Journal of Science and Engineering Research
Vol. 1, No. 2, 2022, pp. 1-20
eISSN: 2957-7187

Simulation of Vehicle Target Detection based on Embedded System

Stephanie M Y Poon

The Hong Kong Polytechnic University
mypoon@dsd.gov.hk

DOI: http://dx.doi.org/10.56828/jser.2022.1.2.1
Article history: *Received (February 15, 2022); Review Result (March 24, 2022); Accepted (May 10, 2022)*

Abstract: Vehicle detection is an important part of environmental awareness in the Advanced Driver Assistance System (ADAS). Traditional vehicle detection is usually performed on a personal computer, with the rapid improvement of computer software and hardware, deep learning, and target detection algorithms continue to develop, and vehicle detection is gradually applied to embedded devices. Embedded devices have the advantages of small size, low price, and good stability. If accurate and real-time vehicle detection is realized on embedded devices, the development cost of the ADAS system will be significantly reduced. Firstly, according to the distribution of vehicle data and the characteristics of embedded devices, the following improvements were made to the original Solid State Drive network (SSD): (1) Using the K-Means clustering algorithm to re-set the regional candidate frames of SSD, making them more in line with the scale distribution of vehicle data and improving the detection accuracy of the model; Based on the original SSD loss function, the rejection loss is increased to improve the detection performance of overlapping vehicles. The Mobile Net V1 deep neural network is used as the feature extraction network of SSD, which can greatly reduce the amount of computation without reducing the detection accuracy, to meet the real-time requirements of running vehicle detection algorithms in embedded devices. The embedded system takes Samsung Exynos 4412 microprocessor-based on ARM architecture as the hardware platform, and transplants Linux operating system and all related drivers on this platform. In application development, QT graphical interface library and Caffe deep learning framework library are transplanted.

Keywords: Vehicle detection, SSD, Embedded system, Mobile net

1. Introduction

In recent years, with the rapid development of today's social economy, the automobile industry has been driven to grow rapidly [1][2][3][4][5]. In today's society, cars are not only a means of transportation but also a manifestation of happiness and a sense of gain. At present, the automobile industry is in the advanced development stage, which also makes the road traffic face tremendous pressure. Especially for some economically developed cities, road construction has taken shape, and it is extremely difficult to increase the width and length of roads, which leads to a series of problems such as road congestion, related accidents, and disputes, deterioration of urban road management and pollution of ecological resources. Road congestion has become an urgent problem in many modern cities. With the innovation and

continuous development of science and technology, intelligent transportation system technology has become recognized as the most effective means to solve traffic problems. According to different application objects, an intelligent transportation system can be roughly divided into two directions: one is the intellectualization of vehicles themselves, and the other is the intellectualization of traffic management facilities. An intelligent transportation system uses computer vision technology to process and analyze traffic surveillance video and realizes effective traffic control based on vehicle detection, identification, and tracking.

1.1. Research significance

The main significance of vehicle detection [1][2][3][4][5] is to judge whether there is a detection vehicle target in a given image or video by using the ability of computer automatic analysis and if there is a vehicle, determine and give the specific position of the detection vehicle. Vehicle detection is the basis and premise of vehicle position tracking, vehicle license plate recognition, vehicle type recognition, etc. Only robust vehicle detection algorithms can provide stronger support and guarantee the latter's further research. Vehicle detection applications mainly include the following aspects:

(1) Application in road infrastructure

The vehicles on urban traffic roads are detected, the current traffic flow on the roads is counted, whether the traffic flow will form road congestion is analyzed, and the traffic information on the roads is obtained in cooperation with broadcast networks and other media so that drivers can know the road conditions and adjust the traffic routes in time. It can also plan road traffic through vehicle detection, optimize road monitoring management, detect illegal vehicles and strengthen traffic safety supervision.

(2) Application in transportation

Used in the automatic driving system and auxiliary driving system. Vehicle detection is the most important part of an automatic driving system. The automatic driving system can detect the relevant vehicles on the road where its vehicle runs in real-time, obtain the vehicle-related information within a certain distance around its vehicle, and feed the obtained peripheral information back to the automatic control system of the vehicle in time, to ensure that various necessary basic operations such as keeping the distance between vehicles, changing lanes and adjusting the speed of vehicles can be realized during driving. In the current intelligent transportation system, vehicle detection is the foundation of vehicle behavior recognition and the core and key technology of its in-depth research. In this paper, a low-cost driving assistance scheme is proposed. The image information in front of the car is collected by a common visual camera on a common embedded device, and the vehicle target in the image is detected by using the target detection technology based on deep learning, and the specific position of the detected vehicle is displayed on the screen in the form of a rectangular frame. After recognizing all the vehicles ahead, the driving assistance system can judge which vehicles have potential threats. If dangerous vehicles are found, the driver can be reminded of the dangers by a voice prompt.

2. Related work

The research on vehicle detection for decades has greatly improved the accuracy and speed of vehicle detection in a given video or image. However, compared with human eye recognition, there is still a big gap in current vehicle detection and tracking technology, and there are still many problems in solving practical problems. In recent years, with the

increasing demand for vehicle detection and tracking technology, many scholars have devoted themselves to improving the performance of vehicle detection and tracking. The related achievements have been published in international top academic conferences and journals, such as IEEE Journal of Pattern Analysis and Machine Intelligence (PAMI), IEEE Conference on Computer Vision and Pattern Recognition (CVPR), European Conference on Computer Vision (ECCV), and European Conference on Computer Vision (ECCV) International Computer Vision Conference (ICCV), etc. This section mainly introduces the technical difficulties of vehicle detection and tracking and the research status of vehicle detection.

2.1. Research status

Foreign research teams engaged in vehicle detection and tracking mainly include the University of California, Berkeley, University of Southern California, Artificial Intelligence Robot Laboratory of Stanford University, and Visual and Cognitive Model Team of Bonn University. Before 2012, most target detection methods trained shallow classifiers through manually designed features, and the best algorithm was based on the improved DPM detection framework. With the coming of the deep learning era, unprecedented progress has been made in the field of vehicle detection. Geoffrey Hinton (2012) led the students to participate in the famous visual recognition competition and won the first prize for many tasks through CNN Deep Convolution Neural Network [6]. Since 2015, the 152-layer ResNet [7] network developed by Microsoft Research Center has reduced the error rate of the image net dataset to 3.57%. In the past 2013, R-CNN [8] network will deeply study the pioneers in the application field of target detection, and propose to select some classical algorithms, such as selective search and other classical algorithms, and then, efficient and fast universal target detectors such as Fast R-CNN[9][10], SPP-NET [11] and R-FCN [12] appeared, which achieved good results in vehicle detection tasks. YOLO [13] is also the vehicle detection algorithm of the deep learning method, and the application of these algorithms has made great progress in vehicle detection speed

2.2. Technical research difficulties

At present, the academic research on vehicle detection and tracking has made some achievements, but there are still great challenges and it is difficult to get a large number of applications in complex life scenes. This is mainly due to the arbitrariness, complexity, and uncertainty of natural life scenes, which leads to the fact that the existing vehicle detection and tracking algorithms cannot meet the actual needs. Therefore, the research of vehicle detection and tracking algorithm in intelligent transportation systems is also the future research direction in the field of computer vision. In today's unpredictable real road situation, vehicle detection and tracking technology still face the following problems:

(1) Scene complexity: In a complex road environment, many factors will bring challenges to vehicle detection and trackings, such as the change of lighting intensity in natural scenes (sunny days, cloudy days, day and night) and the influence of different weather conditions (rainy days, foggy days and snowy days), which are easily interfered and lead to vehicle detection and tracking errors.

(2) Vehicle occlusion: In real scenes, the size and shape of vehicles are different, and there is often a problem of mutual occlusion between vehicles. For example, in real scenes, when

traffic jams, intersections, and other red lights appear, and when there is occlusion between vehicles, it is difficult for general algorithms to correctly detect the occluded vehicles and track them accordingly.

(3) Motion blur: Because vehicles sometimes travel in a high-speed environment, distortion and motion blur easily occur when the camera of the monitoring system acquires the vehicle image, which increases the difficulty of detection and tracking.

(4) Vehicle diversity: Vehicles may appear in various states, such as walking, standing still, or running at high speed. There are many types and brands of automobiles, including automobiles, sports cars, and coaches. Different brands of cars have different appearances. Because of the variety of shapes, colors, and styles of vehicles, it is difficult to detect and track vehicles.

(5) Economical practicability: A vehicle detection and tracking system is a facility serving the transportation department, which not only considers the practicability of the system but also considers the cost of the system.

2.3. The application of traditional methods in vehicle detection

The detection process of traditional methods can be roughly divided into three stages. The first stage is to generate candidate regions. The second stage is feature extraction. The third stage is classifier classification, as shown in Figure 1.

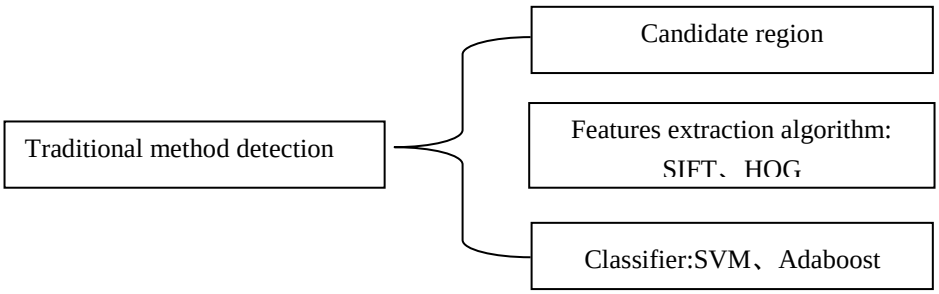

Figure 1: Vehicle inspection process

First, when the surveillance video is converted into an image, a region to be selected is generated in the image, and then traditional methods such as SIFT and HOG can be used for feature extraction of the region to be selected. Finally, these extracted features are classified by a classifier. At this time, SVM and Adaboost can be used.

The candidate area can be obtained by using the algorithm, mainly because the size of the image and its position cannot be completely determined, but the traditional method chooses again by constantly setting new parameters for verification. One of the most important parts of traditional methods is feature extraction. If the feature extraction is not good, the algorithm that uses these features for classification will have a high misjudgment rate. At present, HOG [14] and SIFT [15] are widely used in feature extraction. The final stage of target detection is classifier classification, including Adaboost [16] and SVM [17]. Feature extraction, which is the most important part of vehicle detection, and the target information extracted by the feature at this time, can be passed into the classification model algorithm to get the category of the moving target in the search window.

Using the HOG algorithm to extract features and SVM classifier can detect vehicles. Firstly, the surveillance video is converted into images and made into positive samples and negative samples, so that the HOG algorithm can be used to extract features in images. The next step is to use the SVM classifier to train and simulate the feature information extracted by the HOG algorithm so that the classifier model for the target vehicle can be obtained. Of course, there is no end here. We need to further test and verify the vehicle information, so we need to input images and aggregate all the windows with positive model judgment results. When extracting vehicle features, the appearance of the vehicle can be well described by the directional density distribution of edges or gradients, and HOG has become an important means of describing vehicle features. The steps of extracting the HOG feature are shown in Figure 2.

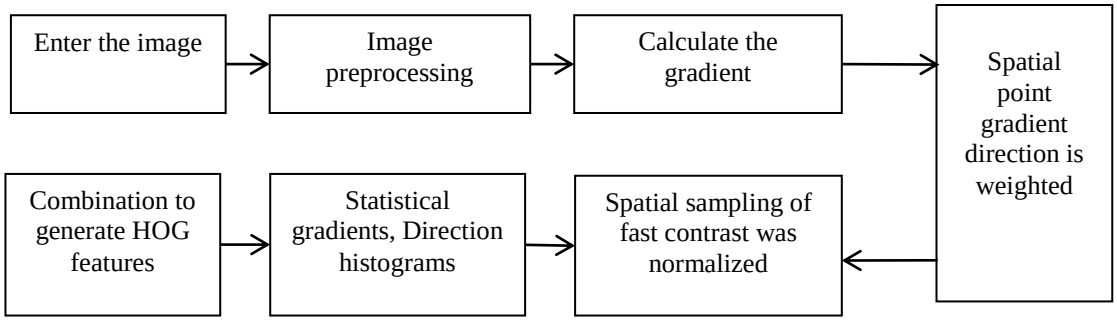

Figure 2: Flow chart of hog feature extraction

Traditional detection algorithms have many problems and great limitations. The inter-frame difference method described above [18] is similar to the background difference method [19]. The algorithm is only applicable to the actual scene where the camera is in a fixed state, that is, the light has little influence. It cannot detect stationary or slow-moving vehicles, and it is easy to miss the detection. In the three-dimensional object space, the optical flow method is more easily affected by noise such as illumination conditions, high algorithm complexity, poor real-time, fast speed, or overlapping vehicles, which easily leads to detection errors. Although the accuracy of the HOG+SVM method is higher than the previous two methods, its effect is not ideal due to factors such as insufficient illumination, shape change, complex background, false alarm rate, and false alarm rate high, and its real-time performance is poor.

It takes 728 milliseconds to process the frame, and the detection speed is not satisfactory. As far as the effect of detection and recognition is concerned, deep learning technology has far surpassed the traditional methods. The next step will introduce the deep learning technology in detail.

3. Vehicle target detection based on SSD

Using single-layer SSD for vehicle detection, its features are single and dense connections that can connect multiple feature layers closely so that the detection layers contain context semantic information at the same time. Therefore, this paper combines the features of different detection layers to form a feature layer with rich content for detection, thus realizing multi-feature fusion. FRN normalization can avoid the dependence on batch size in the SSD model training process, and ensure that the scale of the input feature map of each detection

layer is limited within a certain range, which is beneficial for the model to quickly learn vehicle features and improve the accuracy of pedestrian detection.

3.1. SSD algorithm principle

3.1.1. SSD network structure and default box

As a single-stage multi-scale target detection framework, SSD is much simpler in execution steps than a two-stage target detection framework, and its algorithm principle is relatively simple. Figure 3 is a structural diagram of the SSD300 network model, and the steps to realize target detection are as follows:

(1) Input images to ensure the unity of scale and size;

(2) The basic features are extracted by VGG16 whole convolution. For the convenience of calculation, FC_6 and FC_7 full connection layers are changed into two convolution network layers.

(3) Because the network is too shallow, after VGG16, different convolution kernels are used again to act on the additional convolution blocks;

(4) Under the condition that each detection layer contains classification and regression detectors, all detection results are synthesized, and the most suitable candidate frame is selected as the final target detection frame.

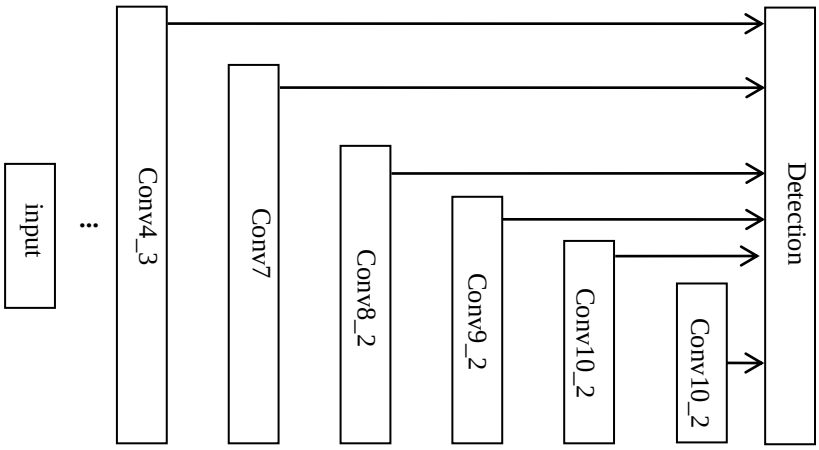

Figure 3: SSD300 network model structure diagram

According to Figure 3, to detect pedestrians, it is usually necessary to set default boxes corresponding to the original image with the same area and different aspect ratios for each detection layer with different scales. The process is as follows:

Step 1: First, the original image is mapped to the scaling factor of each detection layer, which can be expressed as:

$$S_{\mathrm{k}} = S_{\min} + \frac{S_{\max} - S_{\min}}{m-1}(k-1), k \in [1,5] \tag{1}$$

In formula (1), taking $S_{\min} = 0.2, S_{\max} = 0.9, m = 5$ means that the scaling factors of the original map to Conv7 and Conv11_2 are 0.2 and 0.9 respectively, and the corresponding scaling factor for Conv4_3 is 0.1;

Step2: Use different aspect ratios to calculate the width and height of the default box:

$$w_k = W_{input} \times s_k \times \sqrt{r} \tag{2}$$

$$\mathrm{h}_k = H_{input} \times s_k / \sqrt{r} \tag{3}$$

In formulas (2) and (3), sum W_{input} and H_{input} is the width and height of the training image, and r represents the ratio of width to height;

Step3: When r=1', the default box becomes a square with a larger scale:

$$s_k^{'} = \sqrt{s_k \times s_{k+1}} \tag{4}$$

$$w_k = W_{input} \times \sqrt{s_k \times s_{k+1}} \tag{5}$$

[Table 1] shows the specific dimensions of the default box. Because each pixel point needs to predict c categories and 4 placement coordinates, according to the data in [Table 1]. It is known that one will generate the number of silent recognition frames $(38\times38\times4+19\times19\times6+10\times10\times6+5\times5\times6+3\times3\times4+1\times1\times4)=8732$, and for pedestrian detection, it will generate $(8732\times(4\times1))=43660$ results.

Table 1: Specific size table of SSD default frame

	Conv4_3	Conv7	Conv8_2	Conv9_2	Conv10_2	Conv11_2
Size	38*38	19*19	10*10	5*5	3*3	1*1
Number	four	six	six	six	four	four
α_r □	{1,2,1/2,1}	{1,2,3,1/2,1/3,2,1}	{1,2,3,1/2,1/3,2,1}	{1,2,3,1/2,1/3,2,1}	{1,2,1/2,1}	{1,2,1/2,1}
s_k	30	60	111	162	213	264
$s_k^{'}$	forty-two	82	134	186	237	288
Default box pair should be the original in Fig actually size	(30,30)	(60,60)	(111,111)	(162,162)	(213,213)	(264,264)
	(21,42)	(85,42)	(157,78)	(229,115)	(300,151)	(300,187)
	(42,21)	(42,85)	(78,157)	(115,229)	(151,300)	(187,300)
	(42,42)	(103,34)	(192,64)	(281,94)	(237,237)	(288,288)
	one	(34,103)	(64,192)	(94,281)		
	one	(82,82)	(134,134)	(186,186)		

3.1.2. SSD network training

(1) The output SSD of the prediction frame uses a 3*3 convolution kernel to act on different detection layers to obtain two prediction frames: classification and regression:

$$l^{cx} = \left(b^{cx} - d^{cx}\right) / d^{w} \tag{6}$$

$$l^{cy} = \left(b^{cy} - d^{cy}\right) / d^{h} \tag{7}$$

$$l^{w} = \log\left(b^{w} / d^{w}\right) \tag{8}$$

$$l^{h} = \log\left(b^{h} / d^{h}\right) \tag{9}$$

Because the predicted value output by SSD algorithm is actually the converted value of the bounding box relative to the default box, the position of the bounding box can be obtained by inverse derivation from formulas (6)~(9):

$$b^{cx} = d^{w} \times l^{cx} + d^{cx} \tag{10}$$

$$b^{cy} = d^{h} \times l^{cy} + d^{cy} \tag{11}$$

$$b^{w} = d^{w} \times \exp\left(l^{w}\right) \tag{12}$$

$$b^{h} = d^{h} \times \exp\left(l^{h}\right) \tag{13}$$

In formulas (6)~(13), the position of each rectangular box contains four values $\left(cx, cy, w, h\right)$, namely, center point coordinate, width, and height. The default box position is expressed by $d = \left(d^{cx}, d^{cy}, d^{w}, d^{h}\right)$, and the position of the boundary box and prediction box are $b = \left(b^{cx}, b^{cy}, b^{w}, b^{h}\right)$ $l = \left(l^{cx}, l^{cy}, l^{w}, l^{h}\right)$ and respectively.

3.1.3. Determine the positive and negative samples

The sum of positive and negative samples in the SSD300 model framework is 8732. To improve the robustness of the algorithm, the difficult sample mining technology is adopted, and the positive and negative samples are determined according to certain rules and then the model is trained. Rules are as follows:

Step1: Firstly, calculate the intersection ratio of the first label and all default boxes, select the default box with the largest intersection ratio to correspond to the first label, and call the default box a positive sample corresponding to the foreground class;

Step2: Other label execution steps are the same as Step1 principle, and eventually every real box label will match the default box;

Step3: For the remaining default frames, first calculate the intersection ratio between the first default frame and all real frames to see if it is greater than the threshold. If the condition is met, the default frame is determined as a positive sample of the same category as the real frame, otherwise, it is a negative sample;

Step4: Other default boxes that do not correspond to labels follow the same principle as Step3, and finally all positive and negative samples are determined. Note: If a default box matches multiple labels, select the real box with the largest intersection ratio. Ultimately, each default box can only match one label, and one label can correspond to multiple default boxes.

Step5: Using difficult sample mining technology, the ratio of positive and negative samples is determined to be 1:3. In the process of network training, compared with the number of negative samples, the number of positive samples is very small. Therefore, on the

premise of determining the number of positive samples, the default boxes with a small confidence and large error in negative samples are screened out, and then the model is trained to update the parameters faster.

3.1.4. SSD network test

Although SSD trains the model with the ratio of positive and negative samples being 1:3, there will still be redundant frames during detection. To prevent the same target from being framed by multiple detection rectangles, a post-processing technique, Non-Maximum Suppression (NMS), can be used to remove the low frames and keep the detection frames with high confidence, thus reducing the false detection rate. There are two main forms of NMS algorithm, namely standard NMS and Soft-NMS, and their expressions are:

$$S_i = \begin{cases} s_i, iou(M, b_i) < N_t \\ 0, iou(M, b_i) \geq N_t \end{cases} \tag{14}$$

The m value in formula (14) represents the prediction frame with the highest confidence N_t. The prediction frame with the highest confidence b_t in NMS, and other prediction frames. The defect of standard NMS is that for two or more targets with close distance, because of the large intersection ratio between them, the phenomenon of cross occlusion may lead to missed detection. While the Soft-NMS algorithm complements NMS, instead of deleting it, it reduces the confidence score.

4. Construction of the embedded system

In this paper, vehicle target detection is carried out on embedded devices; therefore, the hardware and software environment of the whole embedded system should be set up first. The embedded system uses Samsung Exynos 4412 chip as the hardware platform, and transplants the Linux system, QT library, and Caffe on this hardware platform [20].

4.1. Hardware system

For embedded systems, the performance of the microprocessor determines the execution efficiency of the system. At present, there are microprocessor chips based on ARM, Power PC, X86, MIPS, and other architectures in the market, but the chips based on ARM architecture occupy a dominant position in the global embedded chip market. Typical microprocessor chips based on ARM architecture in the market include Samsung 2410 and 6410 and Qualcomm's 835 series processors. With the improvement of microprocessor performance, embedded devices are used more widely. Samsung's Exynos 4412 chip, one of the most powerful embedded chips in the market, is widely used in smartphones, tablet computers, video processing, and other embedded devices. Exynos 4412 chip is the core processing chip of the development board used in this paper. The Development board includes 2GB DDR3 memory; 16g MMC storage; Graphics card resources and interfaces; Camera interface; Sound card resources and interfaces; Hd MIPI screen interface; Ethernet and WIFI USB interface; TF/SD card interface and other rich hardware devices.

4.2. Boot loader porting

4.2.1. Introduction to the boot loader

The software development of this embedded system is based on Linux operating system. The Linux system distribution is shown in Figure 4. The complete Linux system consists of Boot Loader, start-up parameters, kernel, and root file system. The microprocessor Exynos 4412 Boot Loader is divided into two stages. When it is powered on, the program burned by Samsung in i ROM is executed first. this program copies part of the Boot Loader code into i RAM for execution, and then jumps to the c language part of the Boot Loader code in memory to complete loading and booting Linux.

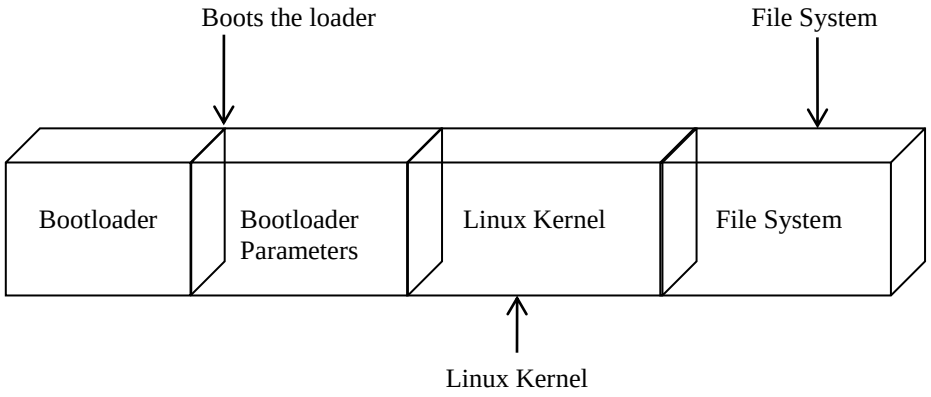

Figure 4: Typical partition structure of embedded Linux system

Generally, the startup of the operating system requires not only loading the kernel image into the memory but also that the SOC (System on Chip) has certain conditions before the operating system can run normally. However, when the CPU is powered on or reset, the memory controller is not initialized, and the memory cannot be used, resulting in the CPU cannot meet the start-up conditions of the kernel. Boot Loader is a program that runs before the kernel. Its main function is to initialize the CPU and other peripherals, establish the mapping relationship between the virtual address and physical address, make the system hardware and software meet the kernel running conditions, then load the kernel from the storage device into memory, set the parameters required by the kernel at a fixed position in memory, and pass the memory address of the parameters to the kernel so that the kernel can correctly address these parameters. The transplantation of Boot Loader should not only consider the architecture of the CPU, but also consider the device configuration of the development board, which makes it difficult to transplant Boot Loader, and boot loader cannot be shared among different devices. To overcome the problem of poor portability of Boot Loader, U-Boot is developed, which supports ARM, Power PC, and MIPS architectures at the same time.

U-Boot (Universal Boot Loader) is the most popular Boot Loader in the embedded system at present. U-Boot was first initiated by Wolfgang Denk, and then all the enthusiasts on the whole Internet participated in the research and development, maintenance, and development of an open-source project. U-Boot can be transplanted to processors with various kernel architectures, such as ARM, x86, Power PC, etc., and can also load various operating systems, such as Linux, Windows CE, Palm OS, etc. The execution of U-Boot can be divided into two

stages: assembly and C language. The main tasks completed in the assembly stage include: shielding interrupts, disabling MMU, reading start-up information, initializing peripherals such as memory controller, LED, and serial port, copying U-Boot from corresponding storage devices to memory, setting stack pointer, and jumping to the start-up function of C language. The main tasks of the C language stage are: initializing hardware devices such as timers, SD/TF cards, and network cards, setting parameters passed to the kernel, reading environment variables, copying the kernel from storage devices to memory, and finally starting the kernel. When the kernel is successfully started, all hardware resources are managed by the kernel, the mission of U-Boot is completed, and the memory space occupied by U-boot is reclaimed.

4.3. Linux kernel transplantation

The kernel manages all hardware resources, provides a file system to facilitate users to operate storage devices, and controls processes to access hardware in an orderly manner. The kernel is a part of the operating system, and the commonly used Linux system (such as Ubuntu) refers to a distribution that integrates the Linux kernel, root file system, various applications, and corresponding library files based on it, and provides a graphical interface with desktop functions. The kernel is also the most important part of an embedded system, which is widely transplanted to various embedded devices. The Linux kernel features are as follows:

(1) Memory management. Reasonable and efficient memory management by the kernel is very important for the whole embedded system, which not only ensures the efficiency of program execution but also ensures the correct operation of the system. For a 32-bit processor, a total of 4G memory address space can be accessed. The memory management system divides it into kernel space and user space, which are independent of each other to prevent users from modifying the kernel. The mapping relationship between the physical address and the virtual address is established, and the user process memory is managed.

(2) Task scheduling. Multitasking operating systems are divided into non-pre-emptive and pre-emption. Like most modern operating systems, Linux adopts a pre-emptive multitasking mode. This means that the CPU occupation time of the program is determined by the scheduler in the operating system. The scheduler decides when to stop a process to let other processes have the opportunity to run, and at the same time, picks out another process to start running. There are three scheduling methods in the kernel: SCHED_OTHER, SCHED_FIFO, and SCHED_RR. Their strategies are different. SCHED_OTHER guarantees that all application processes can be executed. SCHED_FIFO strategy is that only the process with higher execution authority than the currently executing process can pre-empt the CPU for execution, otherwise, other processes will not be executed until the current process is executed; SCHED_RR strategy is to allocate a certain execution time to each process, and put the process at the end of the process queue when its execution time is used up, to execute each process circularly.

(3) File system.

The file system in the Linux kernel is essentially a set of software composed of some codes, which allows users to access disks in the form of directories or file names.

(4) Equipment management.

The core philosophy of the Linux system is "everything is a file", which treats all devices such as character devices, block devices, sockets, processes, threads, and pipes as ordinary files.

4.4. QT transplantation

QT is a multi-platform C++ graphical interface application framework developed by Trolltech. Since its birth, its version has been continuously updated and its functions have become more and more powerful. Now it has become the foundation of KDE (K Desktop Environment), a Linux desktop environment. The advantages of QT are as follows:

(1) The portability is strong.

Operating systems supported by QT include Windows, Win CE, Unix/X11, Linux, Sun Solaris, Android, and many other X11 platforms.

(2) Easy to use.

QT is a C++ toolkit, which consists of hundreds of C++ classes, which can be directly used in programs. Because C++ is an object-oriented programming language and QT is based on C++, QT has all the advantages of OOP (Object Oriented Programming).

(3) Fast running speed.

QT's ease of use and speed are inseparable. QT is a GUI simulation toolkit, that is, it does not use the local toolkit for calling but uses the low-level drawing functions on its platform, thus improving the program speed. The QT versions used in PC and embedded systems are different. QTE is used in embedded systems. QTE directly interacts with LCD video memory to avoid the use of the Xlib library, thus saving resources and reducing program complexity. The development model of QT on PC and the embedded system is shown in [Figure 5].

During the development of QTE, its name has been changed several times, from QTE to Qtopia-core, and finally to qt-everywhere-opensource. The source code is downloaded from the official website and compiled by the cross-compiling tool arm-Linux-GCC-4.3.2. The specific transplant steps are as follows:

Before porting QTE, the embedded Linux system must have a touch screen driver, LCD driver, and tslib library file. Tslib supports touch screen verification and filtering, which is an indispensable component in the QTE4 compilation configuration option. it is the adaptation layer of the touch screen driver and provides a unified interface for the upper application.

4.5. Caffe framework transplantation

Caffe, the full name of convolutional architecture for fast feature embedding, is a common deep learning framework, which is mainly used in video and image processing. It is a convolutional neural network framework based on C++/CUDA/Python developed by Berkeley Visual and Learning Center (BVLC).

Caffe is based on modularization, and its main modules are as follows:

(1) Blob module

Blob is a class that stores and transmits data in Caffe. Its essence is a four-dimensional matrix (n,c,h,w), where n represents the batch size of each training data, c represents the number of image channels, and h and w represent the height and width of the image respectively. Member variables data and diff in Blob class point to the memory space addresses of forwarding operation data and reverse operation gradient, respectively.

(2) Layer module

The layer is the concrete realization of the "Layer" in the convolutional neural network. Each layer accepts Blob-type data transmitted by the previous layer and outputs Blob after operation in this layer. The Forward and Backward functions defined in Layer complete specific calculation tasks. Dozens of commonly used layers have been packaged in the Caffe framework: Conv_layer, relu_layer, pooling_layer, softmax_layer, and so on. (3) Net module.

Net is the whole network, which is the collection of all layers that make up a network. The user defines the specific structure of the Network through the prototxt text file, and Caffe creates a net by reading and parsing the prototxt file. Net is responsible for initializing all Blob and Layers, and calling feedforward and feedback functions of the layer to realize the operation of the whole network.

(4) Solver module

Solver defines the training strategy of neural network (gradient update strategy), which is commonly used as SGD, Adadelta, Adagrad, and so on. Training policy is also specified by the prototxt file. A complete flow chart of Caffe running a convolutional neural network is shown in Figure 5.

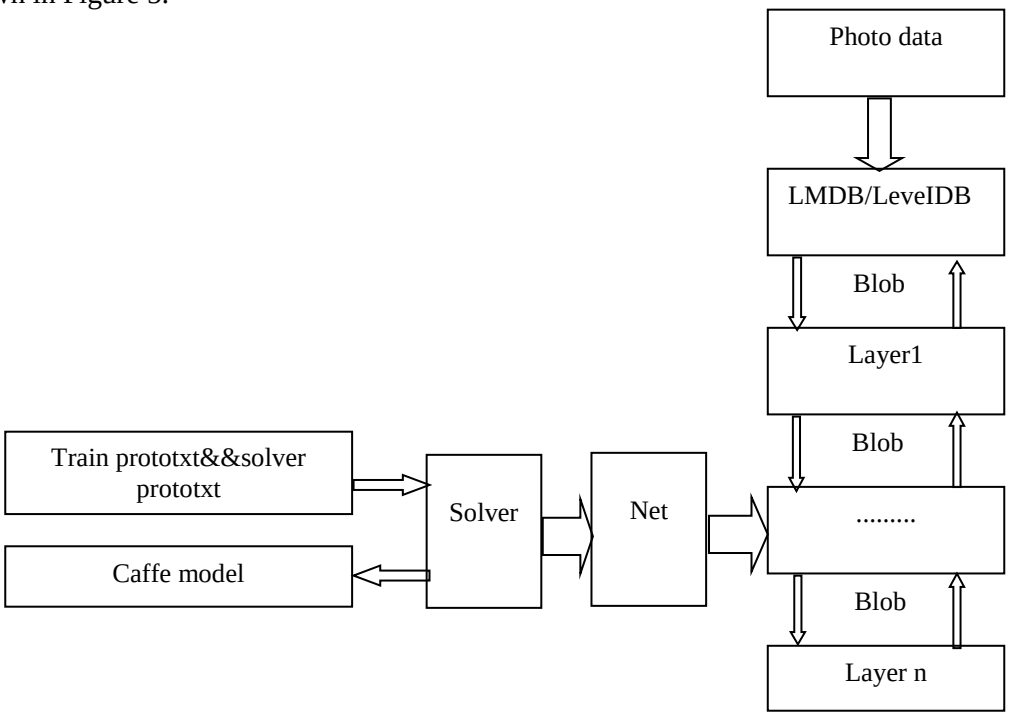

Figure 5: Flow chart of CAFFE running convolutional neural network

Caffe depends on the third-party library, so the third-party library must be transplanted to the ARM platform first. Caffe relies on the following third-party libraries:

Proto Buffer library: a protocol interface developed by Google that can realize the exchange of memory and non-volatile storage media.

Open CV library is the most popular open-source computer vision library in the world, which contains a large number of image processing functions.

Boost library: it is the general name of some C++ libraries that extend the standard library of the C++ language, and it is developed and maintained by the Boost community.

GFlags library: an open-source library developed by Google for processing command line parameters.

GLog library: a library developed by Google for recording application logs.

Blas Library: Linear Algebra Library.

LMDB library: lightning-fast memory-mapped database.

Snappy Library: It is a C++ library used to compress and decompress development packages.

At present, most advanced target detection networks use some recognized image classification networks with superior performance as basic networks to extract image features. For example, the SSD network described in Section 2 uses VGG-16 as a basic feature extraction network. The feature extraction of the SSD network structure is a modification of the standard image classification network. Some layers of the basic network output feature map and then continue convolution operation to reduce the image scale and increase the receptive field, thus obtaining feature maps of different scales and receptive fields. The SSD takes VGG-16 as the basic network and conv4_3 as the low-level feature map to inspect small targets. The two fully connected layers in the sixth part of the original VGG-16 are replaced by two fully rolled layers that do not change the image scale, and the dropout layer is removed. On this basis, four additional convolution layers are added to gradually reduce the image scale, and each convolution layer outputs feature maps, so feature maps with different scales and different receptive fields can be obtained. The VGG-16 contains five groups of convolution layers, each of which uses a 3×3 convolution kernel to extract features, and each group finally reduces the image size by a pooling layer. This network structure is very compact and has a strong feature extraction ability, but its forward speed and backward speed are slightly slower, so if the SSD algorithm uses VGG-16 as the basic feature extraction network, it will reduce the training and detection efficiency of the whole SSD model. In this paper, vehicle detection is carried out on embedded devices, which limits its hardware resources. If the feature extraction of the SSD network takes too much time, it will greatly reduce the speed of target detection, and it is difficult to ensure the real-time performance of detection. The function of the feature extraction network is only to extract image features and input the features into the prediction layer to predict the target position. The feature extraction network can be any other fast and efficient image classification network structure, such as Inception and Mobile Net.

The convolutional neural network has achieved great success in the field of image classification and target detection, and its application in computer vision tasks is becoming more and more common. At present, the development trend of a deep learning algorithm in image classification tasks is to use deeper and more complex networks to improve accuracy, but the cost of these networks to obtain high accuracy is that they consume a lot of memory and run slowly. The embedded system has limited hardware resources and computing capacity. For example, the ARM development board used in this paper needs a lightweight network structure with acceptable accuracy. For a network model with a full convolution structure like SSD, its main computation is convolution operation, so the key to reducing the complexity of a full convolution network is a more efficient convolution layer design. Running an SSD network on an ARM development board with poor performance requires that the SSD feature extraction network must be able to extract image features efficiently and fully.

In this paper, Mobile Netv1 is used as the feature extraction network of SSD, so that the SSD algorithm can run on embedded devices. Google put forward a lightweight deep neural network model called Mobile Net for the application of mobile and embedded vision. This

kind of network greatly reduces the amount of network computation and parameters while maintaining the accuracy of image classification. Based on a streamlined structure, Mobile Net uses depth separable convolution to construct lightweight weighted depth neural networks. At the same time, the network has two global parameters, width factor, and resolution factor, to adjust the size of the network so that it can meet some places with extreme requirements on running speed and memory with minimal changes without redesigning the network model. Mobile Net has good effects in a wide range of visual tasks, such as target detection, image classification, face recognition, fine-grained classification and so on.

Netv1 uses deep separable convolution to improve the standard convolution operation, to reduce the computation and parameters. The principle of depth separable convolution is to divide the common convolution operation into two steps, namely depth convolution and point-by-point convolution. Depth convolution applies a single convolution kernel to each input channel, and point-by-point convolution uses a 1×1 convolution kernel combined with all depth convolutions to obtain the output. In Mobile Netv1, the computation and parameters of the 1×1 convolution kernel account for 95% and 75% of the whole network respectively, and the whole network is composed of deeply separable convolution modules, which is of full convolution structure. The netv1 network structure is shown in the following table (if DW is included in the table, it means that this layer adopts the method of deep separable convolution combined with 1x1).

Table 2: The netv1 network structure

Type / Stride	Filter Shape	Input Size
Conv / s2	3*3*3*32	224*224*3
Conv dw / s1	3*3*32 dw	112*112*32
Conv / s1	1*1*32*64	112*112*32
Conv dw / s2	3*3*64 dw	112*112*64
Conv / s1	1*1*64*128	56*56*64
Conv dw / s1	3*3*128dw	56*56*128
Conv / s1	1*1*128*128	56*56*128
Conv dw / s2	3*3*128dw	56*56*128
Conv / s1	1*1*128*256	28*28*128
Conv dw / s1	3*3*256 dw	28*28*256
Conv / s1	1*1*256*256	28*28*256
Conv dw / s2	3*3*256 dw	28*28*256
Conv / s1	1*1*256*512	14*14*256
5×Conv dw / s1	3*3*512 dw	1*1*512*512
Conv / s1	14*14*512	14*14*512
Conv dw / s2	3*3*512 dw	14*14*512
Conv / s1	1*1*512*1024	7*7*512
Conv dw / s2	3*3*1024 dw	7*7*1024
Conv / s1	1*1*1024*1024	7*7*1024
Avg Pool / s1 Pool	7*7	7*7*1024
FC / s1	1024*1000	1*1*1024
Softmax / s1	Classifier	1*1*1000

In netv1, all convolution layers are followed by a batch norm layer and a Re LU layer (nonlinear activation function). The comparison between the module composed of the ordinary convolution layer, batch norm layer, and Re LU layer and the depth separable convolution module is shown in Figure 6.

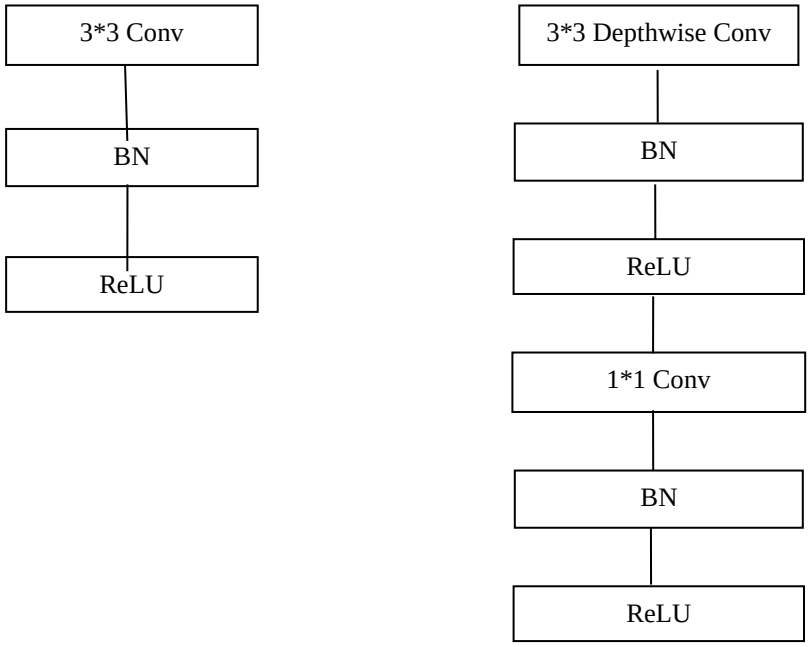

Figure 6: Comparison between standard convolution and depth separable convolution

For the convolution layer with m input channels and n output channels, the number of convolution operations required by the left standard convolution layer in Figure 6 is m*n, while the number of separable modules on the right is only m, which reduces the chance of SIMD (single instruction multiple data) inter-bank memory access in structural design. Meanwhile, 1×1 convolution is implemented by multiplying numbers by vectors, which is convenient for SIMD big data memory access and makes CPU and bandwidth effectively utilized. Therefore, the design of the Mobile Netv1 deeply separable module not only reduces the amount of network computation and parameters but also improves the computing power and data access efficiency of the CPU.

5. Analysis of experimental results

In this paper, Mobile Netv1 is used as the feature extraction network of SSD, and the method of setting the region candidate box increases the rejection loss based on the original SSD loss function, and trains and evaluates the network with the vehicle data in Pascal VOC data set and KITTI data set respectively. The m AP pairs of the original SSD model and the modified SSD model in this paper are shown in [Table 3] (only some kinds of AP data are included in the table). In this paper, Mobile Net V1 is used as the feature extraction network, and the K-Means algorithm is used to determine the region candidate frame of SSD and increase the rejection loss.

Table 3. Comparison of detection effects of two models

Algorithm	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	dog
SSD	80	81.9	87.3	78.7	71.5	50.3	89.9	89	92.3	61	84.7	89.9
Mobile-SSD	74.6	75.5	83.6	71.4	60.9	43.8	84.3	82	89	57.5	73.1	83.2

After testing and comparing the detection accuracy of the two models with the test data set on the computer, it is necessary to test the running speed of the two models on the embedded ARM development board. The image data read by the USB camera is directly input into the model for running detection. In the detection thread, the system time before the image is input into the network is obtained first, and the current system time is obtained after the image input network is detected. The difference between the two is the millisecond time consumed by the model for detecting a frame of the image. In the actual test, it takes an average of 210ms for Mobile Net-SSD to detect a frame of images, and the overall running effect can be 5fps. A comparison of testing speed between original SSD and Mobile Net-SSD running on ARM development board is shown in table 3. Considering that the SOC of the ARM development board is Exynos 4412, which is not the best mobile processor on the market at present, and it is the test result in the environment without GPU, using a more advanced CPU and GPU, running the network model can get higher detection speed.

Table 4: Running speed of different algorithms on ARM development board

Algorithm	Detection speed
SSD	1fps
Mobile-SSD	5fps

The above two models use the same training data and the same training strategy, and after 120,000 iterative pieces of training, they are tested in the same test environment. From the results, it can be concluded that the detection accuracy of using Mobile Netv1 as the feature extraction network does not drop significantly compared with VGG-16, but the running speed is greatly improved.

In practice, it is found that illumination has a great influence on the detection results, and when illumination is weak, there are obvious cases of missing detection. The influences of different illumination conditions on the detection results are shown in Table 5.

Table 5. The missing detection rate of target objects under different illumination conditions

Illumination condition	Vehicle omission rate
weak	24%
good	7%

5. Conclusions

Embedded devices have the advantages of portability, low price, and stable performance, and have a wide range of applications. At first, this paper uses the current mainstream target detection technology based on deep learning to detect vehicles. Based on the SSD network, according to some latest target detection ideas and technical solutions in specific application scenarios, the original SSD network is improved, and an efficient Caffe framework is used to train and test models. In addition, combined with the limited resources of the embedded development board, the feature extraction layer in the SSD network is simplified so that it can be applied to embedded devices. After network training, the image data is acquired by a USB

camera and input into the network for detection. And the detection results are displayed on the LCD. To further speed up the operation of the network, Mobile Net V1 is used as the feature extraction network of SSD. Netv1 uses depth separable convolution to split the standard convolution operation into two steps: depth convolution and point-by-point convolution, which effectively reduces the amount of network computation and parameters, and does not affect the accuracy of the network. The performance of the modified target detection network is tested. The test results show that the detection accuracy of Mobile Net-SSD is slightly lower than that of the original SSD network, but compared with the running speed on the ARM development board, the running speed of Mobile Net-SSD is much faster than that of the original SSD network. In the actual road test, the influence of different illumination conditions on Mobile Net-SSD detection is tested, and it is found that it is greatly influenced by illumination.

References

[1] M. –H. Heo, J. Lee, V. T. Tran, E. –Y. Lee, M. Kim, and D. Kim, "Comparison of user experiences of augmented reality-based car maintenance contents using mobile services and Hololens," Asia-pacific Journal of Convergent Research Interchange, SoCoRI, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.3, no.4, December, pp.57-63, (2017), DOI:10.14257/apjcri.2017.12.11

[2] D. Kim, "Normalization of input vectors in deep belief networks (DBNs) for automatic incident detection," Asia-pacific Journal of Convergent Research Interchange, SoCoRI, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.4, no.4, December, pp.61-70, (2018), DOI:10.14257/apjcri.2018.12.07

[3] D. Kim, "Deep learning neural networks for automatic vehicle incident detection," Asia-pacific Journal of Convergent Research Interchange, SoCoRI, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.4, no.3, September, pp.107-117, (2018), DOI:10.14257/apjcri.2018.09.11

[4] S. –K. Choi, and D. –S. Baik, "Analytical approach on fuel tank design for LPG vehicle," Asia-pacific Journal of Convergent Research Interchange, FuCoS, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.6, no.6, June, pp.55-63, (2020), DOI:10.21742/apjcri.2020.06.06

[5] R. M. Fikri and M. Hwang, "Electric vehicle waiting time prediction with OpenCV and support vector regression," Asia-pacific Journal of Convergent Research Interchange, FuCoS, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.6, no.10, October, pp.137-146, (2020), DOI;10.47116/apjcri.2020.10.11

[6] R. T. Schirrmeiste, J. T. Springenberg, and L. Fiedere, "Deep learning with convolutional neural networks for EEG decoding and visualization," Human Brain Mapping, (2017).

[7] H. C., A. Q. D., A. L. Y., "VoxResNet: Deep Voxelwise residual networks for brain segmentation from 3D MR images," Sciencedirect, pp.446-455, (2018)

[8] Chen, M. Y. Liu, O. Tuzel, "R-CNN for small object detection," Asian Conference on Computer Vision, (2016)

[9] R. Qian, Q. Liu, and Y. Yong, "Road surface traffic sign detection with hybrid region proposal and fast R-CNN," 2016 12th International Conference on Natural Computation and 13th Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). IEEE, (2016)

[10] L. Wang and H. Zhang, “Application of faster R-CNN model in vehicle detection,” Journal of Computer Applications, (2018)

[11] P. Purkait, C. Zhao, and C. Zach, “SPP-Net: Deep absolute pose regression with synthetic views,” British Machine Vision Conference (BMVC 2018), (2017)

[12] H. Liu, L. Peng, and J. Wen, “Multi-occluded pedestrian real-time detection algorithm based on preprocessing R-FCN,” Opto-Electronic Engineering, (2019)

[13] M. J. Shafiee, B. Chywl, and F. Li, "Fast YOLO: A fast you only look once a system for real-time embedded object detection in video," Journal of Computational Vision and Imaging Systems, vol.3, no.1, (2017)

[14] B. H. Kwon and J. K. Kim, “Image searching using a cascade of HOG-kNN,” Itc-ccc: International Technical Conference on Circuits Systems, (2015)

[15] H. Song, W. Yang, and J. Yang, “The improved SIFT algorithm based on rectangular operator and its parallel implementation,” Journal of information technology research, vol.12, no.1, pp.1-17, (2019)

[16] Y. N. Lin, T. Y. Hsieh, and J. J. Huang, “Fast iris localization using Haar-like Features And Adaboost Algorithm,” Multimedia Tools and Applications, vol.79, no.12, (2020)

[17] S. Park and H. Park, “Performance comparison of multi-class SVM with oversampling methods for imbalanced data classification,” International Conference on Broadband and Wireless Computing, Communication and Applications. Springer, Cham, (2020)

[18] S. Georgios, S. Charalampos, H. Arman, "Reverse adaptive Krill Herd locally weighted support vector regression for forecasting and trading exchange-traded funds," European Journal of Operational Research, vol.263, no.2, (2017)

[19] P. Anastasios, P. Sotirios, Chatzis, and S. Vasilis, “A stacked generalization system for automated FOREX portfolio trading,” Expert Systems With Applications, (2017)

[20] S. Garea, D. B. Heras, and F. Arguello, “Caffe CNN-based classification of hyperspectral images on GPU," Journal of Supercomputing, vol.75, no.3, pp.1065-1077, (2019)

This page is empty by intention.