International Core Journal of Engineering Volume 7 Issue 1, 2021
ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0021

Real-Time Target Detection based on Raspberry Pi and Tiny-YOLO

Jun Liu, Wenting Feng, Lei Liao, Tao Yang

College of information science and technology, Chengdu University of Technology (College of network security, Oxford Brooks College), Chengdu, 610059, China.

Abstract

With the advent of the information age and the continuous development of deep learning technology, more and more intelligent applications emerge as the times require. However, the traditional hardware equipment used for training and computing is usually based on the graphics processing unit (GPU), which often faces the disadvantages of high hardware procurement cost and energy consumption in practical application. Therefore, to balance the cost and algorithm availability in the existing deep learning system, this paper mainly introduces a real-time target recognition and detection based on a Raspberry Pi (RPi) and Tiny-YOLO algorithm after a simplified YOLO algorithm. Through the training model and test tuning in the actual environment, it is found that the final result meets the actual use, and the processing speed is 2 frames. The experiment shows that by deploying the Raspberry Pi mobile platform, combined with the Tiny-YOLO algorithm, the running cost can be greatly reduced in the case of practical application.

Keywords

Raspberry Pi; Tiny-YOLO; Target detection; VOC2012.

1. Introduction

The importance of vision is self-evident for human beings, 80% of human information is obtained through vision. Similarly, with the increasingly mature application of deep learning technology, vision is also very important for the computer. Computer vision refers to imitating human vision. First, the camera and other sensor equipment are used to obtain the surrounding image, and then the obtained visual image is analyzed and processed, such as analysis, storage, representation, compression, etc., to realize human biological vision can be regarded as the simulation of biological vision by computer [1].

Traditional target detection needs powerful computing power to support, so it needs a PC with powerful computing power as the main computing platform. However, the control of hardware cost and power consumption can not meet the ideal requirements, which hinders the development of real-time target detection to miniaturization and lightweight. In recent years, due to the rapid development of big data and cloud computing technology, images are sent to the cloud for detection, and then the results are sent back after the cloud detection. The computing power level has been further improved, and the deep learning technology has also ushered in rapid development. However, more and more real-time data generated by edge devices need to be processed in time, and the data transmission requires higher bandwidth and real-time processing the data transmission of the principle wastes a long time and can not meet the requirements of real-time performance. At the same time, security also has dangers [2].

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0021

This paper is mainly based on the Raspberry Pi 4B development board as the basic computing platform. The platform uses a 64-bit 1.5GHz Cortex A72 architecture four-core processor, which has three times faster than the previous generation and provides a variety of memory options. The latest Raspberry Pi 4B version supports 8G running memory.

Figure 1. Raspberry Pi development board

The onboard function is also very powerful, with Bluetooth 5.0 BLE, Wi-Fi, Gigabit Ethernet, two USB 3.0 ports, two USB 2.0 ports, and a micro HDMI port, which can support a dual-screen display with 4K 60Hz resolution. As shown in Figure 1.

Deep learning can greatly promote the target detection algorithm. At present, the mainstream target detection algorithms based on deep learning are mainly divided into one-stage and two-stage. The representatives of two-stage are mainly R-CNN[3] series, and one stage representatives are mainly SSD[4] and YOLO (you only look once)[5] series. In general, the accuracy of the two-stage target detection algorithm is higher than that of the one-stage algorithm, but the speed of the one-stage algorithm is faster. YOLO is one of the most excellent target detection frameworks. Its principle is to directly predict the object boundary and category probability with CNN. Its 3-degree accuracy based on other models has been maintained well, compared with other models. YOLO has made a lot of simplification, but this paper applies Raspberry Pi and other mobile platforms, and the deployment of YOLO is still quite difficult. The framework selected in this paper is Tiny-YOLO, which has a simpler structure and lower computational complexity than YOLO. After experimental verification, the network can achieve an average processing speed of 2 frames / s on the Raspberry Pi 4B Development Board [6]. In this paper, based on the VOC2012 data set, combined with the user-defined data set, the data training set suitable for application scenarios is made. By using the data set training, this paper has achieved a good recognition effect and has been verified in practical application.

2. Construction of target detection model

2.1 Target detection network framework

As the hardware used in this paper is mainly Raspberry Pi 4B as the running environment of the algorithm, storage resources and computing resources are very limited, so deep level algorithms such as R-CNN, Fast R-CNN, Faster RCNN [7] are not applicable. Therefore, to ensure that the final recognition accuracy does not drop too much, we should try our best to improve the operation speed and reduce the hardware requirements.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0021

2.2 Tiny-YOLO framework

YOLO is an object recognition and location algorithm based on a deep neural network. Its biggest characteristic is that it runs very fast and can be used in a real-time system. Now YOLO has developed to the v4 version. Tiny-YOLO[8] is the basic network of YOLOv2[9], but compared with YOLOv2, it has a simple structure and low computational complexity. The network is a forward network composed of 9 convolution layers and 6 maximum pooling layers alternately, which leads to the loss of information in layer to layer transmission, and the lack of detection accuracy due to the failure to make full use of the characteristic information of the convolution layer. After multiple convolution operations, the image features become smaller, which is easy to cause missing and false detection of distant objects or small targets. The specific structure is shown in Figure 2.

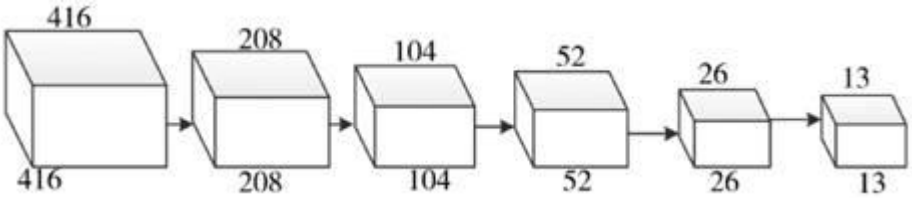

Fig. 2. Structure of Tiny-YOLO

Table 1. Parameters of Tiny-YOLO

Layer	Filters	Size	Input	Outputs
0 conv	16	3*3/1	416*416*3	416*416*16
1 max		2*2/2	416*416*16	208*208*16
2 conv	32	3*3/1	208*208*16	208*208*32
3 max		2*2/2	208*208*32	104*104*32
4 conv	64	3*3/1	104*104*32	104*104*64
5 max		2*2/2	104*104*64	52*52*64
6 conv	128	3*3/1	52*52*64	52*52*128
7 max		2*2/2	52*52*128	26*26*128
8 conv	256	3*3/1	26*26*128	26*26*256
9 max		2*2/2	26*26*256	13*13*256
10 conv	512	3*3/1	13*13*256	13*13*512
11 max		2*2/1	13*13*512	13*13*512
12 conv	1024	3*3/1	13*13*512	13*13*1024
13 conv	1024	3*3/1	13*13*1024	13*13*1024
14 conv	30	1*1*1	13*13*1024	13*13*30
15 detection				

As shown in Table 1, from the input to the final output of the network, the characteristic scale decreases by 32 times. The overall network model belongs to the end-to-end network, causing a lot of losses in the middle.

2.3 Improving network structure

To optimize the shortcomings of Tiny-YOLO proposed in the previous paper, we add the upper sampling layer to fuse the features of the previous convolution layer to improve the feature learning of the target. In the experiment, we use two feasible methods to verify, such as output 1 and output 2

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0021

shown in Figure 3. The first method is to perform a 2 * deconvolution operation on the convolution layer with the size of 13 * 13, and then add the corresponding pixels of the previous feature layer 26 * 26 for feature fusion. Finally, the fused features are convoluted, to normalize the final scale to 13 * 13 size. Method 2 is to deconvolute the 13 * 13 convolution layer twice, then add the corresponding pixels of the previous feature layer 52 * 52, and then perform the convolution operation with a step size of 2. To improve the final positioning accuracy, this paper adds a horizontal line (that is, the feature after the previous convolution operation is fused with the 26 * 26 feature layer), and finally normalizes the final feature scale to the size of 13 * 13, make the final prediction.

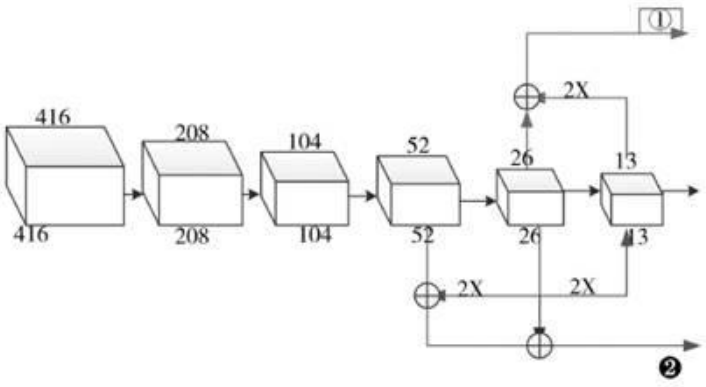

Fig. 3. Improved Tiny-YOLO

In the above feature fusion, to avoid the overlapping effect after feature fusion, it is necessary to carry out 1 * 1 convolution operation after each pixel superposition to eliminate the negative impact of feature fusion.

3. Target detection model deployment

3.1 Raspberry Pi and software program

After the training of the model described above, it is finally deployed and applied to Raspberry Pi. The version of Raspberry Pi used in this paper is the 4B development board. The processor of Raspberry Pi is ARM cortex A72 1.5GHz and memory is 4G RAM. Raspberry Pi OS, the official and deeply customized hardware driver and software program of Raspberry Pi OS, has general computing power, but it has low cost and strong applicability. The software uses Python 3.4, NNPACK neural network acceleration library, and OpenCV3 as data processing.

3.2 Making of training data set

To improve the accuracy of the training data, this paper uses the VOC2012 data set and uses the LabelImage to label the data. The VOC2012 dataset is an upgraded version of the VOC2007 dataset, with a total of 11540 images. For the detection task, there are 11540 images in total, and the trainval / test of VOC2012 includes all corresponding images from 2008 to 2011. Trainval has 11540 images with 27450 objects, which are divided into 20 classes. As shown in Figure 4.

Figure 4. Classification of VOC2012 dataset

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0021

3.3 Development and training process

In the development and training phase, the data needs to be preprocessed, and the algorithm model is compiled into the format that Raspberry Pi can execute at the end of the final training. The specific process is shown in Figure 5.

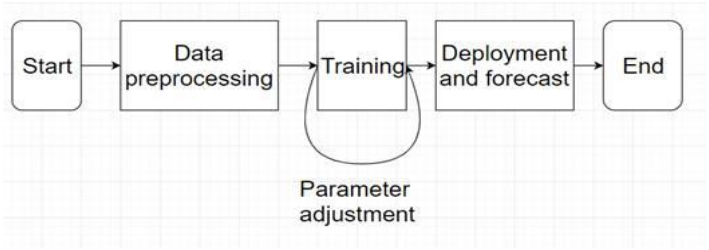

Figure 5. development and training process flow chart

Step 1. Data preprocessing

The data set is made into a training set and test set, and then the data set is transformed into an LMDB format, which is convenient for the subsequent algorithm to read and use.

Step 2. Model training

After the data set is made, the target detection algorithm is designed based on the Caffe[10] framework, and the server with GPU operation ability is used for training. Finally, the detection model in Caffe format, namely caffemodel weight file, is obtained. In the process of algorithm training, the server equipped with 11GB RTX2080Ti is used for training. The proposed model is trained by stochastic gradient descent (SGD), with a momentum of 0.9 and attenuation factor of 0.0005. The number of training iterations set is 250000. To improve the recognition accuracy of the algorithm, a multistep learning rate attenuation strategy is adopted, and the learning rate is set to 0.006.

Step 3. Deployment and forecasting

The detection model is deployed in the development board 4B of Raspberry Pi. Raspberry Pi is connected to the local area network through Wi-Fi. The computers in the local area network can remotely log in to the Raspberry Pi desktop through VNC. The operation of Raspberry Pi can be viewed in real-time on the computer side, and the target location, category, and score can be inferred.

4. Experimental results and analysis

In this paper, the Tiny-YOLO algorithm is mainly used in the Raspberry Pi development board with limited computing resources and storage resources, to achieve lower cost real-time target detection and recognition. To improve the practicability of the algorithm, this paper mainly uses the data set labeled by LabelImage after being collected by VOC2012 and camera. As shown in Fig. 6 is the loss curve of the data set training, we can see that the stable convergence results are obtained by using the data set training.

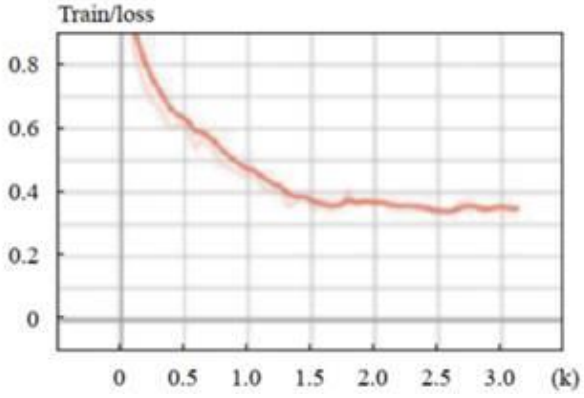

Figure 6. Training test data set

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0021

To compare the difference between the method adopted in this paper and the traditional GPU detection and recognition algorithm, the main indicators used in this paper are FPS and mAP, FPS is the number of recognition frames per second, mAP is the average accuracy. Mean average precision (mAP) is used to measure the proportion of correct targets among the detected targets. The test results are shown in Table 2.

Table 2. Comparison test results of Tiny-YOLO algorithm and other algorithms

Algorithm	FPS	mAP
HOG+SVM (Raspberry Pi)	0.5	55
SSD (GPU)	20	87.9
Tiny-YOLO (Raspberry Pi)	2	75.8

In the comparative experiment, this paper mainly analyzes from two aspects: firstly, in the same computing environment, the traditional machine learning algorithm HOG-SVM is compared with the algorithm in this paper. Through the comparative experiment, we can see that the algorithm in this paper is better than the previous algorithm and stronger in FPS; besides, by comparing the SSD algorithm deployed in GPU environment with the algorithm in this paper, It can be seen that FPS can reach 20 in GPU environment, while there is a gap between mAP and this algorithm, but compared with the power consumption and procurement cost in GPU environment, the algorithm, and architecture proposed in this paper have more advantages. Figure 7 shows the result of Tiny-YOLO algorithm recognition in the Raspberry Pi computing environment.

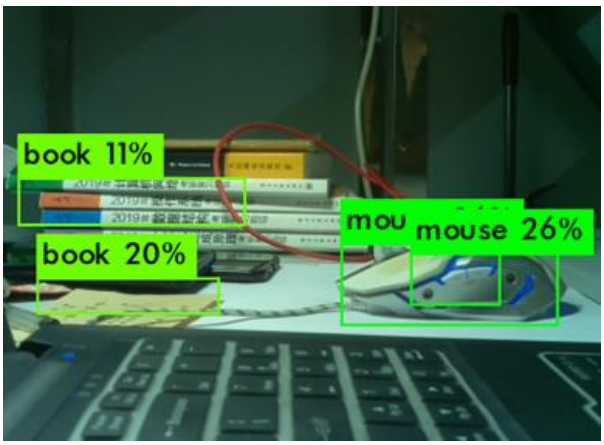

Figure 7. Recognition effect of Tiny-YOLO algorithm

5. Epilogue

In this paper, based on Raspberry Pi as the main computing platform, by embedding the Tiny-YOLO algorithm into Raspberry Pi, we can run real-time target detection and recognition algorithm on the platform with limited computing resources, and verify it in practical application. The experimental results show that the algorithm achieves the design goal, achieves satisfactory detection speed and high accuracy, and can be used as an important basis for unmanned driving environment perception, and has great significance in edge calculation.

References

[1] Huang J, Rathod V, Sun C, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 7310~7311.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0021

[2] Dalal N, Triggs B. Histograms of oriented gradients for human detection. Proceedings of 2018 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Diego, CA, USA. 2005. 886–893.

[3] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition. Columbus, OH, USA. 2014.580–587.

[4] Liu W, Anguelov D, Erhan D, et al. SSD: Single shot MultiBox detector. Proceedings of the 14th European Conference on Computer Vision. Amsterdam, the Netherlands. 2016. 21–37.

[5] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection. Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, NV, USA. 2016. 779–788.

[6] Ionica MH, Gregg D. The Movidius myriad architecture's potential for scientific computing. IEEE Micro, 2015, 35(1):6–14.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun.Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497 [cs] 2015.

[8] Francois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); IEEE,2017 1800–1807.

[9] ZHANG Junfei,BI Zhisheng,WU Xiaoling. Two-way LSTM emotion analysis based on word vector doc2vec[J]. Computer and digital engineering,2018,46(12):2385-2389,2399.

[10] Bahrampour S, Ramakrishnan N, Schott L, et al.Comparative study of deep learning software frameworks. arXiv: 1511.06435, 2015.