VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMMING INTEGRATION PROJECT (CO3101)

Report

DETECTING COMMON TRAFFIC SIGNS USING YOLOv4

Advisor: Băng Ngọc Bảo Tâm
Student: Nguyễn Phan Trí Đức - 2152528

HO CHI MINH CITY, JUNE 2023

Contents

1 Introduction

Traffic signs play a crucial role in ensuring the safety and efficiency of road users, they come in a wide variety, distinguishable by their shapes and colors, each carrying its own specific meaning. In Vietnam, traffic signs can be categorized into several types, including prohibit signs, indication signs, mandatory signs, warning signs, and additional panels: Prohibit signs clearly indicate actions that are forbidden through text. Indication signs provide instructions that drivers must follow. Mandatory signs tell important details about directions, area names, and additional information. Warning signs serve as alerts to drivers, notifying them of potential hazards they should be cautious about when proceeding on the road. Additionally, there are additional panels, which provide optional supplementary information accompanying a traffic sign. These panels offer more specific details regarding factors such as the types of vehicles affected, time restrictions, or the extent to which the main sign's instructions apply.

Figure 1. Vietnamese traffic signs system

In today's rapidly advancing automotive industry, the demand of a reliable system to detect and interpret traffic signs has become increasingly essential for modern vehicles. Such a system offers numerous advantages that contribute to the safety, convenience, and overall driving experience. A traffic sign detection system enhances road safety by providing real-time information to the driver. By utilizing computer vision and machine learning, the system can accurately identify and relay important traffic signs, ensuring that drivers are informed about speed limits, prohibitions, directions, and other essential information. This helps drivers make the right decisions and adapt their driving behavior accordingly, reducing the risk of accidents or violations.

Moreover, the system acts like an extra set of eyes. Rather than relying on the driver's attention to spot and "digest" every traffic sign, the detection system continuously monitoring the surroundings. This improves the cognitive load on the driver, allowing them to focus more on the road and potential hazards. As a result, drivers experience reduced stress during long journeys or complex traffic situations. By accurately detecting traffic signs, the system can be utilize with autopilot technology to assist in optimizing vehicle speed and acceleration, ensures that drivers maintain a consistent and appropriate speed while staying on the right lane, minimizing fuel consumption and reducing environmental impact. This integration creating a more advanced driving experience, enhancing both safety and comfort, promising to evolve the development of a future of transportation.

In this project, I am interested in identifying 30 types of common Vietnamese traffic signs. Using the object detection algorithm that fits with the requirements.

2 Approach

2.1 Model

Object detection algorithms are broadly classified into two categories based on how many times the same input image is passed through a network.

Single-shot object detection uses a single pass of the input image to make predictions about the presence and location of objects in the image. It processes an entire image in a single pass, making them computationally efficient. However, single-shot object detection is generally less accurate than other methods, and it's less effective in detecting small objects. Such algorithms can be used to detect objects in real time in resource-constrained environments.

Two-shot object detection uses two passes of the input image to make predictions about the presence and location of objects. The first pass is used to generate a set of proposals or potential object locations, and the second pass is used to refine these proposals and make final predictions. This approach is more accurate than single-shot object detection but is also more computationally expensive.

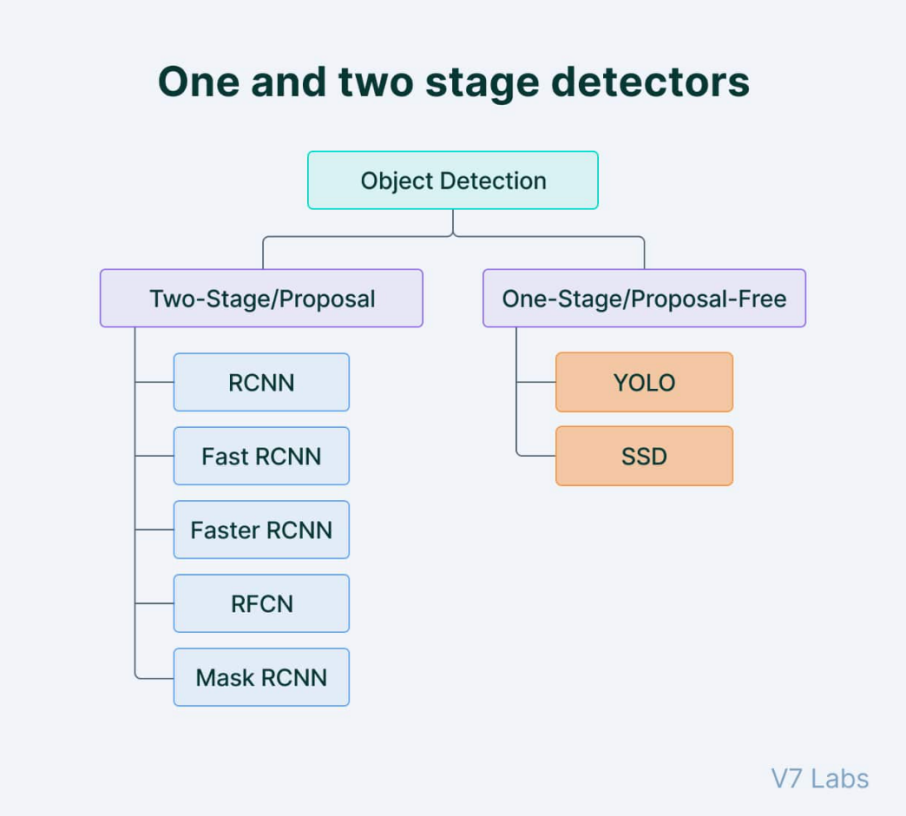


Figure 2. Object detection models

The choice between single-shot and two-shot object detection depends on the specific requirements and constraints of the application. Single-shot object detection is better suited for real-time applications, while two-shot object detection is better for applications where accuracy is more important. For identifying the most suitable object detection algorithm for real-time traffic

sign detection, I have analyzed the two single-shot object detection algorithm: SSD (Single Shot MultiBox Detector), and YOLOv4 (You Only Look Once version 4).

YOLO is a single-shot detector that uses a fully convolutional neural network (CNN) to process an image, YOLO's architecture allows it to detect objects at different scales and aspect ratios, making it perfect for a variety of scenarios and more accurate in detecting objects in general scenes that it has not been trained on.

SSD in the other hand, is known for its balanced trade-off between speed and accuracy. By employing a series of convolutional layers with various aspect ratios, SSD efficiently detects objects at different scales. However, it may face challenges in accurately localizing small objects, such as traffic signs, due to the fixed anchor box sizes used in its default configuration.

After a comprehensive evaluation of these algorithms, I ultimately selected YOLOv4 as the optimal choice for real-time traffic sign detection in this project. YOLOv4 has gained reputation for its exceptional speed-accuracy trade-off, making it suit for real-time applications. It incorporates a powerful backbone network to capture multi-scale information, and leverages advanced attention mechanisms. Moreover, YOLOv4 has superior object detection capabilities, accurately localizing and classifying traffic signs while maintaining impressive processing speed. These qualities make YOLOv4 as an outstanding algorithm for real-time traffic sign detection, ensuring timely and reliable results on the road.

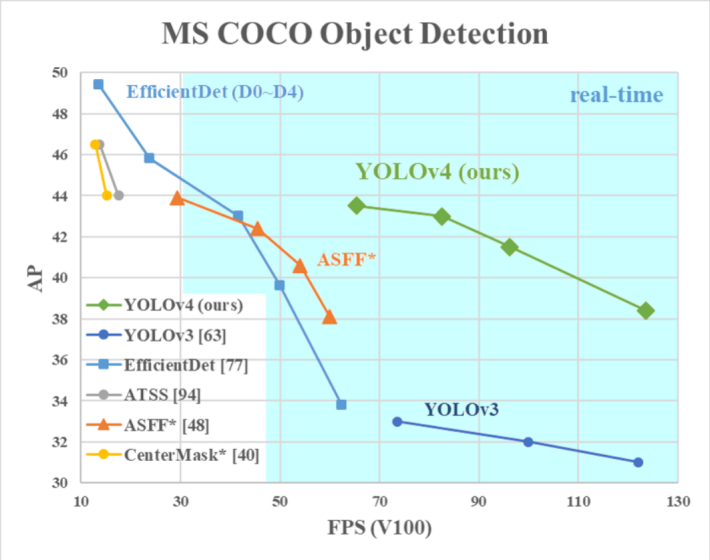


Figure 3. Comparing YOLOv4 with other models

The YOLOv4 model utilizes a powerful darknet backbone network to extract features from input images, enabling the model to capture multi-scale information and effectively detect and classify 30 different types of traffic signs.

YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image, it divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks

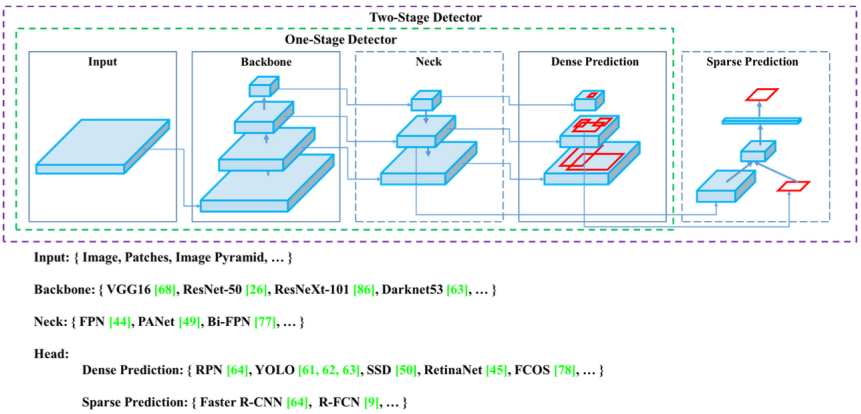


Figure 4. YOLOv4 architecture

the predicted box is.

One key technique used in the YOLO models is non-maximum suppression (NMS). NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

2.2 Dataset

To handle the complexity and diversity of traffic signs, I gather a large image dataset of Vietnamese traffic signs from Kaggle[1], carefully annotated with bounding box coordinates and corresponding class labels. As data augmentation techniques such as random scaling, rotation, and flipping can be applied for other object detection problems, my traffic sign recognizing model is designed for a fixed camera on the vehicle so it's not necessary. In the other hand, the dataset should have many situation when a sign is partially blocked by a building, structure, tree or another sign.

I also add some image from dataset of just the sign itself in[2], although this is not recommended because YOLO also learns the surrounding. I use a small amount of the dataset here for the classes that appear not often in the original dataset, and for the model can recognize easier between similar signs.

LabelImg is used for labelling dataset as it provides ease of use and fast. I have use this tool to label 1709 images for training, 677 from the main source, 319 from online searching, and 512 images of the sign itself; and 201 images gather from all around Ho Chi Minh City by myself for testing.

Figure 5. Using LabelImg tool to label data

The images labelled are scaled to 608x608 pixels before being zipped to sent to train.

[1]Zalo traffic signs dataset www.kaggle.com/datasets/loitranv/zalo-traffic-sign.
[2]Public Google Drive folder of Vietnamese traffic signs drive.google.com/drive/folders/1RLRSq72hey2wiTAq00DuPmtyF2NwgOTy.

Figure 6. Traffic sign classes

These are the 30 classes trained on the model.

2.3 Challenges

Our model aims to enhance the safety and efficiency of traffic sign detection systems, enabling real-time identification of various traffic sign types. With its robustness, accuracy, and speed, our model can contribute to improved traffic management, driver assistance systems, and autonomous vehicle applications, ultimately fostering safer and more intelligent transportation systems.

To achieve this, I have to analyze on how the driver or the model on the car view the sign. There are several situation that needs more relevant data to increase the effectiveness of the model.

- **Similar signs:** Some sign classes are quite similar.

(a) (b)

Figure 7: (a) No motorcycles and No bikes (b) No Stopping and No Parking

- **Occluded signs:** There maybe situations where a sign is occluded by a building, structure, tree or another sign. Training with these data helps the model can still recognize the sign.

(a) (b) (c)

Figure 8: (a) Occluded by a tree (b) Occluded by wires (c) Occluded by another board

- **Various distances:** Training with the signs in various distances or sizes helps the model more compatible with street environment, therefore the driver can have information to make decision more effectively.

(a) (b)

Figure 9: (a) Height limit sign on the bridge (b) No 3 wheels sign at distance

- **Various angles:** Training with the signs in various angles helps the model more compatible with street environment.

(a) (b)

Figure 10: Some signs viewed from side angle

3 Training

3.1 Approach

Training must utilize the YOLOv4 loss function, which combines localization loss, classification loss, and confidence loss to optimize the model's performance.

$$L_{classification} = \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in class} (p_i(c) - \hat{p}_i(c))^2$$

$$\begin{aligned} L_{localization} &= \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ &+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \end{aligned} \quad \text{(Loss components)}$$

$$\begin{aligned} L_{confidence} &= \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{obj} (C_i - \hat{C}_i)^2 \\ &+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{noobj} (C_i - \hat{C}_i)^2 \end{aligned}$$

Classification loss is referred to the error in the classification of the object, it measures the difference between the predicted and the actual class probabilities of an object. $\mathbb{1}_i^{obj}$ will equal to 1 if the cell has an object, else 0.

Localization loss is referred to the error in the ability to accurately locate the object in the image, that is the loss of predicted boundary box included the object's center coordinate, width, height compared with the actual labelled data. Localization loss is calculated by the sum of predicted coordinate error (x, y) error and predicted bounding box (w, h) error with the actual bounding box. At each cell has an object, one highest IOU (Intersect over union value) boundary box will be chosen, the loss is calculated base on these boundary boxes.

Confidence loss is referred to the uncertainty of the predictions made by the model. It measures the accuracy of the confidence in the predictions, calculated based on the predicted probability that an object exists in a bounding box and the IOU between the predicted and actual bounding boxes, both on the cells that include an object or not include any object. Similarly, $\mathbb{1}_{i,j}^{obj}$ will equal to 1 if the box of that cell has an object, else 0. And $\mathbb{1}_{i,j}^{noobj}$ will equal to 0 if the box of that cell has an object, else 1.

According to YOLO paper, most grid cells in an image do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on. Remedy this by increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects, therefore set $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$.

Total loss of the model:

$$L_{total} = L_{classification} + L_{localization} + L_{confidence}$$

I employ transfer learning by initializing the model with pre-trained weights on a large-scale object detection dataset, allowing the model to benefit from prior knowledge and accelerate convergence.

3.2 Environment setup

For accelerating training speed, I use Google Colab VM to use fast GPU utilize with CUDA. The ability to use fast GPU with CUDA helps to reduce the training time required to complete machine learning tasks. This is particularly useful when working with large datasets and complex models that require a lot of computing power.

Google Colab also offers a range of pre-installed libraries and packages which can be easily imported into the notebook reduce the time for installing and configuring all the necessary software. Google Colab connect to Google Drive helps easy of use, saving progress, and can helps remote working.[3]

Prepared data for uploading to VM:[4]

- **obj.zip:** All images and corresponding label text file for training.
- **test.zip:** All images and corresponding label text file for testing.
- **yolov4-obj.cfg:** Configure file of the model, use input size of 416x416.
- **yolov4-obj_last.weights:** Latest weight trained.[5]
- **obj.names:** List of names of 30 classes.

```
Closed
Prohibited
...
Expressway
Roundabout
```

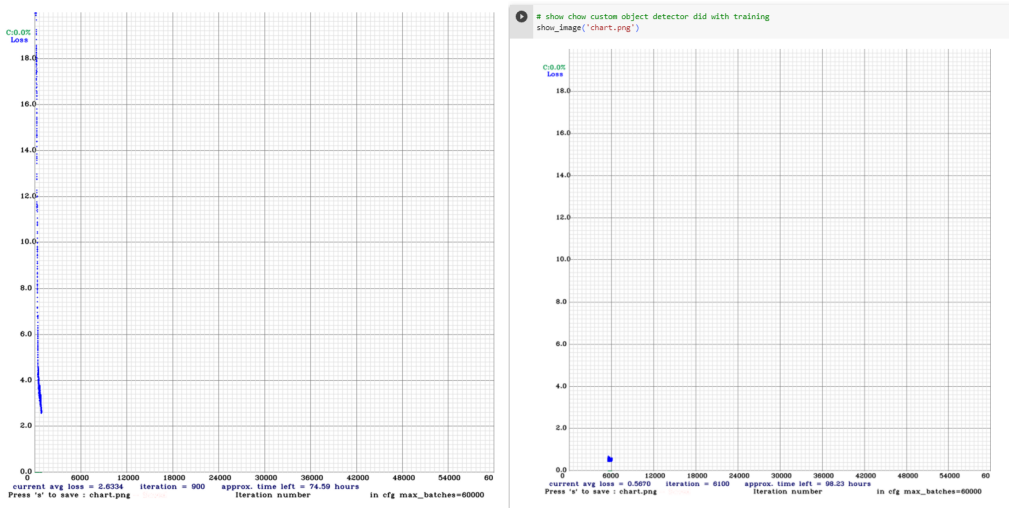
- **obj.data:** Configure information for training process.

```
classes = 30
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = /mydrive/yolov4/backup
```

[3]Google Colab working notebook colab.research.google.com/drive/1q5cnH1faiLOsj4TINRpMseu_OgMog3k-?usp=sharing.
[4]Working folder drive.google.com/drive/folders/1svD1Z_QHqrGfeZ8uMhM4MLlipBXVMA4o?usp=share_link.
[5]Trained weight file drive.google.com/file/d/19YVeCErGCOVTAd5DZe2bHEDKtXScqx4_/view?usp=share_link.

3.3 Progress

Training on Google Colab last for about 3 hours before running out of computation units, as a result, I don't have a full loss chart from the first to last iteration. The below figures show how loss function of the model converges down over time.



(a) (b)

Figure 11: (a) Loss function value after first 900 iterations (b) Loss function at over 6000 iterations

The training process has been successful, with the average loss steadily decreasing over time. After extensive training, the model has reached a convergent value of approximately 0.55 for the average loss. This value is regarded as a well-trained model that has undergone significant learning to effectively detect patterns and make accurate predictions.

Reducing the average loss shows the model's ability to generalize from the training data and effectively capture the underlying patterns and relationships in the input. The model has continuously refining its parameters to minimize the difference between its predictions and the actual outcomes, it has acquired the ability to identify relevant features, meaningful information, and make accurate predictions.

4 Result

To determine and compare the predictive performance of different object detection models, we need standard quantitative metrics.

In object detection algorithms, **IOU (Intersection over Union)** is the key metric, measures the overlap between the predicted bounding box and the actual bounding box. An IOU value $>$ 0.5 is taken as a positive prediction, while an IOU value $<$ 0.5 is a negative prediction.

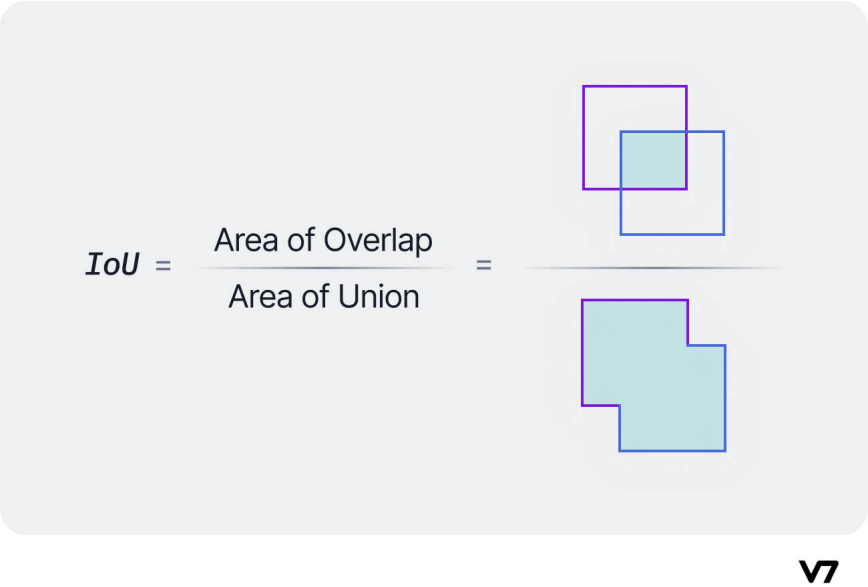


Figure 12. Intersection over Union metric

Another two important metrics, **Precision** and **Recall** are used to evaluate the performance of models. **Precision** is the ratio of the number of true positives (TP) to the total number of positive predictions (TP+FP), literally saying *"if your model predicts, how often does it predicts correctly?"*. **Recall** is the ratio of the number of true positives (TP) to the total number of actual objects (TP+FN), literally saying *"has your model predicted every time that it should have predicted?"*.

Precision and **Recall** offer a trade-off that is graphically represented into a curve by varying the classification threshold. The area under this precision versus recall curve gives us the **Average Precision** per class for the model. The average of this value, taken over all classes, is called mean Average Precision (mAP).

Therefore, the most common two evaluation metrics for real-time object detection algorithms are mean Average Precision (mAP) metrics and FPS.

- **mAP - Accuracy:** Average precision is calculated for a single object class across all test images, and finally mean Average Precision – a single value that can be used to compare models handling detection of any number of object classes.
- **FPS - Speed:** The number of frame in which the model can handle per second.

The recorded values is from using Google Colab Tesla V4 GPU.

4.1 mAP

Figure 13. Detection of an image of many signs

Test the model on the test dataset gives the following result: mAP 84.14%.

```
for conf_thresh = 0.25, precision = 0.80, recall = 0.78, F1-score = 0.79
for conf_thresh = 0.25, TP = 834, FP = 212, FN = 235, average IoU = 63.44 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.841413, or 84.14 %
```

4.2 Video FPS

The model can run smoothly and quite stable at average 20FPS. Video of running on a video

Figure 14. Real-time webcam view

4.3 Webcam

Finally, let's put the model to the test in real-time. Using webcam, FPS is lower due to the significant amount of data being processed from the webcam to the model within the virtual machine, as well as the additional functions required to display the results on the screen. And the color is not perfect so there will be misdetected boxes.

Figure 15. Real-time video view

4.4 Conclusion

In conclusion, the YOLOv4 model trained to detect Vietnamese traffic signs has shown impressive performance throughout the training process. It exhibits accurate and simultaneous recognition of numerous learned signs, as demonstrated by the evaluation results.

The evaluation result, with a confidence threshold of 0.25, has a precision of 0.80, recall of 0.78, and an F1-score of 0.79. The average intersection over union (IoU) was measured at 63.44%, indicating the model's ability to accurately localize the detected signs. The mean average precision (mAP@0.50) of **84.14%**, suggests a relatively high performance level for the model.

(a) (b)

Figure 16: (a) Differentiate between similar signs (b) Just detect the trained signs

One notable strength of the YOLOv4 model lies in its accurate feature extraction capabilities, allows differentiating between similar signs. And the ability to generalize from the training images has contributed to its accurate detection of signs in the images not in trained dataset.

It also almost always ignores the other sign types that not being trained, there are some cases giving false detection highly because there's lack of training images that have the other sign types than the trained one so it may get confused when an another type of sign that looks quite similar to one of the trained sign appears. There are some limitations when the model's performance may not show best result when dealing with occluded signs or signs that are too far or too small in the input image.

In conclusion, despite the model's impressive accuracy and detection performance, there is still needs for improvement. Expanding the training dataset including a wider variety of sizes and occlusion scenarios, could enhance the model's ability to generalize and detect signs more effectively. With further refinement, this YOLOv4 model holds the potential to be utilized in camera systems to assist drivers by providing reliable and accurate traffic sign detection.

(a) (b)

Figure 17: (a) View from side angle (b) The occluded speed limit sign is detected, while the narrow road sign is not

(a)

(b)

Figure 18: Signs far away are challenging to detect