International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

MCS-YOLO A Multiscale Object Detection Method for Autonomous Driving Road Environment Recognition

P.SIVA PRASAD, Assistant Professor, Dept of MCA, Chirala Engineering College, Chirala, Lakshmiprasad8216@gmail.com

SHAIK SUBHAN ASHARAF ALI, PG student -MCA Dept of MCA, Chirala Engineering College, Chirala, skasharafali76@gmail.com

Abstract: This research addresses critical challenges in autonomous driving technology, focusing on the improvement of object detection algorithms' accuracy and speed. Introducing the MCS-YOLO algorithm, our approach incorporates a coordinate attention module into the backbone, enhancing the aggregation of spatial coordinate and cross-channel information in feature maps. Additionally, a multiscale small object detection structure is designed to heighten sensitivity to dense small objects, complemented by the integration of the Swin Transformer structure for CNNs to prioritize contextual spatial information. Through extensive evaluation on the BDD100K autonomous driving dataset, the MCS-YOLO algorithm outperforms the YOLOv5s counterpart in mean average precision and recall rates. Remarkably, our algorithm achieves a real-time detection speed of 55 frames per second in actual driving scenarios. Further exploration with YoloV5x6 demonstrates promising results, showcasing a potential improvement in mean average precision to 0.798%. This research offers a robust and efficient solution for advancing object detection capabilities in autonomous driving, contributing to the continual evolution of intelligent transportation systems.

Index terms *-Coordinate attention mechanisms, autonomous driving, road environmental object detection, swin transformer, YOLOv5.*

1. INTRODUCTION

In the 21st century, the escalating prevalence of automobiles as a fundamental mode of transportation globally has led to a surge in new vehicle registrations and licensed drivers. However, this rapid increase in motor vehicles has brought about challenges such as traffic accidents, congestion, and environmental concerns. Addressing these issues, autonomous driving technology emerges as a pivotal solution, contributing significantly to safety enhancement and informed decision-making in route planning during vehicular travel [1], [2].

The cornerstone of autonomous driving lies in the environmental perception system, tasked with precise and rapid identification of objects within the road environment. This identified information is crucial for informing decision systems to optimize route planning [3]. Early in the development of autonomous driving, expensive single or multi-sensor fusion methods were employed, requiring manual adjustment of vehicle parameters and extensive human involvement. However, with advancements in deep learning, sensing, and hardware technologies, computer vision (CV) and natural language processing (NLP) have flourished, offering more efficient solutions.

Girshick et al. introduced the Regions with Convolutional Neural Networks Features (R-CNN) model, enhancing recognition efficiency [5]. Subsequent innovations such as Spatial Pyramid Pooling (SPP) by He et al. [6], Fast R-CNN [7], and Faster R-CNN [8] with a Region Proposal Network (RPN) have significantly improved detection accuracy and computational efficiency. Mask R-CNN [9] further extends capabilities by enabling high-quality detection and segmentation tasks. These advancements underscore the transformative potential of deep learning-based object detection algorithms in realizing real-time, accurate, and efficient environmental sensing for autonomous vehicles.

The You Only Look Once (YOLO) series of algorithms [10], [11], [12], [13], [14], [15], [16] and the Single Shot MultiBox Detector (SSD) series of algorithms [17], [18], [19] adopt regression methods for object classification and bounding box prediction. The YOLO algorithm takes the entire image as input and regresses the position and class of the bounding box directly in the output layer. The YOLO and SSD algorithms are widely used in industry for their faster real-time detection than the R-CNN algorithm. Liu et al. used the Transformer as the backbone of a convolutional neural network for dense vision tasks.

The success of the Swin Transformer [20], [21] demonstrates the powerful potential of the transformer for classification, detection and segmentation tasks. ConvNext [22] uses the same optimisation strategy as Swin Transformer to train the convolutional neural network. With the same FLOPs, ConvNext has faster inference and higher accuracy than Swin Transformer. Chen et al. [23] proposed a DW-YOLO algorithm that improves vehicle object detection performance by increasing the depth and width of the network. Zhou et al. [24] proposed a lightweight MobileYOLO algorithm that reduces the number of parameters and improves detection speed. Wang et al. [25] applied MobileNet to a YOLOv4 network for driving scenarios and achieved a detection speed of 35 FPS. Tian et al. [26] proposed a SA-YOLOv3 detector that strikes a better compromise between detection speed and accuracy. Gupta et al. [27] applied both detection and segmentation to the task of road environment object detection to enhance the intelligent adaptive behaviour of self-driving cars. Wang et al. [28] propose an autonomous driving detection network for foggy weather that improves the accuracy of object detection in foggy weather scenarios as well as the speed of detection. Li et al. [29] designed a Res-YOLO network model that significantly reduced the missed-detection rate and improved the detection accuracy of vehicle object detection.

2. LITERATURE SURVEY

In the pursuit of achieving autonomous operation in urban environments with unpredictable traffic, this paper presents a comprehensive overview of recent research conducted by the authors towards enabling safe and robust autonomous driving. Focusing on the development of a fully functional vehicle platform, the study integrates various real-time systems, including environment perception, localization, planning, and control. [1] The research builds upon the authors' previous work on Junior, Stanford's entry in the 2007 DARPA Urban Challenge, extending its capabilities to address more realistic scenarios encountered in real-world driving conditions. The authors introduce a trio of unsupervised algorithms facilitating the automatic calibration of a 64-beam rotating LIDAR, achieving superior accuracy compared to manual measurements. High-resolution maps of the environment are then generated for online localization with centimeter-level precision. Enhanced perception and recognition algorithms enable the tracking and classification of obstacles, distinguishing between cyclists, pedestrians, and vehicles, and detecting traffic lights [6,29,39]. A novel planning system dynamically generates thousands of candidate trajectories per second, optimizing the vehicle's path based on incoming data. The improved controller ensures optimal throttle, brake, and steering actuations, maximizing comfort and minimizing trajectory error. Demonstrating versatility, these algorithms operate seamlessly in various weather conditions, day or night. The integration of these systems has enabled Junior to successfully log hundreds of miles of autonomous operation across diverse real-life conditions.

The burgeoning advancements in AI, computer vision, machine learning, and autonomous vehicles were addressed in this paper [2]. Recognizing the challenge of staying current in this rapidly evolving field, the authors aim to bridge the gap for both seasoned researchers and beginners by offering a comprehensive survey. Unlike previous works focusing on specific sub-problems, this book presents a holistic overview of problems, datasets, and methods in computer vision for autonomous vehicles. The survey encompasses a thorough examination of historically significant literature and the latest advancements in various domains crucial for autonomous driving, including recognition, reconstruction, motion estimation, tracking, scene understanding, and end-to-end learning. Notably, the authors assess the state of the art using benchmarking datasets such as KITTI, MOT, and Cityscapes, providing valuable insights into algorithmic performance. The inclusion of open problems and ongoing research challenges enhances the survey's relevance to the current landscape of autonomous vehicle technology [2,4,27]. To enhance accessibility and address missing references, the authors provide a dedicated website for seamless navigation through topics and methods, offering additional context and information. This comprehensive survey serves as a valuable resource for researchers, practitioners, and newcomers seeking a nuanced understanding of the dynamic landscape of computer vision in the context of autonomous vehicles.

The imminent release of autonomous vehicles into the market and the importance of ensuring not only safety and reliability but also a comfortable user experience for widespread acceptance. Acknowledging the subjective nature of user comfort in driving styles, ranging from sporty to relaxed preferences, the authors propose a learning from demonstration approach to personalize autonomous vehicle behavior. [4] In this approach, users can manually drive the vehicle to demonstrate their desired driving style, avoiding the tedious and error-prone task of manually tuning numerous parameters like acceleration profiles, distances to other cars, and speed during lane changes. The individual driving style is modeled through a cost function, and feature-based inverse reinforcement learning is employed to identify the model parameters that best match the observed style. Once learned, the model efficiently computes trajectories for the vehicle in autonomous mode, allowing it to replicate and adapt to diverse driving styles. The efficacy of this approach is demonstrated by its capability to learn cost functions and reproduce various driving styles using real driver data. This user-centric approach not only enhances the autonomous vehicle's adaptability to individual preferences but also contributes to a more seamless integration of autonomous technology into diverse user experiences.

the stagnation in object detection performance on the PASCAL VOC dataset and introduces a novel, simple, and scalable detection algorithm that significantly enhances mean average precision (mAP). The proposed algorithm [5] achieves a remarkable mAP of 53.3%, surpassing the previous best result by more than 30%. The approach combines two key insights: the utilization of high-capacity Convolutional Neural Networks (CNNs) to process bottom-up region proposals, enabling precise object localization and segmentation, and the effectiveness of supervised pre-training for an auxiliary task followed by domain-specific fine-tuning, particularly in scenarios with limited labeled training data. Termed R-CNN (Regions with CNN features), the method exploits these insights to yield substantial performance improvements. Comparing R-CNN to OverFeat, a sliding-window detector based on a similar CNN architecture, the authors find that R-CNN outperforms OverFeat by a significant margin on the ILSVRC2013 detection dataset, which consists of 200 classes [5,7,8,17,18]. The success of R-CNN highlights its efficacy in overcoming the limitations of prior methods, providing a promising advancement in object detection, and showcasing the potential of combining region proposals with CNNs for improved performance.

A limitation in existing deep convolutional neural networks (CNNs) that require fixed-size input images, leading to a loss in recognition accuracy for images or sub-images of varying sizes. The proposed solution introduces a novel pooling strategy called "spatial pyramid pooling" in the form of a new network structure named SPP-net [6]. This architecture enables the generation of a fixed-length representation regardless of the input image's size or scale, making it more versatile. SPP-net exhibits robustness to object deformations, enhancing CNN-based image classification methods. Demonstrating its efficacy on the ImageNet 2012 dataset, the paper illustrates that SPP-net improves the accuracy of various CNN architectures. Furthermore, on the Pascal VOC 2007 and Caltech101 datasets, SPP-net achieves state-of-the-art classification results with a single full-image representation and no fine-tuning. The power of SPP-net is also evident in object detection, where it significantly accelerates the computation of feature maps and achieves competitive or superior accuracy compared to the R-CNN method [39]. In the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014, the proposed methods using SPP-net rank impressively, securing the #2 position in object detection and #3 in image classification among 38 participating teams. The manuscript introduces key improvements made for this competition, showcasing the versatility and efficiency of SPP-net in addressing challenges in visual recognition tasks.

3. METHODOLOGY

i) Proposed Work:

Introducing an innovative MCS-YOLO algorithm for object detection and recognition in autonomous driving environments, our proposed system incorporates a coordinate attention module, multiscale small object detection structure, and Swin Transformer. This algorithm aims to significantly enhance accuracy and speed in detecting objects. Through rigorous ablation experiments and comparative trials on the BDD100K dataset [41], our MCS-YOLO algorithm demonstrated superior performance with a mean average precision (mAP) of 53.6%. To further elevate detection capabilities, our proposed system explores advanced techniques by considering Yolov5x6 and YoloV8. These additional methodologies aim to push the mAP beyond 60%, ensuring robust and efficient object detection. Each algorithm, including Faster RCNN, AD-Faster RCNN, YoloV3, YoloV3-tiny, YoloV4, YoloV5s, YoloV5s Improved Version, Yolo V7 - tiny, Yolo V5x6, Yolo V8, and MCS YoloV5s, contributes to the comprehensive evaluation of detection capabilities in various scenarios [12,13,14,15,23,24]. This diversified approach aims to optimize the autonomous driving environmental perception system, ensuring heightened accuracy and efficiency in identifying objects crucial for safe and reliable autonomous vehicle navigation.

ii) System Architecture:

The system architecture is a meticulously designed framework that seamlessly processes data, starting with input and progressing through image processing, incorporating sophisticated data augmentation techniques. At its core lies model building, where an extensive suite of advanced models, including YoloV5s, YoloV5s Improved Version, MCS YoloV5s, Yolo V5x6, YoloV4, YoloV3, YoloV3-tiny, Yolo V7, Yolo V8, Faster RCNN, and AD-Faster RCNN, is deployed [12,13,14,15,23,24]. The models undergo thorough evaluation, assessing performance metrics such as precision, recall, and mean average precision (mAP) [40]. The most effective model, as determined by these metrics, is selected for object detection. This comprehensive architecture ensures a streamlined and efficient process, elevating autonomous driving by providing robust and accurate perception in diverse road environments. The inclusion of a diverse model set allows for adaptability, enabling the system to excel across a spectrum of scenarios, contributing to the advancement of autonomous vehicle technology.

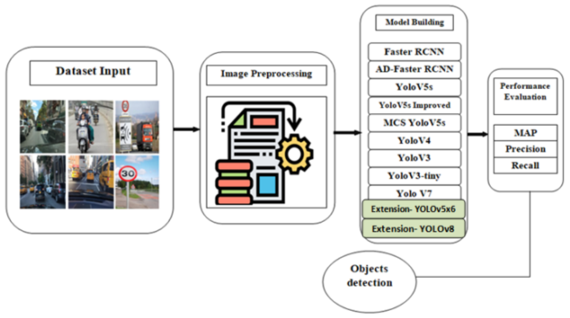


Fig 1 System Architecture

iii) Dataset collection:

The evaluation of the MCS-YOLO algorithm in autonomous driving perception utilizes the BDD100K dataset [41], renowned for its authenticity and comprehensiveness. Collected in real-life scenarios, this authoritative public dataset encompasses diverse weather conditions, driving scenarios, and times of the day, featuring ten target categories. With a substantial size of 100,000 images, the dataset includes six distinct weather conditions: sunny, cloudy, rainy, snowy, and foggy. To enhance model validation, 20,000 images without labels were removed, and the remaining data was partitioned at an 8:1:1 ratio for training, validation, and testing sets, respectively. The training set comprises 64,800 images, the validation set includes 7,200 images, and the test set consists of 8,000 images. Notably, object center points predominantly concentrate in the central image region, ensuring a uniform distribution of objects and a substantial representation of small targets within the dataset, providing a robust foundation for evaluating the MCS-YOLO algorithm's effectiveness in autonomous driving perception.

Fig 2 Dataset images

iv) Image Processing:

Image processing plays a pivotal role in object detection within autonomous driving systems, encompassing several key steps. The initial phase involves converting the input image into a blob object, optimizing it for subsequent analysis and manipulation. Following this, the classes of objects to be detected are defined, delineating the specific categories that the algorithm aims to identify. Simultaneously, bounding boxes are declared, outlining the regions of interest within the image where objects are expected to be located. The processed data is then converted into a NumPy array, a critical step for efficient numerical computation and analysis.

The subsequent stage involves loading a pre-trained model, leveraging existing knowledge from extensive datasets. This includes reading the network layers of the pre-trained model, containing learned features and parameters vital for accurate object detection. Additionally, output layers are extracted, providing final predictions and enabling effective object discernment and classification.

Further, in the image processing pipeline, the image and annotation file are appended, ensuring comprehensive information for subsequent analysis. The color space is adjusted by converting from BGR to RGB, and a mask is created to highlight relevant features. Finally, the image is resized, optimizing it for further processing and analysis. This comprehensive image processing workflow establishes a solid foundation for robust and accurate object detection in the dynamic context of autonomous driving systems, contributing to enhanced safety and decision-making capabilities on the road.

v) Data Augmentation:

Data augmentation is a fundamental technique in enhancing the diversity and robustness of training datasets for machine learning models, particularly in the context of image processing and computer vision. The process involves three key transformations to augment the original dataset: randomizing the image, rotating the image, and transforming the image.

Randomizing the image introduces variability by applying random modifications, such as changes in brightness, contrast, or color saturation. This stochastic approach helps the model generalize better to unseen data and diverse environmental conditions.

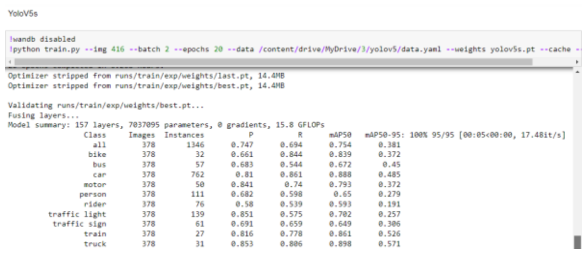
Rotating the image involves varying the orientation of the original image by different degrees. This augmentation technique aids in teaching the model to recognize objects from different perspectives, simulating variations in real-world scenarios.

Transforming the image includes geometric transformations such as scaling, shearing, or flipping. These alterations enrich the dataset by introducing distortions that mimic real-world variations in object appearance and orientation.

By employing these data augmentation techniques, the training dataset becomes more comprehensive, allowing the model to learn robust features and patterns. This, in turn, improves the model's ability to generalize and perform effectively on diverse and challenging test scenarios. Data augmentation serves as a crucial tool in mitigating overfitting, enhancing model performance, and promoting the overall reliability of machine learning models, especially in applications like image recognition for autonomous driving systems.

vi) Algorithms:

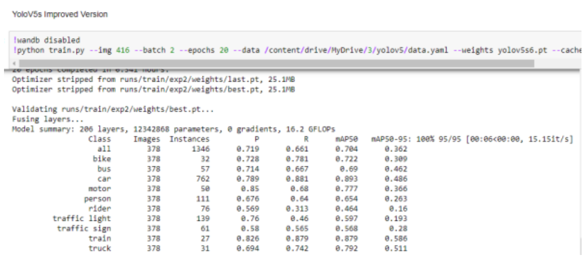
YoloV5s: YoloV5 (You Only Look Once) is an object detection algorithm known for its speed and accuracy. It divides an image into a grid and predicts bounding boxes and class probabilities for each grid cell. YoloV5s refers to the small version of YoloV5, balancing performance and efficiency.



```
!wandb disabled
!python train.py --img 416 --batch 2 --epochs 20 --data /content/drive/MyDrive/3/yolov5/data.yaml --weights yolov5s.pt --cache -
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7037095 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 95/95 [00:05<00:00, 17.48it/s]
                   all        378       1346      0.747      0.694      0.754      0.381
                  bike        378         32      0.661      0.844      0.839      0.372
                   bus        378         57      0.683      0.544      0.672       0.45
                   car        378        762       0.81      0.861      0.888      0.485
                 motor        378         50      0.841       0.74      0.793      0.372
                person        378        111      0.682      0.598       0.65      0.279
                 rider        378         76       0.58      0.539      0.593      0.191
         traffic light        378        139      0.851      0.575      0.702      0.257
          traffic sign        378         61      0.691      0.659      0.649      0.306
                 train        378         27      0.816      0.778      0.861      0.526
                 truck        378         31      0.853      0.806      0.898      0.571
```

Fig 3 YOLOV5s

YoloV5s Improved Version: This includes enhancements over the base YoloV5s, in terms of architecture modifications, training strategies, hyperparameter tuning. Improvements aim to boost accuracy and efficiency in object detection tasks.



```
!wandb disabled
!python train.py --img 416 --batch 2 --epochs 20 --data /content/drive/MyDrive/3/yolov5/data.yaml --weights yolov5s6.pt --cache
Optimizer stripped from runs/train/exp2/weights/last.pt, 25.1MB
Optimizer stripped from runs/train/exp2/weights/best.pt, 25.1MB

Validating runs/train/exp2/weights/best.pt...
Fusing layers...
Model summary: 206 layers, 12342868 parameters, 0 gradients, 16.2 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 95/95 [00:06<00:00, 15.15it/s]
                   all        378       1346      0.719      0.661      0.704      0.362
                  bike        378         32      0.728      0.781      0.722      0.309
                   bus        378         57      0.714      0.667       0.69      0.462
                   car        378        762      0.789      0.881      0.893      0.486
                 motor        378         50       0.85       0.68      0.777      0.366
                person        378        111      0.676       0.64      0.654      0.263
                 rider        378         76      0.569      0.313      0.464       0.16
         traffic light        378        139       0.76       0.46      0.597      0.193
          traffic sign        378         61       0.58      0.565      0.568       0.28
                 train        378         27      0.826      0.879      0.879      0.586
                 truck        378         31      0.694      0.742      0.792      0.511
```

Fig 4 YOLOV5s improved version

MCS YoloV5s: MCS YoloV5s, introduced in this project, incorporates a coordinate attention module for spatial and cross-channel information

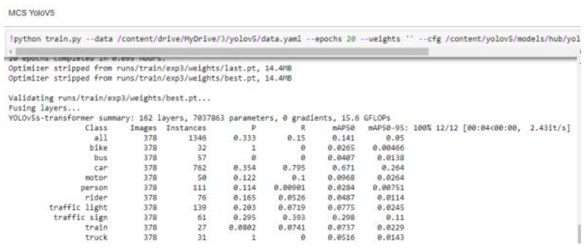
International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

aggregation. Additionally, it employs a multiscale small object detection structure for improved sensitivity, enhancing dense small object recognition. Integration of the Swin Transformer structure further enhances the network's focus on contextual spatial information [40].

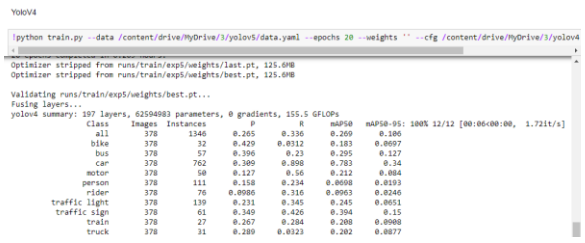


MCS YoloV5

```
!python train.py --data /content/drive/MyDrive/3/yolov5/data.yaml --epochs 20 --weights '' --cfg /content/yolov5/models/hub/yol
Optimizer stripped from runs/train/exp3/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp3/weights/best.pt, 14.4MB

Validating runs/train/exp3/weights/best.pt...
Fusing layers...
YOLOv5s-transformer summary: 162 layers, 7037863 parameters, 0 gradients, 15.6 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:04<00:00,  2.43it/s]
                   all        378       1346      0.333       0.15      0.141       0.05
                  bike        378         32          1          0     0.0265    0.00466
                   bus        378         57          0          0     0.0407     0.0138
                   car        378        762      0.354      0.795      0.671      0.264
                 motor        378         50      0.122        0.1     0.0968     0.0264
                person        378        111      0.114    0.00901     0.0284    0.00751
                 rider        378         76      0.165     0.0526     0.0487     0.0114
         traffic light        378        139      0.203     0.0719     0.0775     0.0245
          traffic sign        378         61      0.295      0.393      0.298       0.11
                 train        378         27     0.0802     0.0741     0.0737     0.0229
                 truck        378         31          1          0     0.0516     0.0143
```

Fig 5 MCS YOLOV5s

YoloV4: YoloV4 is an evolution of the Yolo series, emphasizing speed and accuracy. It introduces features like CSPDarknet53 as a backbone, PANet, and SAM block for improved object detection.

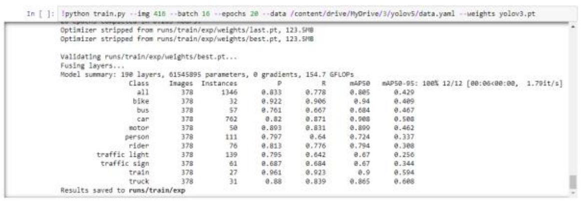


YoloV4

```
!python train.py --data /content/drive/MyDrive/3/yolov5/data.yaml --epochs 20 --weights '' --cfg /content/drive/MyDrive/3/yolov4
Optimizer stripped from runs/train/exp5/weights/last.pt, 125.6MB
Optimizer stripped from runs/train/exp5/weights/best.pt, 125.6MB

Validating runs/train/exp5/weights/best.pt...
Fusing layers...
yolov4 summary: 197 layers, 62594983 parameters, 0 gradients, 155.5 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:06<00:00,  1.72it/s]
                   all        378       1346      0.265      0.336      0.269      0.106
                  bike        378         32      0.429     0.0312      0.183     0.0697
                   bus        378         57      0.396       0.23      0.295      0.127
                   car        378        762      0.309      0.898      0.783       0.34
                 motor        378         50      0.127       0.56      0.212      0.084
                person        378        111      0.158      0.234     0.0698     0.0193
                 rider        378         76     0.0986      0.316     0.0963     0.0246
         traffic light        378        139      0.231      0.345      0.245     0.0651
          traffic sign        378         61      0.349      0.426      0.394       0.15
                 train        378         27      0.267      0.284      0.208     0.0908
                 truck        378         31      0.289     0.0323      0.202     0.0877
```

Fig 6 YOLOV4

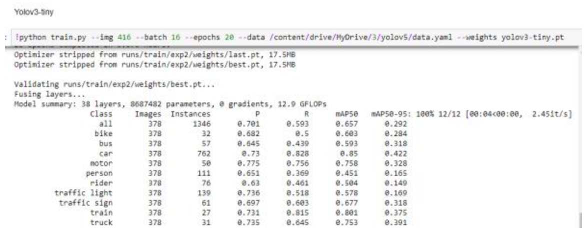
YoloV3: YoloV3 is an earlier version of the Yolo series, characterized by a three-stage detection process. It employs a Darknet-53 backbone and predicts bounding boxes at different scales. YoloV3 balances accuracy and speed in object detection tasks.



```
In [ ]: !python train.py --img 416 --batch 16 --epochs 20 --data /content/drive/MyDrive/3/yolov5/data.yaml --weights yolov3.pt
        Optimizer stripped from runs/train/exp/weights/last.pt, 123.5MB
        Optimizer stripped from runs/train/exp/weights/best.pt, 123.5MB

        Validating runs/train/exp/weights/best.pt...
        Fusing layers...
        Model summary: 190 layers, 61545895 parameters, 0 gradients, 154.7 GFLOPs
                         Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:06<00:00,  1.79it/s]
                           all        378       1346      0.833      0.778      0.805      0.429
                          bike        378         32      0.922      0.906       0.94      0.409
                           bus        378         57      0.761      0.667      0.684      0.467
                           car        378        762       0.82      0.871      0.908      0.508
                         motor        378         50      0.893      0.831      0.899      0.462
                        person        378        111      0.797       0.64      0.724      0.337
                         rider        378         76      0.813      0.776      0.794      0.308
                 traffic light        378        139      0.795      0.642       0.67      0.256
                  traffic sign        378         61      0.687      0.684       0.67      0.344
                         train        378         27      0.961      0.923        0.9      0.594
                         truck        378         31       0.88      0.839      0.865      0.608
        Results saved to runs/train/exp
```

Fig 7 YOLOV3

YoloV3-tiny: YoloV3-tiny is a lightweight version of YoloV3, optimized for faster inference on resource-constrained devices. It sacrifices some accuracy for improved speed, making it suitable for real-time applications.



Yolov3-tiny

```
: !python train.py --img 416 --batch 16 --epochs 20 --data /content/drive/MyDrive/3/yolov5/data.yaml --weights yolov3-tiny.pt
Optimizer stripped from runs/train/exp2/weights/last.pt, 17.5MB
Optimizer stripped from runs/train/exp2/weights/best.pt, 17.5MB

Validating runs/train/exp2/weights/best.pt...
Fusing layers...
Model summary: 38 layers, 8687482 parameters, 0 gradients, 12.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:04<00:00,  2.45it/s]
                   all        378       1346      0.701      0.593      0.657      0.292
                  bike        378         32      0.682        0.5      0.603      0.284
                   bus        378         57      0.645      0.439      0.593      0.318
                   car        378        762       0.73      0.828       0.85      0.422
                 motor        378         50      0.775      0.756      0.758      0.328
                person        378        111      0.651      0.369      0.451      0.165
                 rider        378         76       0.63      0.461      0.504      0.149
         traffic light        378        139      0.736      0.518      0.578      0.169
          traffic sign        378         61      0.697      0.603      0.677      0.318
                 train        378         27      0.731      0.815      0.801      0.375
                 truck        378         31      0.735      0.645      0.753      0.391
```

Fig 8 YOLOV3-tiny

Yolo V7: YOLOv7, an advanced iteration in the YOLO series, integrates features from YOLOv4, Scaled YOLOv4, and YOLO-R. Its core, the Extended Efficient Layer Aggregation Network (E-ELAN), enhances learning, and Compound Model Scaling allows independent adjustment of width, depth, and resolution. YOLOv7 excels in real-time object detection, making it a fitting choice for the

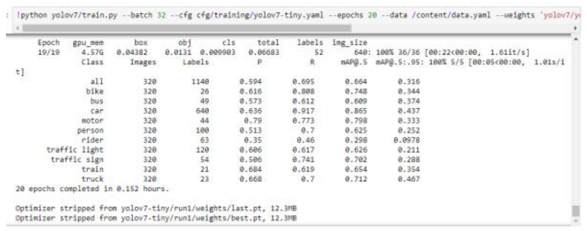
International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

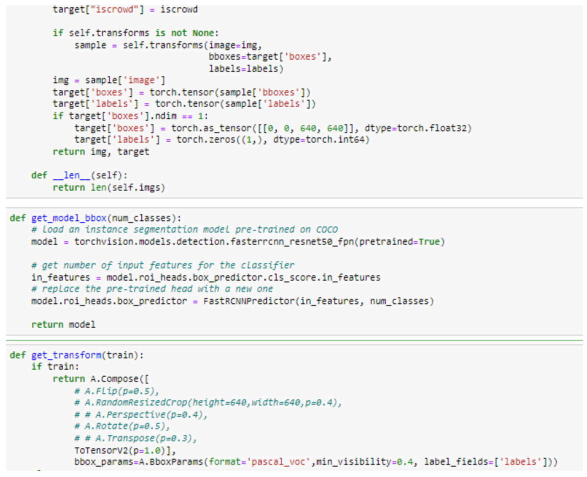
project's demands, with its balance of efficiency, adaptability, and accuracy.



```
: !python yolov7/train.py --batch 32 --cfg cfg/training/yolov7-tiny.yaml --epochs 20 --data /content/data.yaml --weights 'yolov7/y

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     19/19     4.57G   0.04382    0.0131  0.009903   0.06683        52       640: 100% 36/36 [00:22<00:00,  1.61it/s]
               Class    Images    Labels         P         R    mAP@.5  mAP@.5:.95: 100% 5/5 [00:05<00:00,  1.01s/i
t]
                 all       320      1140     0.594     0.695     0.664     0.316
                bike       320        26     0.616     0.808     0.748     0.344
                 bus       320        49     0.573     0.612     0.609     0.374
                 car       320       640     0.636     0.917     0.865     0.437
               motor       320        44      0.79     0.773     0.798     0.333
              person       320       100     0.513       0.7     0.625     0.252
               rider       320        63      0.35      0.46     0.298    0.0978
       traffic light       320       120     0.606     0.617     0.626     0.211
        traffic sign       320        54     0.506     0.741     0.702     0.288
               train       320        21     0.684     0.619     0.654     0.354
               truck       320        23     0.668       0.7     0.712     0.467
20 epochs completed in 0.152 hours.

Optimizer stripped from yolov7-tiny/run1/weights/last.pt, 12.3MB
Optimizer stripped from yolov7-tiny/run1/weights/best.pt, 12.3MB
```

Fig 9 YOLOV7

Faster RCNN: Faster RCNN (Region-based Convolutional Neural Network) is a two-stage object detection framework. It proposes regions of interest using a Region Proposal Network (RPN) and classifies those regions.



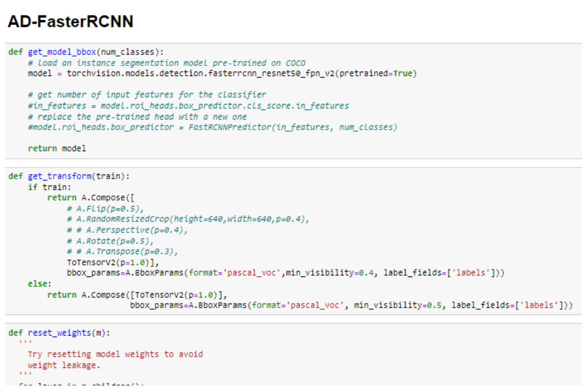
```
        target["iscrowd"] = iscrowd

        if self.transforms is not None:
            sample = self.transforms(image=img,
                                     bboxes=target['boxes'],
                                     labels=labels)
        img = sample['image']
        target['boxes'] = torch.tensor(sample['bboxes'])
        target['labels'] = torch.tensor(sample['labels'])
        if target['boxes'].ndim == 1:
            target['boxes'] = torch.as_tensor([[0, 0, 640, 640]], dtype=torch.float32)
            target['labels'] = torch.zeros((1,), dtype=torch.int64)
        return img, target

    def __len__(self):
        return len(self.imgs)
```

```
def get_model_bbox(num_classes):
    # load an instance segmentation model pre-trained on COCO
    model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)

    # get number of input features for the classifier
    in_features = model.roi_heads.box_predictor.cls_score.in_features
    # replace the pre-trained head with a new one
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

    return model
```

```
def get_transform(train):
    if train:
        return A.Compose([
            # A.Flip(p=0.5),
            # A.RandomResizedCrop(height=640,width=640,p=0.4),
            # # A.Perspective(p=0.4),
            # A.Rotate(p=0.5),
            # # A.Transpose(p=0.3),
            ToTensorV2(p=1.0)],
            bbox_params=A.BboxParams(format='pascal_voc',min_visibility=0.4, label_fields=['labels']))
```

Fig 10 Faster RCNN

AD-Faster RCNN:AD-FRCNN (Adaptive Dynamic Faster R-CNN) enhances the Faster R-CNN object detection by incorporating a dynamic region proposal network, a visual attention scheme for better feature generation, and an adaptive dynamic training module, enhancing adaptability and overall object detection performance [42].



AD-FasterRCNN

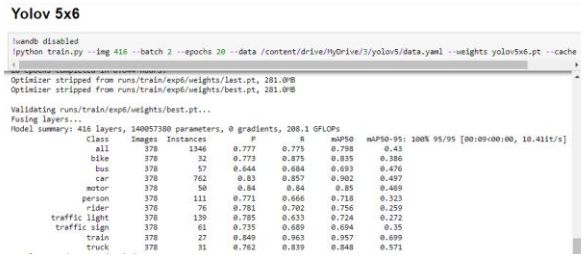
```
def get_model_bbox(num_classes):
    # load an instance segmentation model pre-trained on COCO
    model = torchvision.models.detection.fasterrcnn_resnet50_fpn_v2(pretrained=True)

    # get number of input features for the classifier
    #in_features = model.roi_heads.box_predictor.cls_score.in_features
    # replace the pre-trained head with a new one
    #model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

    return model
```

```
def get_transform(train):
    if train:
        return A.Compose([
            # A.Flip(p=0.5),
            # A.RandomResizedCrop(height=640,width=640,p=0.4),
            # # A.Perspective(p=0.4),
            # A.Rotate(p=0.5),
            # # A.Transpose(p=0.3),
            ToTensorV2(p=1.0)],
            bbox_params=A.BboxParams(format='pascal_voc',min_visibility=0.4, label_fields=['labels']))
    else:
        return A.Compose([ToTensorV2(p=1.0)],
                         bbox_params=A.BboxParams(format='pascal_voc', min_visibility=0.5, label_fields=['labels']))
```

```
def reset_weights(m):
  '''
    Try resetting model weights to avoid
    weight leakage.
  '''
```

Fig 11 AD-FasterRCNN

Yolo V5x6, a variant of the YOLO object detection model, is tailored for this project, excelling in rapid and accurate object detection. Utilizing a grid-based approach for predicting bounding boxes and class probabilities, it distinguishes itself with six times the processing capacity. This computational boost is pivotal for meeting the project's demands, emphasizing swift inference and precise object detection crucial for advancing autonomous driving technology in diverse road scenarios.



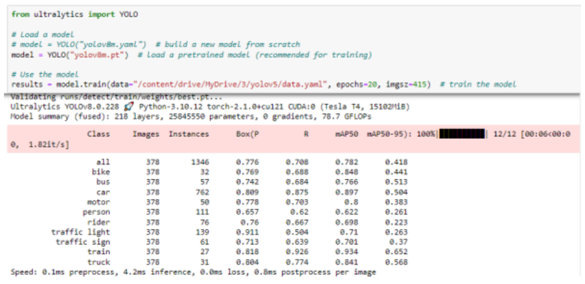
Yolov 5x6

```
!wandb disabled
!python train.py --img 416 --batch 2 --epochs 20 --data /content/drive/MyDrive/3/yolov5/data.yaml --weights yolov5x6.pt --cache
```

```
Optimizer stripped from runs/train/exp6/weights/last.pt, 281.0MB
Optimizer stripped from runs/train/exp6/weights/best.pt, 281.0MB

Validating runs/train/exp6/weights/best.pt...
Fusing layers...
Model summary: 416 layers, 140057380 parameters, 0 gradients, 208.1 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 95/95 [00:09<00:00, 10.41it/s]
                   all        378       1346      0.777      0.775      0.798       0.43
                  bike        378         32      0.773      0.875      0.835      0.386
                   bus        378         57      0.644      0.684      0.693      0.476
                   car        378        762       0.83      0.857      0.902      0.497
                 motor        378         50       0.84       0.84       0.85      0.469
                person        378        111      0.771      0.666      0.718      0.323
                 rider        378         76      0.781      0.702      0.756      0.259
         traffic light        378        139      0.785      0.633      0.724      0.272
          traffic sign        378         61      0.735      0.689      0.694       0.35
                 train        378         27      0.849      0.963      0.957      0.699
                 truck        378         31      0.762      0.839      0.848      0.571
```

Fig 12 YOLOV5x6

YOLOv8, a pinnacle in the YOLO series, excels in simultaneous object detection by dividing images into a grid and predicting bounding boxes and class probabilities. Recognized for superior accuracy and speed, it offers a user-friendly API, supporting Object Detection, Instance Segmentation, and Image Classification. Its novel architecture, including C2f modules and an anchor-free head, enhances efficiency and adaptability. Chosen for this project, YOLOv8 aligns with the goal of robust and real-time object detection.



```
from ultralytics import YOLO

# Load a model
# model = YOLO("yolov8m.yaml")  # build a new model from scratch
model = YOLO("yolov8m.pt")  # load a pretrained model (recommended for training)

# Use the model
results = model.train(data="/content/drive/MyDrive/3/yolov5/data.yaml", epochs=20, imgsz=415)  # train the model
```

```
Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.228 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25845550 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 12/12 [00:06<00:0
0,  1.82it/s]
                   all        378       1346      0.776      0.708      0.782      0.418
                  bike        378         32      0.769      0.688      0.848      0.441
                   bus        378         57      0.742      0.684      0.766      0.513
                   car        378        762      0.809      0.875      0.897      0.504
                 motor        378         50      0.778      0.703        0.8      0.383
                person        378        111      0.657       0.62      0.622      0.261
                 rider        378         76       0.76      0.667      0.698      0.223
         traffic light        378        139      0.911      0.504       0.71      0.263
          traffic sign        378         61      0.713      0.639      0.701       0.37
                 train        378         27      0.818      0.926      0.934      0.652
                 truck        378         31      0.804      0.774      0.841      0.568
Speed: 0.1ms preprocess, 4.2ms inference, 0.0ms loss, 0.8ms postprocess per image
```

Fig 13 YOLOV8

4. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

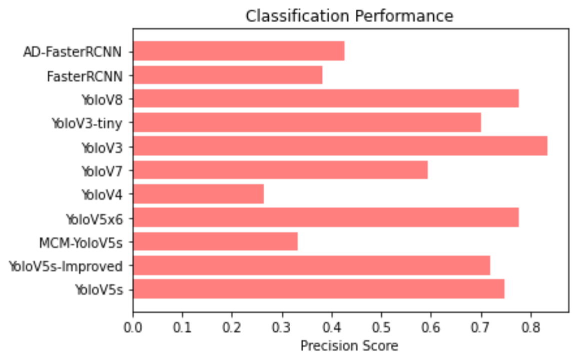


Fig 14 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$Recall = \frac{TP}{TP + FN}$$

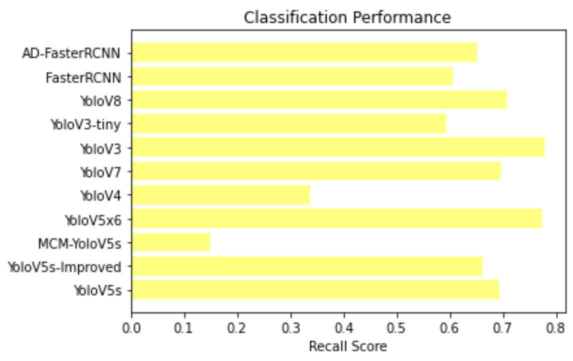


Fig 15 Recall comparison graph

mAP: Mean Average Precision (MAP) is a ranking quality metric. It considers the number of relevant recommendations and their position in the list. MAP at K is calculated as an arithmetic mean of the Average Precision (AP) at K across all users or queries.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{the AP of class } k$$

$$n = \text{the number of classes}$$

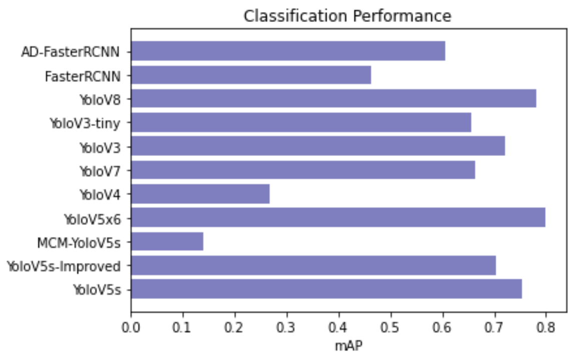
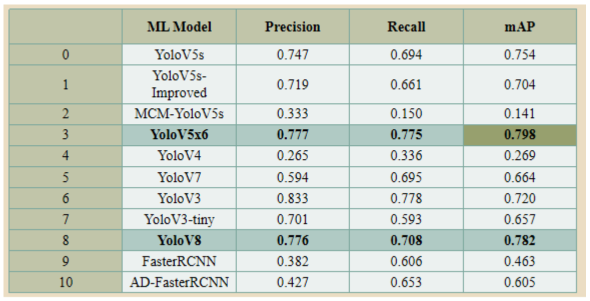
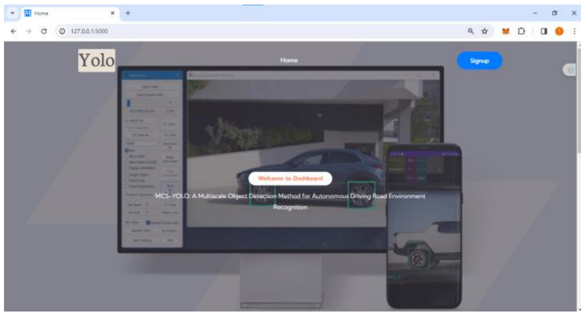


Fig 16 mAP comparison graph



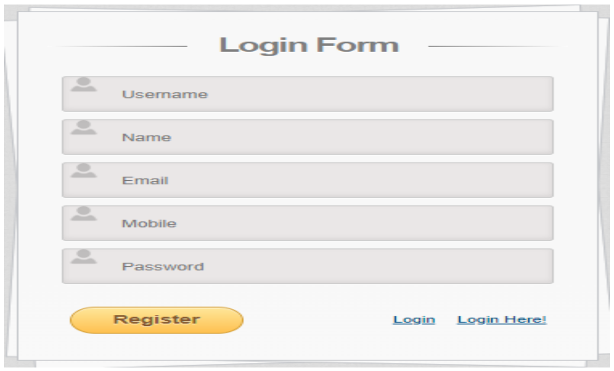
	ML Model	Precision	Recall	mAP
0	YoloV5s	0.747	0.694	0.754
1	YoloV5s-Improved	0.719	0.661	0.704
2	MCM-YoloV5s	0.333	0.150	0.141
3	**YoloV5x6**	**0.777**	**0.775**	**0.798**
4	YoloV4	0.265	0.336	0.269
5	YoloV7	0.594	0.695	0.664
6	YoloV3	0.833	0.778	0.720
7	YoloV3-tiny	0.701	0.593	0.657
8	**YoloV8**	**0.776**	**0.708**	**0.782**
9	FasterRCNN	0.382	0.606	0.463
10	AD-FasterRCNN	0.427	0.653	0.605

Fig 17 Performance Evaluation table



MCS-YOLO: A Multiscale Object Detection Method for Autonomous Driving Road Environment Recognition

Fig 18 Home page


Login Form

Username

Name

Email

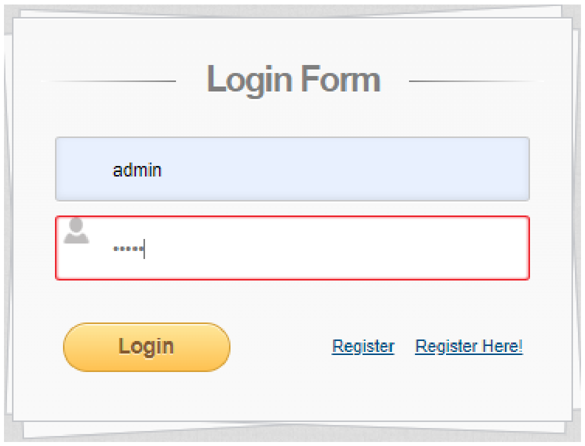
Mobile

Password

Register

Login Login Here!

Fig 19 Registration page


Login Form

admin

•••••

Login

Register Register Here!

Fig 20 Login page

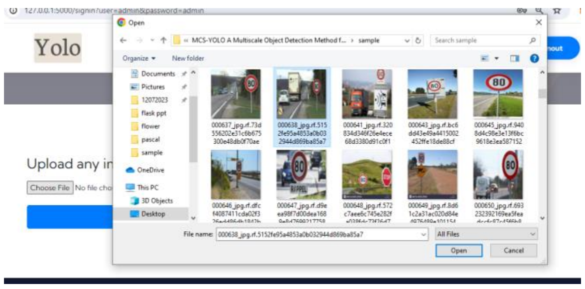
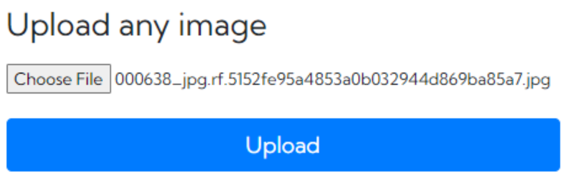


Fig 21 Input image folder


Upload any image

Choose File 000638_jpg.rf.5152fe95a4853a0b032944d869ba85a7.jpg

Upload

Fig 22 Upload input image

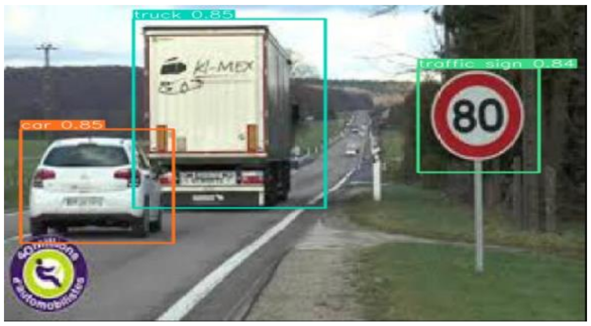


Fig 23 Predict result for given input

5. CONCLUSION

In conclusion, our work introduces the MCS-YOLO algorithm, showcasing its effectiveness and superiority in autonomous driving object detection. Leveraging a coordinate attention module, multiscale small object detection structure, and Swin Transformer, the algorithm significantly improves detection accuracy and speed. Ablation experiments and comparative trials on the BDD100K dataset [41]underscore its remarkable performance enhancements over existing algorithms. Future endeavors involve applying MCS-YOLO to the Multiple Object Tracking (MOT) task, ensuring its adaptability and robustness in varied autonomous driving scenarios. This project addresses the imperative need to enhance autonomous driving safety amid escalating challenges of accidents and congestion. By revolutionizing environmental perception through deep learning algorithms, including YOLOv5s, an improved version, and the innovative MCS-YOLOv5s [25,46], we advance autonomous transportation. Comparative analyses against benchmarks, exploration of advanced models, and integration with the Flask framework and SQLite for user testing showcase our commitment to technological excellence. Ultimately, the beneficiaries include users and communities, as our project contributes to safer transportation, enhanced efficiency, reduced pollution, and broader advancements in autonomous driving technology.

6. FUTURE SCOPE

Future endeavors include enhancing object detection capabilities by incorporating radar and LiDAR sensors for comprehensive environmental understanding. Optimization of real-time processing involves leveraging advanced hardware acceleration, parallel processing, and model compression to cater to dynamic scenarios. Exploring seamless integration of edge computing aims to decentralize processing, reduce latency, and enhance adaptability, especially in resource-constrained or time-sensitive scenarios. Staying at the forefront of advancements involves continuous exploration and integration of the latest algorithms and architectures, ensuring adaptability to emerging challenges in autonomous driving technology [42].

REFERENCES

[1] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, ''Towards fully autonomous driving: Systems and algorithms,'' in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2011, pp. 163–168.

[2] J. Janai, F. Güney, A. Behl, and A. Geiger, ''Computer vision for autonomous vehicles: Problems, datasets and state of the art,'' Found. Trends Comput. Graph. Vis., vol. 12, no. 1–3, pp. 1–308, 2020.

[3] Y. Wang, ''Overview on key technology of perceptual system on selfdriving vehicles,'' Auto Electr. Parts., vol. 2016, no. 12, pp. 12–16, Dec. 2016.

[4] M. Kuderer, S. Gulati, and W. Burgard, ''Learning driving styles for autonomous vehicles from demonstration,'' in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2015, pp. 2641–2646.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ''Rich feature hierarchies for accurate object detection and semantic segmentation,'' in Proc. IEEE

Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 580–587.

[6] K. He, X. Zhang, J. Sun, and S. Ren, ''Spatial pyramid pooling in deep convolutional networks for visual recognition,'' IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904–1916, Jan. 2015.

[7] R. Girshick, ''Fast R-CNN,'' in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1440–1448.

[8] S. Ren, K. He, R. Girshick, and J. Sun, ''Faster R-CNN: Towards realtime object detection with region proposal networks,'' IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[9] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, ''Mask R-CNN,'' in IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2980–2988.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ''You only look once: Unified, real-time object detection,'' in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 779–788.

[11] J. Redmon and A. Farhadi, ''YOLO9000: Better, Faster, stronger,'' in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Honolulu, HI, USA, Jul. 2017, pp. 6517–6525.

[12] J. Redmon and A. Farhadi, ''YOLOv3: An incremental improvement,'' 2018, arXiv:1804.02767.

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, ''YOLOv4: Optimal speed and accuracy of object detection,'' 2020, arXiv:2004.10934.

[14] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, ''YOLOX: Exceeding Yolo series in 2021,'' 2021, arXiv:2107.08430.

[15] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, ''YOLOv6: A single-stage object detection framework for industrial applications,'' 2022, arXiv:2209.02976.

[16] 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,'' 2022, arXiv:2207.02696.

[17] W. Liu, ''SSD: Single shot multibox detector,'' in Proc. Eur. Conf. Comput. Vis. Amsterdam, The Netherlands, 2016, pp. 21–37.

[18] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, ''DSSD : Deconvolutional single shot detector,'' 2017, arXiv:1701.06659.

[19] Z. Li and F. Zhou, ''FSSD: Feature fusion single shot multibox detector,'' 2017, arXiv:1712.00960.

[20] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, ''Swin transformer: Hierarchical vision transformer using shifted Windows,'' in Proc.

IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 9992–10002.

[21] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo, ''Swin transformer v2: Scaling up capacity and resolution,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 12009–12019.

[22] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, ''A ConvNet for the 2020s,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 11976–11986.

[23] Y. Chen et al., ''DW-YOLO: An efficient object detector for drones and self-driving vehicles,'' Arabian J. Sci. Eng., vol. 48, pp. 1–10, May 2022.

[24] Y. Zhou, S. Wen, D. Wang, J. Meng, J. Mu, and R. Irampaye, ''MobileYOLO: Real-time object detection algorithm in autonomous driving scenarios,'' Sensors, vol. 22, no. 9, p. 3349, Apr. 2022.

[25] H. Wang and W. Zang, ''Research on object detection method in driving scenario based on improved YOLOv4,'' in Proc. IEEE 6th Inf. Technol. Mechatronics Eng. Conf. (ITOEC), Mar. 2022, pp. 1751–1754.

[26] D. Tian, ''SA-YOLOv3: An efficient and accurate object detector using self-attention mechanism for autonomous driving,'' IEEE Trans. Intell. Transp. Syst., vol. 23, no. 5, pp. 4099–4110, May 2022.

[27] A. Gupta, K. Illanko, and X. Fernando, ''Object detection for connected and autonomous vehicles using CNN with attention mechanism,'' in Proc. IEEE 95th Veh. Technol. Conf., (VTC-Spring), Jun. 2022, pp. 1–6.

[28] H. Wang, Y. Xu, Y. He, Y. Cai, L. Chen, Y. Li, M. A. Sotelo, and Z. Li, ''YOLOv5-fog: A multiobjective visual detection algorithm for fog driving scenes based on improved YOLOv5,'' IEEE Trans. Instrum. Meas., vol. 71, pp. 1–12, 2022.

[29] Y. Li, J. Wang, J. Huang, and Y. Li, ''Research on deep learning automatic vehicle recognition algorithm based on RES-YOLO model,'' Sensors, vol. 22, no. 10, p. 3783, May 2022.

[30] J. Hu, L. Shen, and G. Sun, ''Squeeze-and-excitation networks,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 7132–7141.

[31] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, ''CBAM: Convolutional block attention module,'' in Proc. Eur. Conf. Comput. Vis. (ECCV), Sep. 2018, pp. 3–19.

[32] Y. Liu, Z. Shao, and N. Hoffmann, ''Global attention mechanism: Retain information to enhance

channel-spatial interactions,'' 2021, arXiv:2112.05561.

[33] X. Pan, C. Ge, R. Lu, S. Song, G. Chen, Z. Huang, and G. Huang, ''On the integration of self-attention and convolution,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 815–825.

[34] Q. Hou, D. Zhou, and J. Feng, ''Coordinate attention for efficient mobile network design,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 13708–13717.

[35] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, ''Feature pyramid networks for object detection,'' in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 936–944.

[36] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, ''Path aggregation network for instance segmentation,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8759–8768.

[37] G. Ghiasi, T.-Y. Lin, and Q. V. Le, ''NAS-FPN: Learning scalable feature pyramid architecture for object detection,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 7036–7045.

[38] S. Qiao, L.-C. Chen, and A. Yuille, ''DetectoRS: Detecting objects with recursive feature pyramid and switchable atrous convolution,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 10213–10224.

[39] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, ''An image is worth 16×16 words: Transformers for image recognition at scale,'' 2020, arXiv:2010.11929.

[40] J. Park, S. Woo, J. Lee, and I. Kweon, ''BAM: Bottleneck attention module,'' presented at the 29th Brit. Mach. Vis. Conf., Newcastle, U.K., Sep. 2018.

[41] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, ''BDD100K: A diverse driving dataset for heterogeneous multitask learning,'' 2018, arXiv:1805.04687.

[42] Y. Zhou, S. Wen, D. Wang, J. Mu, and I. Richard, ''Object detection in autonomous driving scenarios based on an improved faster-RCNN,'' Appl. Sci., vol. 11, no. 24, Dec. 2021, Art. no. 11630.

[43] L. Li and X. Li, ''H-YoLov3: High performance object detection applied to assisted driving,'' in Proc. Asia Conf. Algorithms, Comput. Mach. Learn. (CACML), Mar. 2022, pp. 462–467.

[44] F. Yang, X. Zhang, S. Zhang, C. Li, and H. Hu, ''Design of real-time vehicle detection based on

YOLOv4,'' in Proc. Int. Conf. Control, Autom. Inf. Sci. (ICCAIS), Oct. 2021, pp. 824–829.

[45] Y. Li and F. Yu, ''CDMY: A lightweight object detection model based on coordinate attention,'' in Proc. IEEE 10th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC), Jun. 2022, pp. 1258–1263.

[46] Q. Luo, J. Wang, M. Gao, Z. He, Y. Yang, and H. Zhou, ''Multiple mechanisms to strengthen the ability of YOLOv5s for real-time identification of vehicle type,'' Electronics, vol. 11, no. 16, p. 2586, Aug. 2022.