Research Advanced in Pedestrian Detection Based on Deep Learning

Shuyue Gong

Department of Mechanical and Electronic Control Engineering, Beijing Jiaotong University, Beijing, China

21222005@bjtu.edu.cn

Abstract. In the field of computer vision, pedestrian detection is a significant research area that is frequently applied in areas like intelligent driving and video surveillance. The reliability and accuracy of subsequent tasks like pedestrian tracking and behavior analysis are directly impacted by the accuracy and effectiveness of pedestrian detection. However, there are still a number of difficulties with pedestrian detection tasks because of the variety of factors, including the shape, scale, and posture of pedestrians, as well as background interference and changes in lighting. Based on research on pertinent domestic and international literature, this article examines the key techniques for identifying pedestrians. Introduce frequently used datasets for pedestrian detection first. The model can then be split into two categories based on its training process: one-stage detection algorithms and two-stage detection algorithms. For each stage type, this study offers a thorough introduction to representative algorithms. A summary of its current state of development and future prospects was then given.

Keywords: Pedestrian detection, object detection, deep learning.

1. Introduction

Pedestrian detection aims to detect pedestrian targets accurately and quickly and label them in a given complex scene with a tight bounding box. In recent years, in the domains of artificial intelligence, computer vision, and video analysis, pedestrian detection has grown to be a research focus and a challenge. It is commonly employed in fields such as autonomous driving and video surveillance. It provides technical support for human motion recognition and human behavior understanding. In the realm of object detection technology, pedestrian detection technology is one of the most significant research areas with vast application potential because humans are a primary concern in a variety of circumstances. Deep research in it has tremendous academic value and practical application value, but it also has great significance for the advancement of other academic disciplines' research and development.

Currently, there has been a consistent enhancement in both the precision of detection and the real-time processing speed for pedestrian detection thanks to decades of progress in the area and the ongoing efforts of several researchers on both the local and international levels. However, compared to other types of detection targets, in actual production and application environments, pedestrian targets are much more vulnerable to different poses, variations in scene lighting intensity, complex environmental backgrounds, levels of occlusion (occlusion between pedestrian targets and scene objects, mutual occlusion between pedestrian targets), and variations in shooting angles. At the same time, the reliability of pedestrian detection algorithms may be impacted by external factors such as camera resolution and other hardware, which must also be taken into account. Therefore, currently, how to effectively reduce or even avoid missed and false detections in pedestrian detection is a challenge in pedestrian detection research.

Focusing on the issues and challenges faced above, this article takes deep learning-based pedestrian detection technology as the research object and conducts a detailed analysis of different pedestrian detection algorithms and their advantages and disadvantages. This article, which focuses on the issues and challenges mentioned above, takes deep learning-based pedestrian detection technology as the research object and analyzes various pedestrian detection algorithms in great detail,

outlining their benefits and drawbacks. At the same time, it also introduces the general dataset for pedestrian detection, and the empirical findings of various algorithms, and anticipates potential future growth directions for the field.

2. Method

Object detection technology has advanced significantly after more than 20 years of study and development. The two main categories of object detection techniques are traditional methods based on physically annotated features and deep learning technology [1]. Among them, item detection research has advanced significantly. Deep learning Methods have advantages over conventional algorithms based on regular feature annotation, including flexible structure, automatic feature extraction, high detection accuracy, and quick detection speed. Currently, there are two main categories of deep learning object detection algorithms: two-stage detection algorithms such as Faster R-CNN, FPN, and Mask R-CNN, which is based on region recommendations; one-stage detection algorithm, regression-based algorithms, such as YOLO series, SSD series, and RetinaNets. In recent years, new algorithms like NAS-FPN, EfficientDet, YOLOF have also come into use [2].

As shown in Figure 1, one-stage detection algorithm transforms the task of target boundary localization into a regression challenge by straight extracting features from the original image to forecast object classification and position. Although the detection speed and algorithm structure are both quick and simple, for scenes with higher target densities or overlaps, the accuracy loss is excessive, frequently leading to missed detection. Two-stage detection algorithm partitions the detection process into two sequential stages. The applicant region is created first, and features are then extracted. The targeted object's category information is then determined using the feature maps produced from each candidate region and input into the classifier. The targeted object's boundary box is then corrected using the regression loss function to ascertain its position. The two-stage detection algorithm can properly identify high-density and small targets, but its detection speed is slower than that of a single stage.

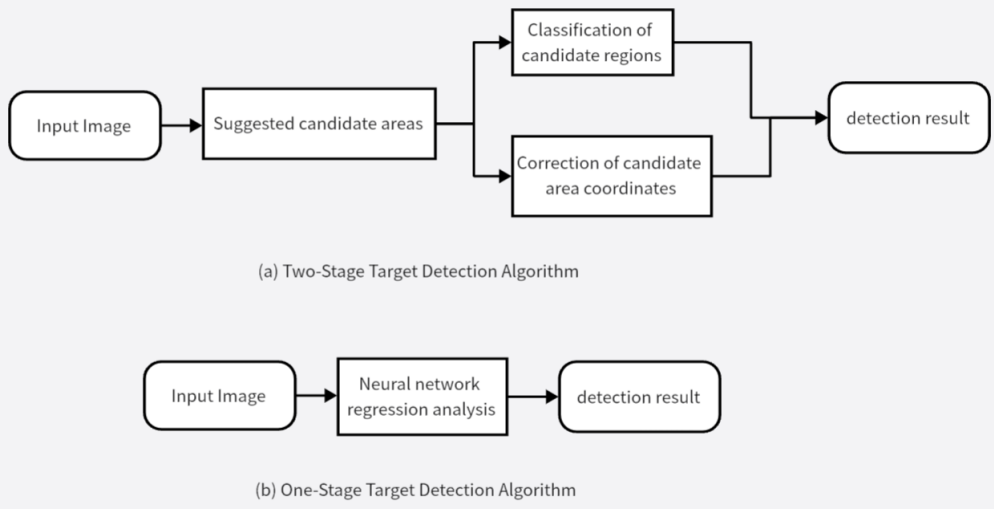


Figure 1. Comparison between one-stage and two-stage detection framewor

2.1. Two-stage detection algorithm

R-CNN is proposed as the first attempt by introducing the convolutional neural network into the object detection tasks. In contrast to conventional object detection techniques, R-CNN adopts a CNN network to extract feature representations. It tackles issues such as training complexity, and potential overfitting with limited data samples, by employing supervised pre-training on extensive datasets and then refining through fine-tuning on smaller datasets [3].

The workflow of R-CNN is as following. Firstly, it is the original image, and then using certain methods to generate some regions of interest on the original image, that is, regions that may contain

targets. Scale the generated candidate region to a consistent size (due to the fixed input requirement of the neural network, convolutional operations can accommodate varying inputs, while fully connected layers necessitate a consistent input size, which has been addressed in subsequent models for enhancement). Input the resized image into a CNN network (which can be a ready-made model, and then fine-tune it), and the CNN network will extract fixed dimensional feature vectors (AlexNet was used in the original text, removing the last output layer to extract 4096-dimensional features, which is much less than traditional methods). Then, input the extracted features into a pre-trained set of SVM classifiers (totaling k, where k is categories and each is a binary classifier) to identify what the target is in the region (and roughly locate the position of the target). The subsequent content will refine the position to make it more accurate.

The candidate regions generated may overlap to a certain extent since the selective search method is still not good enough. This results in a repeated convolutional operation in R-CNN when extracting features from the same region. Moreover, the R-CNN method stores the extracted features and then uses traditional SVM classifiers for classification, resulting in a large storage space requirement. Additionally, R-CNN scales all regions to the same scale for network training while the targeted boxes selected in the actual selective search have various sizes, which may cause deformation of the target.

To solve the above issues, SPP-net and Fast-RCNN are proposed. SPP-net network is based on R-CNN structure improvement and only makes improvements in feature extraction of candidate regions and feature vector size conversion. Due to the slow speed of R-CNN features, each image is divided into 2000 candidate regions of varying sizes, and each candidate region is proposed through CNN features, which is a very time-consuming process. SPP-net proposes two improvements to improve speed for R-CNN:

(1) Reduce convolution operations. An image undergoes a selective search algorithm to obtain 2000 candidate regions. In R-CNN, these 2000 candidate regions undergo size changes and enter the CNN network for feature extraction, requiring 2000 convolution operations. In SPP-net, the image first undergoes convolution operations to obtain a feature map and then maps these 2000 candidate regions one by one to the feature map, only performing one convolution operation. The original image is transformed into a feature map through CNN, and candidate regions are obtained through a selection search (SS). Now, it is necessary to map the candidate regions based on the original image to the feature vectors. Specific flowchart is as follows (the blue area is the candidate region, and the yellow area is the region mapped to the feature map).

(2) Add SPP layer (Spatial Pyramid Pooling). The SPP layer converts the feature map into a feature vector of fixed dimensions. The SPP layer divides each candidate region into three subgraphs: 4×4, 2×2, 1×1. Max pooling is performed on each subgraph and region. Assuming the input of the original image is 224×224 and the CNN convolution output channel is 256, the feature maps are connected together, resulting in (16+4+1) * 256=5376 results. This result is then passed to the feature vector training classifier SVM for subsequent steps.

SPP-Net proposes improvements on the basis of R-CNN, by mapping candidate regions, features, and maps, in conjunction with the use of SPP layers, to achieve shared computing in the CNN layer, reducing computational time. FastR-CNN and others are also inspired by SPP-Net. However, training of SPP-net is still too slow and inefficient, and features need to be written to disk (due to the presence of SVM).

As illustrated in Figure 2, the input of Fast R-CNN comprises two components: the first is the entire image slated for processing, and the second is the candidate region. Fast R-CNN initiates by conducting multiple convolutional kernel pooling operations on the image to generate convolutional feature maps. Given the presence of several candidate regions, the system applies a screening process to identify the region of interest, which is referred to as the Region of Interest (RoI). The RoI pooling layer is a specialized variant of the Spatial Pyramid Pooling (SPP) layer, designed to extract fixed-length feature vectors from the feature map. Each feature vector is fed into a sequence of fully connected (FC) layers, which bifurcates into two output layers of equal hierarchy. One of these layers

is responsible for classification and yields the probability distribution of each RoI across K object classes, encompassing all "background" classes, thereby producing a softmax probability estimation. The other layer is tasked with generating four real values (box regressions) for every class within the K objects. These four values encode precise bounding box coordinates for each of the K classes. The complete architecture is trained in an end-to-end fashion utilizing a multi-task loss, excluding the stage involving Region Proposal extraction.

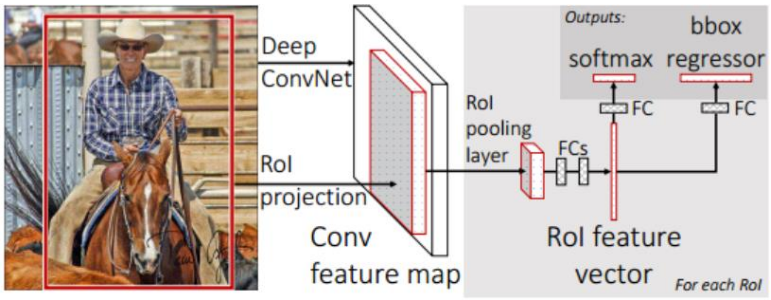


Figure 2. Main process of Fast R-CNN

2.2. One-stage detection algorithm

2.2.1 YOLO series algorithms

The architecture of YOLO [4] is notably straightforward, encompassing convolutional and pooling layers, culminating in the addition of two fully connected layers. In terms of network structure, it closely resembles CNN classification networks, with the primary distinction being the application of a linear activation function in the output layer. This alteration stems from the requirement to predict bounding box positions, involving numerical outputs, as opposed to solely object probabilities. In essence, the YOLO framework entails the transformation of input images through a neural network, yielding an output tensor. YOLO initially leverages the ImageNet dataset to pre-train the first 20 layers of the convolutional network, subsequently conducting complete network training for object recognition and localization using the PASCAL VOC dataset. While the last layer employs a linear activation function, the other layers employ Leaky ReLU activations. Training incorporates techniques such as dropout and data augmentation to mitigate overfitting. Significantly, YOLO achieves notable enhancements in detection speed, capable of processing up to 45 FPS (frames per second), while its streamlined version (utilizing a smaller network) can even achieve a processing speed of 155 FPS. Nonetheless, in comparison to contemporaneous two-stage detection methods, YOLO's detection accuracy for small and closely located targets is comparatively suboptimal.

YOLOV2 [5] has introduced enhancements across three key domains: heightened prediction accuracy, improved speed, and augmented object recognition capabilities, all while maintaining efficient processing. Notably, the extension of object recognition encompasses the identification of a broader array of objects, a notable instance being YOLO9000, capable of detecting up to 9000 distinct objects. Batch normalization, performed separately for each batch, engenders a regularization effect, culminating in improved convergence speed and efficacy. Adding Batch Normalization after convolution in YOLOV2 will increase the network's mAP by 2%.

YOLOV3 [5] marks the third iteration of the YOLO series in object detection algorithms, showcasing significant enhancements in accuracy, particularly for detecting small targets. Moving on to YOLOV4 [5], this iteration achieves a substantial AP and FPS improvement of 10% and 12% respectively on the COCO dataset. It has been officially endorsed by Joseph Redmon and is regarded as one of the most potent real-time object detection models presently available. YOLOV4 effectively consolidates numerous detection techniques, systematically testing, combining, and refining these methods to optimize performance. While YOLOV4 doesn't necessarily introduce entirely new advancements, it amalgamates a wide array of target detection techniques to enhance its overall efficacy.

As shown in Figure 3, YOLOV5 [6] further boosts the detection performance, whose structure mainly includes the backbone, neck and detection head. The main structures in the backbone include the Conv module, C3 module, and SPPF module. The Conv module consists of a Conv2d, a BatchNorm2d, and an activation function, which completes down sampling, dimensionality reduction, normalization, and nonlinearity of feature maps. The C3 module is composed of three Conv modules and one Bottleneck module, named C3. In the backbone, C3 is a more important module for feature extraction. YOLOV5 has improved the SPP to SPPF, which has a faster speed when the output is the same.

In YOLOv5, the Neck module acquires relatively shallower features from the backbone and combines them with deeper semantic features. The diagram illustrates that the Neck employs Upsample sampling to interpolate the feature map on the left, enlarging its scale for improved feature map fusion. This upward fusion process incrementally increases the size of the feature map. Conversely, on the right side of the Neck, downsampling is executed to achieve feature maps at diverse scales. This step enhances the integration of shallow visual features with profound semantic features, moving beyond simple concatenation. The primary role of the Neck layer is to amalgamate shallow visual features with more complex semantic features, yielding more comprehensive and informative features for subsequent processing.

The architecture of the Detect module is notably straightforward, comprising solely three 1×1 convolutions that correspond to three detection feature layers. The purpose of a 1×1 convolution is to either expand or contract the dimensionality of the input. In this context, the number of channels in the 1×1 convolutional kernel aligns with the channel count in the input feature map. Furthermore, the number of output channels corresponds to the quantity of convolutional kernels used. Ordinarily, a 1×1 convolution doesn't alter the size of the feature map; instead, it impacts the channel dimension of the feature map.

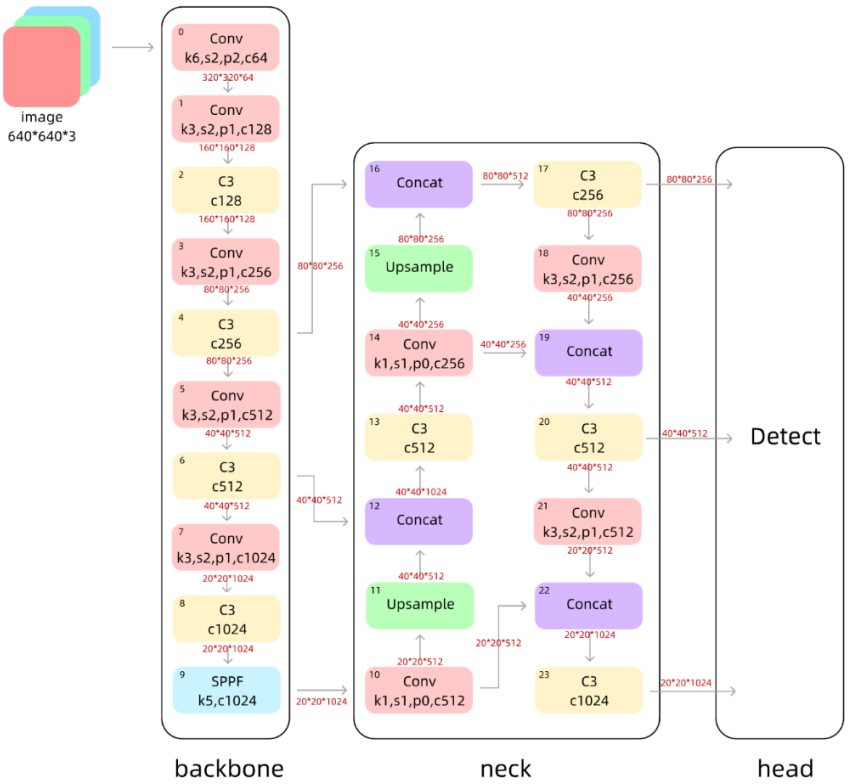


Figure 3. Network structure of YOLOV5 algorithm

2.2.2 SSD algorithm

SSD is a one-stage target detector that was developed after the release of YOLOV1 and before the release of YOLOV2 [7]. SSD employs multi-scale feature maps, and this concept of utilizing multi-scale feature maps is also integrated into YOLOv3's Darknet-53 architecture. On the shallow feature

map, the receptive field of each cell is not very large, so it is suitable for detecting smaller objects. On deeper feature maps, the receptive field of each cell is relatively large, making it suitable for detecting larger objects. There are two types of SSD networks: SSD 300 and SSD 512, used for different input sizes. The following text mainly analyzes SSD 300 as an example.

The network's backbone is based on the VGG16 architecture. The feature map obtained by Conv4-3 is the first 38×38 feature map among multi-scale features (i.e. the first feature map used for position regression and classification calculations in subsequent target detection networks). As this layer is relatively advanced, an L2 Normalization layer has been added after it. Normalize each pixel in the channel dimension. The third-to-last and second-to-last fully connected layers of VGG16 are transformed into a 3×3 convolutional layer named conv6 and a 1×1 convolutional layer named conv7, respectively. Additionally, the final fully connected layer is removed from the architecture. At the same time, the final pooling layer will be changed from 2×2 with a side of 2 to a 3×3 pooling layer with a side of 1. In order to perform subsequent multi-scale feature extraction, some convolutional layers have been added after VGG. Specifically, feature maps obtained from the Conv8_ 2, Conv9_2, Conv10_2, Conv11_2 layer will be sent to the subsequent target detection network for position regression and classification calculations. Based on the six feature maps extracted by Backbone and Extras, position information and classification information are extracted, which are implemented through the three branches: (1) Default Box layer used to generate a prior box, which is equivalent to the anchor box in Fast-RCNN. Assuming that there are 3 prior boxes on each cell of the feature map, and the size of the feature map is 5×5. (2) Localization branch uses a 3×3 convolutions to complete. The output feature map size is $5 \times 5 \times 12$, where 12 represents the coordinate information of the prior box. (3) Category confidence bransh uses a 3×3 convolutions to complete. The size of the output feature map is $3 \times 3 \times 63$, where 63 represents the category information of the prior box. Because each prior box has 21 category prediction results (VOC dataset, a total of 20 categories plus a background), 3 prior boxes are generated on each cell of the feature map, with a total of 3x21=63 prediction results. Finally, the final prediction result is obtained through boundary correction, classification, and NMS filtering.

2.2.3 RetinaNet

RetinaNet surpassed the two-stage network for the first time, despite its lack of disruptive contributions in the network structure. As shown in Figure 4, on the backbone network framework, unlike FPN, RetinaNet does not use conv2_X because the feature maps closer to the bottom will consume more computing resources. The generation of the P6 layer is not achieved by down sampling with Maxpool, but by using a convolutional layer with a step of 2. At the same time, a P7 layer has been added. In terms of anchors, three scales and proportions are used in different feature layers, totaling nine anchors. For the problem of scales larger than feature maps, in fact, this corresponds to the original map, not the feature map. Because ultimately, it needs to be mapped back to the original image, not the feature map. The reason behind the larger size of the top-level feature map is attributed to the convolutional operations, which result in an increased receptive field corresponding to the original image. The predictor uses the same and weights share parameters. The predictor is divided into two branches: a class subnet used to predict the category to which each anchor belongs, and a box subnet used to predict its bounding box regression parameters for each anchor.

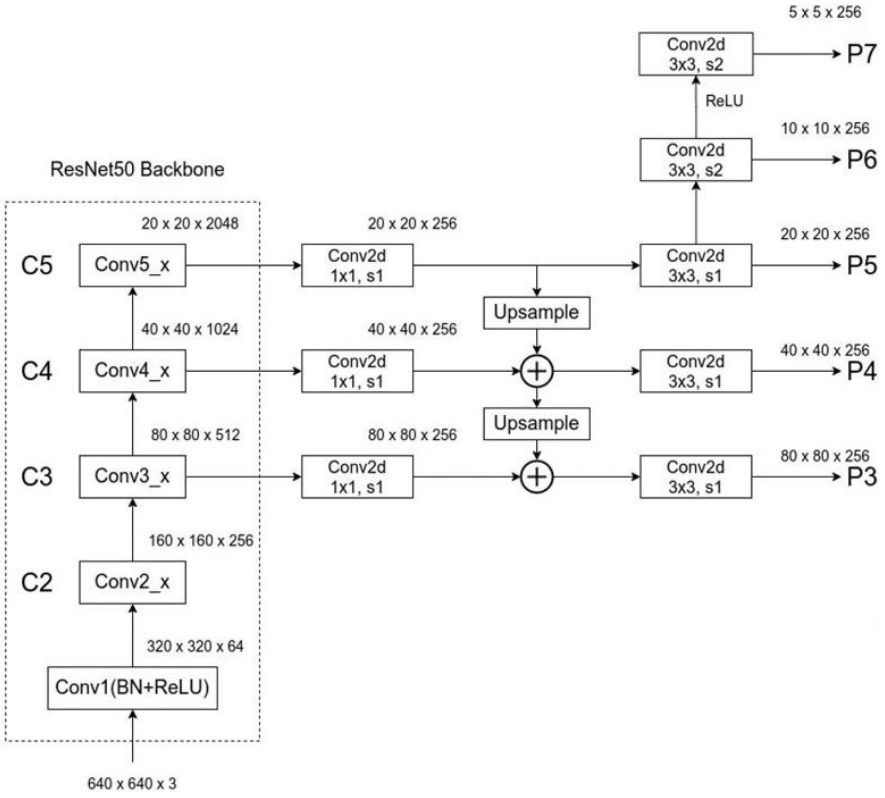


Figure 4. Network structure of RetinaNet algorithm

3. Dataset

The commonly used datasets in the field of pedestrian detection include PASCAL VOC, ImageNet, CrowdHuman, INRIA Person Dataset, etc.

PASCAL VOC2007 and PASCAL VOC2012 training have a total of 16551 images, including 20 common types of targets [8]. Mainly including humans, animals, transportation, etc. And the images in the dataset have corresponding XML files to annotate the location and category of the target [9]. ImageNet is a dataset to address overfitting and generalization issues in machine learning.

CrowdHuman contains a training set of 15000 images, a validation set of 4370 images, and a testing set of 5000 images, each with approximately 20 instances. This dataset is mainly used for pedestrian detection [10]. INRIA Person Dataset is a static pedestrian database, which includes training and testing sets. The pedestrian background of this dataset is relatively complex, there are many types of environmental changes (lighting), and there are also many human postures. Belongs to a dataset that is relatively close to real-life scenarios. The USC Pedestrian Detection dataset contains three sets of datasets. The images of USC-A come from the internet, with a total of 205 images containing 313 standing pedestrians. These images are shot from a front or back angle with no mutual obstruction between pedestrians. The images of USC-B are mainly from the CAVIAR video library. Including pedestrians from various perspectives, with some pedestrians obstructing each other, a total of 54 images and 271 pedestrians. USC-C has 100 images from the network, 232 pedestrians (from multiple angles), and there is no mutual obstruction between pedestrians.

4. Performance comparison

With the aim of facilitating more effective comparisons regarding the detection performance of various methods, we report the results of different single-stage and two-stage methods on different datasets. As shown in Table 1, the change from the R-CNN algorithm to the Fast-RCNN algorithm increases the mAP from 58.5% to 68.0% on the VOC dataset. The quantitative results show that the two-stage object detection algorithms are rapidly evolving and their accuracy is improving. For the

single-stage algorithms, similar performance growth trends are equally evident on the VOC and COCO datasets. It can be seen that two-stage object detection algorithms typically have higher accuracy but slower speed when compared to other algorithmic methods. Although there is a loss of accuracy, one-stage target detection algorithms typically move more quickly than two stages of the algorithm.

Table 1. Performance comparison of object detection algorithms

Algorithm	Backbone	Dataset	mAP (%)
R-CNN	AlexNet	**PASCAL VOC2007**	58.5
SPP-net	ZF-5	**PASCAL VOC2007**	59.2
Fast-RCNN	VGG16	**PASCAL VOC2007**	68.0
YOLO	VGG16	PASCAL VOC2007	63.4
YOLOV2	Darknet-19	PASCAL VOC2007	78.6
YOLOV3	Darknet-53	COCO	31.0
YOLOV4	CSPDarknet-53	COCO	43.5
SSD	VGG16	PASCAL VOC2007	79.8
RetinaNet	ResNet-101	COCO	34.4

5. Conclusion

This paper primarily centers around an in-depth exploration of advancements in the realm of pedestrian detection tasks, particularly emphasizing the progress achieved through deep learning approaches. In detail, we summarize the representative algorithms from the aspects of general object detection, which can be applied to solve the pedestrian detection directly. We also introduce the common datasets that can be used to train the detection model and compare the detection performance of various methods on these datasets. The accuracy and speed of pedestrian detection technology will continue to rise as deep learning-based object detection technologies are developed further. Strong generalization skills, outstanding results in complex scenarios, and wide potential applications are just a few examples. The design of adaptive detectors, multi-perspective and occlusion issues, pedestrian detection issues in unique situations, and other areas require additional study. The application areas of object detection technology will expand, benefiting social development more than just pedestrian detection.

References

[1] Lu Jian, He Jinxin, Li Zhe, et al. Overview of Object Detection Based on Deep Learning [J]. Electro Optics and Control, 2020, 27(5): 56-63.

[2] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, Brian Lee, A survey of modern deep learning based object detection models, Digital Signal Processing, Volume 126, 2022, 103514, ISSN 1051-2004.

[3] Bai Mengxuan, Li Shuaiyang, Qi Liping. Overview of Object Detection Based on Deep Learning [J]. Technology Perspective, 2020(9): 153-154.

[4] Hafiz Mughees Ahmad, Afshin Rahimi, Deep learning methods for object detection in smart manufacturing: A survey, Journal of Manufacturing Systems, Volume 64, 2022, Pages 181-196, ISSN 0278-6125.

[5] Shao Yanhua, Zhang Duo, Chu Hongyu, et al. Overview of YOLO Object Detection Based on Deep Learning [J]. Journal of Electronics and Information Technology, 2022, 44(10): 3697-3708.

[6] Liu Chunlei, Li Zhihua, Wang Chao, et al. A pedestrian detection method combining RepVGG and YOLOv5 [J]. Science Technology and Engineering, 2023, 23(7): 2945-2951.

[7] Zhang Lun, Tan Guangxing. Pedestrian detection algorithm based on improved SSD [J]. Journal of Guangxi University of Science and Technology, 2023, 34(3): 93-98, 107.

[8] Xie Fu, Zhu Dingju. Overview of Deep Learning Object Detection Methods [J]. Computer System Applications, 2022, 31(2): 1-12.

[9] Cheng Xu, Song Chen, Shi Jingang, et al. A Review of General Object Detection Based on Deep Learning [J]. acta electronica sinica, 2021, 49(7): 1428-1438.

[10] Luo Huilan, Chen Hongkun. A Review of Research on Object Detection Based on Deep Learning [J]. acta electronica sinica, 2020, 48(6): 1230-1239.