

License Plate Detection and Recognition using YOLO v4

Meena Deshpande¹, Veena M.B.²

¹Research Scholar Electronics and Communication Department BMS College of Engineering,
Bangalore

Sr. Member, IEEE Associate Professor Electronics and Communication Department BMS
College of Engineering, Bangalore

Abstract: -Object identification, one of the most fundamental and difficult tasks in computer vision, aims to find object instances, particularly small objects, in natural images from a variety of specified categories. Detection frameworks, object feature representation, and object proposal are all aspects of generic object detection. The proposed work performs small object detection like locating and recognizing the number plate, color of the number plate and character on the number plate by using Yolov4 and feature fusion approach. Proposed method can overcome different challenges in object detection and shows competitive results for small object detection with 86% detection accuracy at 45fps.

Keywords: - Small Object Detection, YOLOv4, LPR, feature fusion

I. INTRODUCTION

In computer vision, small object identification is a fascinating problem. With the fast advancement of deep learning, it has attracted the interest of academicians, scholars, who have come up with novel techniques. Region proposals, split grid cells, multi scale feature maps, and a novel loss function are among the suggested advances. As a result, object detection performance has recently improved significantly. Despite recent breakthroughs, accuracy of small object identification accuracy remains significantly lower than that of bigger objects. As a result, one of the most difficult issues in object detection is detecting small objects. Since certain applications, such as autonomous driving, only need the identification of small objects within a larger area and not precise localization. localization criteria may need to be generalised as a function of size.

In this study, we propose a novel approach to detect the object for LPR dataset.
Key contributions of the work following

1. First, we partition the original image into blocks (block sizes range from 400 to 400 pixels), which may be used to detect small objects, and then we scale each block to a set size (300 to 400 pixels) for training.
2. Next, the original image is down sampled into multiples of 1/2. For the LPR dataset, we suggest a unique strategy to detecting the object. In a single image block, large objects may be entirely covered.

Third, when the YOLO model's feature map decreases, the small object's distinctive information vanishes. As a result, a feature fusion mechanism is included in the YOLO model to ensure that tiny objects in a big picture are detected precisely.

II. RELATED WORK

Since fully convolutional networks are difficult to train at the time, Overfeat has a considerable speed advantage but is less accurate than RCNN [1]. The performance benefit comes from the fully convolutional network's convolution process being divided between overlapping windows. YOLO splits an image into a $S \times S$ grid, with each predictor predicting C class probabilities, B bounding box locations, and confidence scores. YOLO is designed to be quick, with real-time speeds of 45 frames per second and Fast YOLO [2] speeds of 155 frames per second. YOLO automatically encodes contextual information about object classes and is less likely to forecast false positives in the background since it views the complete image when making predictions. Because to the coarse division of bounding box location, scale, and aspect ratio, YOLO causes more localization mistakes than Fast RCNN. According to [2], YOLO may fail to locate some objects, particularly little ones, due to the coarseness of the data. On datasets with multiple items per image, such as MS COCO, it is unknown to what degree YOLO can convert to high performance. YOLO9000 and YOLOv2 are improved versions of YOLO in which the custom Google Net[3] network is replaced with the simpler DarkNet19, plus batch normalisation [4], removing the fully connected layers, and using good anchor boxes learned via k-means and multiscale training, as proposed by Redmon and Farhadi [2][5]. YOLOv2 beat the competitors in routine detection tasks. Redmon and Farhadi in [5] suggested a hybrid optimization technique that uses Word Tree to integrate data from both datasets to produce YOLO9000, which can identify over 9000 item classes in real time. YOLO9000 can perform weakly supervised detection, i.e., recognising object classes without bounding box annotations, because to this cooperative training. SSD Liu et al. [6] presented SSD (Single Shot Detector) to retain real-time speed without losing detection accuracy. SSD is quicker than YOLO [2] and has an accuracy comparable to region-based detectors like Faster RCNN (Ren et al. 2015). To achieve rapid detection speed while maintaining high detection quality, SSD successfully incorporated principles from RPN in Faster RCNN[7], YOLO, and multiscale CONV features [8]. SSD, like YOLO, predicts a set number of bounding boxes and scores, then performs an NMS step to generate the final detection. SSD's CNN network is completely convolutional, with early layers based on conventional architectures like VGG[11], followed by multiple auxiliary CONV layers that get smaller as the network grows. Because the information in the final layer may be too coarse in terms of spatial resolution to allow exact localization, SSD uses numerous CONV

feature maps to detect over different scales, each of which predicts category scores and box offsets for bounding boxes of suitable sizes. On the VOC2007 test at 59 FPS, SSD scores 74.3 percent mAP versus Faster RCNN 7 FPS / mAP 73.2 percent or YOLO 45 FPS/mAP 63.4%.

The most popular frameworks [12], Fast RCNN[13], Faster RCNN[7], YOLO [2], SSD[6] have consistently boosted detection accuracy and speed, with the CNN architecture playing a key role. As a result, research into the building of innovative networks has accounted for most recent increases in detection accuracy. They performed object feature representations, such as developing invariant features to account for geometric variations in object scale, pose, viewpoint, and part deformation, and performing multiscale analysis to improve object detection. The detection accuracy of SSD model is lower than Faster R-CNN. Small objects are eventually lost as the network deepens all through the SSD model's convolution phase. as VGG [11], followed by a series of smaller additional CONV layers[11]. Because the information in the final layer may be too coarse in terms of spatial resolution to allow exact localization, SSD uses numerous CONV feature maps to detect over different scales, each of which predicts category scores and box offsets for bounding boxes of suitable sizes. SSD scores 74.3 percent mAP on the VOC2007 test at 59 FPS for a 300300 input, compared to Faster RCNN 7 FPS / mAP 73.2 percent or YOLO 45 FPS / mAP 63.4 percent. Yolo-V3 achieved a mAP of 37 on the COCO-2017 validation set with input resolution 608x608, whereas the competing Mobile net-SSD architecture received a Map of 30. YOLOv4 was formulated to overcome YOLOv2 and YOLOv3's inability to identify small objects.

III. PROPOSED WORK

The process of detecting the tiny objects involves the following steps:

- i. Collection of Image: First an image of the tiny object is taken, and a dataset will be formed for further process.
- ii. Image Pyramid
- iii. Image Training: Image training is done to simplify the processing and to be able to identify the tiny object in an image.
- iv. Image Acquisition: The number of tiny objects is grouped together in the form of data sets

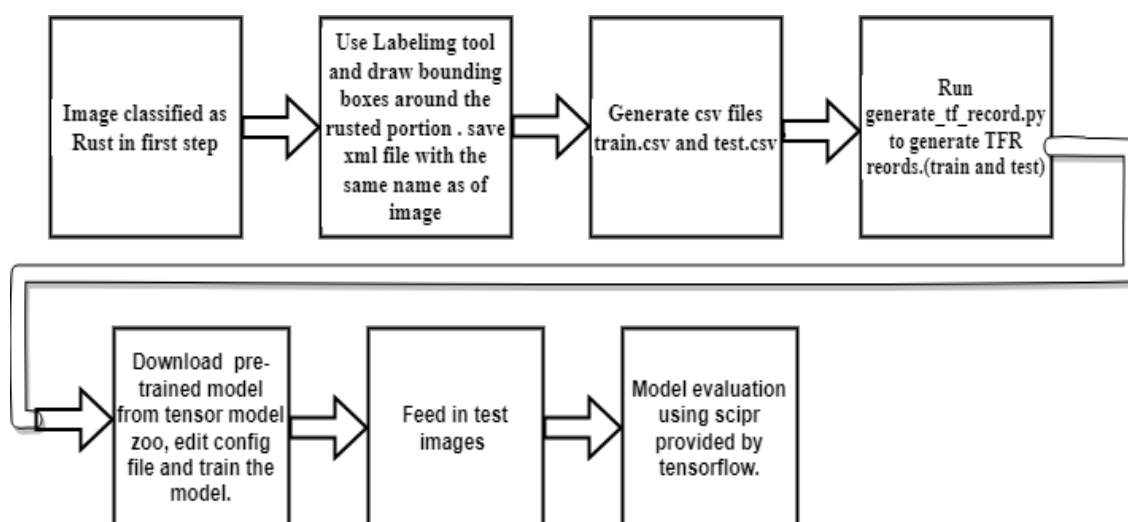


Figure. 1 Methodology of object detection using YOLOv4

First, a dataset composed of 1000 images of cars that contains Indian license plate, for each image is prepared. An xml/txt file using a desktop application called LabelImg is done. Load `xml_to_csv.py` to change labels (xml/txt files) into the .csv format. Then run and generate the `tf_records.py` to generate `tf_records`. Next download the pretrained model from Tensor Flow model zoo, and edit configuration file, and train the model. Next feed in the test images and check rust detection at work. Then evaluate the model, using script provided by Tensor Flow. LabelImg is a tool for annotating graphical images. It's developed in Python and has a graphical user interface built using Qt. Annotations are saved as XML files in the PASCAL VOC format, which is the same format that ImageNet uses. It also supports the YOLO format. A downloaded network that has been previously trained on a large dataset, often for large-scale image classification tasks, is referred to as a pre-trained model. To modify this model to a specific purpose, you can employ a pre-trained model or transfer learning. Transfer learning for image classification is based on the idea that if a model is trained on a big and general enough dataset, it may successfully serve as a generic model of the visual world. In AI, pre-training is when a model is trained on one task to help it create parameters that may be applied to subsequent tasks.


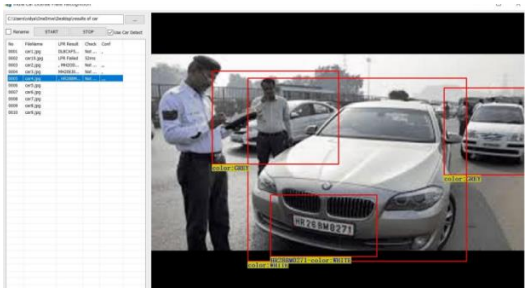

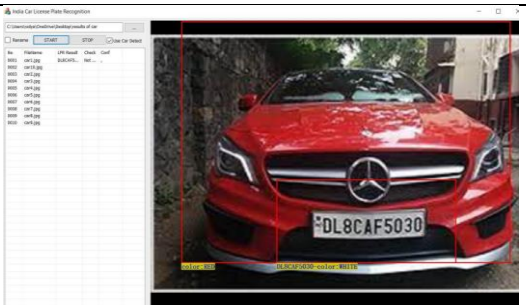
Dataset is sorted into train, test, and validation, second by their class. This repository contains a collection of pre-trained models that have been adapted to TensorFlow.js. The models are available through NPM and unpkg and may be used in any project right away. With TensorFlow.js, they may be utilised directly or in a transfer learning context. Many frequent instances for importing data into a model are handled by TensorFlow's Dataset API. This is a high-level API for reading input and converting it into a training format. `tf.data.experimental` is used since the dataset is a CSV-formatted text file. To parse the data into a suitable format, use the `_csv` dataset function. Since this function creates data for training models, the default behaviour is to shuffle the data (`shuffle=True`, `shuffle buffer size=10000`) and to repeat the dataset indefinitely (`num epochs=None`).

IV. EXPERIMENTAL RESULTS

Experimental results obtained by running India LPRdemo executable Applications for detecting multiple images and India video LPR executable application for detecting multiple objects in video is given below. It can be observed from the observation table that test1, test2, test3 and test4 images the LPR is detected and for test5 image LPR is not detected due to environmental issues. Detection results consists of the following details

Table 1. Segmentation result reading filename, LPR number and confidence score

Sl. No	Filename	LPR Result	Conf.
1	192.168.0.141_1568621723.png,	KA51AB9116	99.3%_
2	192.168.0.141_1568621724.png,	KA51AB9116	99.5%_
3	192.168.0.141_1568621798.png,	KA03AH1355	99.9%_
4	192.168.0.141_1568621843.png,	KA05AH0695	99.9%_
5	192.168.0.141_1568621897.png,	KA449619_,99	.8%_
6	192.168.0.141_1568621904.png,	KA01AJ8004	80.5%_

	Realtme Input image captured by camera	Small Object (LPR) detected
1		
2		

Licence plate number with the confidence score is read. This can be used to recognize fake licence plate



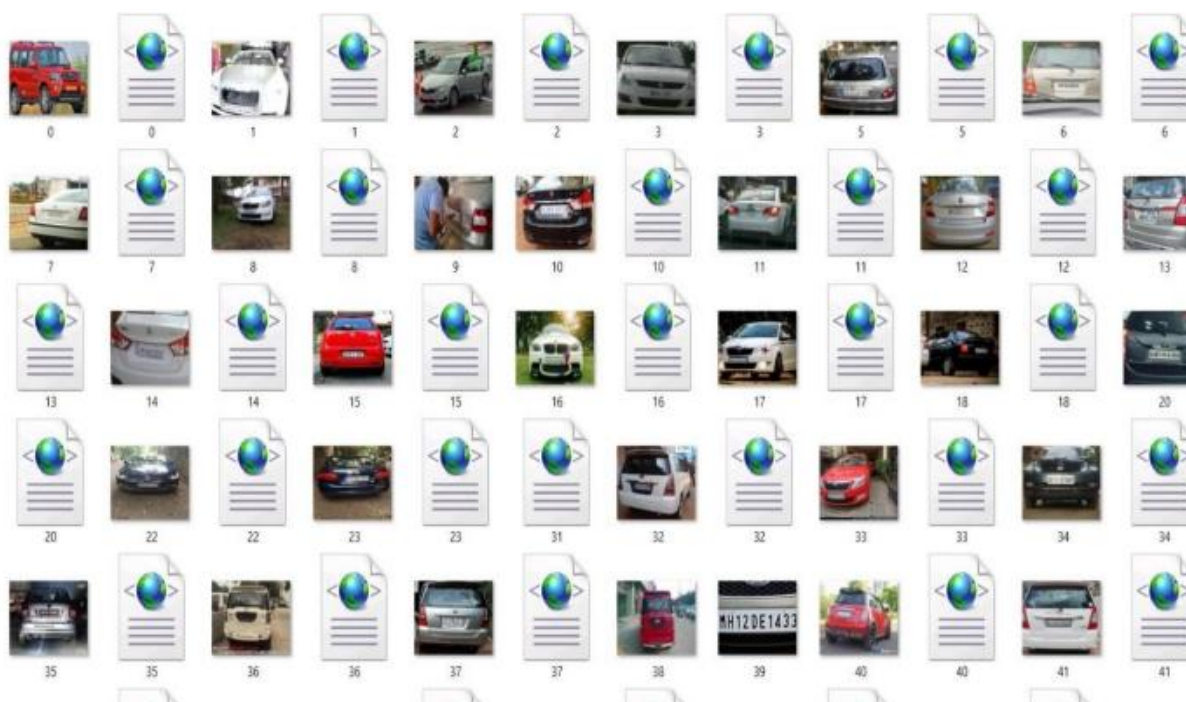
Figure 2. Licence plate detection and recognition

V. CONCLUSION AND FUTURE WORK

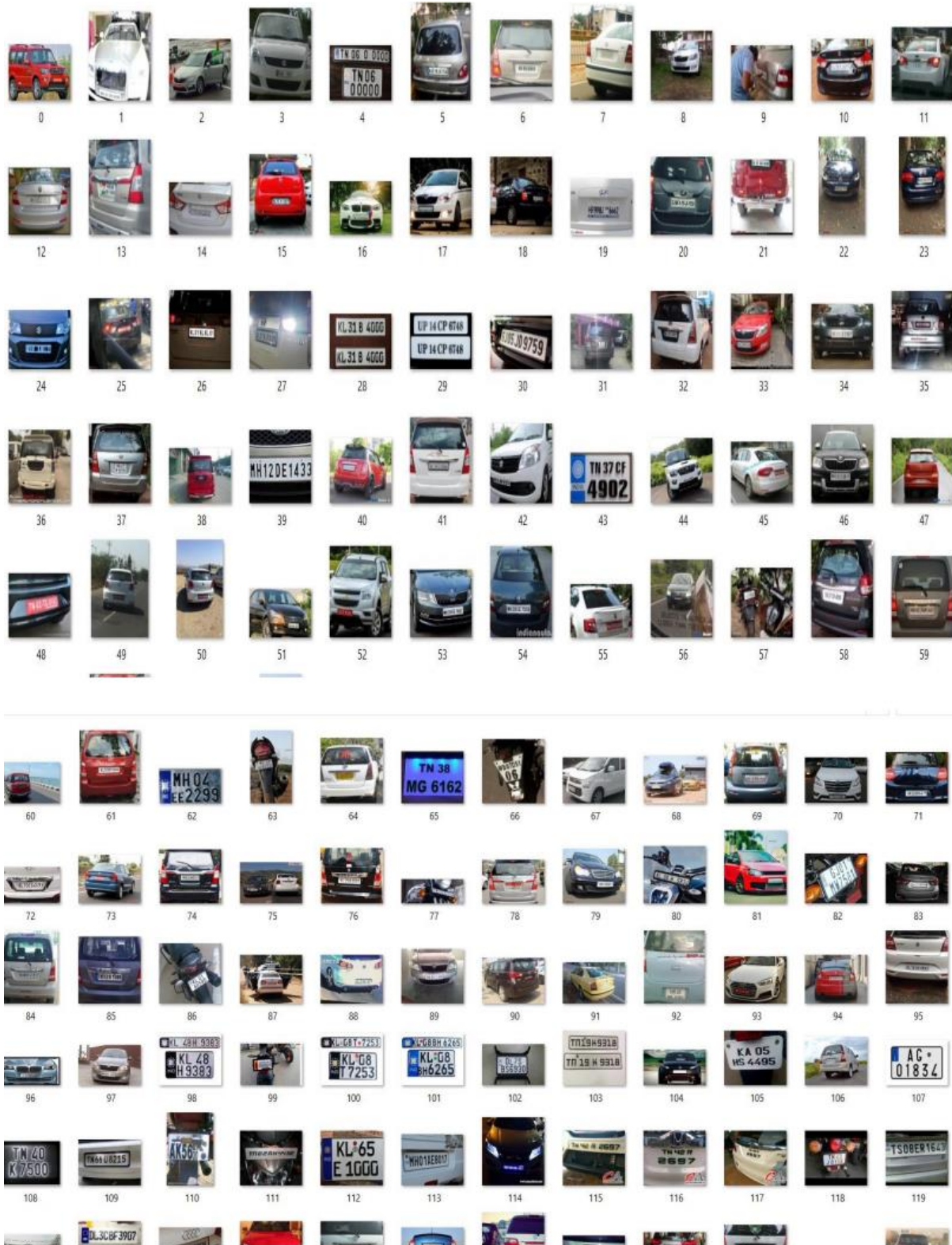
In this paper, we propose a YOLOv4 model-based object identification system for detecting licence plates. The successful transmission of object characteristics is maintained by the feature fusion of the convolution layers. A technique of image block approach was introduced to the training process to increase the detection performance of tiny objects. Furthermore, we suggest an image pyramid for large objects, which successfully addresses the problem of large object feature loss due to picture segmentation. However, to develop a new dataset for LPR detection methods, we specified labelling criteria and object types. The suggested detection method for LPR detection has high detection performance, according to our experimental results. However, the system requires more datasets and training in a variety of settings. The low resolution of the number plates on vehicles in video frames and extreme weather conditions under typical surveillance systems pose a problem in detecting number plate. So, at that time to overcome this problem we need large datasets for training purpose.

Bibliography

- [1] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR (pp. 580–58)
- [2] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real time object detection. In CVPR (pp. 779– 788).
- [3] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In CVPR (pp. 1–9)
- [4] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In ICCV (pp. 1026–1034)
- [5] Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In CVPR
- [6] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., & Berg, A. (2016). SSD: Single shot multibox detector. In ECCV (pp. 21–37).
- [7] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real time object detection with region proposal networks. In NIPS (pp. 91–99)
- [8] Hariharan, B., Arbeláez, P., Girshick, R., & Malik, J. (2016). Object instance segmentation and fine-grained localization using hypercolumns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4), 627–639.
- [9] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real time object detection. In CVPR (pp. 779– 788).
- [11] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large scale image recognition. In ICLR
- [12] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR (pp. 580–587)
- [13] Ren, S., He, K., Girshick, R., Zhang, X., & Sun, J. (2016). Object detection networks on convolutional feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7), 1476–1481



xml files of annotated images



Dataset