Research Article

Embedded YOLO: A Real-Time Object Detector for Small Intelligent Trajectory Cars

WenYu Feng,¹ YuanFan Zhu,² JunTai Zheng,¹ and Han Wang¹

¹School of Transportation and Civil Engineering, Nantong University, 9 SeYuan Road, Nantong, China

²School of Management Science and Engineering, Chongqing Technology and Business University, Chongqing, China

Correspondence should be addressed to Han Wang; hanwang@ntu.edu.cn

Received 12 August 2021; Accepted 23 August 2021; Published 2 September 2021

Academic Editor: Xiao Chen

Copyright © 2021 WenYu Feng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

YOLO-Tiny is a lightweight version of the object detection model based on the original “You only look once” (YOLO) model for simplifying network structure and reducing parameters, which makes it suitable for real-time applications. Although the YOLO-Tiny series, which includes YOLOv3-Tiny and YOLOv4-Tiny, can achieve real-time performance on a powerful GPU, it remains challenging to leverage this approach for real-time object detection on embedded computing devices, such as those in small intelligent trajectory cars. To obtain real-time and high-accuracy performance on these embedded devices, a novel object detection lightweight network called embedded YOLO is proposed in this paper. First, a new backbone network structure, ASU-SPP network, is proposed to enhance the effectiveness of low-level features. Then, we designed a simplified version of the neck network module PANet-Tiny that reduces computation complexity. Finally, in the detection head module, we use depthwise separable convolution to reduce the number of convolution stacks. In addition, the number of channels is reduced to 96 dimensions so that the module can attain the parallel acceleration of most inference frameworks. With its lightweight design, the proposed embedded YOLO model has only 3.53M parameters, and the average processing time can reach 155.1 frames per second, as verified by Baidu smart car target detection. At the same time, compared with YOLOv3-Tiny and YOLOv4-Tiny, the detection accuracy is 6% higher.

1. Introduction

Intelligent autonomous driving cars, also known as smart cars, have quickly become a research hotspot because of their small size, flexibility, and low energy consumption [1]. Smart cars can perform automatic tracking, obstacle avoidance, positioning and parking, and remote image transmission [2]. They have broad application prospects not only in transportation but also in the military, medicine, and aerospace fields. With the COVID-19 pandemic, demand for unmanned vehicles is growing.

Automatic obstacle avoidance and path planning are two hot research topics related to smart analysis, and processing of the data stream from onboard terminal equipment still faces many challenges. Rapid detection of a target is the basic problem of autonomous driving and obstacle avoidance. Visual object detection based on a convolutional neural

network (CNN) can automatically identify landmarks, pedestrians [3], and other vehicles [4] and also can measure distance and speed [5] through relative positions. Because of its strong algorithm generalization and high practicability, it has received extensive attention from researchers [6, 7].

Traditional object detection methods, which are based on a two-stage framework that consists of region selection and then classification, include RCNN [8] and fast-RCNN [9]. In 2016, YOLO [10] was proposed as a new one-stage framework for target detection. The main idea was to use the entire graph as the input of the network and directly return to the position of the bounding box and the category to which the bounding box belongs in the output layer. To enhance the accuracy of original YOLO, Redmon proposed a new joint training method in YOLOv2 [11] that trained the detector simultaneously on the COCO detection dataset [12] and the ImageNet classification dataset [13]. As a result,

YOLOv2 achieved a 19.7 mean average precision (mAP) score on ImageNet. To improve the processing speed, the feature extraction backbone network Darknet53 was proposed in the YOLOv3 [14]. Experimental results show that Darknet53 is similar in accuracy to the ResNet network but has a faster processing speed. To further balance the relationship between detection accuracy and processing speed, Bochkovskiy sorted out the model structure of the YOLO series in YOLOv4 [15], that is, the feature extraction module “backbone network,” the feature enhancement module “neck network,” and the detection module “head network.” YOLOv4 [15] enlarged the receptive field through the CSPDarknet53 CNN and used the PANet [16] feature pyramid structure for multiscale feature fusion. The experimental results on the COCO dataset [12] proved that the detection accuracy of the YOLOv4 model is improved on the basis of a guaranteed processing speed of 30 frames per second (fps).

Although the performance of the optimized YOLOv4 model has been improved, the application of the mobile terminal environment for smart cars still has two contradictory problems. (1) The YOLO series models have high requirements for hardware configuration. In the smart car terminal environment, the detection speed still cannot meet real-time requirements. (2) Excessive simplification of the network structure will reduce the effectiveness of features, causing a significant decrease in detection accuracy while increasing the calculation speed, which will decrease the model’s target detection capability.

In order to balance the above contradictions, this paper formulates the following optimization strategies for the YOLO model. The backbone network and neck network in the YOLO model determine the detection accuracy of the model, and its parameters account for more than 60% of the total model. The backbone network is responsible for the extraction of underlying features. To maintain the detection accuracy, the complexity of the backbone network must be preserved or even increased to ensure the effectiveness of its extracted features. The neck network is responsible for feature enhancement, which can simplify the network structure and calculation process, reduce the volume of parameters, and increase the calculation speed while ensuring its basic functions.

On the basis of the above optimization strategy, this paper proposes an ultralightweight target detection network model, called embedded YOLO, for the smart car mobile terminal environment.

First, the feature extraction module ASU-SPP network is proposed. In the three-scale feature extraction branch, the attention mechanism is used to adjust the feature channel weight of the lightweight network ShuffleNet v2 and then SPP is used to enhance the feature of multiscale pooling. Second, we design the PANet-Tiny network, a lightweight feature fusion module. By excluding all large-scale convolution operations in PANet, only 1×1 convolution is retained for feature channel dimension alignment. Both up-sampling and down-sampling are implemented using linear interpolation. The multiscale fusion method uses an element addition operation that significantly reduces the amount of

calculations and number of parameters in the entire feature fusion module. Finally, the lightweight head structure is adopted, and the depth separable convolution is used to reduce the number of convolution stacks. At the same time, the number of channels is reduced to 96 dimensions so that the model can obtain the parallel acceleration of most inference frameworks.

To verify the effectiveness of the proposed model, we selected a smart car based on an edge processor board as the verification environment. After online real-world testing in the mobile terminal environment, embedded YOLO achieved high-accuracy and real-time capability.

2. Proposed Lightweight Network

Figure 1 shows the embedded YOLO network model architecture. It can be seen from Figure 1 that the model follows the three-module tandem frame design of YOLOv4, which is composed of the underlying feature extraction module ASU-SPP network, the multiscale feature fusion module PANet-Tiny, and the detection head module Head-Tiny.

2.1. ASU-SPP Backbone Network. The traditional YOLOv4 low-level feature extraction module CSPDarknet53 CNN has so many parameters and high computational complexity that its network operation speed is only 30 fps, which makes it difficult to meet the real-time requirements of the smart car mobile terminal equipment environment. However, the simplistic network structure of the underlying feature extraction module CSPDarknet53-Tiny in YOLOv4-Tiny [17] results in weak feature extraction capabilities. In addition, the environment in the front-view image of the smart car is more complex, with multiple target objects with large-scale differences often appearing, as well as mutual occlusion phenomena. Therefore, it is difficult for a one-stage detector to accurately capture all targets at the same time.

In order to compress the network volume, enhance the effectiveness of features, and achieve multiscale bottom-level feature extraction, this paper proposes the multiscale low-level feature extraction module ASU-SPP network model. As shown in Figure 1, the model uses the lightweight ShuffleNet v2 network as the backbone structure for feature extraction, adjusts the weight of its important feature channels by introducing the attention mechanism squeeze excitation (SE) unit, and uses SPP to obtain multiscale pooling branch features.

Figure 2 shows the attentive ShuffleNet unit (ASU) network structure. After inputting the feature map for channel splitting, two branches are formed in the channel dimension. The lower branch is equivalently mapped, and the upper branch contains three consecutive convolutional layers, including two 1×1 nongroup convolution. After the BN layer, the squeeze excitation (SE) attention mechanism module is employed to adjust and sort the channel feature values. To reduce the time loss caused by the introduction of the SE structure, the channel of the expansion layer is reduced to the original $1/8$ so as to improve the detection

FIGURE 1: Architecture of the embedded YOLO lightweight network model.

FIGURE 2: Structure of attentive ShuffleNet unit (ASU).

accuracy without increasing the time loss. Finally, the feature maps extracted from two branches are fused by the concatenate layer and sent to channel shuffle layer.

In this paper, the upper branch is selected to introduce the SE mechanism so that the ASU model has two advantages: combining the concatenate and channel shuffle at the end of each ASU unit with the channel split of the next module unit to form an element-level operation, avoiding the design concept of adding element-level operations, and introducing the SE model in the upper branch to reduce the amount of calculation.

In order to increase the acceptance field, the SPP structure is introduced at the output end of each ASU unit to perform multiscale pooling fusion processing, as shown in Figure 1. The output features are pooled in four scales (1×1 , 5×5 , 9×9 , and 13×13) and then spliced and fused so that the ASU-SPP backbone network can realize the multiscale feature fusion of each branch output feature. Experiments

show that the pooling design of multiscale windows not only improves the accuracy of target detection but also significantly speeds network convergence during the training process.

Table 1 shows the structural characteristics of this paper's backbone network and of traditional YOLO series models. Although the backbone network proposed in this paper introduces the attention model SE and the SPP structure, through its lightweight design, the parameter volume of the network module is greatly reduced compared with the traditional YOLO model so that the model can improve the calculation speed while ensuring no loss in detection accuracy.

2.2. PANet-Tiny Neck Network. One type of feature map pyramid network [19] (FPN) for target detection is BiFPN [20]. Although the BiFPN network has high performance, its

TABLE 1: Comparison of the structural characteristics of different backbone networks.

Model name	Backbone network	Parameters (M)
YOLOv3-Tiny [18]	DarkNet53	86
YOLOv4-Tiny [17]	CSPDarknet53	6.06
Our Embedded YOLO	Our ASU-SPP	2.6

large number of stacked feature fusion operations causes a decrease in operating speed. The PANet network [16] is used as a feature fusion module in YOLOv4 because of its simple structure. However, there is still an excessive amount of calculation in the PANet network structure because of using stride = 2 convolution to perform multiple scaling of feature map with different scales and because feature fusion based on concatenate operation multiplies the number of channels.

To solve these problems, this paper proposes a simplified multiscale feature scaling and fusion feature map pyramid network structure, called PANet-Tiny, shown in Figure 3. The network removes all large-scale convolution operations in the original PANet and only retains 1×1 convolution for feature channel dimension alignment. Both up-sampling and down-sampling are implemented using linear interpolation, and the multiscale fusion method is adopted with the element addition operation. The amount of calculation and number of parameters of PANet-Tiny are significantly reduced compared with the original PANet.

Table 2 shows a comparison of the structural characteristics of different neck networks. It can be seen that the PANet-Tiny neck network proposed in this paper has the fewest parameters.

2.3. Head-Tiny Network. The traditional FCOS [21] detection head uses four 256-channel convolutions as a branch, and there are eight channels of 256 convolutions in the two branches of border regression and classification. This creates a huge calculation overhead.

In response to these problems, this paper selects each layer of features to use a set of convolutional structures, uses depth separable convolution to replace ordinary convolution, and reduces the number of convolution stacks from four groups to two. The 3×3 convolution can be replaced by a depthwise (DW) convolution with a 5×5 convolution kernel and a 1×1 convolution, and they have the same receptive field. The number of parameters of DW convolution and 1×1 convolution is only one-ninth of 3×3 convolution and at the same time significantly reduces the computational overhead. The channel dimension is then compressed, and the feature dimension is compressed to 96 dimensions, which allows the parallel acceleration of most inference frameworks. Finally, referencing the YOLO series model, the bounding box regression and classification are calculated using the same set of convolutions and then split into three branches. This makes the optimized lightweight detection head structure very small, as shown in Figure 4.

2.4. Loss Function. The three basic elements of object detection are quality prediction, classification, and positioning. If the problem of the inconsistency of quality prediction in

training and prediction is solved, then in the case of complex scenes, the positioning of objects tends to be uncertain and random. To ensure the consistency of training and prediction while also taking into account the classification score and quality prediction score, all positive and negative samples can be trained. We combine the representations of the two, retain the classification vector, and use the loss function:

$$QFL(\sigma) = -|y - \sigma|^\beta ((1 - y)\log(1 - \sigma) + y \log(\sigma)), \quad (1)$$

where y is the quality label of 0-1, which is the prediction. The global minimum solution of QFL is $\sigma = y$. In this way, the cross-entropy component becomes complete cross-entropy, and the adjustment factor becomes a power function of the absolute value of the distance. Considering that the real distribution is usually not too far from the marked position, another loss is added:

$$DFL(s_i, s_{i+1}) = -((y_{i+1} - y)\log(s_i) + (y - y_i)\log(s_{i+1})). \quad (2)$$

The above formula optimizes the probability that the label y is closest to the left and right positions in the form of cross-entropy so that the network can quickly focus on the distribution of the adjacent area of the target position. Finally, QFL and DFL are uniformly expressed as generalized focal loss (GFL), and their specific forms are as follows:

$$GFL(p_{yt}, p_{yr}) = -|y - (y_t p_{yt} + y_r p_{yr})|^\beta ((y_r - y)\log(p_{yt}) + (y - y_t)\log(p_{yr})). \quad (3)$$

3. Analysis of Experimental Process and Results

3.1. Experimental Environment and Evaluation Indicators. In the experimental environment of this paper, the training environment uses an Nvidia GeForce RTX 3090 24G GPU, and the test environment uses the edgeboard, a computing card based on PGA (the Xilinx Zynq ZU3), 2 GB of DDR4 memory, 6 bit OS, and 1.2 TOPS of computing power. The software consists of the Ubuntu v20.04 operating system, Cuda v11.1, Python v3.6.12, and PyTorch v1.7.0 deep learning framework. Mean average precision or mAP (0.5: 0.95) and frames per second (fps) are used as the evaluation indicators of model detection accuracy and speed, respectively.

3.2. Backbone Network Ablation Experiment. To verify the effectiveness of the proposed backbone network, an ablation experiment of the ASU-SPP network structure was carried out. Our collected dataset includes basic traffic signs, such as traffic lights, zebra (pedestrian) crossings, speed limits, and parking, which are captured from a smart car, as shown in Figure 5. Table 3 shows the detection results based on our collected dataset.

From Table 3, it can be seen that compared with the traditional ShuffleNet v2 [22], the ASU-SPP network has

FIGURE 3: Network structure of PANet-Tiny.

TABLE 2: Comparison of structural characteristics of different neck networks.

Model name	Neck network	Parameters (M)
YOLOv3-Tiny [18]	FPN [19]	14.02
YOLOv4-Tiny [17]	PANet [16]	0.17
Our Embedded YOLO	Our PANet-Tiny	0.11

reduced the number of parameters to 2.6M while resulting in an improvement of 2.2% in the Top-1 accuracy (i.e., the accuracy rate of the first category consistent with the actual results). The experimental results show that the introduction of the attention mechanism SE and the multiscale feature fusion structure SPP enhances the characterization ability of the underlying features.

3.3. Neck Network Ablation Experiment. To verify the effectiveness of the neck network PANet-Tiny, Mish [23] was selected as the neck network activation function, PANet-Tiny was combined with different backbone networks, and its parameter size was compared with that of traditional PANet networks along with detection accuracy. The experimental results are shown in Table 4.

From Table 4, it can be seen that under the COCO dataset, the combined parameters of PANet-Tiny and the four feature extraction networks are all lower than those of the traditional PANet network [16], and the detection accuracy is not compromised. In addition, under the same feature fusion module conditions, the backbone network ASU-SPP network in this paper has the best detection effect, with an accuracy of 23.3, which is 0.1% higher than ThunderNet [25] and 0.4% and 1.7% higher than MobileNet v3 [24] and ShuffleNet v2 [22], respectively. This also verifies

that the accuracy of the ASU-SPP network is significantly enhanced compared with that of traditional feature extraction networks.

3.4. Comparison of Lightweight Detection Networks. To compare the performance of lightweight target detection models, this paper selected the MS COCO dataset [12] for the experimental data to train and test the accuracy and real-time performance of target detection models. The experiment sets the input image size to 416×416 , the initial learning rate of the stage to 0.14, and the attenuation coefficient to 0.0001. The Multi-Step-LR learning rate method was used, milestones were set to [240, 260, 275], and gamma was set to 0.1. The SGDM gradient optimization algorithm was used, momentum was 0.9, weight decay was 0.0001, batch size was 32, workers per GPU were 2, and training epochs were set to 280 rounds. The results of the comparative experiment are shown in Table 5.

Table 6 shows the comparison results of target detection speed for four YOLO models and ours in the GPU 3090 environment. It can be seen that the target detection speed of the model in this paper is much better than that of the traditional lightweight YOLO series models.

Figure 6 shows a histogram of the training convergence process and a performance comparison of different models. Figure 6(a) depicts the loss curve of the three YOLO series models during the training process. It can be seen that in the case of 280 rounds of model training, the embedded YOLOv4 proposed in this paper has the lowest loss—about 3%—compared with embedded YOLO and YOLOv3-Tiny. To better compare the accuracy of the detection performance of each model, Figure 6(b) shows the performance (AP-FPS) broken line histogram of the four models. It can be seen that

FIGURE 4: Network structure of Head-Tiny.

FIGURE 5: Our collected dataset from the view of a smart car.

TABLE 3: Parameters of different backbone networks and detection results.

Backbone model	Parameters (M)	Top-1 Acc (%)	Top-5 Acc (%)
ShuffleNet v2 [22]	3.07	69.4	88.9
Our ASU-SPP	2.6	72.4	90.3

TABLE 4: Feature fusion network comparison experiment (PANet vs. PANet-Tiny).

Neck network	Backbone network	Parameters (M)	Accuracy (%)
PANnet [16]	ShuffleNet v2 [22]	14.33	21.7
Our PANet-Tiny		3.71	21.7
PANnet [16]	Our ASU-SPP	12.62	23.3
Our PANet-Tiny		3.25	23.3
PANnet [16]	MobileNetV3 [24]	11.8	22.9
Our PANet-Tiny		4.84	22.9
PANnet [16]	ThunderNet [25]	10.96	23.2
Our PANet-Tiny		4.52	23.2

the proposed embedded YOLO model can significantly increase the processing speed while ensuring the accuracy of target detection.

Figure 7 shows examples of target detection of the lightweight YOLO series models. It can be seen that the accuracy of the model in this paper is better than that of other traditional YOLO series models for small targets in complex environments.

Figure 8 provides the comparison results of each kind of target detection and Precision-Recall performance curve between embedded YOLO and YOLOv4-Tiny model based on VOC dataset [26]. Figures 8(a) and 8(b) show three object detection results of embedded YOLO and YOLOv4-Tiny model, respectively. It can be seen that YOLOv4-Tiny model misses some objects in the complex environments but embedded YOLO provides good detection results. Figures 8(c) and 8(d) show the detection accuracy of each kind of target and Precision-Recall performance based on embedded YOLO and YOLOv4-Tiny, respectively. The mAP of our proposed embedded YOLO is 77.89%, while the mAP of YOLOv4-Tiny is 71.20%. The Precision-Recall performance also illustrates that the object detection accuracy of the proposed embedded YOLO is higher than that of YOLOv4-Tiny.

In addition, we also conducted simulation target detection experiments from the perspective of the smart car for the YOLOv3-Tiny, YOLOv4-Tiny, and embedded YOLO models, and the test results for each model are shown in Figure 9. Figure 9(a) shows the detection results of simple markers in three models, Figure 9(b) shows the detection results of fuzzy markers in three models, and Figure 9(c) shows the detection results of complex markers of the three

TABLE 5: Results of comparison of light target detection models (edgeboard environment).

Model name	mAP	Parameters (M)	Latency (ms)	Size (MB)	FPS
YOLOv3-Tiny [18]	16.6%	8.86	12.89	33.7	77.6
YOLOv4-Tiny [17]	21.7%	6.06	10.92	23.0	91.5
Embedded YOLO	23.3%	3.53	6.4	7.9	155.1

TABLE 6: Comparison results of lightweight models (GPU3090 environment).

	YOLOv3 [14]	YOLOv3-Tiny [18]	YOLOv4 [15]	YOLOv4-Tiny [17]	Ours
FPS	30.32	40.62	24.9	49.06	86.29

FIGURE 6: Loss curve and AP-FPS performance results.

FIGURE 7: Comparison of target detection of YOLO series models.

FIGURE 8: Examples of target detection results for YOLO series models. (a) Object detection results of embedded YOLO. (b) Object detection results of embedded YOLOv4-Tiny. (c) Object detection results. (d) Precision-Recall performance.

FIGURE 9: Comparison of target detection results of YOLO series models. (a) Simple markers detection results. (b) Fuzzy markers detection results. (c) Complex markers detection results.

models. In Figure 9, the red box indicates the detection result and the white box indicates the missed target.

As can be seen from Figure 9, for simple markers, each model has a better recognition effect. For fuzzy markers, the YOLOv3-Tiny model displayed a certain degree of misjudgment or missed detection. The YOLOv4-Tiny model and embedded YOLO model performed better without misjudgment; for complex markers, the detection results were compared with the original YOLO series. It was difficult for the network to find a suitable target frame in small target detection, and there are more misjudgments and lower confidence. The embedded YOLO network can more accurately give the correct target frame, improve the confidence of target discrimination, and reduce the misjudgment of target categories. It can be seen that the embedded YOLO model perfectly recognizes the category and location of each marker compared with other YOLO series of lightweight models, which reflects that the improvement of

the underlying feature extraction method and the GFL loss function in this paper improves the detection accuracy of the model and the quality of the detection frame.

3.5. Online Experiment Results in Smart Terminal Environments. Lastly, we applied the proposed model to the small intelligent trajectory car terminal system. The small intelligent trajectory car is equipped with an edgeboard, a computing card based on PGA (Xilinx Zynq ZU3), 2GB DDR4 memory, 64 bit OS, and 1.2 TOPS. The experimental car and its main accessories are shown in Figure 10. The model in this paper achieves 155.1 fps on the smart car mobile terminal, which meets the real-time performance requirements of the vehicular system. Figure 11 shows the target detection results and automatic driving schematic diagram from the perspective of the car after loading the proposed model.

FIGURE 10: Examples of the small intelligent trajectory car's main accessories.

FIGURE 11: Examples of smart car first-view target detection and its autonomous driving trajectory.

4. Conclusions

In response to the requirement for fast target detection in a smart mobile terminal environment, this paper proposes an ultralightweight target detection network model, called embedded YOLO. First, the attention mechanism SE and the spatial pyramid pooling SPP structure are used to optimize the ShuffleNet network. Then, a multiscale feature fusion network module, called PANet-Tiny, is proposed to further lighten the detection network. Lastly, a lightweight structure is used on the detection head to simplify calculations.

After verification by the COCO test dataset and online experiments, the embedded YOLO model was compared with the traditional lightweight model, and it was found that while maintaining the mAP performance, the proposed model compresses the volume of model parameters and increases the computing speed. The attention mechanism SE and SPP structure introduced in this paper can enhance the information expression ability of the network neck feature map and significantly improve the detection accuracy of small targets. In the smart car environment, the processing speed is 155.1 fps, which meets the requirements of the smart car mobile terminal environment for the performance of the target detection network.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Natural Science Foundation of China under grant 61872425.

References

- [1] Q. Xu, B. Wang, F. Zhang, D. S. Regani, F. Wang, and K. J. R. Liu, "Wireless AI in smart car: how smart a car can Be?" *IEEE Access*, vol. 8, pp. 55091–55112, 2020.
- [2] T. Okuyama, T. Gonsalves, and J. Upadhyay, "Autonomous driving system based on deep Q learning," in *Proceedings of the International Conference on Intelligent Autonomous Systems (ICoIAS)*, pp. 201–205, Singapore, March 2018.
- [3] S. Ghosh, P. Amon, A. Hutter, and A. Kaup, "Reliable pedestrian detection using a deep neural network trained on

- pedestrian counts,” in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pp. 685–689, Beijing, China, September 2017.
- [4] W. Dong, Z. Yang, W. Ling, Z. Yonghui, L. Ting, and Q. Xiaoliang, “Research on vehicle detection algorithm based on convolutional neural network and combining color and depth images,” in *Proceedings of the International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pp. 274–277, Dalian, China, September 2019.
 - [5] A. Chetouani and M. Pedersen, “On the use of a convolutional neural network to predict perceptual quality of images without reference for different viewing distances,” in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pp. 1009–1013, Taipei, Taiwan, September 2019.
 - [6] S. f. Zhang, C. Chi, Y. Q. Yao, Z. Lei, and S. Z. Li, “Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9756–9765, Seattle, Washington, June 2020.
 - [7] D. F. Zhou, J. Fang, X. B. Song et al., “Joint 3D instance segmentation and object detection for autonomous driving,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1836–1846, Seattle, WA, USA, June 2020.
 - [8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, Columbus, OH, USA, June 2014.
 - [9] R. Girshick, “Fast R-CNN,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, Santiago, Chile, December 2015.
 - [10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, Las Vegas, NV, USA, June 2016.
 - [11] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, Honolulu, HI, USA, July 2017.
 - [12] T.-Y. Lin, M. Maire, S. Belongie et al., “Microsoft COCO: common objects in context,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 740–755, Zurich, Switzerland, September 2014.
 - [13] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and Li Fei-Fei, “ImageNet: a large-scale hierarchical image database,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, Miami, FL, USA, June 2009.
 - [14] J. Redmon and A. Farhadi, “YOLOv3: an incremental improvement,” 2018, <https://arxiv.org/abs/1804.02767>.
 - [15] A. Bochkovskiy, C. Y. Wang, H. Yuan, and M. Liao, “YOLOv4: optimal speed and accuracy of object detection,” 2020, <https://arxiv.org/abs/2004.10934>.
 - [16] S. Liu, L. Qi, H. F. Qin, J. P. Shi, and J. Y. Jia, “Path aggregation network for instance segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8759–8768, Salt Lake City, UT, USA, June 2018.
 - [17] <https://github.com/AlexeyAB/darknet>.
 - [18] P. Adarsh, P. Rathi, and M. Kumar, “YOLO v3-Tiny: object Detection and Recognition using one stage improved model,” in *Proceedings of the International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 687–694, Coimbatore, India, March 2020.
 - [19] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, Honolulu, HI, USA, July 2017.
 - [20] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: scalable and efficient object detection,” in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10778–10787, Seattle, WA, USA, June 2020.
 - [21] Z. Tian, C. Shen, H. Chen, and T. He, “FCOS: fully convolutional one-stage object detection,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 9626–9635, Seoul, Korea (South), November 2019.
 - [22] N. Ma, X. Y. Zhang, H. T. Zheng, and J. Sun, “ShuffleNet V2: practical guidelines for efficient CNN architecture design,” 2018, <https://arxiv.org/abs/1807.11164>.
 - [23] D. Misra, “Mish: a self regularized nonmonotonic neural activation function,” 2019, <https://arxiv.org/abs/1908.08681>.
 - [24] A. Howard, M. Sandler, G. Chu et al., “Searching for MobileNetV3,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1314–1324, Seoul, Korea (South), November 2019.
 - [25] Q. Zheng, Z. Li, Z. Zhang et al., “ThunderNet: towards real-time generic object detection on mobile devices,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 6717–6726, Seoul, Korea (South), November 2019.
 - [26] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: a retrospective,” *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.