

License Plate Detection and Recognition

Project Report

Team Members:

Karthik Boddupalli (200001016)

Mudavath Bhanu Prakash (200001048)

Submitted to:

Prof. Aruna Tiwari (Instructor – CS304N)

1. Problem Statement

When security personnel chase a car or are unable to apprehend one that violated traffic laws, it is known that the authorities have difficulties. Also, in a parking lot, officials/employees find it difficult to physically record the vehicle numbers in a parking lot on a busy day. So, in order to make the entire process autonomous, we can install an automatic license detection and recognition system to automatically detect the vehicle which breaks the traffic rules, take a picture of it and store the number in the database so as to find the respective owner afterwards. The project can also be used to photograph the vehicle when parked and record the vehicle number in a database. This technology reduces unnecessary manual labor required on busy days, saves labor costs, and is far more efficient than humans.

2. Analysis and Design of problem

Looking at the bigger picture:

License plate detection and recognition (LPDR):

As the name suggests, we have two clear objectives: -

- ▶ 1. License plate detection (Object detection)
- ▶ 2. Character Recognition (letters & numbers from the plate obtained above)

But in order to do any of this, we need DATA i.e., images of vehicles along with their license plates.

i) Data Collection and Pre-processing

After a pretty tiring research, we were able to find a dataset that best suits our requirements i.e., properly structured and labelled data:

Overview

The License Plates dataset is a object detection dataset of different vehicles (i.e. cars, vans, etc.) and their respective license plate. Annotations also include examples of "vehicle" and "license-plate". This dataset has a train/validation/test split of 245/70/35 respectively.



Given dataset contains about 350 images but that's not good enough

Not all image files have same resolution (i.e., length and width) but we need 640x640 as standard input for YOLOv8.

So, we need to pre-process the data by making:

- ▶ 1. Resizing images to a particular resolution that will be accepted by our "object detection model".
- ▶ 2. Performing few augmentations like Horizontal flip to increase the dataset.

The required/mentioned pre-processing is done.

Summary of final dataset along with the split is as follows:

Training Set 80%		Validation Set 13%	Testing Set 7%
421 images		70 images	35 images
PREPROCESSING		Auto-Orient: Applied	
		Resize: Stretch to 640x640	
AUGMENTATIONS		Outputs per training example: 3	
		Flip: Horizontal	
DETAILS		Version Name: 2023-04-04 2:39pm	
		Version ID: 1	
		Generated: Apr 4, 2023	
		Annotation Group: licence-plate	

ii) Algorithm

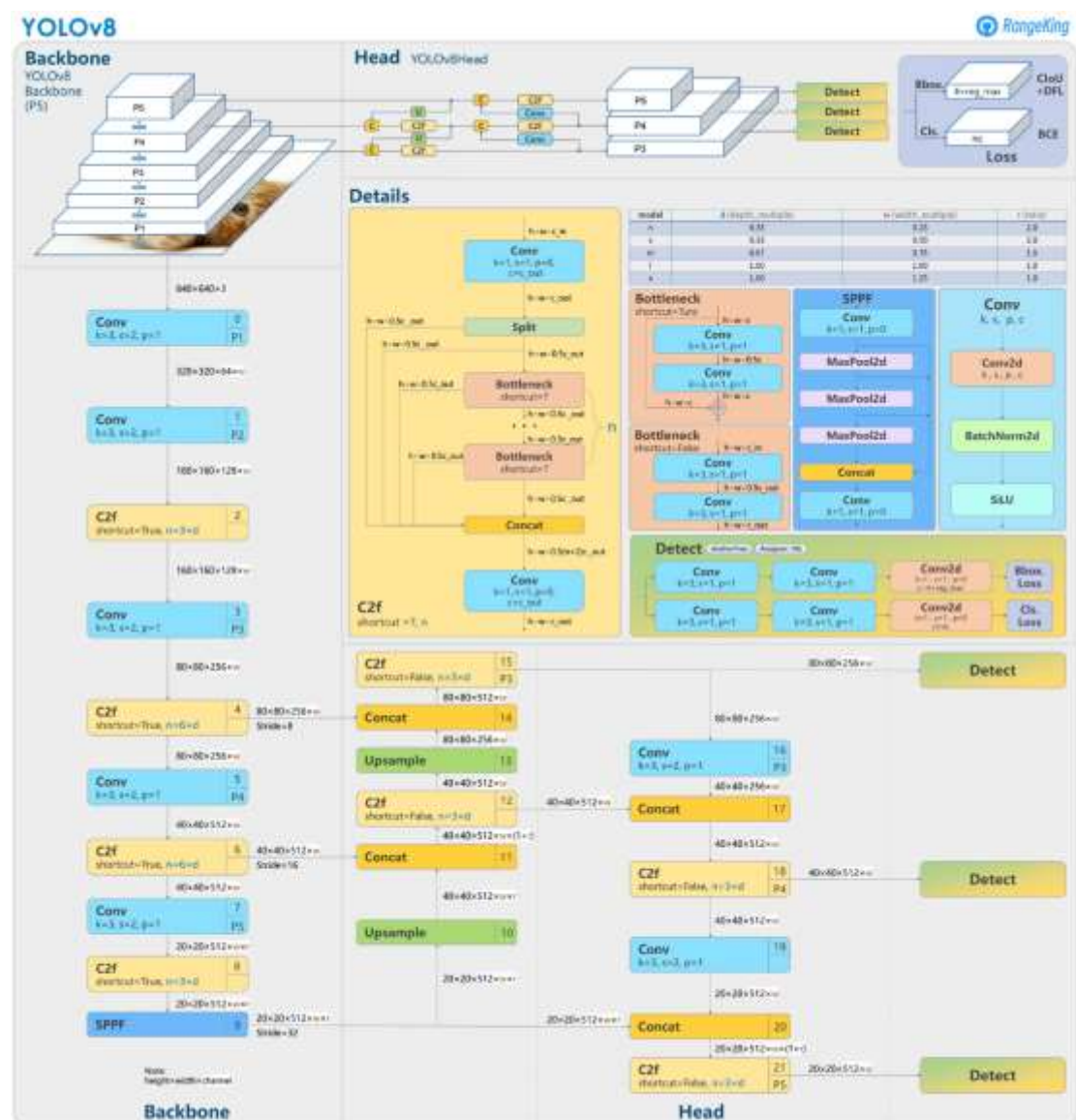
Now that we have our data, it's time to use it. But how?

As said earlier, we need to

- 1) Detect the license plate in an image and then
- 2) Recognize the characters in it.

First let's see how we can detect the license plate. This can be done by developing an **"Object detection model using YOLO algorithm"**. The working/idea behind the YOLO algorithm is as follows:

YOLO Algorithm – Architecture:



YOLO (You Only Look Once) is a CNN based algorithm used to detect objects (i.e., region of interests) from an image.

The latest industrial standard of this algorithm is the YOLO v8 developed by Ultralytics.

► YOLOv8 architecture has 225 layers, 11136374 parameters.

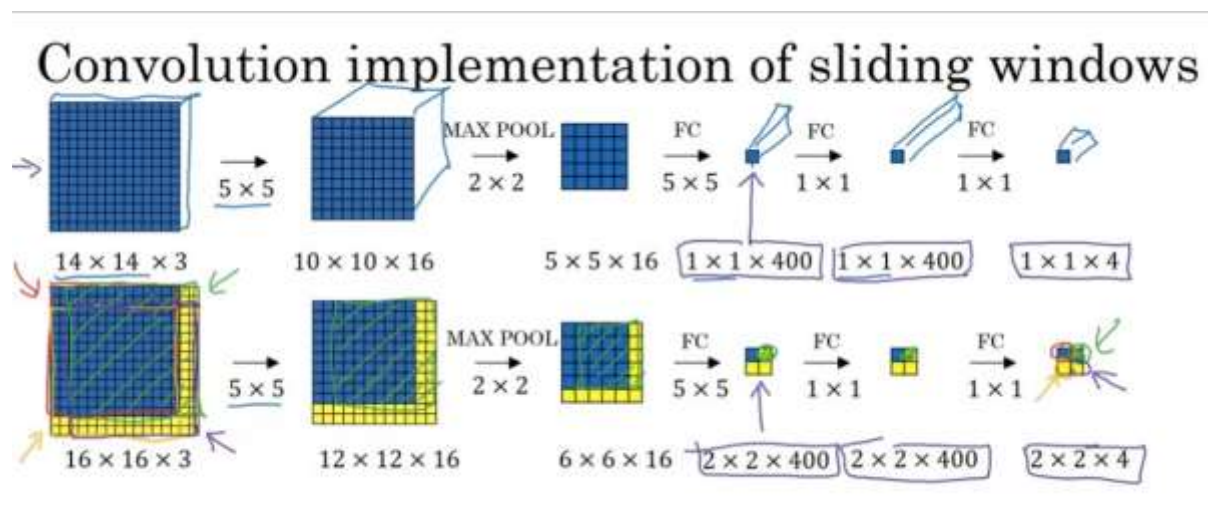
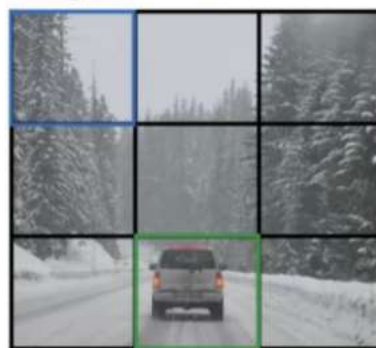
This consists of a backbone which extracts features from an image and the head plays role in the detection phase i.e., giving bounding box using the extracted features

How this works?

(Object Detection Example)

First the given image is divided into several parts/frames.

Now in each frame, we try to detect the object using the sliding window approach:

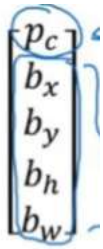


Convolution Implementation of Sliding windows:

A frame (image) is fed into the above CNN. First, it'll be passed through a conv.net (using say a 5x5 filter) to extract the features from it. Then it'll be sent through a MAXPOOL which reduces the image size but not the channels.

This will be repeated several times extracting one feature at a time. Finally, we'll be left with a 1x1xn which will be fed into a Fully Connected NN (can be implemented using CNN only) as the input layer.

Now just like in an ANN, this i/p layer will be passed through conv. layers (adjusting weights) and finally through a 'softmax function' to reach the output layer which will be $1 \times 1 \times 5$ as shown in the fig.



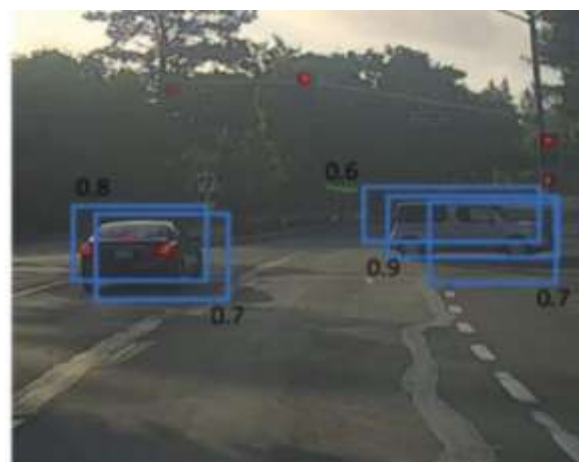
- Bounding Box (p_c -probability of presence of obj., (b_x, b_y) co-ordinates of midpt. of bounding box, (b_h, b_w) height and width of bounding box).

Note that we don't have to do the above process sequentially for each frame sized partition but can pass the whole image into the prev. CNN to get outputs corresponding to each frame in a matrix.



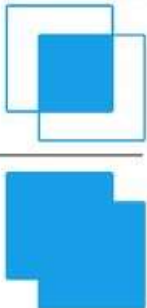
Non-max suppression:

Now after the said process is done, we'll have several bounding boxes given by diff. frames. In order to chose the best one, we'll follow NON-MAX SUPPRESSION:



1. Discard all boxes with $p_c \leq \text{threshold_p}$ (say 0.6)
2. Pick the box with largest p_c . Output that as a prediction.
3. Discard any remaining box with $\text{IoU} \geq 0.5$ with the box output in the prev. step.

NOTE: IoU is defined as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Final steps after detection

Now that we got a bounding box corresponding to our license plate, we can obtain the co-ordinates of the end-points of the license plate using our output vector (p_c , b_x , b_y , b_h , b_w).

After computing the co-ordinates, we extract the plate out of the image and scale it uniformly to an appropriate size which will be later used for recognition of characters present in it.



Now, that we obtained our “region of interest”, we now need to “recognize” each character present in it to get the license number.

Optical Character Recognition (OCR)

Text Recognition:

One of the most popular algorithm used for character recognition is ‘Feature Extraction or Pattern Matching’:

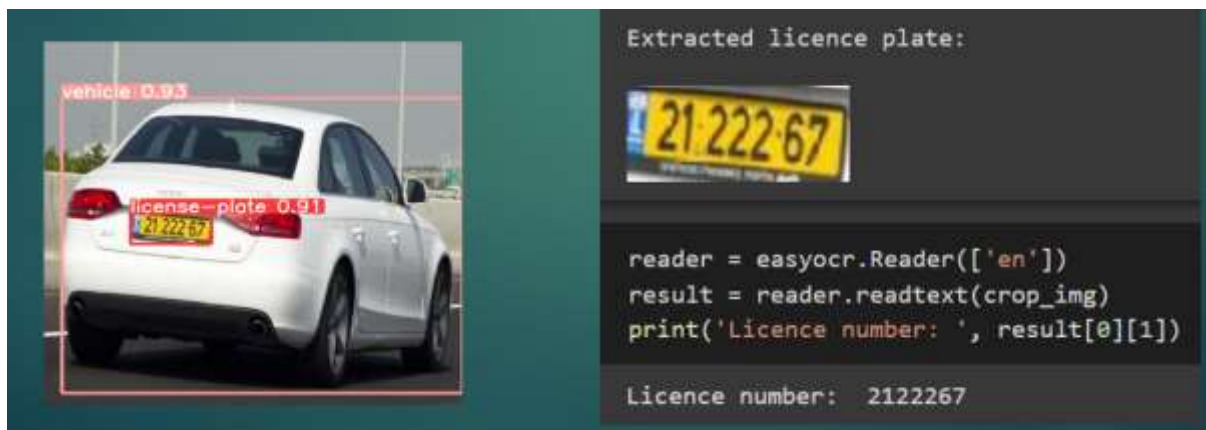
Feature extraction (Pattern Matching)

Feature extraction breaks down or decomposes the glyphs (isolated character image) into features such as lines, closed loops, line direction, and line intersections. It then uses these features to find the best match or the nearest neighbor among its various stored glyphs.

EasyOCR:

EasyOCR is a Python library for Optical Character Recognition (OCR) that allows you to easily extract text from images and scanned documents.

We'll be using EasyOCR to read the text from the detected licence plate in order to obtain the 'text version' of the licence number.



3) Implementing – Training, Validation & Testing

Detection part:

Loading and Training phase:

Now that we've seen how our algorithm works, it's time to implement it on our dataset.

First, we load the YOLOv8 model (along with the pretrained weights) for object detection.

We train the loaded model on our custom dataset in order to make the model serve our purpose i.e., licence-plate detection.

We perform this on our 'train-split' of our dataset for 50 epochs as shown:

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
50/50	7.03G	0.4179	0.3138	0.9419	14	800: 100% 27/27 [00:13<00:00, 2.01it/s]	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 3/3 [00:03<00:00, 1.33s/it]
	all	70	218	0.842	0.815	0.865	0.678
	license-plate	70	84	0.925	0.869	0.923	0.715
	vehicle	70	134	0.758	0.761	0.806	0.641

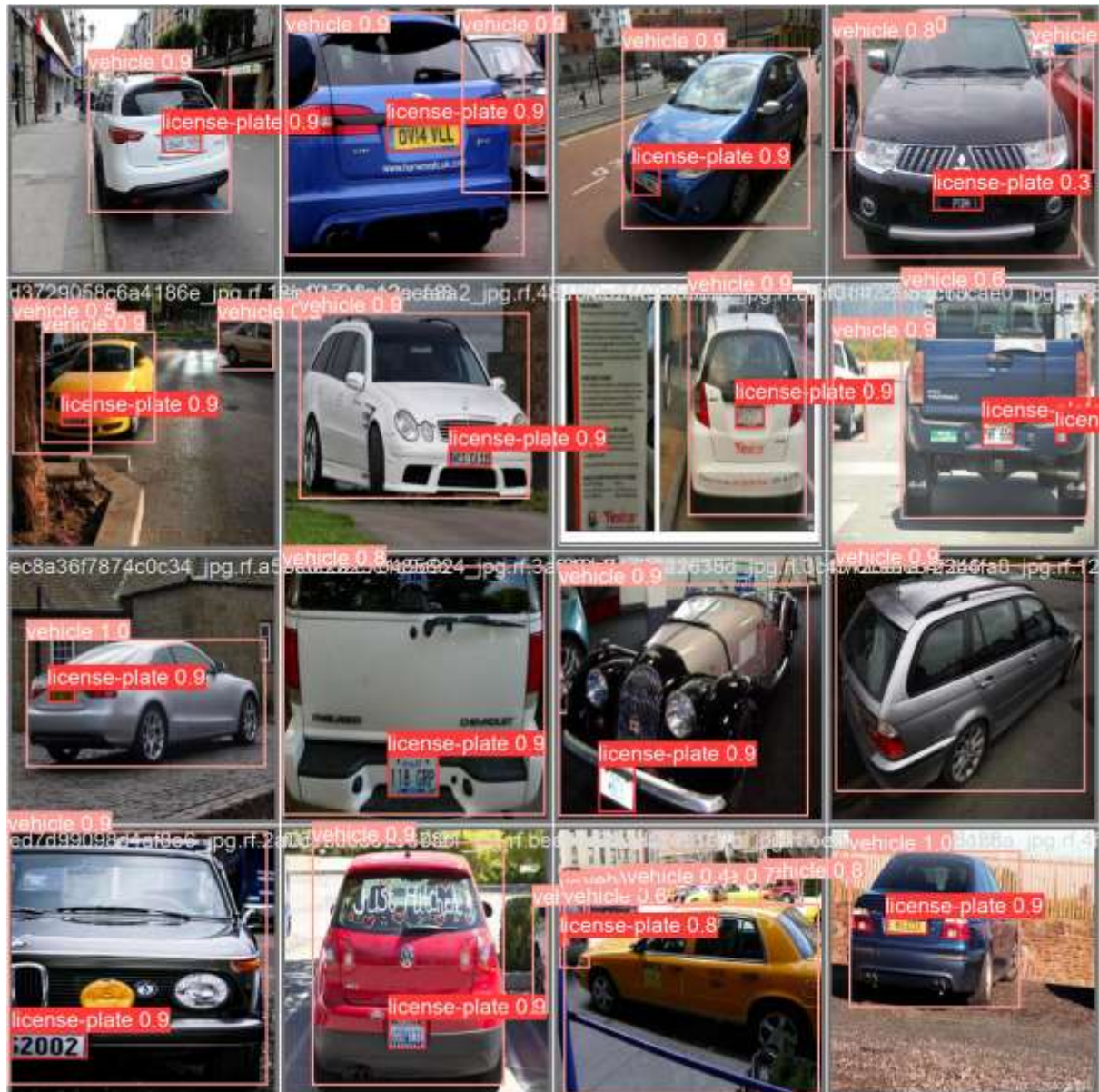
50 epochs completed in 0.346 hours.

Validation:

Now that we've trained our model. It's time to validate it.

When we run detection on one of the validation batches, this is what we got.

We can see our model is able to predict both the vehicle and licence-plate pretty accurately.



Testing phase: Images

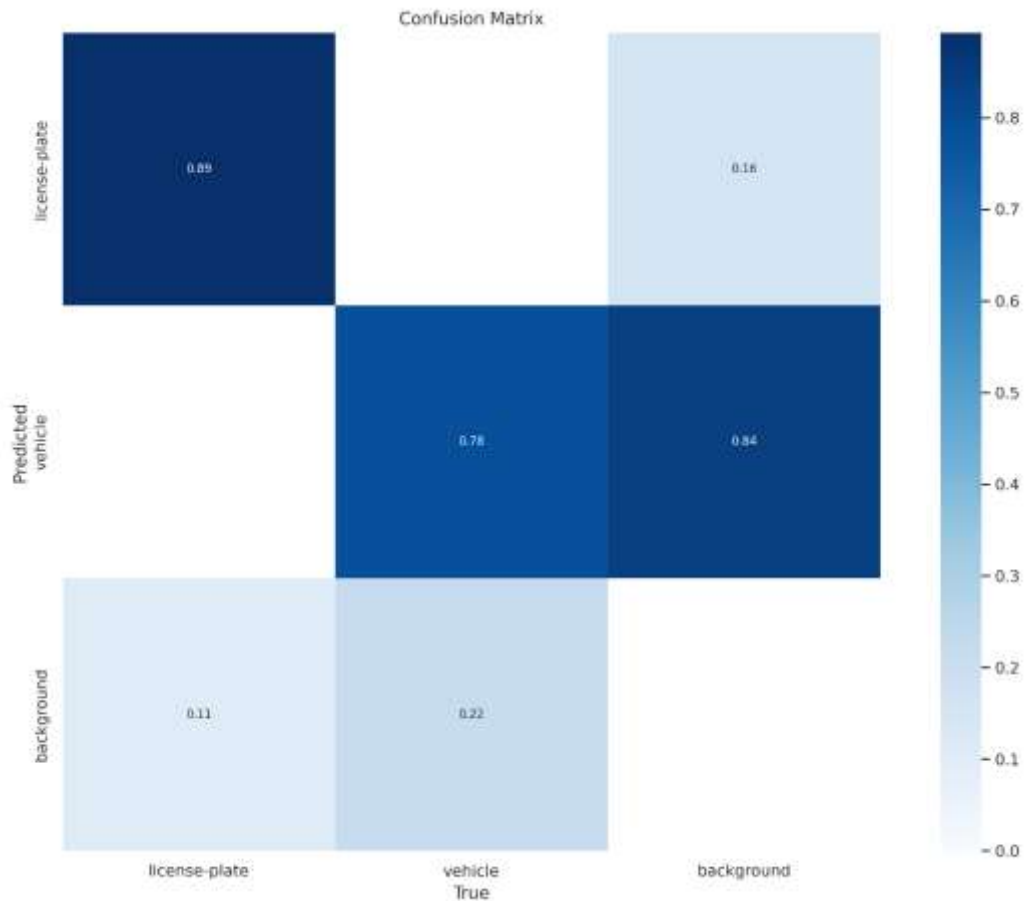
```
image 22/35 /content/datasets/licence-plate-detection-1/test/images/c18b13b1b6e95a.jpg.rf.6a1fa3a8bda2e2b1e43bf11ac0511.jpg: 800x800 1 licence-plate, 2 vehicles, 11.7ms
image 23/35 /content/datasets/licence-plate-detection-1/test/images/c778e22ac987e69.jpg.rf.14824c7286644fcdb1b145d8f7f1829.jpg: 800x800 2 vehicles, 11.2ms
image 24/35 /content/datasets/licence-plate-detection-1/test/images/2c028f18d1e8e75b1.jpg.rf.8c592467e4d8625697c68ffcc53e933.jpg: 800x800 2 licence-plates, 4 vehicles, 11.2ms
image 25/35 /content/datasets/licence-plate-detection-1/test/images/c78f5232201406a3.jpg.rf.19b2828c3e848461ba68c49f4e4119.jpg: 800x800 1 licence-plate, 1 vehicle, 11.3ms
image 26/35 /content/datasets/licence-plate-detection-1/test/images/c81e9321216a8b8b.jpg.rf.cdf4825c11a8766548652874c561f0af.jpg: 800x800 1 licence-plate, 2 vehicles, 11.2ms
image 27/35 /content/datasets/licence-plate-detection-1/test/images/c846754537789d22.jpg.rf.6fca29810d0b8e89135e1981039961+f.jpg: 800x800 2 licence-plates, 3 vehicles, 11.2ms
image 28/35 /content/datasets/licence-plate-detection-1/test/images/c8b8c75fc7cc773.jpg.rf.ca808b8d6aefc9061285ab3c74b5b8d.jpg: 800x800 1 licence-plate, 1 vehicle, 11.3ms
image 29/35 /content/datasets/licence-plate-detection-1/test/images/cc1a244a290388.jpg.rf.50f5814cfc24e13c506de412e9e1ad.jpg: 800x800 2 licence-plates, 1 vehicle, 12.6ms
image 30/35 /content/datasets/licence-plate-detection-1/test/images/cdb1e8621d1c624e.jpg.rf.4c182879ffac45979f13ff765e7346c.jpg: 800x800 1 licence-plate, 1 vehicle, 12.1ms
image 31/35 /content/datasets/licence-plate-detection-1/test/images/ce9f77bc940e97289.jpg.rf.92d2dad2803d7a676a0029f2b6b11.jpg: 800x800 2 licence-plates, 2 vehicles, 12.1ms
image 32/35 /content/datasets/licence-plate-detection-1/test/images/d827c8c37db86de3c.jpg.rf.8117c797f26d3c58cdc4fdaa2ffc197e.jpg: 800x800 1 licence-plate, 1 vehicle, 12.1ms
image 33/35 /content/datasets/licence-plate-detection-1/test/images/d27e894c9b124ff8.jpg.rf.ea338e5d88b52b86c7c663b717fcd0.jpg: 800x800 1 licence-plate, 3 vehicles, 12.3ms
image 34/35 /content/datasets/licence-plate-detection-1/test/images/d28d71c2e90c16ed.jpg.rf.3a7852479528c73aed64f8b391dc4ca.jpg: 800x800 2 licence-plates, 10 vehicles, 12.1ms
image 35/35 /content/datasets/licence-plate-detection-1/test/images/d2fe2b47668efdb9.jpg.rf.5129599401434aa13e21cfa3956dabc.jpg: 800x800 1 licence-plate, 2 vehicles, 11.2ms
Speed: 0.7ms pre-process, 16.5ms inference, 4.0ms postprocess per image at shape (1, 3, 800, 800)
Results saved to runs/detect/predict3
```



Performance Metrics

Confusion Matrix:

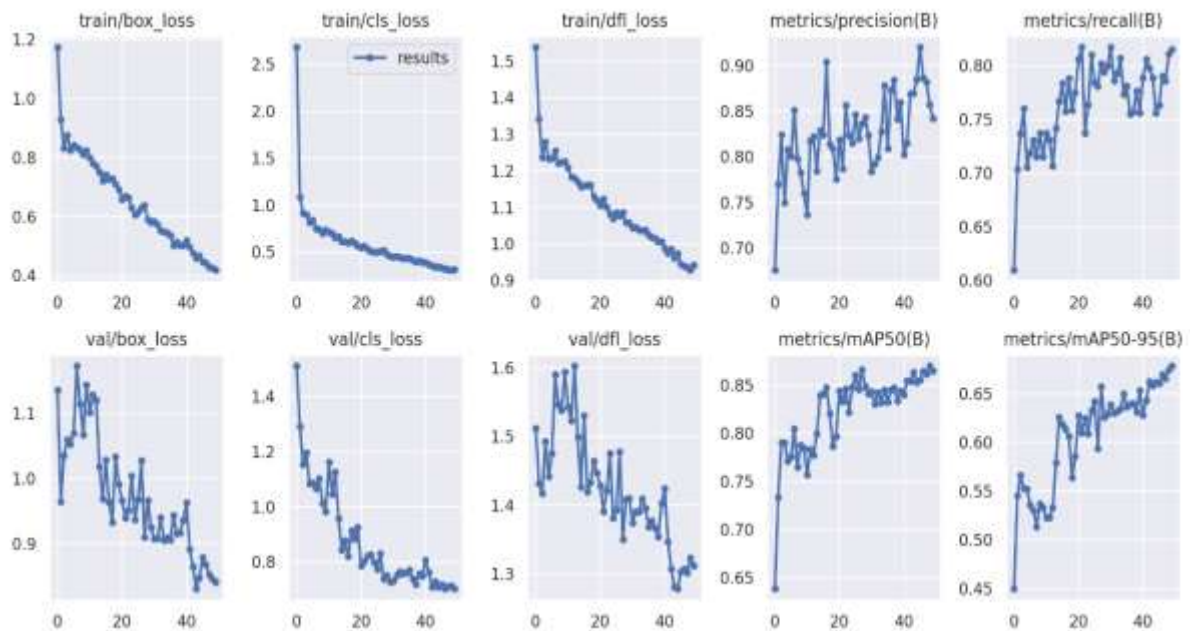
The suitable performance metric for our object-detection problem is the 'Confusion Matrix'.



We can see that after training, our model is able to detect the licence plate correctly (True-Positive) 89% of the time i.e., around 90% accuracy.

Similarly, the same for object 'vehicle' is 0.78 (around 80%).

Loss graphs:



Also while training, we can see the losses (i.e., box_loss, cls_loss and dfl_loss) are decreased over each epoch and eventually converged.

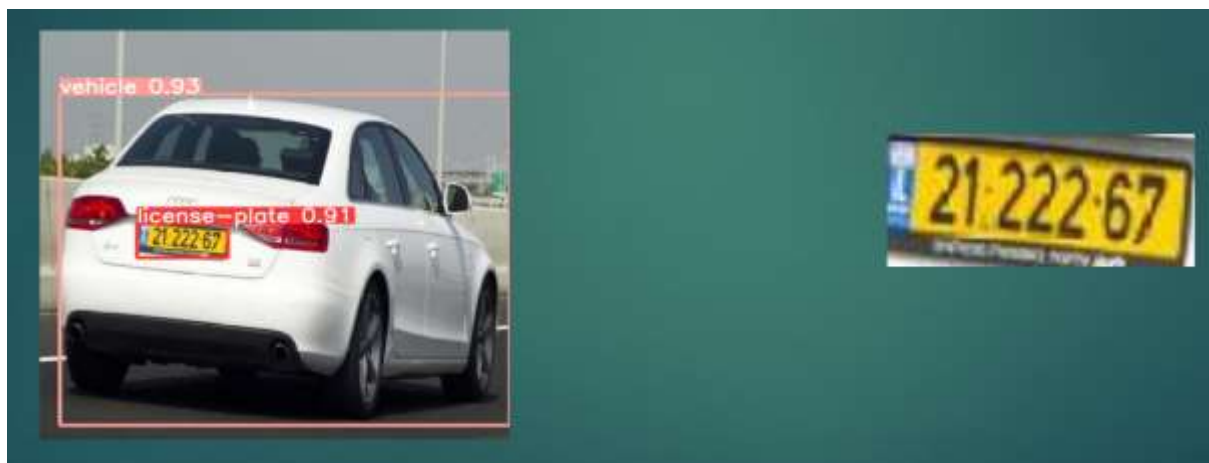
This indicates that our model has 'learned' and is able to predict more accurately at the last few epochs.

Same applies for metric (precision) which gradually increased over each epoch.

Implementing Recognizing part:

Extracting bounding box

- From our detection output, we can have the co-ordinates of the bounding box and we should extract/crop that part out of the image before passing it onto EasyOCR.



Applying Recognition (EasyOCR)

Now we can just pass on the 'cropped_image' into the easyocr.reader to obtain the text:

```
Extracted licence plate:  
  
  
reader = easyocr.Reader(['en'])  
result = reader.readtext(crop_img)  
print('Licence number: ', result[0][1])  
  
Licence number: 2122267
```

References:

Convolutional Neural Networks

DeepLearning.AI

- By Andrew NG

Ultralytics YOLOv8 Docs (for training on custom dataset)