Note: the final notebook is **yolo\_implementation\_final.ipynb**

**YOLO: System Description and Implementation**

The YOLOv5 deep learning model forms the core of the proposed system, designed for detecting and classifying vehicles. The model has been trained on a dataset of 7,500 images, collected using a suitable camera capturing the diverse array of vehicles on the streets of Iraq. These images have been pre-classified into five categories: personal cars, taxi cars, minibus cars, trucks, and tuk-tuks.

Google Colaboratory (<https://colab.research.google.com/>) provides the platform for the training process. After 100 epochs, the model achieves an Odds Ratio mAP of 0.924, indicative of the model's high accuracy in identifying and categorizing vehicles.

The YOLOv5 network architecture is composed of three primary components:

1. Backbone: The CSPDarknet, which is tasked with the initial extraction of features from the input data.
2. Neck: The PANet, which performs the fusion of the extracted features.
3. Head: The YOLO Layer, which generates the output results of the detection process, including class, score, location, and size of the detected vehicles.

**Our Network: System Description and Implementation**

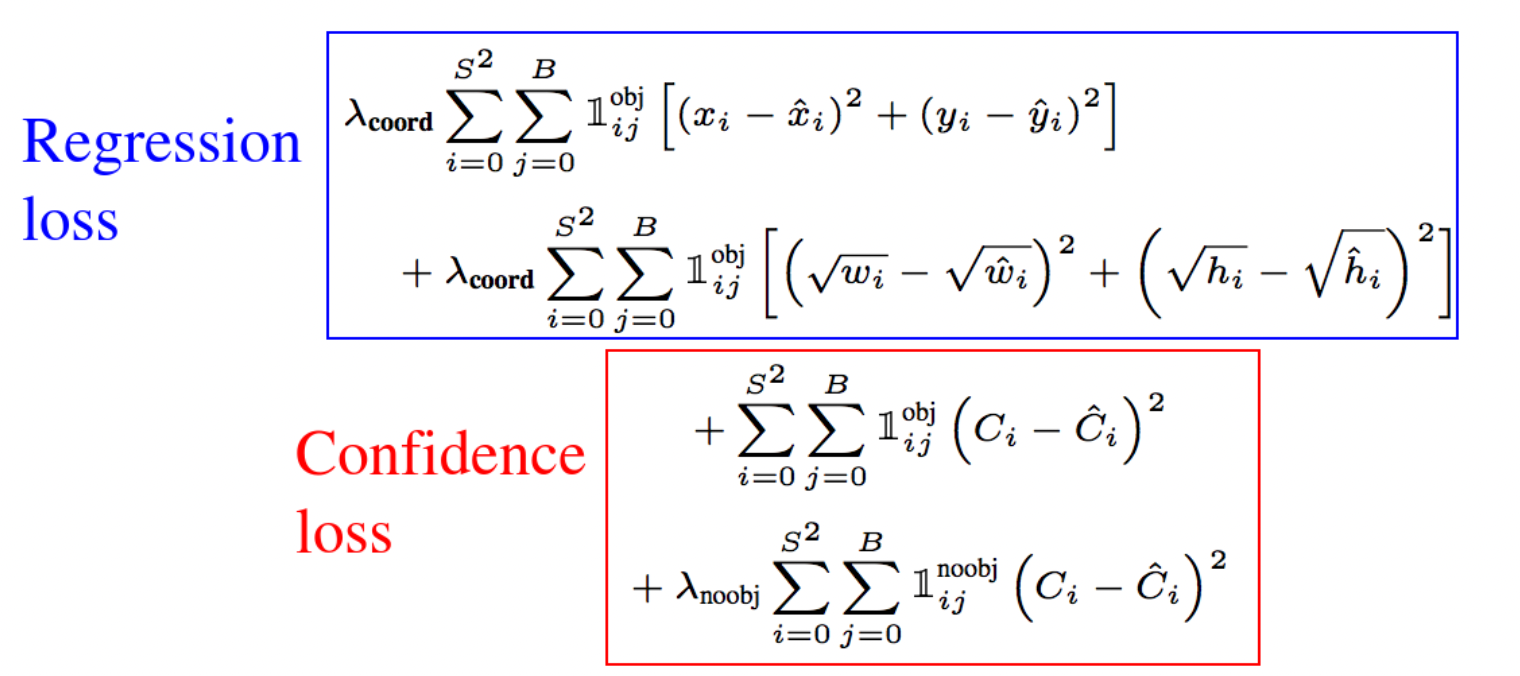
We had the general structure of the YOLO network in mind when creating our own object detection network. There are some differences in our data that allow us to implement a slightly simpler version of the YOLO model; the most apparent of which is that there is only one class, cars. This simplifies our loss function, as we no longer have the classification loss component.

Generally speaking, our model does the same thing as a YOLO network; it takes an image and creates a grid, where each element in that grid is a 2d tensor. Then, for each grid cell, it calculates a specific number of bounding boxes in that cell, along with how confident it is that an object actually lies within each bounding box.

Input data are images of shape (338, 190, 3)[[1]](#footnote-0). The model outputs a tensor of size (5, 10, 5, 6). Let’s explain the meaning of each number in the output size.

* The first 5 is the grid width.
* The 10 is the grid height
* The third 5 is the number of bounding boxes per grid cell. In our case, for each cell, the model calculates 5 bounding boxes. So, for example, even if only one object lies within a cell, the model will still calculate 5 bounding boxes for that cell – ideally, only one of them will have a high objectness score.
* The 6 contains information about each calculated bounding box. Specifically, the elements here are (x\_center, y\_center, width, height, objectness, class). x\_center and y\_center are normalized between [0, 1) and are **relative to the grid cell they are in**[[2]](#footnote-1). The width and height are normalized by the whole image. The objectness is a measure of how confident the model is that an object is within the specific bounding box, and it is used to filter out the relevant bounding boxes and calculate the objectness loss. The class is irrelevant here and is an artifact of the more general YOLO network.

We use a custom loss function, which factors in both bounding box accuracy and objectness accuracy.



Our model architecture is actually quite simple, and we chose to keep it this way primarily for performance reasons – training the model for 10 epochs, for example, already takes a relatively long time. The model summary is given below.

Model: "model\_11"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

input\_12 (InputLayer) [(None, 190, 338, 3)] 0

conv\_1 (Conv2D) (None, 190, 338, 16) 432

max\_pooling2d\_55 (MaxPoolin (None, 95, 169, 16) 0

g2D)

conv\_2 (Conv2D) (None, 95, 169, 32) 4608

max\_pooling2d\_56 (MaxPoolin (None, 47, 84, 32) 0

g2D)

conv\_3 (Conv2D) (None, 47, 84, 64) 18432

max\_pooling2d\_57 (MaxPoolin (None, 23, 42, 64) 0

g2D)

conv\_4 (Conv2D) (None, 23, 42, 128) 73728

max\_pooling2d\_58 (MaxPoolin (None, 11, 21, 128) 0

g2D)

conv\_5 (Conv2D) (None, 11, 21, 256) 294912

max\_pooling2d\_59 (MaxPoolin (None, 5, 10, 256) 0

g2D)

conv\_6 (Conv2D) (None, 5, 10, 512) 1179648

conv\_7 (Conv2D) (None, 5, 10, 1024) 4718592

conv\_8 (Conv2D) (None, 5, 10, 256) 2359296

conv\_9 (Conv2D) (None, 5, 10, 30) 7710

reshape\_11 (Reshape) (None, 5, 10, 5, 6) 0

=================================================================

Total params: 8,657,358

Trainable params: 8,657,358

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Results and Discussion**

The proposed system, based on YOLOv5, demonstrates excellent results in the field of vehicle detection and classification. It accurately identifies and classifies different types of vehicles with an impressive mAP score of 0.924 after 100 epochs of training.

The system outperforms previous models in the literature. For example, the YOLOv4-based system proposed by V Sowmya and R Radha achieved an accuracy of 96.54 percent (mAP) for detection, but the dataset consisted only of 3,500 images of buses and trucks.

The proposed system offers an efficient solution for vehicle detection and classification, with potential applications in traffic monitoring and autonomous driving.

**Efficiency Metrics of the Proposed System**

The proposed system's efficiency is demonstrated in Table 2 below:

| **Classes** | **TP** | **FP** | **FN** | **F1** | **Recall** | **Precision** |
| --- | --- | --- | --- | --- | --- | --- |
| CAR-F | 0.92 | 0.13 | 0.08 | 90 | 92.000 | 87.619 |
| CAR-R | 0.9 | 0.9 | 0.09 | 91 | 90.909 | 90.909 |
| CAR-S | 0.9 | 0.1 | 0.1 | 90 | 90.000 | 90.000 |
| TAXI-F | 0.98 | 0.08 | 0 | 96 | 100.000 | 92.453 |
| TAXI-R | 0.97 | 0.07 | 0.01 | 96 | 98.980 | 93.269 |
| TAXI-S | 0.93 | 0.06 | 0.02 | 96 | 97.895 | 93.939 |
| MINIBUS-F | 0.93 | 0.11 | 0.06 | 92 | 93.939 | 89.423 |
| MINIBUS-R | 0.94 | 0.06 | 0.05 | 94 | 94.949 | 94.000 |
| MINIBUS-S | 0.9 | 0.06 | 0.07 | 93 | 92.784 | 93.750 |
| TRUCK | 0.93 | 0.11 | 0.03 | 93 | 96.875 | 89.423 |
| TUK-TUK | 0.93 | 0.13 | 0.06 | 91 | 93.939 | 87.736 |

**Conclusion**

Determining vehicle direction yields several benefits, particularly for people with special needs. This includes determining the direction of travel and identifying vehicles moving in the wrong direction. Such technology is particularly useful for visually impaired individuals, aiding them in understanding traffic direction.

The proposed system highlights the utility of the YOLO network in accurately detecting objects. This reliability bodes well for real-time systems that require a high degree of accuracy and fast response times.

1. The actual images are of shape (776, 380, 3), but for performance reasons we compressed them by a factor of 2. [↑](#footnote-ref-0)
2. For example, if a car is in the grid cell (2, 3) and is halfway across that grid cell and a fourth of the way down that grid cell, its (x\_center, y\_center) would be (0.50, 0.25) [↑](#footnote-ref-1)