JOURNAL of
MATERIALS and MECHATRONICS:A

e-ISSN 2717-8811
JournalMM, 2022, 3(2), 275-289
https://doi.org/10.55546/jmm.1196409

Araştırma Makalesi / Research Article

Real-Time Application of Traffic Sign Recognition Algorithm with Deep Learning

Faruk Emre AYSAL[1]*, Kasım YILDIRIM[2], Enes CENGİZ[3]

[1] Afyon Kocatepe Üniversitesi, Teknoloji Fakültesi, Mekatronik Mühendisliği Bölümü, Afyonkarahisar, Türkiye,
ORCID ID: https://orcid.org/0000-0002-9514-1425, faysal@aku.edu.tr
[2] Afyon Kocatepe Üniversitesi Teknoloji Fakültesi, Mekatronik Mühendisliği Bölümü, Afyonkarahisar, Türkiye,
ORCID ID: https://orcid.org/0000-0002-5393-3889, kasimm.yildirimm@outlook.com
[3] Sinop Üniversitesi, Ayancık Meslek Yüksek Okulu, Elektronik ve Otomasyon Bölümü, Sinop, Türkiye,
ORCID ID: https://orcid.org/0000-0003-1127-2194, ecengiz@sinop.edu.tr

Geliş/ Received: 29.10.2022; **Kabul / Accepted:** 30.11.2022

ABSTRACT: Autonomous vehicles are one of the increasingly widespread application areas in automotive technology. These vehicles show significant potential in improving transportation systems, with their ability to communicate, coordinate and drive autonomously. These vehicles, which move from source to destination without human intervention, appear to be a solution to various problems caused by people in traffic, such as accidents and traffic jams. Traffic accidents and traffic jams are largely due to driver faults and non-compliance with traffic rules. For this reason, it is predicted that integrating artificial intelligence (AI)-based systems into autonomous vehicles will be a solution to such situations, which are seen as a problem in social life. Looking at the literature, VGGNet, ResNet50, MobileNetV2, NASNetMobile, Feed Forward Neural Networks, Recurrent Neural Networks, Long-Short Term Memory, and Gate Recurrent Units It is seen that deep learning models such as these are widely used in traffic sign classification studies. Unlike previous studies, in this study, a deep learning application was made for the detection of traffic signs and markers using an open-source data set and models of YOLOv5 versions. The original data set was prepared and used in the study. Labeling of this data set in accordance with different AI models has been completed. In the developed CNN models, the training process of the data set containing 15 different traffic sign classes was carried out. The results of these models were systematically compared, and optimum performance values were obtained from the models with hyper parameter changes. Real-time application was made using the YOLOv5s model. As a result, a success rate of 98-99% was achieved.

Keywords: Deep Learning, Autonomous Vehicle, Real-Time, YOLOv5.

*Sorumlu yazar / Corresponding author: faysal@aku.edu.tr

Bu makaleye atıf yapmak için /To cite this article

Aysal, F. E., Yıldırım, K., Cengiz, E. (2022). Real-Time Application of Traffic Sign Recognition Algorithm with Deep Learning. Journal of Materials and Mechatronics: A (JournalMM), 3(2), 275-289.

1. INTRODUCTION

The growth in total number of vehicles in the world has increased exponentially over the last ten years. Population growth in the world causes increased traffic density, increasingly congested roads, transportation problems, air pollution, accidents with injury or loss of life (Jain et al., 2018). These events are undesirable but important consequences of modern transportation systems (Satria and Castro, 2016). The approach of existing rules and regulations to prevent traffic accidents is often not effective enough. Raising awareness, strict enforcement of traffic rules, and scientific engineering measures are seen as the needs of the society to prevent this public health problem (Gopalakrishnan, 2012). These reasons necessitate the optimization of the traffic network to meet the transportation needs of the city (Dorokhin et al., 2020).

The adoption of autonomous vehicle technology is known to have many benefits compared to conventional vehicles, such as higher reliability, better traffic flow, and reduction in traffic congestion (Fagnant et al., 2015). AI, the basis of autonomous vehicle technology, is a general term that can interact with the environment, aims to simulate human intelligence, and implies the use of a computer to model intelligent behavior with minimal human intervention (Glikson and Woolley, 2020). In other words, it is defined as the ability of a system to correctly interpret external data, learn from such data, and use these learnings to achieve specific goals and tasks through flexible adaptation (Kaplan and Haenlein, 2019). Due to the exponential increase in popularity and use of AI-based techniques in various applications over the past few years, there has been a thriving increase in the use of autonomous vehicles around the world (Miglani and Kumar, 2019). Thanks to this increase, it is expected that traditional vehicles will be replaced by smart vehicles that can decide on their own and perform driving tasks in the near future. (Kulkarni et al., 2018). These vehicles, which move to a specific destination without human intervention, emerge as solutions to various problems such as accidents and traffic jams (Sarkar et al., 2018; Rajabli et al., 2020). Deep learning-based detection of traffic signs, which have an important place in the regulation of traffic flow, plays a key role in the decision mechanism of autonomous cars. Therefore, it is predicted that integrating deep learning-based systems into autonomous vehicles will be a solution to the problems seen in social life. In the studies to be carried out for this purpose, image processing-based leather learning applications come to the fore. Image processing is a method of performing some operations on an image to obtain an enhanced image or extract useful information from it. It is a type of signal processing where the input is an image, and the output can be an image or features associated with that image (Kour et al., 2012; Wiley and Lucas, 2018; Bayram, 2020). In the field of image processing, feature and feature determination operations play a very important role. Before the features are obtained, binarization, thresholding, resizing, normalization, etc. are performed on the sampled image. Various image preprocessing techniques are performed. Then, feature extraction techniques are applied to obtain features that will be useful in classification and recognition of images. (Kumar and Bhatia, 2014). Autonomous decision support systems that provide enhanced image recognition performance can be developed by using the data obtained by image processing techniques as training data in various deep learning algorithms (Guo et al., 2016; Zhou et al., 2019; Khan et al., 2018; Hasan et al., 2020; Shrestha et al., 2019; Sampedro et al., 2017; Zhao et al., 2019; Huval et al., 2015; Kulkarni et al., 2018; Miglani et al., 2019; Eraqi et al., 2017).

Lim et al. proposed a method that uses real-time traffic sign detection based on GPGPU (General-Purpose Graphics Processing Unit) and performs zone detection and recognition using a hierarchical model. This method produces stable results in low-illumination environments. It performed both detection and hierarchical recognition in real-time, and the proposed method reached an F1 score of 0.97 in the aggregated data set. (Lim et al., 2017). Çetin and Ortataş have created different datasets of six signs related to traffic signs and markers used in Turkey. Trained the model with the Haarcascade machine learning algorithm and tested the training data by simulating the behavior of the autonomous vehicle on the track. (Cetin and Ortatas, 2021). Palandz and Bayrakçı trained using the data set consisting of 4000 traffic signposts, ResNet50, MobileNetV2, and NASNetMobile models. They achieved an accuracy of 97.62% in the ResNet50 model, 97.8% in the MobileNetV2 model, and 98.56% in the NASNetMobile model. (Palandiz and Bayrakçı, 2021). Swaminathan et al. performed image recognition application for the detection of traffic signs using Convolutional Neural Network (CNN). They tried to design a system that could respond in real-time with an Arduino-controlled autonomous car. To examine the performance of this road sign recognition system, they conducted various experiments using the Belgian Traffic Signs dataset and achieved 83.7% accuracy with this approach (Swaminathan et al., 2019).

In this study, the accuracy and execution time performances of different deep learning models in real-time were systematically compared on autonomous vehicles. The original data set of various traffic signs prepared by us was included in the study and examined more comprehensively. The data was systematically learned in current deep learning models such as YOLOv4, YOLOv4-tiny, YOLOv5s, YOLOv5m, YOLOv5x, Faster R-CNN, Mask R-CNN. In the classification process, it was seen that the YOLOv5x model performed better than other models.

The rest of the work is organized as follows. In Chapter 2, the material and method part is given. In this section, general information about the dataset and models used is given. As a result of the different model trainings carried out in Chapter 3, the confusion matrix was examined, and the model efficiencies were compared. 4. Finally, the conclusion section is concluded by reporting the results of the most efficient model and talking about the general outlines of the study.

2. MATERIALS AND METHODS

The flow chart of the work carried out is presented in Figure 1. In the first stage, there is the process of preparing the data set containing the traffic signs, which will constitute an important part of the learning stage for different classes (Stop, Green Light, Park, etc.). The next step is to perform different label operations for the data set containing 26 different classes, for the deep learning models (YOLOv5s, YOLOv5m, YOLOv5x) to be used. The labeling process distinguishes itself in the form of rectangles, polygons, points, lines, etc. The reason for this distinction is that the desired output form from deep learning models is not the same. The training outputs may differ with applications such as masking, segmentation, and detection. For this reason, it is important to carry out labeling processes in accordance with the deep learning models that are desired to be used. The learning phase begins with the introduction of the data set, which has been labeled and made ready for the learning phase, to different deep learning models. Then there is a systematic comparison of the learning performances of the models. Among the models that are systematically compared, the best model is determined according to the output data. TensorRT technology is used to make this model compatible with the hardware. As a result of this process, it is aimed to test the model suitable for the hardware in real-time in the field and to report the results.

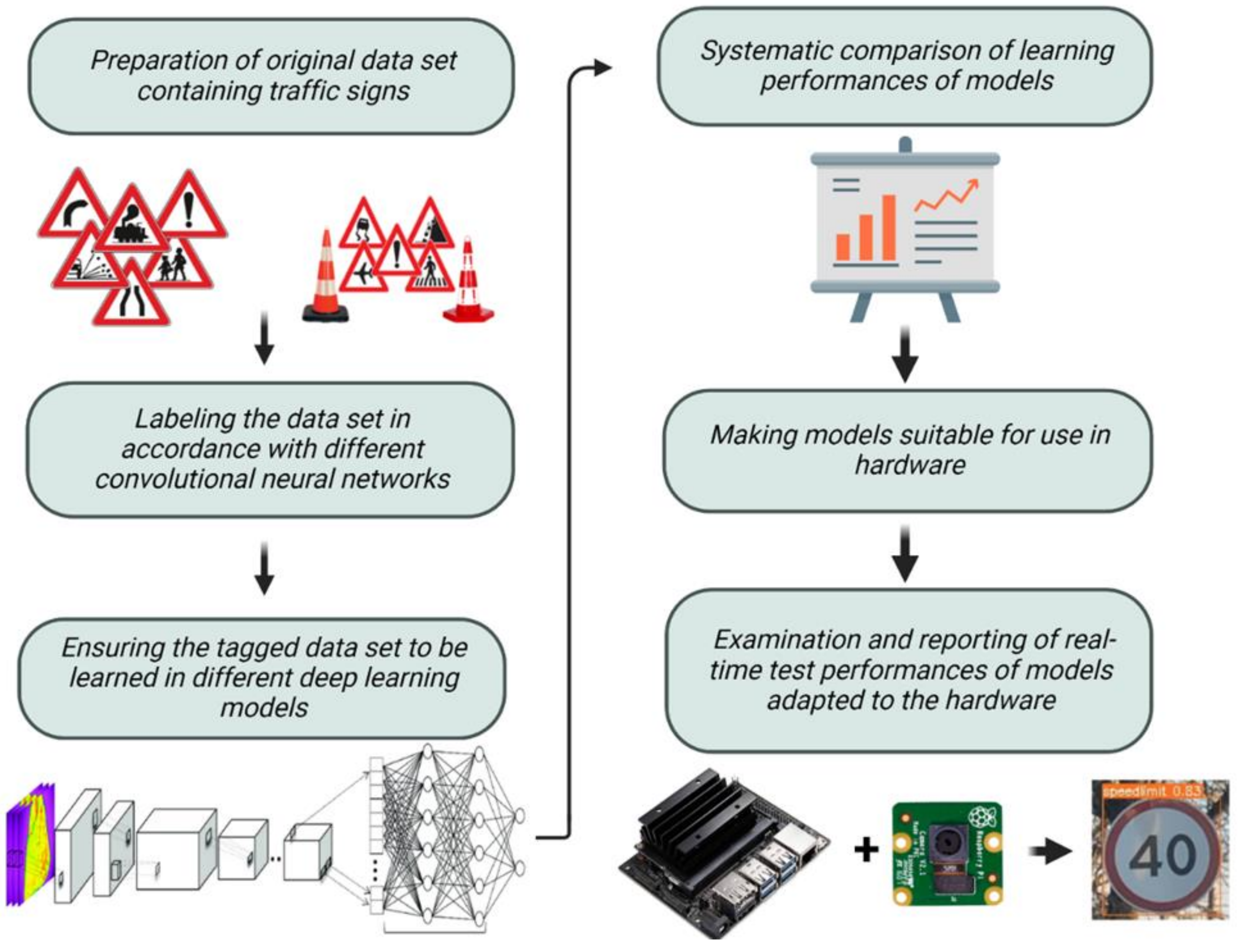

Preparation of original data set containing traffic signs
Labeling the data set in accordance with different convolutional neural networks
Ensuring the tagged data set to be learned in different deep learning models
Systematic comparison of learning performances of models
Making models suitable for use in hardware
Examination and reporting of real-time test performances of models adapted to the hardware
speedlimit 0.83
40

Figure 1. 1 Deep learning process for traffic sign classification

In this study, the traffic sign images were taken in Afyonkarahisar province, and a data set containing 1249 traffic sign images in total was used. Video recording was taken in Afyonkarahisar province by using the camera placed on a standard road vehicle. A total of 1249 traffic sign images were obtained from this video recording and a data set was created. For the preprocessing, the outlier images and tags in the data set were separated, and then the labeling process was carried out on the "makesensei.ai" site. More than 5000+ labeling processes (Figure 9 in Appendix) were performed for 26 different classes in 1249 images. 80% of this images data (1000 pieces of data) was used for model training (Figure 10 in Appendix), and the remaining 20% (249 pieces of data) was used in the testing phase. The distribution of the data set by classes is shown in Table 1

Figure 2 shows an example CNN architecture. Convolutional neural networks are formed by the combination of certain layers. The CNN algorithm consists of layers such as pooling, convolution, full connection, and activation. The convolution layer is the main building block of the neural network architecture. It is responsible for detecting the features of the picture. This layer applies some filter operations to the image to extract the features found in the image. In this way, a new image is obtained by extracting essential features from the image. The activation layer is the layer where the activation function is applied. Activation functions determine how neuron outputs must undergo a change. There are various functions that can be used in the activation layer. Some of these are step, sigmoid, tanh, ReLU, softmax, softplus, swish, and hyperbolic functions.

The pooling layer is generally preferred to reduce the size of the images. The size reduction process causes some information to be lost and, therefore to a decrease in performance. However, it prevents the model from memorizing and reduces the computational load. There are multiple processes in the pooling layer. These are Minimum pooling, maximum pooling, and average pooling. Among these methods, it is observed that the maximum pooling process is preferred more frequently. Full link layers are usually found at the end of the convolutional neural network architecture and are used to optimize results such as class prediction. In this layer, each neuron is connected to the next neuron.

Table 1. Distribution of data set by classes

“Traffic Sign”	“Train + Test Data”
No Park Stopping	64
50 Km/h	51
30 Km/h	37
70 Km/h	26
Stop Give Way	57
Main Road	45
Stop	60
Red Light	55
Green Light	43
Park	48
80 Km/h	30
60 Km/h	35
Pedestrian Crossing	52
Station	57
40 Km/h	45
No Entry	54
Uneven Road	53
Risk of Ice	34
Bend to Right	44
School Crossing	48
Caution	59
No Turn Right	38
No Turn Left	36
Side Road	44
Level Crossing with Gate	37
Bend to Left	27

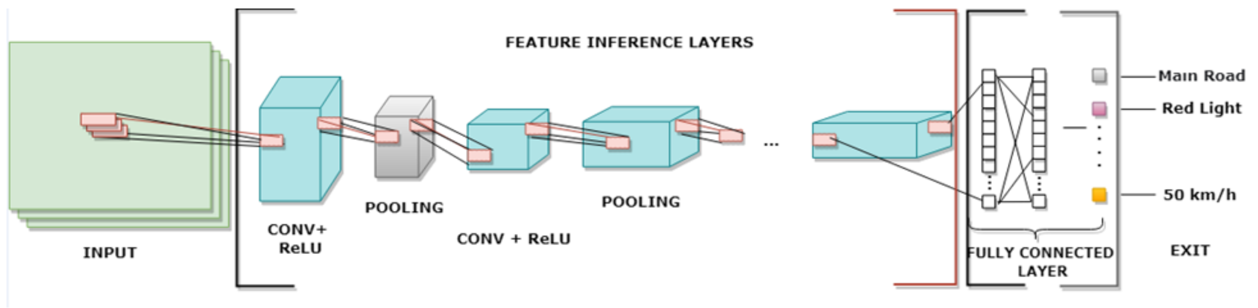


Figure 2. CNN Architecture

One of the important factors affecting the processing performance of convolutional neural network layers is hyper parameters. These parameters are; data size, batch size, dropout, epoch, learning coefficient, etc. It is aimed controlling the sufficient and insufficient learning status of the model by changing these parameters.

The reason for using the YOLO algorithm is that while the YOLO algorithm operates with the one stage method, the other discussed algorithms operate with the Two Stage Method. For example, unlike the prediction filter applied by the R-CNN algorithm to the image data, the YOLO algorithm tries to predict all objects by passing the picture through the neural network at once. In other words, it treats the detection process as a single regression problem. The most prominent feature that distinguishes the YOLOv5 model, which was chosen in line with these features, from different deep learning models is PyTorch-based development and GPU (Graphics Processor Unit) support. Since the separation of CUDA cores is more problematic and inefficient by image processing libraries such as OpenCV, it has been selected with YOLOv5 to ensure that it works optimally on the mission computer, for example, NVIDIA Jetson Nano. The human eye can see an average of 18-25 FPS (Narang et al. 2015). For this reason, it is aimed to improve the situations where the human eye is inadequate by choosing better hardware (Jetson Nano / 28 FPS) than the performance of the human eye.

Figure 3 shows the comparison of different YOLO models according to GPU speeds and IoU (intersection over union) metrics (Web Site 1). The IoU measures the intersection between 2 predictive values. It is used to measure how much the position of the predicted object coincides with the target position. When the graph in Figure 4 is examined according to these metrics, the GPU and IoU performances of the YOLOv5s, YOLOv5m, YOLOv5x models used for this study are seen. According to these results, it is seen that the YOLOv5m model gives better results than other models.

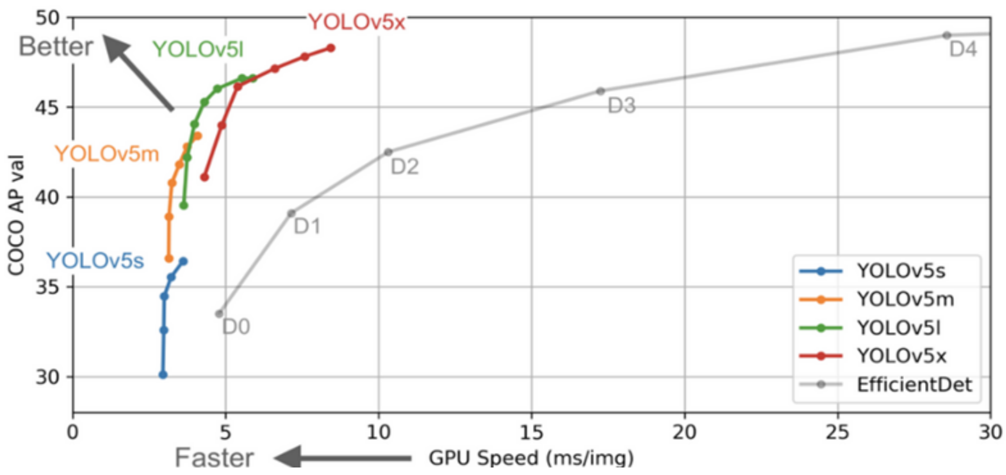


Figure 3. YOLO model comparisons according to GPU Performances and IoU averages (Web Site 1)

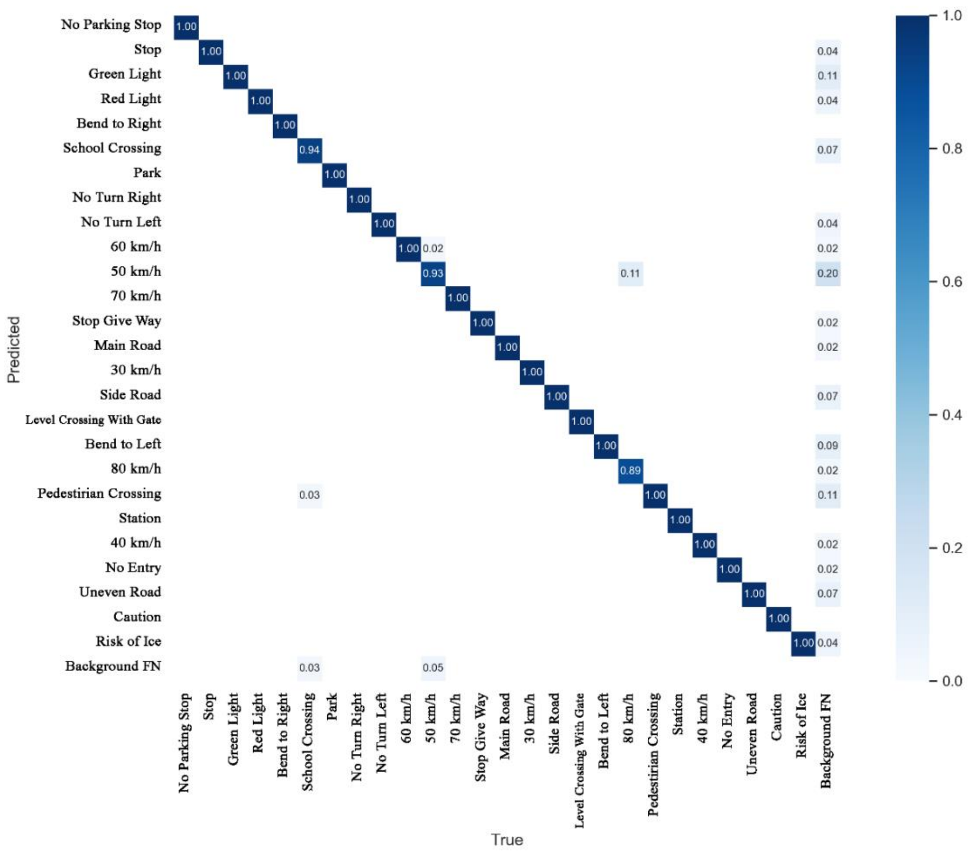
Deep learning models that provide learning produced outputs (F1 score, confusion matrix, accuracy value, etc.). According to these results, calculations are made on learning metrics, especially the confusion matrix and F1 score. Values are ranked according to their height status, and their comparisons are evaluated.

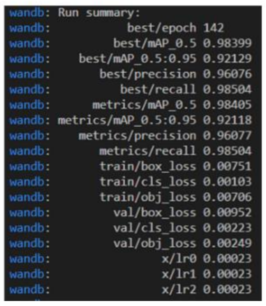
The model weight file is formed as a result of the learning. The weight file contains the architecture of the model, the learning weights, etc. contains information. TensorRT technology aims to improve the model weight file by NVIDIA company with five optimization items such as calibration, layers and tensor fusion, automatic core tuning, dynamic tensor memory, and multi-stream execution (parallel processing). The model weight file, which has evolved into a simpler and more efficient state for the hardware as a result of the optimization, is used for real-time tests. FPS results of the real-time tests are shown in Figure 11 in Appendix.

For real-time studies, operations are carried out on Jetson Nano hardware with TensorRT transformation of the model file. With the execution of the "Engine" extension TensorRT file, the model learning metrics are successful is expected to start the target recognition process. The tests are reported considering the high FPS value, which is one of the important criteria in the real-time test phase, and the extent to which the model has achieved the target during this period. LOGITECH C920 HD Pro is used as camera hardware in real-time studies. The camera used provides a 30 FPS video recording feature with clear and detailed 1080p image clarity. The camera's 78° field of view and HD auto light correction were key factors in real-time testing.

3. RESULTS AND DISCUSSION

When the output data of the YOLOv5m model, which provides learning, are examined, it is seen that a success rate of 99.07% is achieved in the confusion matrix for 150 epochs and 16 batch sizes. When Figure 4 is examined, it is calculated that the f1 score value of the training is 0.97, taking into account the precision and recall values. The training results are presented graphically in Figure 4.





```
wandb: Run summary:
wandb:              best/epoch 142
wandb:            best/mAP_0.5 0.98399
wandb:       best/mAP_0.5:0.95 0.92129
wandb:          best/precision 0.96076
wandb:             best/recall 0.98504
wandb:         metrics/mAP_0.5 0.98405
wandb: metrics/mAP_0.5:0.95 0.92118
wandb:       metrics/precision 0.96077
wandb:          metrics/recall 0.98504
wandb:          train/box_loss 0.00751
wandb:          train/cls_loss 0.00103
wandb:          train/obj_loss 0.00706
wandb:            val/box_loss 0.00952
wandb:            val/cls_loss 0.00223
wandb:            val/obj_loss 0.00249
wandb:                  x/lr0 0.00023
wandb:                  x/lr1 0.00023
wandb:                  x/lr2 0.00023
```

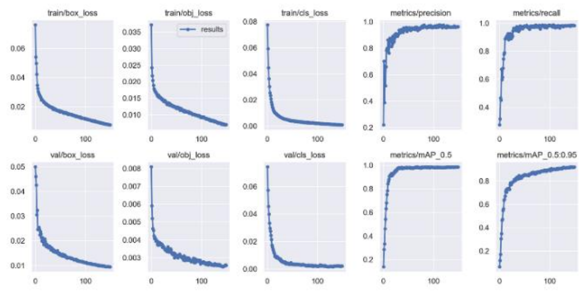
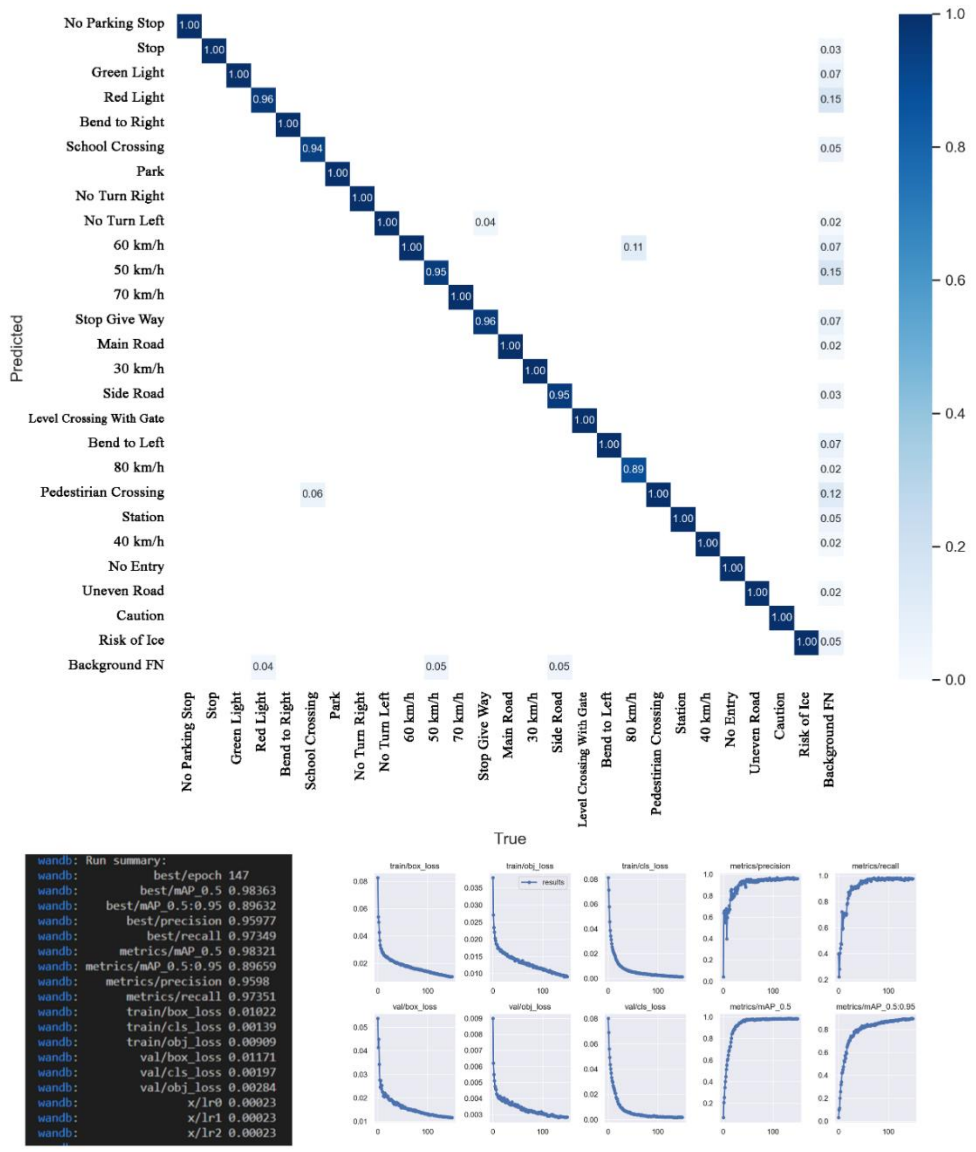


Figure 4. Confusion matrix, model results, and graphs for 150 epochs of the YOLOv5m model

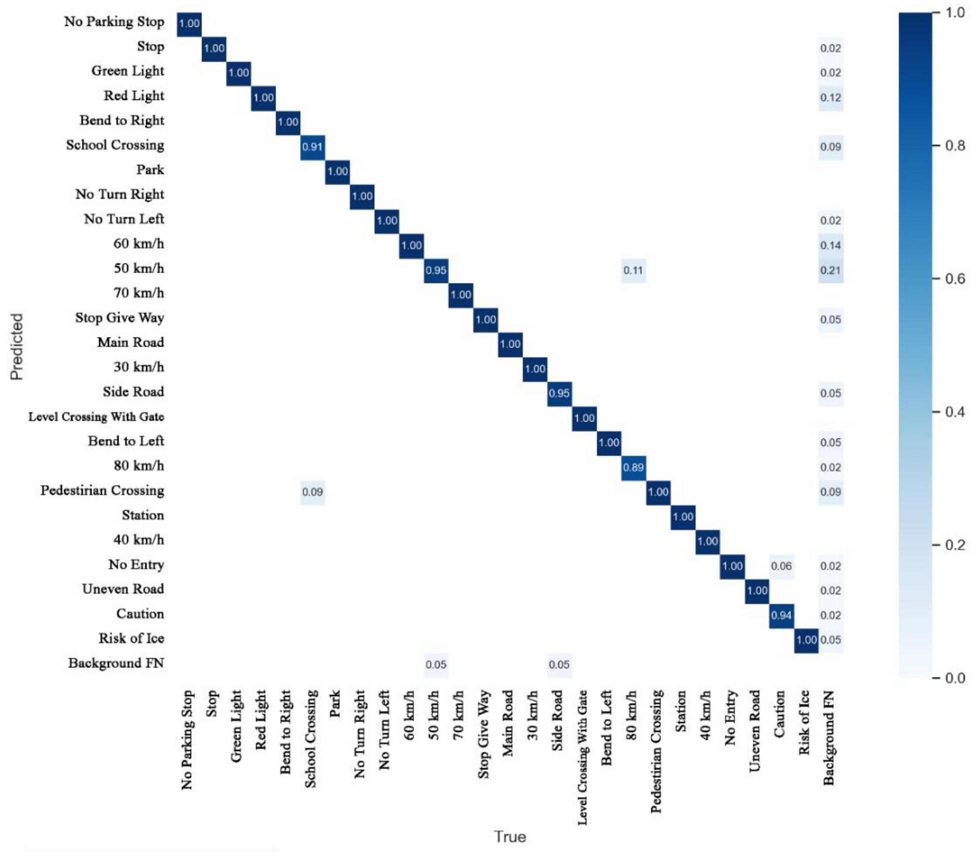
Compared to the YOLOv5m model, the YOLOv5s model, which has a less complex structure, shows success of 98.65% when the confusion matrix is analyzed as a result of the learning performed for 150 epochs and 16 batch "size" values. It is calculated that the training f1 score value is 0.95 as a result of calculating precision and recall values when examined in figure 5. Graphs related to the learning parameters of the training are shown in Figure 5.

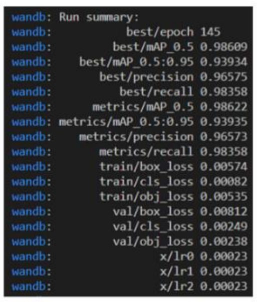


```
wandb: Run summary:
wandb:                best/epoch 147
wandb:              best/mAP_0.5 0.98363
wandb:         best/mAP_0.5:0.95 0.89632
wandb:            best/precision 0.95977
wandb:               best/recall 0.97349
wandb:           metrics/mAP_0.5 0.98321
wandb: metrics/mAP_0.5:0.95 0.89659
wandb:         metrics/precision 0.9598
wandb:            metrics/recall 0.97351
wandb:            train/box_loss 0.01022
wandb:            train/cls_loss 0.00139
wandb:            train/obj_loss 0.00909
wandb:              val/box_loss 0.01171
wandb:              val/cls_loss 0.00197
wandb:              val/obj_loss 0.00284
wandb:                    x/lr0 0.00023
wandb:                    x/lr1 0.00023
wandb:                    x/lr2 0.00023
```

Figure 5. Confusion matrix, model results, and graphs for 150 epochs of the YOLOv5s model

In the YOLOv5x model, which has a more complex structure than the other YOLO models used, a success rate of 98.61% was achieved when the confusion matrix formed as a result of the training for 150 epochs and 16 batch sizes was examined. When Figure 6 is examined, it is calculated that the training f1 score value is 0.97 as a result of processing with precision and recall values. Graphics showing the training status of the train are shown in Figure 6.





```
wandb: Run summary:
wandb:              best/epoch 145
wandb:            best/mAP_0.5 0.98609
wandb:       best/mAP_0.5:0.95 0.93934
wandb:          best/precision 0.96575
wandb:             best/recall 0.98358
wandb:         metrics/mAP_0.5 0.98622
wandb: metrics/mAP_0.5:0.95 0.93935
wandb:       metrics/precision 0.96573
wandb:          metrics/recall 0.98358
wandb:          train/box_loss 0.00574
wandb:          train/cls_loss 0.00082
wandb:          train/obj_loss 0.00535
wandb:            val/box_loss 0.00812
wandb:            val/cls_loss 0.00249
wandb:            val/obj_loss 0.00238
wandb:                  x/lr0 0.00023
wandb:                  x/lr1 0.00023
wandb:                  x/lr2 0.00023
```

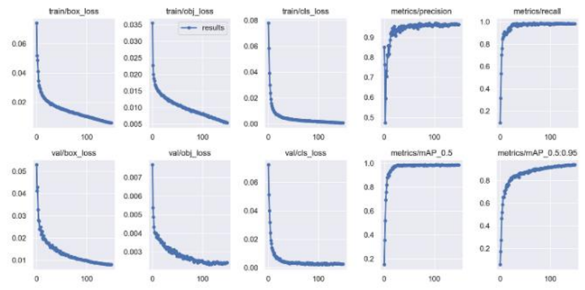


Figure 6. Confusion matrix, model results, and graphs for 150 epochs of the YOLOv5x model

In line with all these results, it has been determined that the YOLOv5m model is the model that gives the most optimum results when the test results and detection times are examined. Experimental studies were carried out in Microsoft Visual Studio Code environment on Intel (R) Core (TM) i7-10700 CPU 2.90GHz, 128 GB RAM, NVIDIA GeForce RTX 3080 Graphics card.

Examining the training results on the YOLOv5s, YOLOv5m, and YOLOv5x models, it was seen that the f1 score and confusion matrix results of the YOLOv5m model were higher than the others. When the YOLOv5s model results are compared with the YOLOv5m model, it is seen that the f1 score of the YOLOv5s model is lower by 0.02. However, YOLOv5s was preferred as the learning model in the real-time study. Considering the weight file size, training and detection time, and FPS (Frame Per Second) of the YOLOv5s model, it is seen that it has high-performance results compared to the YOLOv5m model. Although the learning results of the YOLOv5m model are higher, it has been decided that the YOLOv5s model is more suitable for real-time work. The YOLOv5s model has been tested in real-time on Jetson Nano and Raspberry Pi4B models. When tests were made for different image size values, it was seen that the best FPS values with acceptable accuracy were provided for 320x320 image size on both cards. As a result, for 320x320 image size, a maximum of 8.2 FPS was obtained when using Jetson Nano and TensorRT conversion, and a maximum of 28.1 FPS after TensorRT conversion. As a result of the studies on Raspberry Pi4B 4GB and Raspberry Pi4B 8GB models, it has been seen that the 4GB model gives 0.5 FPS, and the 8GB model gives a maximum of 2.1 FPS for 320 image size. When the real-time test phases were completed, processing on the Jetson Nano mission computer was determined to be suitable in terms of model efficiency, detection time, and accuracy. The steps of the implementation and decision process are shown in the flow chart in Figure 7.

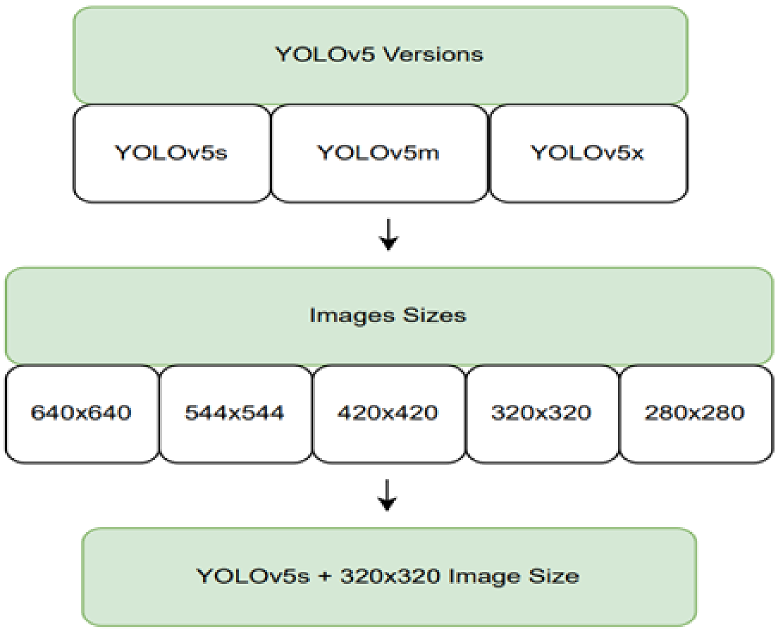


Figure 7. Models and Tested image size values

In Table 2, a comparison of the prominent studies on the classification of traffic markers with deep learning algorithms and the presented study is given. As can be seen from the table, using different CNN models, Lim et al. obtained an F1 score of 0.97. Çetin et al. developed models with a sensitivity of 0.97, and Paladiz et al. developed a sensitivity of 0.83. However, in the presented study, a sensitivity of 0.98 was achieved by using the YOLOv5s model, unlike the previous studies.

Table 2. Comparison of existing studies and presented study

	YOLOv5s Model	YOLOv5m Model	YOLOv5x Model	F1 Score	Accuracy	Raspberry Pİ4B FPS	Jetson Nano FPS
Lim et al.	×	×	×	0.97	×	×	×
Çetin et al.	×	×	×	×	0.97	×	×
Palandız et al.	×	×	×	×	0.83	×	×
This paper	✓	✓	✓	0.97 (YOLOv5m)	0.99 (YOLOv5m)	1.5	23.3
				0.95 (YOLOv5s)	0.98 (YOLOv5s)	2.1	28.2

The outputs showing the training status of the model as a result of the determination made on the test class in the training dataset are presented in Figure 8. The "Stop" class of the model is 98%, the "Stop Parking Forbidden" class is 99%, the "Sharp Corner to the Left" class is 98%, the "Rough Road" class is 98%, the "Give Way" class is 98%, the "No Entry" class is 98%, It is seen that it detects the "Stop" class with an accuracy of 99%, the "Hidden Icing" class with an accuracy of 98%, the "Pedestrian Crossing" class with 98%, and the "Controlled Railway" class with an accuracy of 98%. One of the important criteria not included in other studies is the Frame Per Second (FPS) value. When the model obtained as a result of the training is used with Raspberry Pi4B, it reaches 2.1 FPS, while 28.2 FPS values can be reached with Jetson Nano.

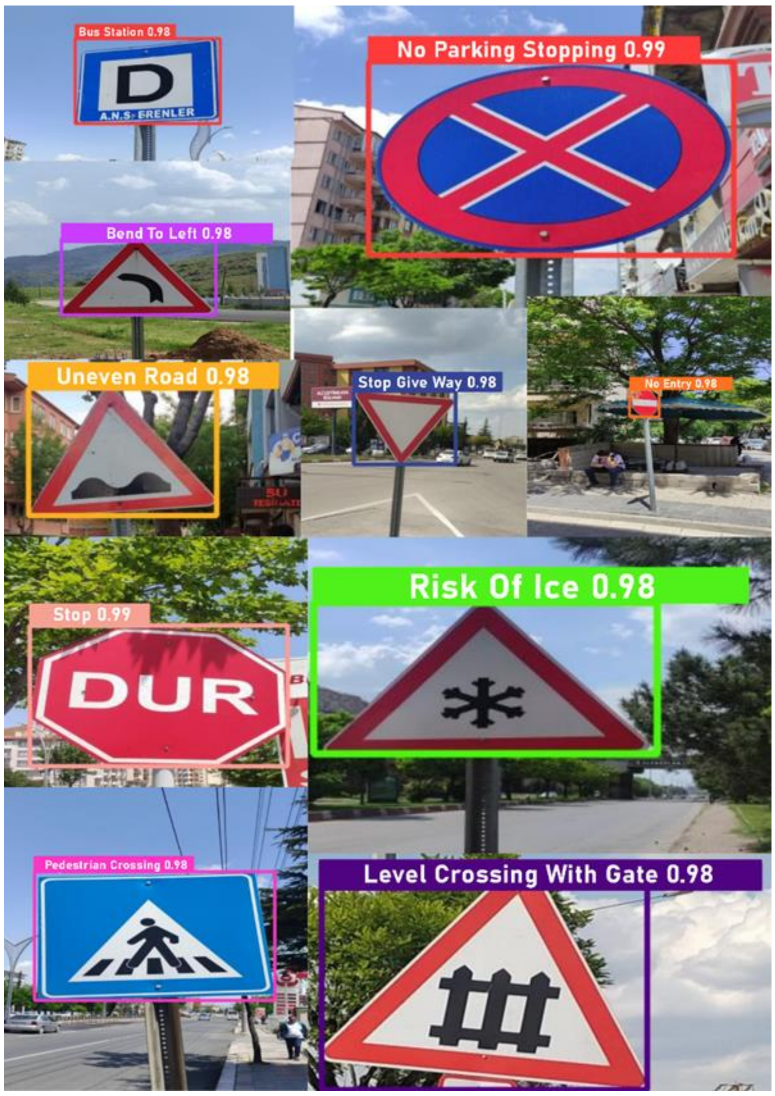


Figure 8. Test results showing the learning status of the model

4. CONCLUSION

In this study, using different YOLOv5 deep learning models, a data set containing 26 different traffic sign images were trained. There are 1249 image data in total in the prepared data set *(Repistority: https://drive.google.com/file/d/1oHkNKT9KZrLrtGcCj7pnjbTd5NkJJXzJ/view?usp=sharing)*. 80% of this data (1000 pieces of data) was used for model training, and the remaining 20% (249 pieces of data) was used in the testing phase. The best results were obtained in the YOLOv5m model in computer studies. Accordingly, the f1 score of the developed model is 97%. However, when real-time testing was performed using YOLOv5m, 1.5 and 23.3 FPS values were obtained on Raspberry PI4B and Jetson Nano, respectively. On the other hand, the f1 score of the YOLOv5s model was calculated as 95%. However, when real-time testing was performed with YOLOv5s, 2.1 and 28.2 FPS values were captured on Raspberry PI4B and Jetson Nano, respectively. For this reason, although the f1 score of the YOLOv5m model is higher, it is more appropriate to prefer the YOLOv5s model in real-time study. It is thought that these success rates obtained with computer technologies will help the traffic movements of autonomous vehicles in detecting traffic sign images with the help of deep learning models.

In addition to this study, adding new images with different backgrounds and angles to the classes in the dataset will help to increase the training and detection rate. Testing the models in real-time on different hardware will be more efficient in terms of the applicability of the study and testing the model efficiency.

5. CONFLICT OF INTEREST

Author approves that to the best of his knowledge, there is not any conflict of interest or common interest with an institution/organization or a person that may affect the review process of the paper.

6. AUTHOR CONTRIBUTION

Faruk Emre Aysal and Enes Cengiz determining the concept and design process of the research and research management, Kasim Yildirim data collection and analysis Faruk Emre Aysal and Kasim Yildirim data analysis and interpretation of results.

7. REFERENCES

Bayram F., Derin öğrenme tabanlı otomatik plaka tanıma. Politeknik Dergisi 23(4), 955-960, 2020.

Çetin E., Ortataş F., Elektrikli ve Otonom Araçlarda Makine Öğrenmesi Kullanarak Trafik Levhaları Tanıma ve Simülasyon Uygulaması. El-Cezeri 8(3), 1081-1092, 2021.

Dorokhin S., Artemov A., Likhachev D., Novikov, A., Starkov E., Traffic simulation: an analytical review. IOP Conference Series: Materials Science and Engineering, 918, 2020.

Eraqi H. M., Moustafa M. N., Honer J., End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA., 2017.

Fagnant D. J., Kockelman K., Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. Transportation Research Part A: Policy and Practice 77, 167-181, 2015.

Glikson E., Woolley A. W., Human Trust in Artificial Intelligence: Review of Empirical Research. Academy of Management Annals 14(2), 627-660, 2020.

Gopalakrishnan S., A Public Health Perspective of Road Traffic Accidents. Journal of Family Medicine Primary Care 1(2), 144-150, 2012.

Guo Y., Liu Y., Oerlemans A., Lao S., Wu S., Lew M., Deep learning for visual understanding: A review. Neurocomputing 187, 27-48. 2016.

Haenlein M., Kaplan A., A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. California Management Review 61(4), 5-14, 2019.

Hasan R. I., Yusuf S. M., Alzubaidi L., Review of the State of the Art of Deep Learning for Plant Diseases: A Broad Analysis and Discussion, Plants 9(10), 1302, 2020.

Huval B., Wang T., Tandon S., Kiske J., Song W., Pazhayampallil J., Ng A. Y., An Empirical Evaluation of Deep Learning on Highway Driving. Computer Science, 2015.

Jain N. K., Saini R. K., Mittal P., A Review on Traffic Monitoring System Techniques, Soft Computing: Theories and Applications, 569-577, 2018.

Kour A., Yadav V. K., Maheshwari V., Prashar D., A Review on Image Processing. International Journal of Electronics Communication and Computer Engineering 4(1), 270-275, 2013.

Kulkarni R., Dhavalikar S., Bangar S., Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning, Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune/India, August 16-18, 2018.

Kumar G., Bhatia P. K., A Detailed Review of Feature Extraction in Image Processing Systems, Fourth International Conference on Advanced Computing & Communication Technologies, Rohtak/India, February 8-9, 2014, pp: 5-12.

Lim K., Hong Y., Choi Y., Byun H., Real-time traffic sign recognition based on a general purpose GPU and deep-learning. PLOS ONE, 2017.

Miglani A., Kumar N., Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges. Vehicular Communications 20, 100184, 2019.

Narang P; Agarwal A, Sanu A. S., Detecting subtle intraocular movements: Enhanced frames per second recording (slow motion) using smartphones. Journal of Cataract & Refractive Surgery 41(6), 1321-1323, 2015.

Palandız T., Bayrakçı H. C., Yapay Zekâ Kullanılarak Trafik İşaret Levhalarının Sınıflandırılması: Denizli İl Merkezi İçin Örnek Bir Uygulama. International Journal of 3D Printing Technologies and Digital Industry 5(3), 645-653, 2021.

Sampedro C., Ramos A. R., Campoy P., A Review of Deep Learning Methods and Applications for Unmanned Aerial Vehicles, Journal of Sensors, 2017.

Sarkar S. B., Mohan B. C., Review on Autonomous Vehicle Challenges. First International Conference on Artificial Intelligence and Cognitive Computing, 593-603, 2018.

Satria R., Castro M., GIS Tools for Analyzing Accidents and Road Design: A Review. Transportation Research Procedia 18, 242-247, 2016.

Shrestha A., Mahmood A., Review of Deep Learning Algorithms and Architectures, IEEE Access 7, 53040-53065, 2019.

Swaminathan V., Arora S., Bansal R., Rajalakshmi R., Autonomous Driving System with Road Sign Recognition using Convolutional Neural Networks, International Conference on Computational Intelligence in Data Science (ICCIDS), Chennai/India, February 21-23, 2019.

Web Site 1: "YOLOv5 New Version- Improvements and Evaluation", Access date: 20/05/2022, https://blog.roboflow.com/YOLOv5-improvements-and-evaluation/

Wiley V., Lucas T., Computer Vision and Image Processing: A Paper Review. International Journal of Artificial Intelegence Research 2(1), 28-36, 2018.

Zhou L., Zhang C., Liu F., Qiu Z., He Y., Application of Deep Learning in Food: A Review, Comprehensive Reviews in Food Science and Food Safety18(6), 1793-1811, 2019.

Appendix

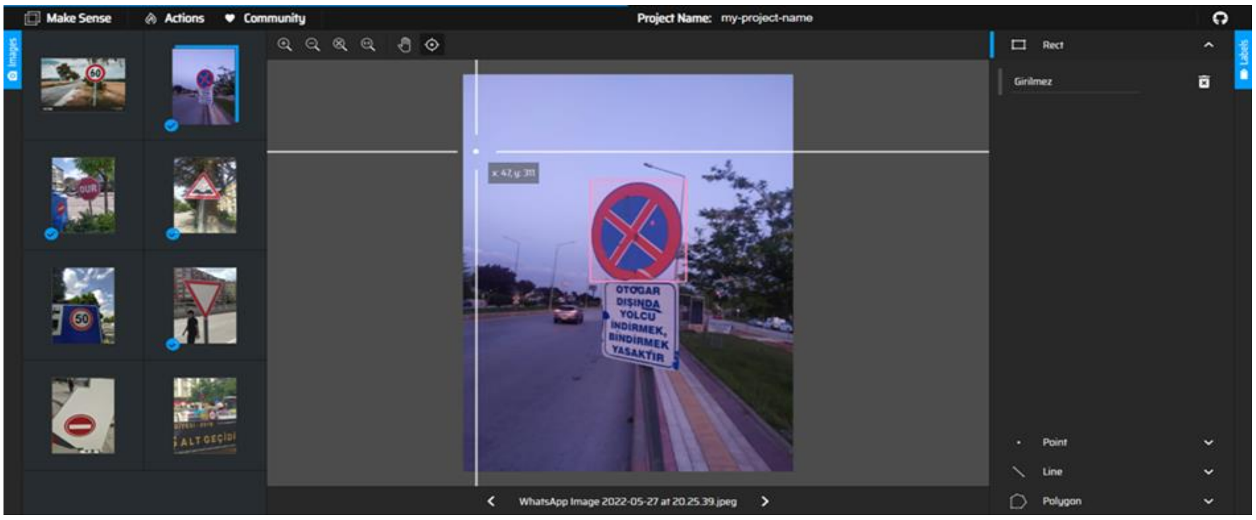
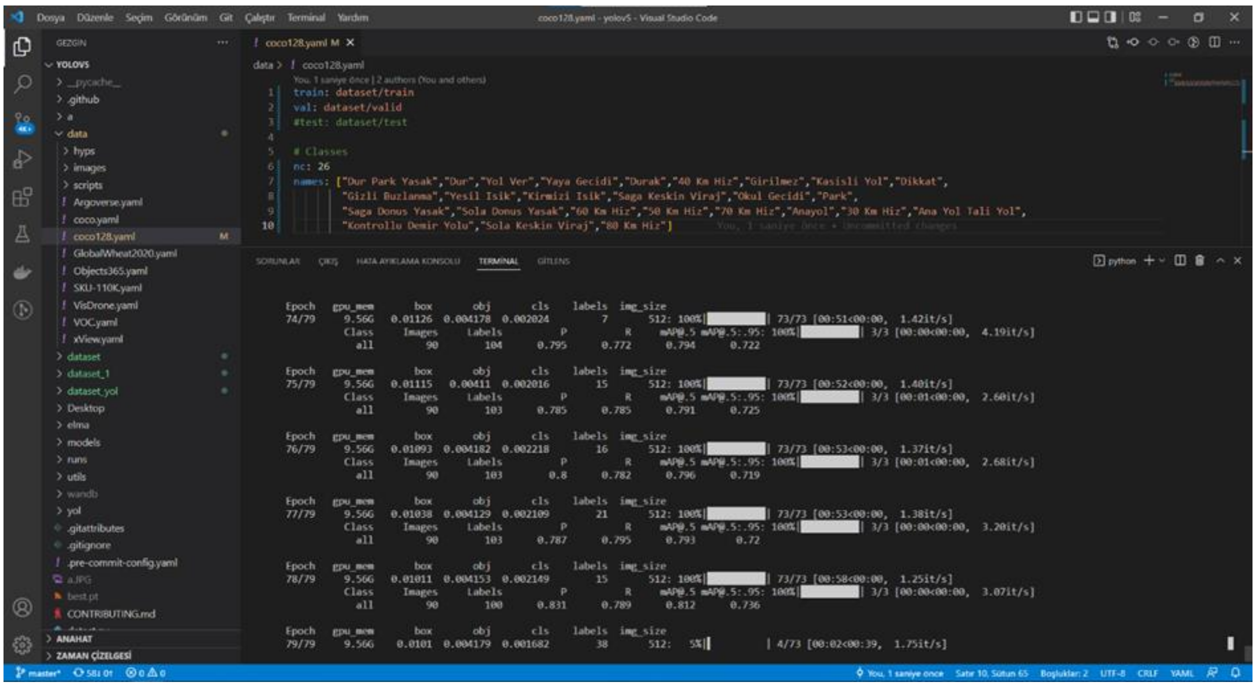

Make Sense
Actions
Community
Project Name: my-project-name
Images
Labels
Rect
Girilmez
Point
Line
Polygon
WhatsApp Image 2022-05-27 at 20.25.39.jpeg

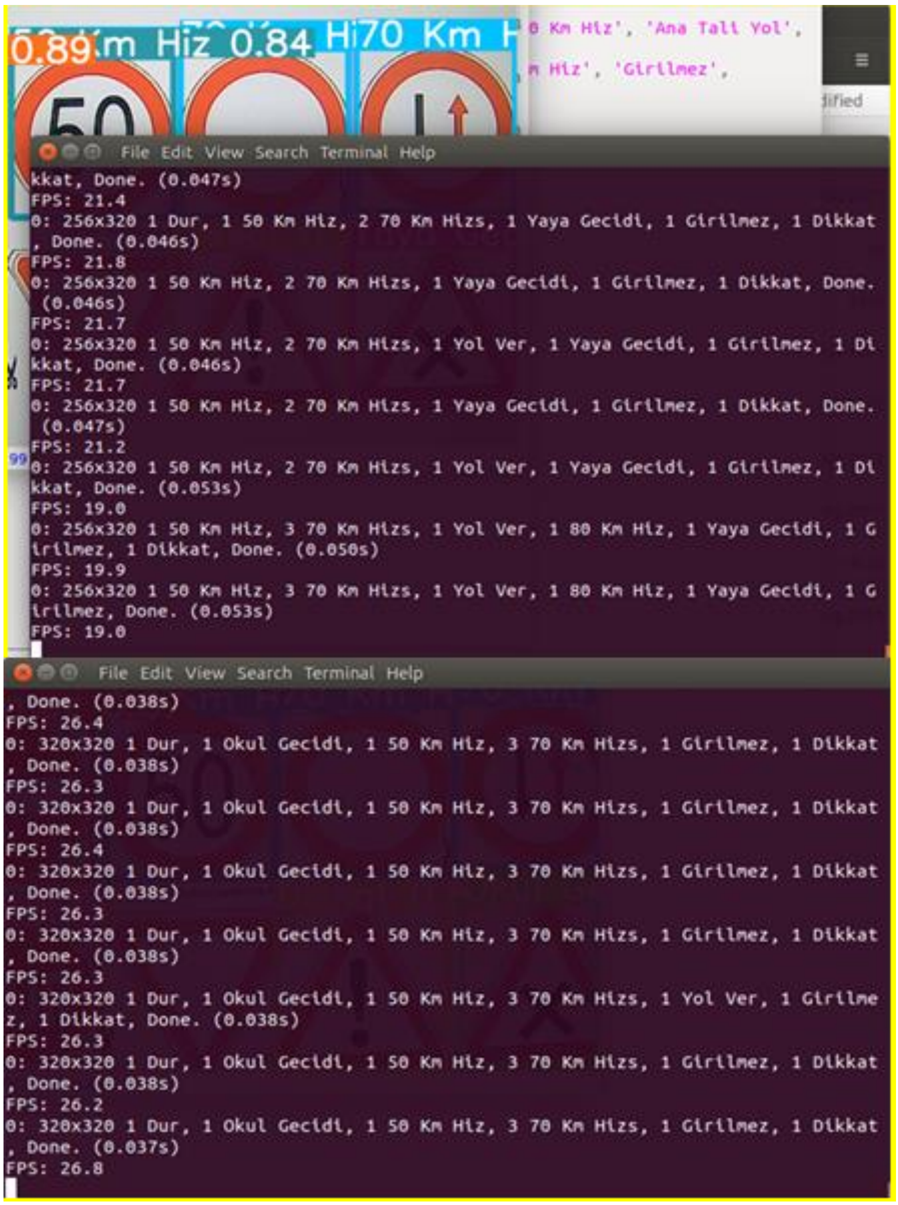
Figure 9. Labelling example


Dosya Düzenle Seçim Görünüm Git Çalıştır Terminal Yardım
coco128.yaml - yolov5 - Visual Studio Code
GEZGİN
YOLOV5
SORUNLAR ÇIKIŞ HATA AYIKLAMA KONSOLU TERMINAL GITLENS
ANAHAT
ZAMAN ÇİZELGESİ

```yaml
train: dataset/train
val: dataset/valid
#test: dataset/test

# Classes
nc: 26
names: ["Dur Park Yasak","Dur","Yol Ver","Yaya Gecidi","Durak","40 Km Hiz","Girilmez","Kasisli Yol","Dikkat",
        "Gizli Buzlanma","Yesil Isik","Kirmizi Isik","Saga Keskin Viraj","Okul Gecidi","Park",
        "Saga Donus Yasak","Sola Donus Yasak","60 Km Hiz","50 Km Hiz","70 Km Hiz","Anayol","30 Km Hiz","Ana Yol Tali Yol",
        "Kontrollu Demir Yolu","Sola Keskin Viraj","80 Km Hiz"]
```

```
     Epoch   gpu_mem       box       obj       cls    labels  img_size
     74/79     9.56G   0.01126  0.004178  0.002024         7       512: 100%|██████████| 73/73 [00:51<00:00,  1.42it/s]
               Class    Images    Labels         P         R     mAP@.5 mAP@.5:.95: 100%|██████████| 3/3 [00:00<00:00,  4.19it/s]
                 all        90       104     0.795     0.772     0.794     0.722

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     75/79     9.56G   0.01115   0.00411  0.002016        15       512: 100%|██████████| 73/73 [00:52<00:00,  1.40it/s]
               Class    Images    Labels         P         R     mAP@.5 mAP@.5:.95: 100%|██████████| 3/3 [00:01<00:00,  2.60it/s]
                 all        90       103     0.785     0.785     0.791     0.725

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     76/79     9.56G   0.01093  0.004182  0.002218        16       512: 100%|██████████| 73/73 [00:53<00:00,  1.37it/s]
               Class    Images    Labels         P         R     mAP@.5 mAP@.5:.95: 100%|██████████| 3/3 [00:01<00:00,  2.68it/s]
                 all        90       103       0.8     0.782     0.796     0.719

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     77/79     9.56G   0.01038  0.004129  0.002109        21       512: 100%|██████████| 73/73 [00:53<00:00,  1.38it/s]
               Class    Images    Labels         P         R     mAP@.5 mAP@.5:.95: 100%|██████████| 3/3 [00:00<00:00,  3.20it/s]
                 all        90       103     0.787     0.795     0.793      0.72

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     78/79     9.56G   0.01011  0.004153  0.002149        15       512: 100%|██████████| 73/73 [00:58<00:00,  1.25it/s]
               Class    Images    Labels         P         R     mAP@.5 mAP@.5:.95: 100%|██████████| 3/3 [00:00<00:00,  3.07it/s]
                 all        90       100     0.831     0.789     0.812     0.736

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     79/79     9.56G    0.0101  0.004179  0.001682        38       512:   5%|▌         | 4/73 [00:02<00:39,  1.75it/s]
```

Figure 10. YOLOv5 model training screenshot


0.89
m Hiz 0.84
Hi70 Km H
0 Km Hiz', 'Ana Tali Yol',
m Hiz', 'Girilmez',
File Edit View Search Terminal Help

```
kkat, Done. (0.047s)
FPS: 21.4
0: 256x320 1 Dur, 1 50 Km Hiz, 2 70 Km Hizs, 1 Yaya Gecidi, 1 Girilmez, 1 Dikkat
, Done. (0.046s)
FPS: 21.8
0: 256x320 1 50 Km Hiz, 2 70 Km Hizs, 1 Yaya Gecidi, 1 Girilmez, 1 Dikkat, Done.
 (0.046s)
FPS: 21.7
0: 256x320 1 50 Km Hiz, 2 70 Km Hizs, 1 Yol Ver, 1 Yaya Gecidi, 1 Girilmez, 1 Di
kkat, Done. (0.046s)
FPS: 21.7
0: 256x320 1 50 Km Hiz, 2 70 Km Hizs, 1 Yaya Gecidi, 1 Girilmez, 1 Dikkat, Done.
 (0.047s)
FPS: 21.2
0: 256x320 1 50 Km Hiz, 2 70 Km Hizs, 1 Yol Ver, 1 Yaya Gecidi, 1 Girilmez, 1 Di
kkat, Done. (0.053s)
FPS: 19.0
0: 256x320 1 50 Km Hiz, 3 70 Km Hizs, 1 Yol Ver, 1 80 Km Hiz, 1 Yaya Gecidi, 1 G
irilmez, 1 Dikkat, Done. (0.050s)
FPS: 19.9
0: 256x320 1 50 Km Hiz, 3 70 Km Hizs, 1 Yol Ver, 1 80 Km Hiz, 1 Yaya Gecidi, 1 G
irilmez, Done. (0.053s)
FPS: 19.0
```

```
, Done. (0.038s)
FPS: 26.4
0: 320x320 1 Dur, 1 Okul Gecidi, 1 50 Km Hiz, 3 70 Km Hizs, 1 Girilmez, 1 Dikkat
, Done. (0.038s)
FPS: 26.3
0: 320x320 1 Dur, 1 Okul Gecidi, 1 50 Km Hiz, 3 70 Km Hizs, 1 Girilmez, 1 Dikkat
, Done. (0.038s)
FPS: 26.4
0: 320x320 1 Dur, 1 Okul Gecidi, 1 50 Km Hiz, 3 70 Km Hizs, 1 Girilmez, 1 Dikkat
, Done. (0.038s)
FPS: 26.3
0: 320x320 1 Dur, 1 Okul Gecidi, 1 50 Km Hiz, 3 70 Km Hizs, 1 Girilmez, 1 Dikkat
, Done. (0.038s)
FPS: 26.3
0: 320x320 1 Dur, 1 Okul Gecidi, 1 50 Km Hiz, 3 70 Km Hizs, 1 Yol Ver, 1 Girilme
z, 1 Dikkat, Done. (0.038s)
FPS: 26.3
0: 320x320 1 Dur, 1 Okul Gecidi, 1 50 Km Hiz, 3 70 Km Hizs, 1 Girilmez, 1 Dikkat
, Done. (0.038s)
FPS: 26.2
0: 320x320 1 Dur, 1 Okul Gecidi, 1 50 Km Hiz, 3 70 Km Hizs, 1 Girilmez, 1 Dikkat
, Done. (0.037s)
FPS: 26.8
```

Figure 11. YOLOv5s real-time FPS tests