Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Enhancing Road Safety: Machine Learning-Based Car Orientation Detection for Code Violation Prevention

Hemza Saidi[1,2], Bassam G.N. Muthanna[3*]

1Embedded Systems Research Unit of Chlef, Research Centre for Scientific and Technical Information (CERIST), Ben Aknoun, Algiers, Algeria

2Electrical Engineering Department, Faculty of Technology, Hassiba Benbouali University of Chlef, Chlef, Algeria

3Department of Mechatronics Engineering, Faculty of Engineering and Computing, University of Science and Technology, Aden, Yemen

ARTICLE INFO

Received : 05 Mar 2025

Revised: 25 Aug 2025

Accepted : 01 Sept 2025

ABSTRACT

The rising number of lives lost and serious injuries resulting from car accidents has underscored the urgent need for driving assistance systems and code violation detection mechanisms. In response to this pressing issue, in this study, we present a novel and advanced device specifically designed to detect road code violations, with a particular focus on the ”no entry” situations. Traditional approaches relying on GPS and offline maps have exhibited limitations in accuracy and real-time performance. To overcome these challenges, we propose a cutting-edge computer vision solution that leverages state-of-the-art machine learning models for accurate car orientation detection. Our system combines high- resolution single camera and on board processing units to continuously analyse the surrounding road environment in real-time. The machine learning model employed has been extensively trained on a vast dataset, enabling it to recognize ‘no entry’ signs and vehicles violating traffic rules.

Keywords : Machine Learning, Object Detection, Object Tracking, Computer Vision, Embedded Systems.

I. INTRODUCTION

Car accidents continue to Constitute a considerable risk to human life, resulting in substantial losses and devastating consequences worldwide, according According to the World Health Organization (WHO), around 1.35 million people lose their lives annually due to road traffic accidents globally [1]. This statistic represents an alarming loss of human lives and emphasizes the urgent need for effective road safety measures and interventions. In recent years, there have been several notable advancements in car detection using machine learning techniques [2] [3], particularly with the rise of advanced deep learning techniques and convolutional neural networks (CNNs) [4] [5]. CNNs have shown remarkable Performance across different computer vision applications, including object detection [6], and have been extensively applied to car detection [7]. One popular architecture for object detection, including car detection, the group of region-based convolutional neural networks (R-CNN)" [8]. This family includes R-CNN [9], Fast R-CNN [10], and Faster R-CNN [11]. These models combine region proposal methods with CNNs to efficiently detect objects in an image. Another significant development in car detection is the introduction of single-shot Object detection models, including YOLO (You Only Look Once) [12] and SSD (Single Shot MultiBox Detector) [13], are renowned for their real-time capabilities and have found widespread use in various applications, such as vehicle detection.

AdditionallyFurthermore, The existence of extensive annotated datasets, like COCO dataset [14] and the KITTI dataset [15], The availability of large-scale annotated datasets, such as COCO, has been instrumental in training and assessing car detection models These datasets provide a diverse range of images with labeled objects, including cars, allowing researchers to efficiently develop and evaluate their algorithms algorithms effectively. Additionally, the use of data augmentation techniques, transfer learning, and ensemble methods has contributed to improving the performance of car detection models. Data augmentation It entails artificially enlarging the training dataset by applying transformations such as rotation, scaling, and flipping. Transfer learning allows models to leverage pre-trained networks on large-scale datasets and adapt them to the car detection task with smaller labeled datasets. Ensemble methods combine the predictions of multiple models to enhance detection accuracy. Overall, the development of machine learning methods, especially deep learning models, combined with large-scale datasets and advanced training methods, has significantly improved the performance of car detection in image processing in recent years. Our method addresses the novel challenge of detecting violations of the ”no entry” road sign by Utilizing convolutional neural networks (CNNs) for car detection and estimating orientation. By training a CNN model on specialized dataset we created that includes examples of cars orientations, we can effectively detect instances where vehicles violate the restriction. The CNN model learns to extract meaningful extracting features from input images and accurately estimating their orientations. This approach offers a ground-breaking solution to the detection of ”no entry” violations, contributing to improved road safety and effective enforcement of traffic regulations in previously unaddressed scenarios.

II.RELATEDWORK

In this section, we provide an overview of the existing studies that have explored car detection using Machine leering and other methods. Several studies have contributed to advancing the field and developing effective methodologies for accurately estimating the orientation of cars. A notable approach called OERFF (Orientation Estimation and Regional Feature Fusion) was proposed by BIN ZHENG et *al.* [16]. The OERFF method presents an architecture Consisting of three primary modules: the orientation estimation module, the regional feature extraction module, and the feature fusion module The orientation estimation module operates in two steps. Initially, thirty vehicle key points are identified using the VGG network. Following this, an orientation estimation model is applied utilizes The coordinates and relative relationships of these key points are used to predict the vehicle's orientation in the image To capture local subtle features and improve the model's classification performance and refinement, the regional feature extraction module is developed. This module constructs and trains a dedicated model for extracting features in specific regions, thereby highlighting the differences in local characteristics. Furthermore, the orientation difference- based feature fusion The module computes the similarity distance and orientation difference between the query and gallery vehicles. To address the challenge posed by significant orientation variations, the module adopts a a weighted distance proportional to the azimuthal difference, adjusting the similarity based on the real orientation difference. Consequently, the OERFF model achieves improved the overall classification performance and identification accuracy notable study by Yuan Zhang et *al.* [17] proposed an improvement of the YOLOv7 algorithm, Res3Unit have been Employed to rebuild the backbone network to enhance its capability to obtain more non-linear feature. ACmix is incorporated after the SPPCSPC layer of the backbone network to strengthen the network's focus on vehicles while minimizing interference from other targets. Additionally, the RFLA (Gaussian-receptive-field-based label assignment) module is applied between the feature fusion area and the detection head to enhance the receptive field of the network model for detecting small objects in the image.

A significant work by Furong Shi et *al.* [18] The authors introduce a one-stage, anchor-free detection method tailored for detecting vehicles with arbitrary orientations in high-resolution aerial images. This method converts the vehicle detection task into a multi-task learning problem by directly predicting high-level vehicle features through a fully convolutional network .The proposed method involves creating a A classification subtask to identify vehicle central points, along with three regression subtasks to predict vehicle orientations, scales, and offsets of the central points. To incorporate multi-scale features, Coarse and fine feature maps from various stages of a residual network are combined using a feature pyramid fusion strategy. Subsequently, four parallel Convolutional layers are added to predict high-level vehicle features. During the training process, The authors leverage task uncertainty, learned from the training data, to assign weights to the loss function within the multitask learning framework. This approach allows the model to effectively handle the inherent uncertainties in vehicle detection. During inference, the predicted vehicle features are used to generate oriented bounding boxes, which capture the orientation of the detected vehicles. The authors employ oriented Non-maximum suppression (NMS) is applied as a post-processing step to remove redundant detection results. The proposed method is evaluated on two publicly available aerial image datasets, with the experimental results showcasing its effectiveness. The approach successfully addresses the limitations of existing methods by accurately detecting vehicles with arbitrary orientations in high-resolution aerial images. Overall, this work presents a novel and efficient vehicle detection method that leverages deep learning techniques and multitask learning for oriented vehicle detection in aerial images, showcasing its effectiveness through comprehensive experiments on real-world datasets.

Paul E. Rybski et *al.* [19], propose training a set of classifiers based on the Histogram of Oriented Gradients (HOG) method. These classifiers are designed to identify various vehicle orientations in the imagery. Through experimentation, the authors report achieving a high classification accuracy of 88% on a test dataset consisting of 284 images. Moreover, the study explores The study explores how combinations of orientation-specific classifiers can be used to differentiate subsets of orientations, such as distinguishing between the driver's side and passenger's side views. This analysis contributes to improving the system's ability to recognize fine-grained orientations. Interestingly, the paper also compares the performance of a vehicle detector built from orientation-specific classifiers with that of an orientation-independent classifier. Surprisingly, the results show that the orientation-independent classifier outperforms the set of orientation-specific classifiers, challenging the initial expectations.

Nozomu Araki et *al.* [20], The paper presents a vehicle orientation measurement system using a single camera. It introduces a method that applies a single-camera calibration technique with a planar pattern to measure the orientation from the camera to a planar object. By using a planar object, like a standard-sized license plate, the orientation from the camera to the object can be determined. The proposed method depends on extracting feature coordinates from more than four points on the image of the planar object. This allows for accurate measurement of the vehicle’s orientation. To track the corner points of the planar object, specifically the license plate, an active contour tracking method is employed. This tracking technique enables the system to continuously monitor and update the position of the corner points throughout the vehicle’s movement. The effectiveness of the proposed method is validated through experiments conducted with real vehicle images. These experiments aim to verify the system's accuracy and reliability in measuring vehicle orientation.

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/
Research Article

III. PROPOSED SOLUTION

In this section, we present our proposed solution for measuring the orientation of vehicles using a single camera. Our approach involves the utilization CNN model to detect the vehicles and recognize their orientation, the figure 1 Illustrate the steps of the proposed solution.

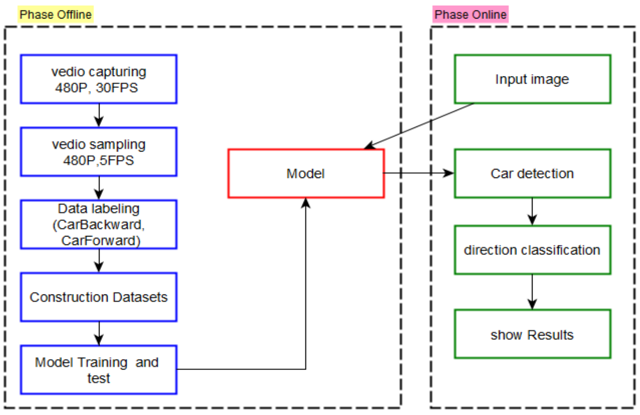

Fig. 1. Proposed Solution Steps.

We proposed an algorithm for vehicle detection and orientation classification, distinguishing between forward and backward directions. Our algorithm is divided into two main phases:

Offline Phase: In this phase, we collected images from various locations and scenarios involving different vehicles. We then labeled these images based on the vehicle orientation (forward/backward). After labeling, we trained our model using this annotated dataset. After the model was trained, it was prepared for real-time testing.

Online Phase: In this phase, a camera captures frames in real-time and sends them to our model. The model first detects the vehicles within the captured frame. The input image may contain one or more vehicles or none at all. After detecting the vehicles, the model classifies the direction of each detected vehicle as either forward or backward. Finally, it displays the results, showing the detected vehicles and their respective orientations.

This two-phase approach ensures that our model is well-trained on a diverse set of scenarios and can accurately perform vehicle detection and orientation classification in real-time applications.

IV. DATA

The purpose of this section is to provide detailed information about the set of data used to develop the algorithm for detecting cars. Additionally, it provides an explanation of how the data was collected, pre-treated, and analysed, which are crucial to evaluating the model's effectiveness and reliability.

4.1 Description

- **Annotation Step**: The relevant annotations are stored in a CSV file, which includes the bounding box coordinates for each car in each image. These annotations provide the bounding box coordinates along with the label orientation for each car in each image. The CSV file contains six columns: frame (image), X, Y, width, height, and rectangle_labels

• Size of the dataset: 1393 images.

The bounding box coordinates (x, y, width, heigh) define the corners of the box that encloses each car in the image, from top-left to bottom-right. These coordinates were used to train and evaluate the object detection model (Table 1).

TABLE I. Sample Annotation

Image	X	Y	Width	Heigh	Rectangle_lables
frame_0013	32.98	65.46	23.75	20.88	forward car
frame_0014	38.7	53.81	14.75	29.91	forward car
frame_0015	62.95	58.43	31.17	31.72	backward car
frame_0016	7.22	56.82	14.75	25.1	backward car
frame_0017	30.42	66.26	10.09	11.04	forward car

4.2 Dataset collection

To train our model, we conducted data collection in the city by capturing video sequences, using 480P (640x480) resolution camera with 30FPS (Frame Per Second) the 3 shows the used car and camera setup, these videos were then converted into individual images, sampling 5 frames per second from the original 30 frames per second to reduce computational requirements (Figure 2).

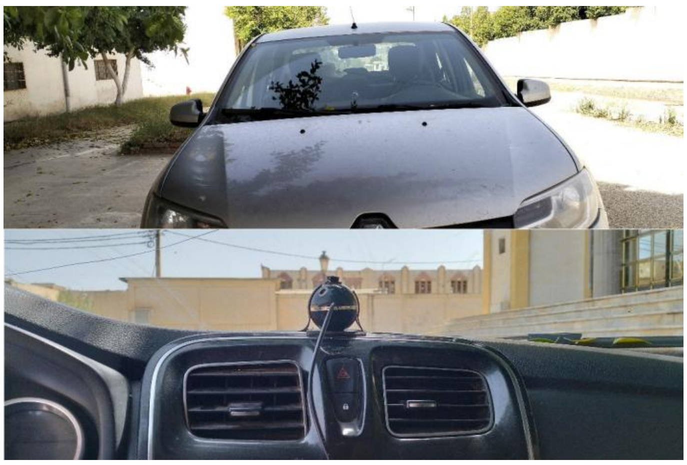

Fig. 2. The used Car and Camera setup

4.3 Dataset variation

This dataset features a diverse collection of images, showcasing cars of different makes, sizes models and colors. It also encompasses a wide range of environments, including urban areas, parking lots and highways, captured under various lighting conditions and from multiple camera angles.

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

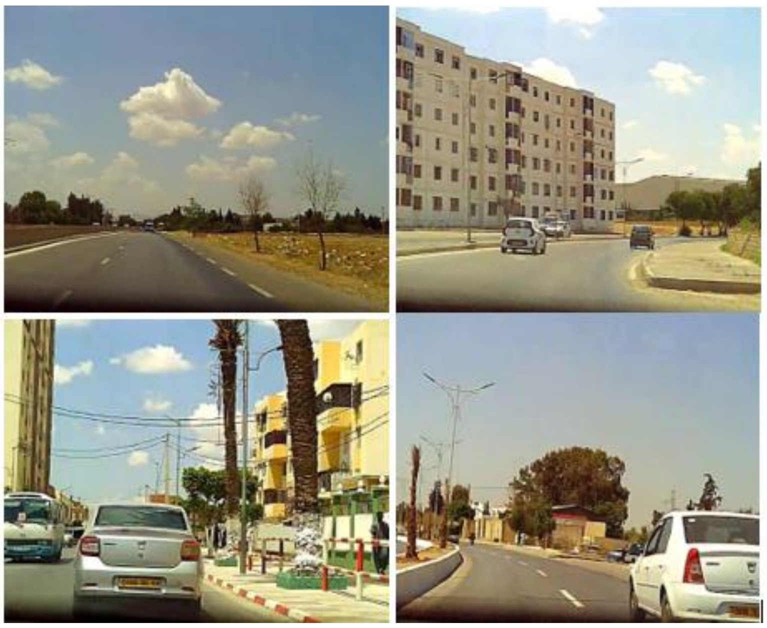

Fig. 3. Dataset Sample

The dataset contains a varying number of car instances per image, with some images featuring multiple cars and others showing only a single car, as illustrated in Figure 3. This diversity is essential for training a robust car detection model that can adapt to different scenarios.

4.4 Data labeling

To recognize vehicle orientation, we first created labels of each vehicle position, for the vehicles heading with the flow of traffic we gave Car Forward label to indicate the cars going in the correct direction without any code violation and for the vehicles heading in the opposite direction of the traffic flow we gave Car Backward label to indicate the for cars traveling in the opposite direction, indicating a violation of the code. Labeling the collected images with the created labels, we utilized Label-Studio free open-source software; the figure 4 shows the created labels in the lable-studio software.

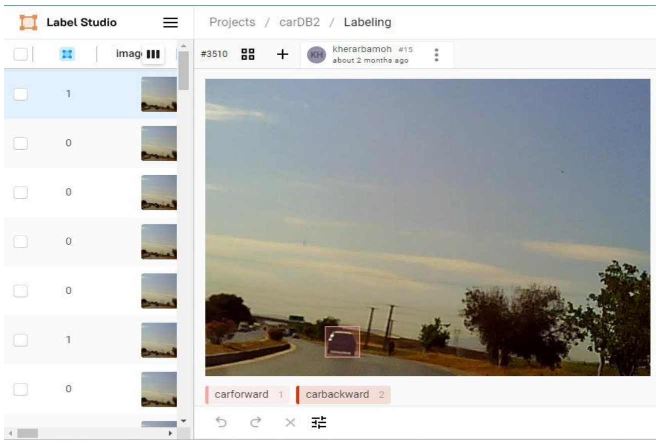

Fig. 4. The lable-studio software

V. ORIENTATION

In this section, we discuss the categorization of vehicle orientations into two primary groups as show in figure 5: backward and forward. This classification is critical for the development of our car orientation detection model. The "backward" group includes orientations where the vehicle is primarily facing away from the camera, encompassing angles that represent the rear or side-rear views of the car. On the other hand, the "forward" group consists of orientations where the vehicle is predominantly facing towards the camera, including front or side-front views. By dividing the dataset into these two groups, we can train a more robust model that accurately detects the orientation of cars. This approach enhances the model's overall performance by considering the various possible poses of vehicles in different road scenarios.

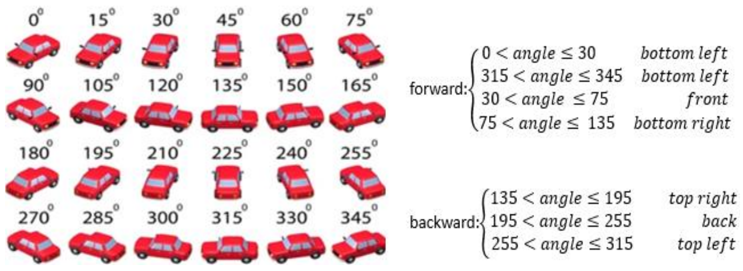

Fig. 5. Car orientation

VI. MODEL CREATION AND TRAINING

5.1 Data Split

To assess the performance and generalization of the proposed model, the dataset presented in Table 2 was split into two subsets: a training set and a testing set. The data was divided in a 2:1 ratio, with roughly two-thirds of the dataset used for training and the remaining one-third reserved for testing. The splitting process was carried out randomly, ensuring that the distribution of samples across different classes was maintained in both the training and testing sets. This randomization reduces the bias and ensures a representative distribution of data during model training and evaluation. By using a data split of 2/3 for training and 1/3 for testing, we aimed to strike a balance between providing sufficient data for the model to learn a evaluate its performance on unseen samples. This partitioning allows for robust model training while enabling an unbiased assessment of its generalization capabilities. It is important to note that throughout the training process, The testing set was kept entirely separate and was not utilized for model training or hyperparameter tuning. This separation guarantees an unbiased evaluation of the model's performance on unseen data, providing a reliable estimate of its effectiveness in real-world scenarios. The data split methodology adopted in this study allows us to Evaluate the model's ability to generalize and make accurate predictions on new, unseen instances, enhancing the overall reliability and validity of the obtained results.

TABLE II. Dataset sample

Object_id	Image_path	label
4-3	cars/4-3.jpg	1
4-4	cars/4-4.jpg	1
5-1	cars/5-1.jpg	1
5-2	cars/5-2.jpg	1
5-3	cars/5-3.jpg	1
5-4	cars/5-4.jpg	1
5-5	cars/5-5.jpg	2
6-1	cars/6-1.jpg	1
6-2	cars/6-2.jpg	1
6-3	cars/6-3.jpg	1
6-4	cars/6-4.jpg	2
6-5	cars/6-5.jpg	1
7-1	cars/7-1.jpg	1
7-2	cars/7-2.jpg	1
7-3	cars/7-3.jpg	1
7-4	cars/7-4.jpg	2
7-5	cars/7-5.jpg	2
7-6	cars/7-6.jpg	1
8-1	cars/8-1.jpg	1
8-2	cars/8-2.jpg	1
8-3	cars/8-3.jpg	1
8-4	cars/8-4.jpg	2
8-5	cars/8-5.jpg	2
9-1	cars/9-1.jpg	1
9-2	cars/9-2.jpg	1
9-3	cars/9-3.jpg	1
9-4	cars/9-4.jpg	2
9-5	cars/9-5.jpg	2
9-6	cars/9-6.jpg	2

5.2 CNN model creation

- Initialize a sequential model object using model = keras.models.Sequential([]). This object allows us to build the model by stacking layers sequentially.

- Add a convolutional layer using keras.layers.Conv2D(...). This layer performs convolutional operations on the input data, extracting features using 16 filters and a kernel size of (3, 3). The ReLU activation function is applied to introduce non-linearity.

- Add another convolutional layer using keras.layers.Conv2D(...). This layer performs additional convolutional operations with 32 filters and the same (3, 3) kernel size and ReLU activation.

- Add a max pooling layer using keras.layers.MaxPool2D(...). This layer down- samples the input data by taking the maximum value within each 2x2 window, reducing spatial dimensions.

- Add a batch normalization layer using keras.layers.BatchNormalization(...). This layer normalizes the activations of the previous layer, improving model performance and reducing over fitting. The normalization is performed along the last axis (-1).

- Repeat steps 2-5 to add more convolutional, pooling, and batch normalization layers. The second set of convolutional layers has 64 filters and 128 filters, respectively.

- Add a flatten layer using keras.layers.Flatten(). This layer reshapes the multi- dimensional output from the previous layers Flattened into a 1D vector, ready for the subsequent fully connected layers.

- Add a dense layer using keras.layers.Dense(...). This layer is a fully connected layer with 5,1 and 2 units, applying the ReLU activation function to introduce non-linearity.

- Add a batch normalization layer using keras.layers.BatchNormalization(). This layer normalizes the activations of the previous dense layer.

- Add a dropout layer using keras.layers.Dropout(rate=0.5). This layer randomly drops 50% of the inputs during training to prevent over fitting.

- Add the final dense layer using keras.layers.Dense(...). This layer has 3 units, representing the number of output classes, and applies the softmax activation function for multi-classclassification.

5.3 Learning configuration

Learning Rate and Epochs:

- learningrate = 0.001: Defines the learning rate for the optimizer, which controls the step size at which the optimizer updates the model's weights during training.

- epochs = 30: defines the number of epochs, which represents the number of complete passes the model makes over the training data during training.

Optimizer Configuration:

- opt = Adam(learningrate=0.001, decay=lr / (epochs * 0.5)): This code initializes the Adam optimizer with the given learning rate and decay. The Adam optimizer is commonly used for gradient-based optimization and adapts the learning rate throughout the training process.

- lr / (epochs * 0.5)calculates the decay rate, which gradually decreases the learning rate over time to help the model converge more effectively.

Model Compilation:

- model.compile(...): Compiles the model by configuring the loss function, optimizer, and metrics.

- loss = ’categorical crossentropy’ Defines the loss function as categorical cross-entropy, which is commonly used for multi-class classification tasks.

- metrics =[’accuracy’]Indicates that the model's performance metric during training and evaluation will be accuracy, which measures the ratio of correctly classified samples.

By setting the learning rate, configuring the optimizer, and compiling the model, we prepare the neural network model for training. The chosen optimizer, learning rate, loss function, and metric are essential components that impact the model’s training process and subsequent performance evaluation.

5.4 Model Training

Data Augmentation:

- aug.flow(...): This line applies data augmentation to the training data by creating augmented batches using a specified batch size. Data augmentation techniques, including rotation, scaling, and flipping, introduce variations in the training samples, thereby expanding the dataset and enhancing the model's ability to generalize.

Training Loop:

- model.fit(...): This line initiates the training process of the model using the augmented data.

- batchsize = 32specifies the number of samples in each batch used for updating the model’s weights.

- epochs indicate the number of complete passes the model will make over the training data during training.

- validationdata = (Xval, yval)refers to the validation dataset, which is used to evaluate the model’s performance after each epoch. X val and y val represent the validation data inputs and labels, respectively.

During the training process, the model iteratively updates its weights based on the computed loss and gradients from the training data batches. The model's performance on the validation dataset is evaluated at the end of each epoch to track its progress and prevent overfitting. The goal of the training process is to optimize the model's parameters by minimizing the loss function and improving its ability to make accurate predictions. The 'history' object can be used to access the training and validation metrics recorded throughout the training process, including loss and accuracy values for each epoch.

VI. RESULTS AND DISCUSSION

In this section, we present the results obtained from training and evaluating the proposed model for car orientation detection figure 6. The performance metrics, including loss and accuracy, These are analyzed to evaluate the model's effectiveness in accurately classifying car orientations.

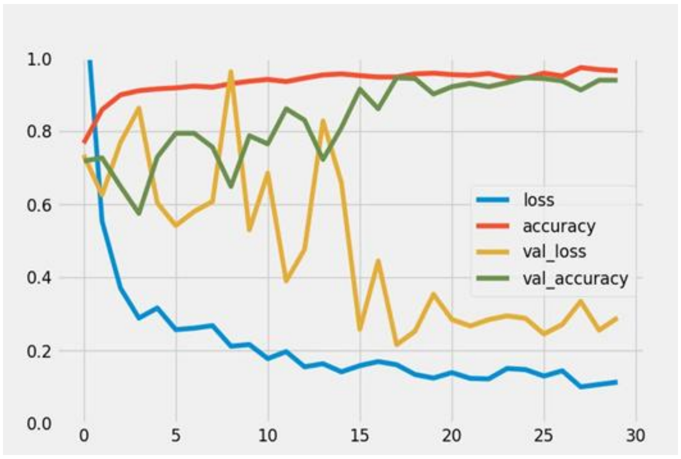

Fig.6. The evaluation of the model

6.1 Training Results

The model was trained on the training dataset over 30 epochs. During this process, the model continually updated its weights to minimize the loss function. Once training was complete, the final results were obtained.

Training Loss: The training loss achieved a value of 0.1131. This represents the average difference between the predicted and actual car orientations during training. A lower value signifies a better fit of the model to the training data.

Training Accuracy: The training accuracy reached a value of 0.9655. This indicates the percentage of car orientations that were correctly classified in the training dataset. A higher accuracy indicates the model's ability to correctly predict the car orientations.

6. 2 Validation results

The model's performance was evaluated using a separate validation dataset. The validation results provide insights into how well the model generalizes to unseen data (Figure 7).

Validation Loss: The validation loss obtained a value of 0.2891. This measures the average difference between the predicted and actual car orientations in the validation dataset. It is a crucial metric for assessing the model's ability to generalize and make accurate predictions on unseen data. A lower validation loss indicates stronger generalization performance.

Validation Accuracy: The validation accuracy achieved a value of 0.9396. This represents the proportion of correctly classified car orientations in the validation dataset. It offers an estimate of the model's accuracy on unseen data. Higher validation accuracy reflects better generalization performance.

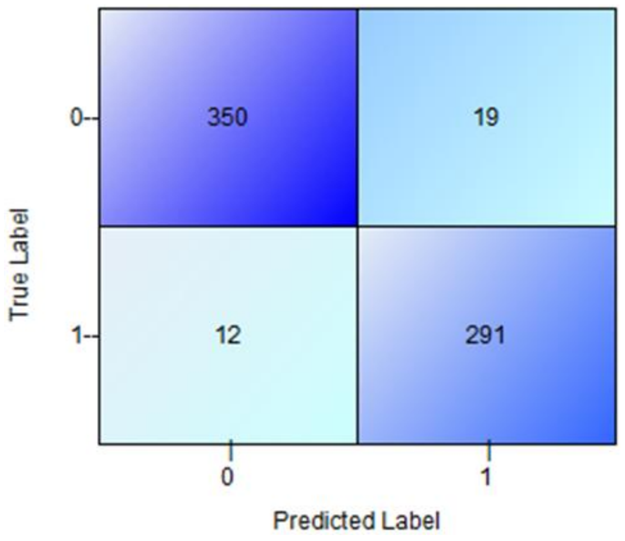

Fig.7. R-CNN Confusion Matrix

The confusion matrix reveals that the model has correctly classified 350 'backward' images and 291 'forward' images. It has made 22 false positives (predicting 'backward' when it was actually 'forward') and 11 false negatives (predicting 'forward' when it was actually 'backward'). These results indicate that the model is performing well in distinguishing between 'backward' and 'forward' vehicles.

The Precision-Recall curve is positioned near the top-right corner of the graph, indicating that the model maintains high precision and recall across various threshold values. This suggests that the model performs effectively on the dataset (Figure 8).

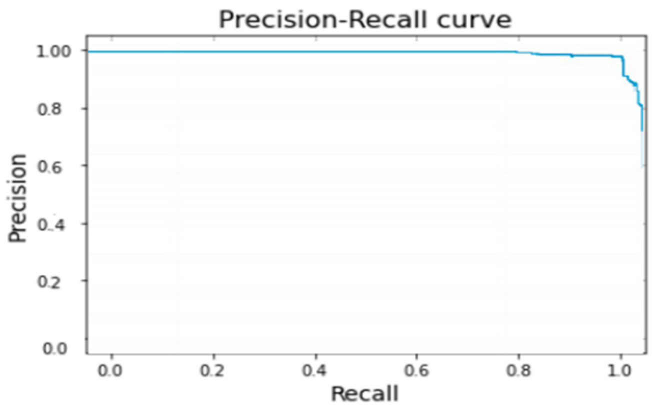

Fig. 8. Precision-Recall Curve

6.3 Discussion

The results demonstrate the effectiveness of the proposed model for car orientation detection. The high training accuracy of 0.9655 indicates that the model has successfully learned the patterns and features associated with car orientations. different car orientations in the training dataset. Similarly, the validation accuracy of 0.9396 suggests that the model can generalize well to unseen data and make accurate predictions on car orientations.

The relatively low training and validation losses of 0.1131 and 0.2891, respectively, indicate that the model achieved a good fit to the training data and minimized the discrepancies between predicted and actual car orientations. This indicates that the model has effectively learned the relevant features and patterns required for accurate classification.

However, additional evaluation on larger and more diverse datasets, along with comparisons to state-of-the-art approaches, is needed to thoroughly assess the model's performance and generalization abilities. In conclusion, the results from the trained model highlight its effectiveness and potential for accurate car orientation detection. The high accuracy and low loss values demonstrate its ability to classify car orientations with both accuracy and reliability. These findings contribute to the advancement of car detection systems and lay the groundwork for enhanced road safety measures and traffic management.

VII. CONCLUSION

In this study, we introduced a machine learning-based approach for car orientation detection, with the goal of reducing car accidents and improving road safety. By utilizing computer vision techniques and a Convolutional Neural Network (CNN) model, we achieved promising results in accurately classifying car orientations.

The experimental results confirmed the effectiveness of our model, achieving an accuracy of 96.55% on the training data and 93.96% on the validation data. The low loss values (0.1131 for training and 0.2891 for validation) further validate the model's capability to accurately classify car orientations.

Our approach offers several advantages. Firstly, it provides a scalable solution that can detect car orientations without relying on GPS devices, making it applicable to a wider range of vehicles. Additionally, it eliminates the need for complex infrastructure or expensive sensors, making it cost-effective and easily deployable. Furthermore, the real-time processing capability of our model, ensures timely detection and response to code violations.

However, it is important to acknowledge the limitations of our study. The evaluation was performed on a specific dataset, and further validation on larger and more diverse datasets is needed to assess the model's robustness and generalization capabilities. Additionally, the model's performance may be affected by environmental factors, such as changes in lighting conditions or weather patterns, which should be addressed in future research.

In conclusion, our machine learning-based approach for car orientation detection shows promising results in accurately classifying car orientations, With the potential to make a significant impact on reducing car accidents and improving road safety, further research and development in this area can enhance performance, scalability, and applicability of our proposed solution, ultimately creating safer and more efficient road environments for all stakeholders.

REFERENCES

[1] S. K. Ahmed, M. G. Mohammed, S. O. Abdulqadir, R. G. A. El-Kader, N. A. El-Shall, D. Chandran, M. E. U. Rehman, and K. Dhama, "Road traffic accidental injuries and deaths: A neglected global health issue," Health science reports, vol. 6, no. 5, p. e1240, 2023.

[2] Y. Zhang, Y. Sun, Z. Wang, and Y. Jiang, "Yolov7-rar for urban vehicle detection," Sen- sors, vol. 23, no. 4, p. 1801, 2023.

[3] P. Mahto, P. Garg, P. Seth, and J. Panda, "Refining yolov4 for vehicle detection," Interna- tional Journal of Advanced Research in Engineering and Technology (IJARET), vol. 11, no. 5, 2020.

[4] E. Arkin, N. Yadikar, X. Xu, A. Aysa, and K. Ubul, "A survey: Object detection methods from cnn to transformer," Multimedia Tools and Applications, vol. 82, no. 14, pp. 21353– 21383, 2023.

[5] P. Song, P. Li, L. Dai, T. Wang, and Z. Chen, "Boosting r-cnn: Reweighting r-cnn samples by rpn's error for underwater object detection," Neurocomputing, vol. 530, pp. 150–164, 2023.

[6] Y. Liu, P. Sun, N. Wergeles, and Y. Shang, "A survey and performance evaluation of deep learning methods for small object detection," Expert Systems with Applications, vol. 172, p. 114602, 2021.

[7] M. A. Berwo, A. Khan, Y. Fang, H. Fahim, S. Javaid, J. Mahmood, Z. U. Abideen, and S. MS, "Deep learning techniques for vehicle detection and classification from im- ages/videos: A survey," Sensors, vol. 23, no. 10, p. 4832, 2023.

[8] M. Maity, S. Banerjee, and S. S. Chaudhuri, "Faster r-cnn and yolo based vehicle detec- tion: A survey," in 2021 5th international conference on computing methodologies and communication (ICCMC), pp. 1442–1447, IEEE, 2021.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587, 2014.

[10] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on com- puter vision, pp. 1440–1448, 2015.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detec- tion with region proposal networks," Advances in neural information processing systems, vol. 28, 2015.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real- time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788, 2016.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Sin- gle shot multibox detector," in Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, pp. 21–37, Springer, 2016.

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dolla´r, and C. L. Zitnick, "Microsoft coco: Common objects in context," in Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, pp. 740–755, Springer, 2014.

[15] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in 2012 IEEE conference on computer vision and pattern recognition, pp. 3354–3361, IEEE, 2012.

[16] B. Zheng, Z. Lei, C. Tang, J. Wang, Z. Liao, Z. Yu, and Y. Xie, "Oerff: A vehicle re- identification method based on orientation estimation and regional feature fusion," IEEE Access, vol. 9, pp. 66661–66674, 2021.

[17] Y. Zhang, Y. Sun, Z. Wang, and Y. Jiang, "Yolov7-rar for urban vehicle detection," Sen- sors, vol. 23, no. 4, p. 1801, 2023.

[18] F. Shi, T. Zhang, and T. Zhang, "Orientation-aware vehicle detection in aerial images via an anchor-free object detection approach," IEEE Transactions on Geoscience and Remote Sensing, vol. 59, no. 6, pp. 5221–5233, 2020.

[19] F. Shi, T. Zhang, and T. Zhang, "Orientation-aware vehicle detection in aerial images via an anchor-free object detection approach," IEEE Transactions on Geoscience and Remote Sensing, vol. 59, no. 6, pp. 5221–5233, 2020.

[20] N. Araki, T. Sato, Y. Konishi, and H. Ishigaki, "Vehicle's orientation measurement method by single-camera image using known-shaped planar object," in 2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC), pp. 193–196, IEEE, 2009.