A Robust Pedestrian Detection Approach for Autonomous Vehicles

Bahareh Ghari  
Department of Computer Engineering  
University of Guilan  
Rasht, Iran  
baharehghari@msc.guilan.ac.ir

Ali Tourani  
Interdisciplinary Centre for Security, Reliability, and Trust (SnT)  
University of Luxembourg  
Luxembourg, Luxembourg  
ali.tourani@uni.lu

Asadollah Shahbahrami  
Department of Computer Engineering  
University of Guilan  
Rasht, Iran  
shahbahrami@guilan.ac.ir

Abstract—Nowadays, utilizing Advanced Driver-Assistance Systems (ADAS) has absorbed a huge interest as a potential solution for reducing road traffic issues. Despite recent technological advances in such systems, there are still many inquiries that need to be overcome. For instance, ADAS requires accurate and real-time detection of pedestrians in various driving scenarios. To solve the mentioned problem, this paper aims to fine-tune the YOLOv5s model for handling pedestrian detection challenges on the real-world instances of Caltech pedestrian dataset. We also introduce a developed toolbox for preparing training and test data and annotations of Caltech pedestrian dataset into the format recognizable by YOLOv5. Experimental results on the Caltech pedestrian dataset have verified that the mean Average Precision (mAP) of our fine-tuned model for pedestrian detection task is equal to 91 percent when performing at the highest rate of 70 FPS. Moreover, our proposed approach can outperform other existing methodologies in terms of accuracy and speed.

Keywords—pedestrian detection; deep learning; object detection; autonomous vehicles;

I. INTRODUCTION

As a well-known issue, many people get injured or lose their lives in traffic accidents every day. There are several causes for these accidents, such as driver tiredness, speeding, and drunk driving. Therefore, autonomous systems have been utilizing in modern vehicles to provide reliable and safe driving and decrease the risk of accidents [1]. These systems use various sensors to analyze the surroundings in real-time and provide automatic decision-making while driving. In this regard, pedestrian detection is a vital component in automotive safety and Advanced Driver-Assistance Systems (ADAS) domains, as it can guarantee safe vehicle passage on the roads [2].

The process of pedestrian detection in autonomous vehicles can lie under object detection tasks. Object detection has recently matured and overwhelmed any other hand-craft or classic detection solutions [3]. Currently, there is a vast number of object detection algorithms, including Histogram of Oriented Gradients (HOG) [4], Scale-invariant Feature Transform (SIFT) [5], Speeded up Robust Features (SURF) [6], Haar feature-based methods [7], and Artificial Neural Networks (ANNs) [8]. Object detection contributes to the Convolutional Neural Networks (CNNs) [9] in the computer vision domain, providing robust outcomes compared to the classic image processing approaches [10]. In this regard, many deep learning-based applications for object detection are divided into either

"one-stage" - e.g., You Only Look Once (YOLO) [11] - or “two-stage” - e.g., Faster Region-based CNN (Faster R-CNN) [12]- methodologies. While in the two-stage methods a set of samples is used to find the candidate objects, one-stage techniques predict the locations as a regression problem.

Among a wide range of algorithms, YOLO is a one-stage object detector that utilizes a real-time end-to-end approach to predict bounding boxes and class labels at once [11]. At the time of writing this paper, many different versions and variations have been introduced. YOLOv2 has improved the performance of the first version, including low recall and small object detection, by using batch normalization on convolutional layers, improved classifier, and anchor boxes [13]. As the fast approach introduced in YOLOv2 decreased the accuracy, YOLOv3 appeared with a robust backbone, allowing it to detect features at three scales [14]. Although later versions of YOLO are not considered official, their developers have changed the architecture of YOLOv3 to improve performance and accuracy. In YOLOv4 [15], the primary focus is on enhancing the object detection stage, resulting in ten and twelve percent gains in accuracy and performance, respectively. In 2021, the fifth version was introduced as compound-scaled object detection models trained on the COCO dataset [16]. YOLOv5 [17] is implemented with the aid of the Ultralytics PyTorch framework and Python programming language, making it a super-fast methodology to train.
In this paper, a pedestrian detection application based on YOLOv5 is introduced. The main contributions of the paper are described below:

- Fine-tuning of YOLOv5 for handling the pedestrian detection challenges,
- Implementing a toolbox for preparing Caltech pedestrian dataset training and test data and annotations into proper YOLOv5 format.

The rest of the paper is organized as follows: Section II reviews recent pedestrian detection approaches. In Section III, we discuss our proposed method and contributions for pedestrian detection in detail. Section IV presents the experimental results and discussions, and finally, the paper concludes in Section V.

II. RELATED WORKS

Many recent publications have recommended deep learning-based solutions for pedestrian detection. Most of these researches have focused on the overall architecture of the detector module for a reliable pedestrian detection and tracking. For instance, Boyuan et al. [18] have proposed a detection model that combines a new type of Spatial Pyramid Pooling (SPP) network and K-means clustering algorithm with YOLOv4 model for easier feature extraction. In [19], DeepParts is proposed, consisting of forty-five fine-tuned part detectors to tackle the occlusion challenge. Each part detector can cover the entire body of pedestrians at different positions and scales with partial occlusion. Similarly, Noh et al. [20] adopted a set of part-based detectors learned and integrated into single-stage networks to solve the occlusion problem. For domain adaptation, Zeng et al. [21] have proposed a new approach to automatically learn domain invariant features in video surveillance with no manual labels. In [22], weighted combination layer and depth-wise separable convolution were combined with YOLO to improve the accuracy of detecting pedestrians in hazy weather.

Moreover, based on the recent advances in the anchor-free based detection networks, Center and Scale Prediction (CSP) [23] is proposed to detect the center and scale of pedestrians with a fixed aspect ratio. In [24], Recurrent Rolling Convolution (RRC) architecture is introduced to gradually aggregate relevant contextual information among the multi-scale feature maps. Based on the channel-wise attention mechanism, Zhang et al. [25] added an attention network to the Faster R-CNN architecture to handle different occlusion patterns in pedestrian detection. Some other studies addressed the role of data and tuning in pedestrian detection applications. For instance, the Boosting-like Deep Learning (BDL) framework [26] is used to prevent the over-fitting problem while training by reducing the average miss rate of the detector. Additionally, and as a baseline, Tumas et al. [27] introduced a dataset for detecting pedestrians in challenging weather conditions.

Unlike the mentioned studies, our method fine-tunes the robust architecture of YOLOv5 object detector for the pedestrian detection task. The fundamental goal of this paper is to provide accurate pedestrian detection in different scenarios, such as profound environmental changes, while keeping the architecture simple and fast. We also aim to provide detection in real-time so that our approach can be used in driverless vehicles.

III. PROPOSED METHOD

Due to the splendid features introduced in YOLOv5, including a lower training time, and mosaic data augmentation, we have utilized this version in our approach.

A. Architecture

The network structure of YOLOv5 model consists of three primary sections, as shown in Fig. 1. Accordingly, the backbone is a CNN that aggregates image features at different granularities. The Focus layer located at the beginning of the backbone. This layer divides the input image data into four pieces. The four pieces are generated by the reduction of width and height to $\frac{1}{2}$ and integrates it into the channel dimension. YOLOv5 adopts Cross-Stage Partial Networks (CSPNets) [28] as its backbone to formulate image features. The CSPNet addresses repetitious gradient problems in deeper CNNs, resulting in fewer parameters and Floating-point Operations-per-Second (FLOPS) of the model, improving the inference speed and and reducing the model size. The network also adds an SPPF block [29] after the CSP to remove the fixed-size input image constraint. The SPPF block is used to enhance the feature extraction ability of the backbone. The Neck of model is a series of layers to combine image features to pass them forward for the detection stage. It employs a Path Aggregation Network (PANet) [30] to boost the information flow process. The Head of the model is mainly used to apply feature anchor boxes and generated the final detection vectors. YOLOv5 implements the same YOLOv3 (anchor based) head for prediction. In addition, YOLOv5 including four models ranging from small to large according to the memory storage size (parameters): YOLOv5s (the smallest, which we have used in this paper), YOLOv5m (medium), YOLOv1 (large), and YOLOv5x (extra-large). All four models were trained on the MS COCO [16] training dataset.

B. Dataset

We have utilized the Caltech pedestrian dataset [31] in this work, which contains more than ten hours of real-world videos. The resolution of the videos is 640×480 and the total number of frames exceeds 250,000. Also, challenges in the Caltech pedestrian dataset such as variation in lighting conditions, pedestrian size, and occlusion, making this dataset a proper choice for the training of pedestrian detectors. Fig. 2, shows some instances of the dataset.

C. Data preparation for YOLOv5

As YOLO does not recognize the Caltech’s training data and annotations, we needed to implement a conversion toolbox for it. The mentioned toolbox contains different tools to prepare proper feed for YOLO architecture and is publicly accessible in
a GitHub repository \footnote{https://github.com/GuilanITS/Caltech-Pedestrian-YOLO}. Using our toolbox, we processed Caltech pedestrian dataset training .seq files and generated squared 640 × 640 .png images. Additionally, the .vbb annotation files are converted into .txt files according to the YOLOv5 standard.

D. Training Process

We picked the most miniature version of the YOLOv5 framework for the experiment, titled YOLOv5s. After fine-tuning the architecture of YOLOv5s based on the application requirements, the network has been trained on 3,000 images of the dataset in 600 epochs with a learning rate of 0.01, a momentum of 0.937 and a batch size of 8. The number of images in the validation and test sets was 280 and 1006, respectively. The total number of pedestrian instances in train, valid and test images are 5,577, 4,26, and 1,993, respectively. Also the total number of parameters in the network was 7,012,822, shaped into 213 layers. Pre-trained weights on COCO dataset are used to initialize the detection model. Among SGD, Adam and AdamW optimization functions implemented in YOLOv5, we chose the Adam algorithm for the optimization of parameters. With this settings, 600 epochs completed in 10.902 hours. Moreover, YOLOv5 can easily trade-off between speed and accuracy by changing the size of input image, without retraining. For instance, at the resolution of 1280, YOLOv5s runs in 4.9 ms at 83.3 mAP on validation test images. So, larger image sizes usually lead to better results, but obviously, take longer to process. It should also be noted that we used only person class in the dataset and ignored the people class, enabling us to detect individuals correctly even if they are in a crowd.

Fig. 2. Some instances of the Caltech pedestrian dataset, the baseline for training and testing in this paper.

IV. EVALUATION

The machine for conducting experiments was equipped with an NVIDIA GeForce RTX3090 Graphics Processing Unit (GPU) with 24,756 Megabytes of memory and an Intel(R) Xeon(R) Gold 6248R processor. The codes were implemented in Python v3, and we used CUDA v.11.6.55 computing platform and PyTorch framework.

A. Evaluation Metrics

We have employed precision, recall, Average Precision (AP), and mAP metrics to evaluate our proposed method. Precision refers to the ratio of all correctly predicted instances (i.e., pedestrians) among all predictions. Recall is used to indicate the number of correctly classified samples in the total number of ground-truth data. In other words, it shows how many samples was the model able to detect out of the total number of pedestrians in the input image. The mentioned metrics are calculated using the equations 1 and 2, respectively.

\[
precision = \frac{TP}{TP + FP} \quad (1)
\]

\[
recall = \frac{TP}{TP + FN} \quad (2)
\]

Where TP is True-Positive, FP refers to False-Positive, and FN refers to False-Negative. The value of precision and recall depends on how many True Positives were detected by the model. A precision-recall curve plots the value of precision against recall for different confidence threshold values. To simplify using precision and recall measure, we use F1-score, which shapes a single metric based on their harmonic mean. Equation 3 shows calculation of F1-score, where Pr and Rc refer to precision and recall, respectively.

\[
F1-score = 2 \times \frac{Pr \times Rc}{Pr + Rc} \quad (3)
\]

AP summarizes the precision-recall curve to one scalar value. The range for AP is between 0 to 1. The metric AP can be calculated from Equation 4, where \( n \) is the number of thresholds.

\[
AP = \sum_{k=0}^{n-1} (Rc(k) - Rc(k+1)) \times Pr(k) \quad (4)
\]

Additionally, mAP is another metric which indicating the average AP for each category. The formula for calculating mAP is shown in Equation 5, in which \( k \) refers to a specific class and \( n \) refers to the total number of classes.

\[
mAP = \frac{1}{n} \sum_{k=1}^{n} AP_k \quad (5)
\]

B. Evaluation Results

Fig. 3 demonstrates eight curves of changes in key indicators according while training during the defined epochs. The top four curves are related to network training and the bottom four curves are related to network validation. In all graphs, the horizontal axis indicates the epochs. The vertical axis refers to the loss (CIOU()) or accuracy (mAP) values, based on the graph.
four curves are related to network training and the bottom four curves are related to network validation. Here, YOLOv5’s loss function is composed of three parts: boxloss is a bounding box regression loss that measures how tight the predicted bounding boxes are w.r.t. to the ground truth object (loss criteria in use is CIoU()). objloss is the objectness loss that represent the confidence of object presence (Binary Cross Entropy()). And finally, clsloss is a classification loss that measures the correctness of the classification. This loss is usually called Cross Entropy loss(). Since our dataset has one class only, the classification error is constantly zero.

In this regard, the precision-recall curve is shown in Fig. 4. With the precision-recall curve, we can see that YOLOv5 model has the high precision and recall values at the pedestrian detection task. Fig. 5, demonstrates the pedestrian detection results of the proposed approach in various scenarios. We can see that even people walking in the throngs have been detected and tracked correctly using our fine-tuned network. Table I presents the accuracy of the proposed pedestrian detection system. Accordingly, the system could achieve 0.935 in precision and 0.84 in the recall. Also, the mean average precision at 50% IoU (cut-off value of 5) was around 91.8%, which indicates our proposed pedestrian detection application is an efficient approach. Among the 1993 pedestrian samples in the test images, the total number of detected pedestrian instances is 1622.

Moreover, to assess the system's performance, we noticed that the detection speed of the YOLOv5s architecture is 14.4ms, which indicates a real-time performance. It can also work properly for pedestrian detection when the frame rate is 69 fps. The rate of frame per second can increases to 277 fps at the test when a batch size of 8 is considered.

C. Discussion

This paper uses YOLOv5s as the pedestrian detector, which can meet the requirements of real-time detection. It can be seen in Table I that, compared with the other pedestrian detectors, the fine-tuned YOLOv5s model has advantages in terms of precision (F1-score) and frame rate on the Caltech dataset. As a result of comparing, the mAP of YOLOv5s model is 7% higher than YOLOv4 [18]. Also, the frame rate of YOLOv5s is 69.4 fps, which is approximately twice the value of YOLOv4 (36.4 fps). For further work, we plan to evaluate our system in an end-to-end application with real-world data and improve the object detection module using other robust approaches, such as transformers.

| Model                  | F1-Score | mAP@.5 | mAP@.5:.95 | Frame-rate (fps) | Inference (ms) |
|------------------------|----------|---------|------------|-----------------|----------------|
| YOLOv5s (proposed)     | 0.885    | 0.918   | 0.663      | 69.4            | 14.4           |
| Improved YOLOv4 [18]   | 0.80     | 0.847   | -          | 36.4            | -              |

V. CONCLUSIONS

Pedestrians are among the paramount objects that autonomous vehicles must detect. Considering the current challenges in pedestrian detection and the prominent features introduced in YOLOv5, including a lower training time, autolearning anchor bounding boxes, and mosaic data augmentation, this paper presented a pedestrian detection application based on YOLOv5s. The framework is written in the Ultralytics PyTorch framework, making it very fast to train. In this approach, we have utilized the Caltech pedestrian dataset and implemented a conversion toolbox to prepare proper feed for the YOLO architecture. The mentioned toolbox converted images and annotations of the Caltech dataset to items identifiable by YOLOv5. According to the experiments, the mAP value of the proposed system is 91.8%, while the framerate is 69.4 frames-per-second.
ACKNOWLEDGMENT

The training and evaluation processes of the proposed work could not been done without the supports of the Amirkabir University of Technology for providing the Simorgh Supercomputer under Contract No. ISI-DCE-DOD-Cloud-9008081700.

REFERENCES

[1] N. Adnan, S. M. Nordin, M. A. bin Bahruddin, and M. Ali, “How trust can drive forward the user acceptance to the technology? in-vehicle technology for autonomous vehicle,” Transportation research part A: policy and practice, vol. 118, pp. 819–836, 2018.
[2] S. Zhang, R. Benenson, M. Omar, J. Hosang, and B. Schiele, “Towards reaching human performance in pedestrian detection,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 4, pp. 973–986, 2017.
[3] M. Hneva and H. Radha, “Object detection under rainy conditions for autonomous vehicles: A review of state-of-the-art and emerging techniques,” IEEE Signal Processing Magazine, vol. 38, no. 1, pp. 53–67, 2020.
[4] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in IEEE computer society conference on computer vision and pattern recognition, vol. 1. 2005, pp. 886–893.
[5] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International journal of computer vision, vol. 60, no. 2, pp. 91–110, 2004.
[6] H. Bay, T. Tuytelaars, and L. V. Gool, “Surf: Speeded up robust features,” in European conference on computer vision. Springer, 2006, pp. 404–417.
[7] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Proceedings of the IEEE computer society conference on computer vision and pattern recognition. vol. 1. Ieee, 2001, pp. 1–1.
[8] I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, “Artificial neural networks,” Cham: Springer International Publishing, vol. 39, 2017.
[9] J. Redmon J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” arXiv preprint arXiv:1804.02767, 2018.
[10] Y. Tian, P. Luo, X. Wang, and X. Tang, “Deep learning strong parts for pedestrian detection,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1904–1912.
[11] J. Noh, S. Lee, B. Kim, and G. Kim, “Improving occlusion and hard negative handling for single-stage pedestrian detectors,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 966–974.
[12] X. Zeng, W. Ouyang, M. Wang, and X. Wang, “Deep learning of scene-specific classifier for pedestrian detection,” in European Conference on Computer Vision. Springer, 2014, pp. 472–487.
[13] G. Li, Y. Yang, and X. Qu, “Deep learning approaches on pedestrian detection in hazy weather,” IEEE Transactions on Industrial Electronics, vol. 67, no. 10, pp. 8889–8899, 2019.
[14] W. Liu, S. Liao, W. Ren, W. Hu, and Y. Yu, “High-level semantic feature detection: A new perspective for pedestrian detection,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 5187–5196.
[15] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, “Accurate single stage detector using recurrent rolling convolution,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5420–5428.
[16] S. Zhang, J. Yang, and B. Schiele, “Occluded pedestrian detection through guided attention in cnns,” in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2018, pp. 6995–7003.
[17] L. Wang and B. Zhang, “Boosting-like deep learning for pedestrian detection,” arXiv preprint arXiv:1505.06800, 2015.
[18] P. Tumas, A. Nowosielski, and A. Serafinski, “Pedestrian detection in severe weather conditions,” IEEE Access, vol. 8, pp. 62775–62784, 2020.
[19] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Chen, and I.-H. Yeh, “Cspnet: A new backbone that can enhance learning capability of cnn,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2020, pp. 390–391.
[20] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” IEEE transactions on pattern analysis and machine intelligence, vol. 37, no. 9, pp. 1904–1916, 2015.
[21] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8759–8768.
[22] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: An evaluation of the state of the art,” IEEE transactions on pattern analysis and machine intelligence, vol. 34, no. 4, pp. 743–761, 2011.