iteratively optimizing its parameters through backpropagation.

With careful adjustment of hyperparameters and continuous monitoring of training metrics like loss, the model learns to accurately detect objects of interest, ultimately achieving high performance on the specified task.

**Step – 7:** Testing

Supported model, testing on a custom dataset involves utilizing the trained model to make predictions on unseen data.

This process evaluates the model's ability to accurately detect objects of interest in real-world scenarios.

Evaluation metrics such as precision, recall, and mAP are commonly used to assess the model's performance and ensure its reliability for deployment.

**Step – 8:** Validation

In YOLOv5 with a custom dataset, validation involves assessing the model's performance on a separate subset of the data not used during training.

This process helps ensure that the model generalizes well to unseen examples, by evaluating metrics such as accuracy, precision, and recall, thereby confirming the model's effectiveness before deployment.

**Step – 9:** Model

In YOLOv5, training a custom dataset involves fine-tuning the model's weights to detect objects specific to the dataset.

Optimize the model for deployment by reducing its size or converting it to a format suitable for inference on specific hardware ( like ONNX, Tensorflow Lite,…..).

**Step – 10:** Application

Integrate the trained YOLOv5 model into your application or system for real-world use cases like object detection in images or videos.

Continuously monitor the model’s performance in the production environment and update it as needed to adapt to changing requirements or data distributions.