Journal of Engineering Science and Technology Review 17 (4) (2024) 198-204

Research Article

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Comprehensive Study of YOLO Versions for Front and Rear-View Classification of Vehicles in Context of Indian Roads

Manas Kumar Rath and Prasanta Kumar Swain*

Department of Computer Application, MSCB University, Baripada, India

Received 14 May 2024; Accepted 3 July 2024

Abstract

Ever since Computer Vision was introduced, humanity has seen various ways to detect or classify objects of various types. Depending upon the context in consideration, the performances of models vary with respect to their evolution or even upon the nature of the data in hand. The classification of front or rear views in vehicles forms an integral part when we go ahead with deciding whether a given vehicle is moving in the correct lane. In the context of Indian streets, we have various challenges like rural unmarked roads, faded markings, shaded situations from poles or trees, etc. Hence instead of detecting lanes, an alternative way is to detect whether the vehicle(s) ahead is facing toward or away from our vehicle. Various deep learning architectures have been proposed in this aspect to detect or classify objects like the networks from Visual Geometry Group, You Only Look Once, Inception Networks, Residual Networks, etc. In this paper, we have performed a comparative analysis of performance on various versions of You Only Look Once for its evolution over time.

Keywords: Vehicle View Classification, Convolutional Neural Networks, Deep Learning, YOLO

1. Introduction

The various problems like faded lane markings, improper rural roads, etc. make it quite challenging to address the problem through the street's view. The real-time classification of vehicles' views in the non-ideal context of Indian lanes is an integral part of detecting whether a vehicle is moving in the correct lane. Hence, we address this problem of correct lane detection using the view of other vehicles(s) ahead. Considering our own vehicle as a reference, the dash-cam acquires the images of the vehicle(s) ahead. Then is image is classified into whether it is the rear view or the front view of the vehicle. If the dashcam detects the front view, it means that our own vehicle is in the wrong lane and vice versa. We have comprehensively surveyed the performances of various main versions of YOLO (You Only Look Once) architecture from version 1 to version 8. Other versions of YOLO like YOLOX [1], YOLOR [2], DAMO-YOLO [3], PP-YOLO [4], etc. have not been considered in this study. "Table 1" shows the evolution of YOLO algorithms that we'll be using ahead in this paper.

Also, the COCO (Common Objects in Consideration) dataset by Microsoft [5] is a widely used one when it comes to training and testing the models. Other datasets include the GTI's vehicle image database [6], Caltech Database [7], and Tu-Graz-02 Database [8]. We've used our own dataset which will be discussed later in one of the subsequent sections.

Throughout our work, we discuss various aspects of each version – architecture, features, strengths, and performance based on mean average precision. Since, for Indian roads and vehicles, it isn't that easy to classify the front or rear of vehicles (due to the conditions mentioned previously), hence, classification based on number plate or structure might need to be addressed to classify the vehicle's front or rear.

In this section, we have introduced the concept of classification of the view of vehicles and its significance. Section 1.1 discussed the system model where we described the dataset that has been manually collected and used for our analysis. The literature survey of previously published works has been carried out in Section 2. Section 3 describes the proposed work, the accuracy metrics used, and its implementation through each of the used versions of YOLO.

Table 1. Evolution of YOLO.

Year	Version	Features	Notable Improvements
2015	YOLO-V1	Real-time detection, initial version	Speed and simplicity
2016	YOLO-V2	Improved localization with anchor boxes	Accuracy enhancements
2018	YOLO-V3	Feature pyramid networks (FPN)	Multi-scale detection
2020	YOLO-V4, YOLO-V5	CSPDarknet53, PANet, lightweight YOLO-V5	State-of-the-art performance
2022	YOLO-V6, YOLO-V7	Unofficial iterations, optimization	Speed and efficiency
2023	YOLO-V8	Efficient Net backbone, competitive performance	Balance of speed/accuracy
2024	YOLO-V9	PGI, GELAN, information bottleneck	Efficiency, accuracy

The results have been discussed in Section 4, followed by which we concluded the work in Section 5. The front and rear views of a training sample of a vehicle are shown in "Fig 1" below.

*E-mail address: prasantanou@mail.com
ISSN: 1791-2377
doi:10.25103/jestr.174.21

Fig. 1. Front and Rear views of a vehicle

System Model

The data for the system model is prepared by keeping the Indian road and driving patterns in the form of images which are taken from different video sources. This is adopted with 28 Frames per second by own installed camera in the vehicle. In the first approach, 30000 images were collected to prepare the dataset. The basic challenges found in general Indian road infrastructure is its unstructuredness, featuring irregular merge points, faded or absent lane markings. Additionally, there are no strict restrictions on vehicle types where one can find all type of vehicles moving, resulting in diverse traffic scenarios. Indian roads exhibit irregular and unpredictable turns and drivers frequently encounter challenges such as illegal parking on the roadside, wrong-side driving and even wrong-way movement. These characteristics make Indian roads a complex and dynamic environment for developing driving models.

The dataset includes both the front and rear captures of different vehicles. Considering the 80:20 ratio we split the train and test set. “Fig 2”, “Fig 3”, and “Fig 4” are some of the glimpses of the captured images.

Fig. 2. Rear view of cars

Fig. 3. Rear view of a bus

It can easily be understood in “Fig 5” and “Fig 6” below that the images are taken during nighttime. This also included making the dataset robust and getting trained with more accuracy and efficiency in different light conditions.

In “Fig 6” it can be observed that there are so many different types of vehicles on the move. The image is also taken at nighttime with different intensity of light. The model is trained well to determine the front and rear of the different vehicles in such a complex traffic environment as well. This outcome can suffice whether our vehicle is on the correct lane or not by aggregating the overall data.

Fig. 4. Front view of Autorickshaws, Truck and Cars

Fig. 5. Rear View of autorickshaw and car during the night

Fig. 6. Rear View of cars during the night

2. Related Work

It has been observed that a lot of complexities are encountered in Indian road traffic. Considering this issue, certain approaches are being addressed. Lane detection in complex Indian environments, addressing the poor road conditions mentioned above, has been worked upon in [9]. Using the CNN architecture, NVIDIA also has proposed to keep track of the steering movements in a real-time environment for automated vehicles [10].

For an end-to-end automated vehicle, steering angles have been predicted [11] using the architecture in [10]. A comparison of performances has been performed in [12] for Jacinto Net, VGG-19, and CNN in [10]. Authors in [13] have improved the Jacinto-Net which shows the same performance for Heterogeneous Multi-core platforms. In [14], the authors have proposed two approaches to execute the objective. The rear-view dimensions and edges are taken into consideration in the first method. In the second approach, considering orientation, position, eccentricity, and other features of its backlights it has been observed that the outcome is 89%.

In the mentioned work [15], the automatic recognition of vehicle makes and model (MMR) using frontal views is addressed, The two-stage vision-based consideration for effective front and rear classification is addressed in[16] using Eigen space and SVM. Authors in The MMR using local-tiled deep networks are addressed by the authors in [17].

In [20], VOC and COCO datasets are taken into consideration. Using all the main YOLO architectures, the front-view and the rear-view classification of vehicles are compared. Following the description of the dataset, the accuracy metric and the version architectures are presented along with performances individually.

The publicly available datasets contain images of roads that are well-marked and maintained. But in the context of Indian roads, the previously implemented works fail to work properly due to the faded lane markings and even unmarked lanes at various places. Hence, we took this approach of analyzing the view of the other vehicles instead of the lanes. The view classification is done after regular intervals of time so that sometimes even if no vehicle is present ahead, it would carry on with the previous flag signal until the algorithm runs for the next time.

3. Proposed Work

We used Average Precision (AP) or Mean Average Precision (MAP), a common metric for object detection techniques. Followed by which a post-processing technique called Non-Maximum Suppression (NMS) has been leveraged which reduces the number of overlapping bounding boxes to improve the detection and classification quality. Both have been explained in detail in [23].

However, in [23] ImageNet dataset [21] and the PASCAL VOC dataset have been employed to train YOLO V1 and YOLO V2. The rest of the versions have been trained using the COCO dataset.

3.1. YOLO V1

This consists of 24 convolutional layers with 2 fully connected layers at the end. The architecture is explained in detail originally in [20]. These convolutional layers are used to extract features from the image followed by which the fully connected layers predict probabilities and the coordinates. It is important to note that the model was originally trained using the PASCAL VOC dataset, which consists of 20 classes (i.e., C = 20). For our dataset, since C = 2, the output dimensions of the tensor turned out to be 7 × 7 × 12. We've tweaked all the other models similarly to meet our requirements. The figure for YOLO-V1 is given in "Fig 7".

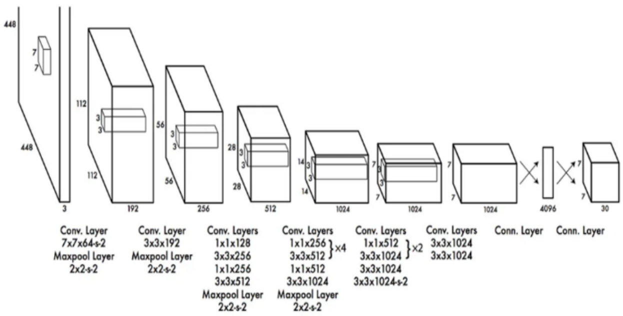

Fig. 7. YOLO V1 architecture.

3.2. YOLO V2

This is an architecture based on Darknet-19 [24] which has 19 convolutional and 5 max-pooling layers. Several other improvements were made to YOLO-V1 so that 9000 categories can be classified using it. The following modifications were made:

- Batch normalization on all the convolutional layers.
- The model was fine-tuned for 10 epochs with a resolution of 448×448 for high-resolution classification.
- The dense layers were removed, and fully convolutional layers were used.
- Prior boxes were used for predicting the bounding boxes.
- k-means clustering was used by the authors to find good priors.
- Trained using images of multiple sizes ranging from 320×320 to 608×608 in batches of 10.
- It predicts local coordinates, unlike offsets as predicted by other methods.
- One pooling layer was removed to obtain a grid of 13×13 from an input size of 416×416. Also, a passthrough layer has been used so that the features aren't lost via spatial subsampling. The architecture of YOLO-V2 [25] has been shown in "Fig 8".

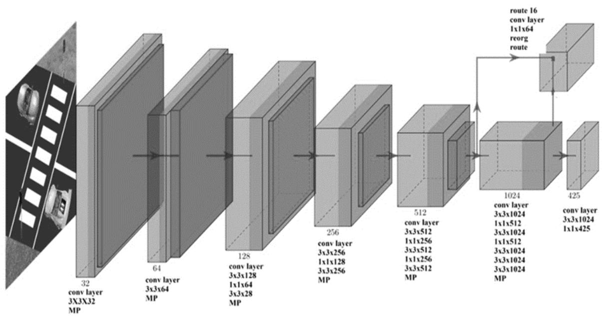

Fig. 8. YOLO V2 architecture.

3.3. YOLO V3

This architecture in "Fig 9" is based on Dark-net-53 [26] which has 53 convolutional layers and the max-pooling layers in YOLO-V2 have been replaced by stride convolutions. In addition to that some residual connections have also been made. Several other improvements were made to YOLO-V2. The following modifications were made:

- Batch normalization and Leaky Rectified Linear Unit activation function on all the convolutional layers.
- Across the whole network, the residual connections connect the input of 1 × 1 with the output of 3 × 3 convolutions in size.
- The predictions are made for multiple grid sizes, hence enabling us to acquire finer detailed boxes, hereby improving the predictions for smaller objects.

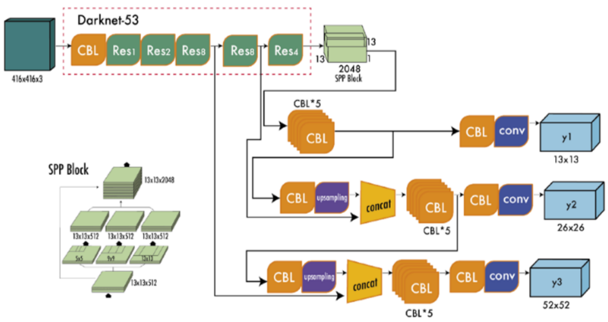

Fig. 9. YOLO V3 architecture.

3.4. YOLO V4

After 2 years, in 2020 the next version of YOLO-V4 was released [28]. This is based on the architecture of modern object detectors which collectively consist of a backbone (CSP DarkNet 53), a neck (SPP + PANet), and a head (YOLO-V3). The input goes to the backbone which is essentially the feature extractor.

The neck is essentially used for multi-resolution feature aggregation. The head generates final predictions as outputs. A schematic architecture [29] is shown in "Fig 10".The key changes [23] in this version include – Bag of Specials (BoS) integration, Bag of Freebies (BoF) integration, Self-Adversarial training, and Genetic algorithms for fine-tuning parameters.

The architecture for YOLO-V4 is shown in "Fig 11". It consists of the following key modules:

- CBM: Convolution, Batch Normalization, Mish Activation.
- CBL: Convolution, Batch Normalization, Leaky ReLU Activation
- UP: Up-sampling
- SPP: Spatial Pyramid Pooling
- PANet: Path Aggregation Network.

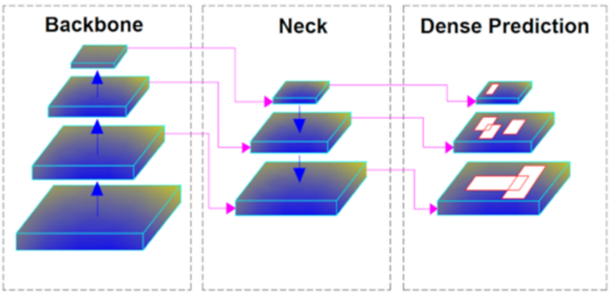

Fig. 10. Modern Object Detection.

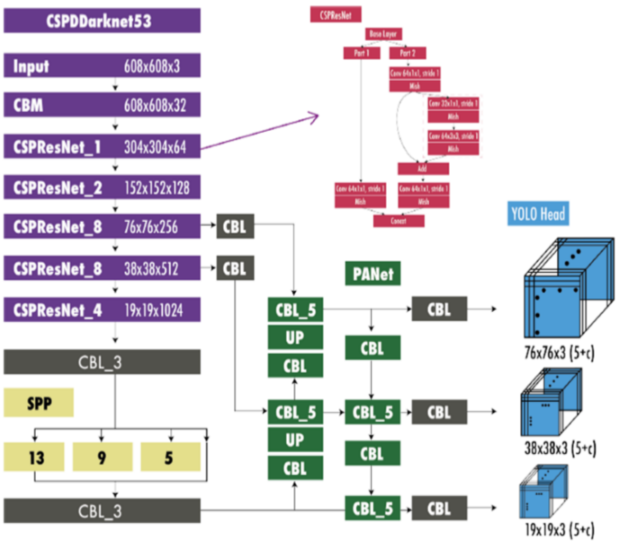

Fig. 11. YOLO-V4 architecture.

3.5. YOLO V5

This has been developed using PyTorch (Bottle Neck CSP) instead of DarkNet. It provides 5 scaled versions namely – nano, small, medium, large, and extra-large versions. As such, there has been no official paper released for YOLO-V5 as such, but Ultralytics actively maintains this open-source model. Authors in [30] have used YOLO-V5 for their image localization and classification tasks. The official architecture as published by Ultralytics is shown in "Fig 12".

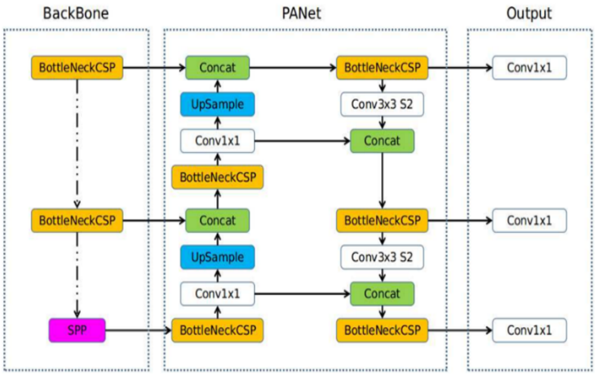

Fig. 12. YOLO V5 architecture.

3.6. YOLO V6

Published in [31] by the Meituan Vision AI Department, this model uses a backbone based on RepVGG (EfficientRep) where there is high parallelism. The neck utilizes PANet integrated with RepBlocks or CSPStackNet. The head is decoupled, inspired by YOLOX. The architecture is shown in "Fig 13".

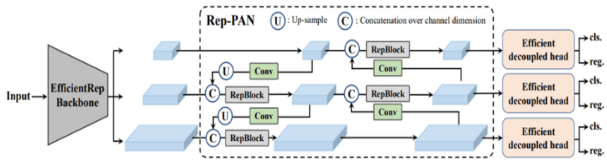

Fig. 13. YOLO V6 architecture.

The new features in this architecture include:

- Label assignment using Task Alignment One-step Object Detection [32].
- VariFocal loss metric [33] for classification and SIoU/GIoU metric loss [34] for regression.
- Self-distillation for both above-mentioned tasks.
- Quantization scheme using re-parameterized optimizers [35] and channel-wise distillation [36] for low latency detection.

3.7. YOLO V7

This version of YOLO [37] was published by the same authors as YOLO-V4. It simply out powered all known object detectors with respect to accuracy and speed in the range of 5 FPS to 160 FPS. The training time increased, but the accuracy improved without affecting the speed much. The major changes in this architecture include:

- Extended Efficient Layer Aggregation Network (E-ELAN) is a way through which models train and fit easily by controlling the shortest longest gradient path.
- Model scaling since YOLO-V7 is an architecture ("Fig 14") based on concatenation. By using techniques such as width or depth scaling, the ratio between input and output channels is changed leading to less hardware usage by the model as shown in "Fig 15".
- The identity connection in RepConv used in YOLO-V6, seemed to affect the concatenation in DenseNet [38] and residual in ResNet [39]. Hence it was removed and renamed to RepConvN.
- Coarse labels for auxiliary (training) and fine

labels for the lead head (output).

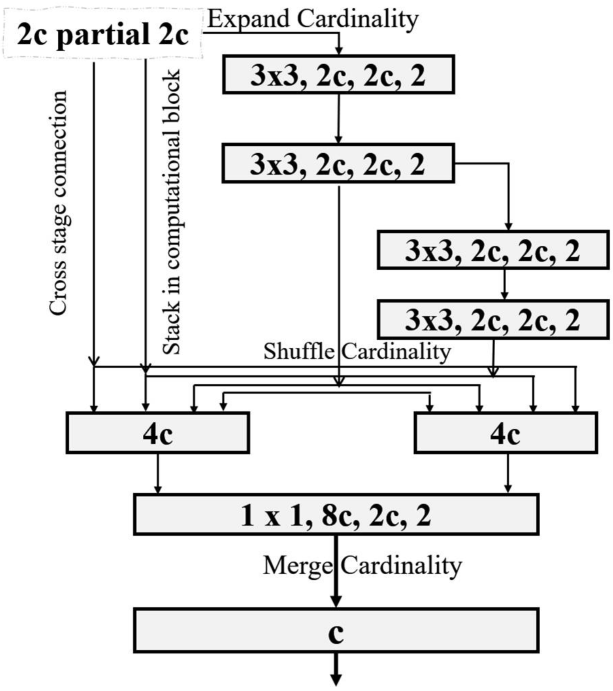

Fig. 14. YOLO V7 architecture.

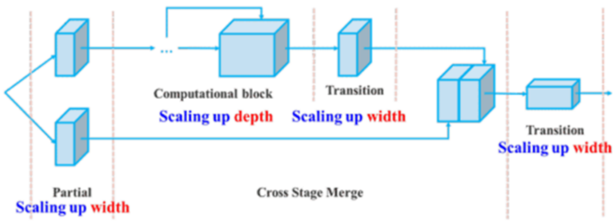

Fig. 15. Scaling in YOLO-V6 architecture.

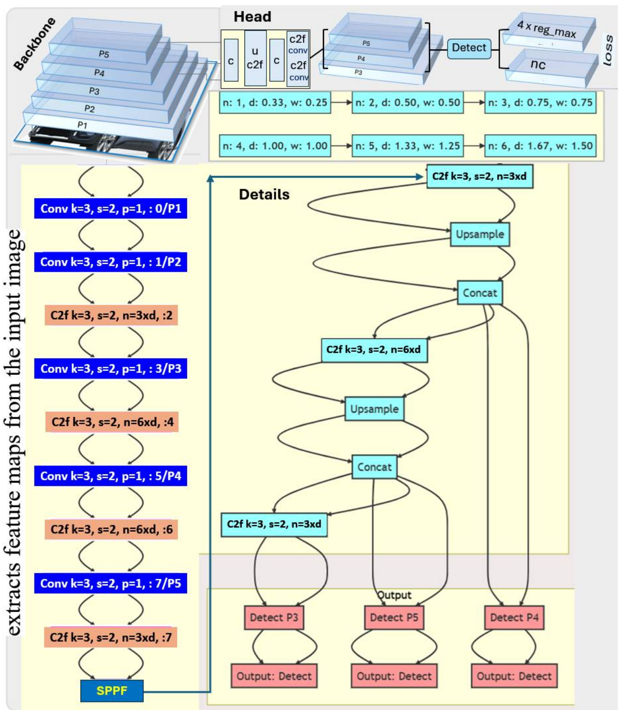

Fig. 16. YOLO V8 architecture.

3.8. YOLO V8

This version was again released by Ultralytics [40] in 2023. In addition to continuing its trend from YOLO-V5, it used mosaic augmentation for training, except in the last 10 epochs so that it doesn't get detrimental.

It also provides 5 scaled versions namely – nano, small, medium, large & extra-large versions. It can be run using a command line interface or also a pip module. The official architecture released by Ultralytics is shown in "Fig 16" for YOLO-V8.

4. Results and Discussion

As discussed in Section 3 above, Average Precision is the metric that has been used to compare these models, trained on our gathered dataset. "Table 2" summarizes the various aspects of the architectures and the accuracies of the model outputs.

Table 2. Evolution of YOLO

Year	Version	Framework	Anchor Box	Backbone	AP
2015	1	Darknet	×	Darknet 24	0.711
2016	2	Darknet	✓	Darknet 19	0.713
2018	3	Darknet	✓	Darknet 53	0.462
2020	4	Darknet	✓	CSPDarknet53	**0.420**
2020	5	Pytorch	✓	Modified CSP V7	0.558
2022	6	Pytorch	×	Efficient Rep	0.515
2022	7	Pytorch	×	RepConvN	0.502
2023	8	Pytorch	×	YOLO V8	0.455

From the table above, we can see that the mean average precision of YOLO-V4 is **0.42**, hence proving to be the best fit for our data and task in hand, followed by YOLO-V8 & YOLO-V3 with AP values of 0.45 & 0.46 respectively. The largest precision values were given by YOLO-V1 & YOLO-V2 with approximate values of 0.71 each.

The results corresponding to YOLO-V4 implementation are shown in figures 17 to 23, indicating the front and rear views of vehicles in the frame. "Fig 17" below shows the rear view of a bus moving in the same lane as that of our vehicle. "Fig 18" shows the front view of the vehicles being detected on a busy street. "Fig 19" shows the rear views of vehicles being detected during the night. "Fig 20" also shows the rear views detected during the daytime. "Fig 21" shows the rear view of a car detected during the night. "Fig 22" shows the front view of cars detected during the night. "Fig 23" shows both the rear and frontal views of cars detected from a distance.

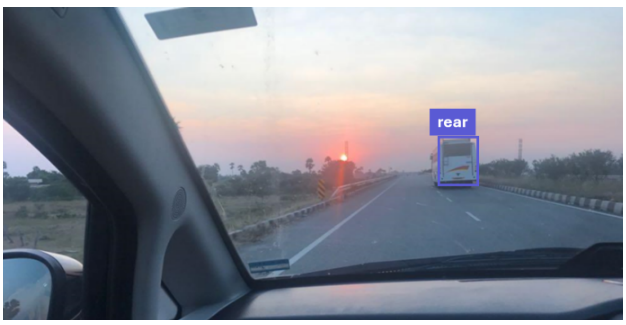

Fig. 17. Rear view detection of a bus moving in a distance

As the challenges discussed in subsection 1.1, with reference to Indian road context we observed that

- The implemented YOLO-V4 model works well even during the night to detect the front and rear views of

vehicles.
- The model also works well to detect the views of distant vehicles.
- The model is successfully able to detect the front and rear views of cars, buses, autorickshaws, and trucks.

Fig. 18. Front views of vehicles detected on a busy road

Fig. 19. Rear view of vehicles detected during the night

Fig. 20. Rear view of vehicles detected on a road

Fig. 21. Rear view of a car detected during the night

Fig. 22. Front view of cars detected during the night

Fig. 23. Rear & front views of distant vehicles detected on the road

If implemented along with an ADAS, the control can automatically detect whether our vehicle is in the correct lane or not based on the views of other vehicles on the street. This can be seen in “Fig. 18” and “Fig. 22” that our vehicle is on the wrong side of the lane. This scheme can be extended and implemented even in Indian lanes where the lane markings are faded, absent, or even sometimes unmarked. This creates a significant milestone in our work, where in the context of Indian lanes we can work ahead after detection of the view of the vehicles in consideration.

5. Conclusion and Future Scope

The 8 main versions of YOLO were studied, trained & tested with our collected dataset aiming to minimize the mean Average Precision metric. After running the codes and validating the results, we found that YOLO-V4 gave us the best AP value. Version 9 is just released and these datasets or more can be further used on it, especially for classification use cases.

Other versions of YOLO such as YOLOR, YOLOX, DAMO-YOLO, etc. can also be trained & tested with our dataset to see if any improvements can be made. In addition to that, YOLO models which are yet to be proposed also find a scope to be studied and trained with the same dataset to look for improvements in the future.

References

[1] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, “YOLOX: Exceeding YOLO Series in 2021,” 2021, *arXiv*. doi: 10.48550/ARXIV.2107.08430.

[2] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, “You Only Learn One Representation: Unified Network for Multiple Tasks,” 2021, *arXiv*. doi: 10.48550/ARXIV.2105.04206.

[3] X. Xu, Y. Jiang, W. Chen, Y. Huang, Y. Zhang, and X. Sun, "DAMO-YOLO : A Report on Real-Time Object Detection Design," 2022, *arXiv*. doi: 10.48550/ARXIV.2211.15444.

[4] X. Long *et al.*, "PP-YOLO: An Effective and Efficient Implementation of Object Detector," 2020, *arXiv*. doi: 10.48550/ARXIV.2007.12099.

[5] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," 2014, *arXiv*. doi: 10.48550/ARXIV.1405.0312.

[6] J. Arróspide, L. Salgado, and M. Nieto, "Video analysis-based vehicle detection and tracking using an MCMC sampling framework," *EURASIP J. Adv. Signal Process.*, vol. 2012, no. 1, p. 2, Dec. 2012, doi: 10.1186/1687-6180-2012-2.

[7] J. Dietrich, "The Caltech–Chile Connection," *Eng. Sci.*, vol. 64, no. 7, pp. 8–17, Apr. 2004.

[8] Y. Li, Y. Guo, J. Guo, Z. Ma, X. Kong, and Q. Liu, "Joint CRF and Locality-Consistent Dictionary Learning for Semantic Segmentation," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 875–886, Apr. 2019, doi: 10.1109/TMM.2018.2867720.

[9] M. K. Rath, P. K. Swain, and S. Banerjee, "An Optimised Deep Learning Approach of Lane Detection in Complex Indian Environment," in *2022 1st IEEE Int. Conf. Industr. Electron.: Developm. & Applicat. (ICIDeA)*, Bhubaneswar, India: IEEE, Oct. 2022, pp. 1–5. doi: 10.1109/ICIDeA53933.2022.9970079.

[10] M. Bojarski *et al.*, "End to End Learning for Self-Driving Cars," 2016, *arXiv*. doi: 10.48550/ARXIV.1604.07316.

[11] M. K. Rath, T. Swain, T. Samanta, S. Banerjee, and P. K. Swain, "Steering Wheel Angle Prediction from Dashboard Data Using CNN Architecture," in *Emerg. Technol. Data Mining Informat. Sec.*, vol. 1348, P. Dutta, A. Bhattacharya, S. Dutta, and W.-C. Lai, Eds., in Advances in Intelligent Systems and Computing, vol. 1348., Singapore: Springer Nature Singapore, 2023, pp. 393–401. doi: 10.1007/978-981-19-4676-9_33.

[12] M. K. Rath and P. Kumar Swain, "Front and Rear Classification of Vehicles in Indian Context using Deep Neural Networks," in *2023 Int. Conf. Commun., Circ., Sys. (IC3S)*, Bhubaneswar, India: IEEE, May 2023, pp. 1–4. doi: 10.1109/IC3S57698.2023.10169515.

[13] S. Chen, H. Yuan, X. Cao, and X. Li, "A Real-time Image Recognition System Based on Improved Jacintonet Convolutional Neural Network," *J. Phys.: Conf. Ser.*, vol. 1576, no. 1, Art. no. 012004, Jun. 2020, doi: 10.1088/1742-6596/1576/1/012004.

[14] D. Santos and P. L. Correia, "Car recognition based on back lights and rear view features," in *2009 10th Worksh. Image Analys. Multimedia Interact. Servic.*, London, United Kingdom: IEEE, May 2009, pp. 137–140. doi: 10.1109/WIAMIS.2009.5031451.

[15] V. S. Petrovic and T. F. Cootes, "Vehicle type recognition with match refinement," in *Proceedings of the 17th Inter. Conf. Pattern Recogn., 2004. ICPR 2004.*, Cambridge, UK: IEEE, 2004, pp. 95-98 Vol.3. doi: 10.1109/ICPR.2004.1334477.

[16] Q. B. Truong and B. R. Lee, "Vehicle Detection Algorithm Using Hypothesis Generation and Verification," in *Emerg. Intell. Comput. Techn. Applic.*, vol. 5754, D.-S. Huang, K.-H. Jo, H.-H. Lee, H.-J. Kang, and V. Bevilacqua, Eds., in Lecture Notes in Computer Science, vol. 5754., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 534–543. doi: 10.1007/978-3-642-04070-2_59.

[17] Y. Gao and H. Lee, "Local Tiled Deep Networks for Recognition of Vehicle Make and Model," *Sensors*, vol. 16, no. 2, p. 226, Feb. 2016, doi: 10.3390/s16020226.

[18] M. Mathew, K. Desappan, P. K. Swami, and S. Nagori, "Sparse, Quantized, Full Frame CNN for Low Power Embedded Devices," in *2017 IEEE Conf. Comput. Vision Patt. Recogn. Worksh. (CVPRW)*, Honolulu, HI, USA: IEEE, Jul. 2017, pp. 328–336. doi: 10.1109/CVPRW.2017.46.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conf. Comp. Vis. Patt. Recogn. (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conf. Comp. Vis. Patt. Recogn.*, Miami, FL: IEEE, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int J Comput Vis*, vol. 88, no. 2, pp. 303–338, Jun. 2010, doi: 10.1007/s11263-009-0275-4.

[23] J. Terven and D. Cordova-Esparza, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," 2023, doi: 10.48550/ARXIV.2304.00501.

[24] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conf. Comp. Vis. Patt. Recogn. (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 6517–6525. doi: 10.1109/CVPR.2017.690.

[25] S. Seong, J. Song, D. Yoon, J. Kim, and J. Choi, "Determination of Vehicle Trajectory through Optimization of Vehicle Bounding Boxes using a Convolutional Neural Network," *Sensors*, vol. 19, no. 19, Art. no. 4263, Sep. 2019, doi: 10.3390/s19194263.

[26] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018, *arXiv*. doi: 10.48550/ARXIV.1804.02767.

[27] A. Ammar, A. Koubaa, M. Ahmed, A. Saad, and B. Benjdira, "Vehicle Detection from Aerial Images Using Deep Learning: A Comparative Study," *Electronics*, vol. 10, no. 7, Art. no. 820, Mar. 2021, doi: 10.3390/electronics10070820.

[28] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020, *arXiv*. doi: 10.48550/ARXIV.2004.10934.

[29] S.-S. Park, V.-T. Tran, and D.-E. Lee, "Application of Various YOLO Models for Computer Vision-Based Real-Time Pothole Detection," *Appl. Sci.*, vol. 11, no. 23, Art. no. 11229, Nov. 2021, doi: 10.3390/app112311229.

[30] M. Horvat, L. Jelečević, and G. Gledec, "A comparative study of YOLOv5 models performance for image localization and classification." *In 33rd Central European Conf. Infor. Intellig. Sys. (CECIIS), Dubrovnik, Hrvatska,* Sep. 2022.

[31] C. Li *et al.*, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," 2022, *arXiv*. doi: 10.48550/ARXIV.2209.02976.

[32] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang, "TOOD: Task-aligned One-stage Object Detection," in *2021 IEEE/CVF Internat. Conf. Comp. Vis. (ICCV)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 3490–3499. doi: 10.1109/ICCV48922.2021.00349.

[33] H. Zhang, Y. Wang, F. Dayoub, and N. Sünderhauf, "VarifocalNet: An IoU-aware Dense Object Detector," 2020, *arXiv*. doi: 10.48550/ARXIV.2008.13367.

[34] Z. Gevorgyan, "SIoU Loss: More Powerful Learning for Bounding Box Regression," 2022, *arXiv*. doi: 10.48550/ARXIV.2205.12740.

[35] X. Ding, H. Chen, X. Zhang, K. Huang, J. Han, and G. Ding, "Re-parameterizing Your Optimizers rather than Architectures," 2022, *arXiv*. doi: 10.48550/ARXIV.2205.15242.

[36] C. Shu, Y. Liu, J. Gao, Z. Yan, and C. Shen, "Channel-wise Knowledge Distillation for Dense Prediction," 2020, *arXiv*. doi: 10.48550/ARXIV.2011.13256.

[37] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in *2023 IEEE/CVF Conf. Comp. Vis. Patt. Recogn. (CVPR)*, Vancouver, BC, Canada: IEEE, Jun. 2023, pp. 7464–7475. doi: 10.1109/CVPR52729.2023.00721.

[38] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *2017 IEEE Conf. Comp. Vis. Patt. Recogn. (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 2261–2269. doi: 10.1109/CVPR.2017.243.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conf. Comp. Vis. Patt. Recogn. (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[40] "Ultralytics YOLOv8," Ultralytics. [Online]. Available: https://docs.ultralytics.com