Field Robot Environment Sensing Technology Based on TensorRT

Bo Dai, Chao Li(✉), Tao Lin, Yong Wang, Dichen Gong, Xiao Ji, and Bosong Zhu

Chengdu University of Technology, Chengdu 610059, China

Abstract. The inference speed of complex deep learning networks on embedded platforms of mobile robots is low, and it is difficult to meet actual application requirements, especially in complex environments such as the wild. This experiment out motion blur processing on the data set to improve the robustness, by using NVIDIA inference accelerator TensorRT to optimize the operation, the computational efficiency of the model is improved, and the inference acceleration of the deep learning model on the mobile quadruped robot platform is realized. The experimental results show that, on the test data set, the method achieves 91.67% mAP of 640 × 640 model on the embedded platform Nvidia Jetson Xavier NX. The reasoning speed is about 2.5 times faster than before, reaching 35 FPS, which provides support for the real-time application of mobile robot environment sensing ability in the field.

Keywords: Embedded platform · Motion blur · YOLOv5s · TensorRT · Environmental sensing

1 Introduction

With the rapid development of mobile robots, robots can be seen everywhere in all walks of life, but as the application area becomes more and more extensive, the problems they face will follow. Among them, mobile robots have an increasing demand for environmental sensing capabilities, especially in complex environments in the wild, robots have very high requirements for the real-time and accuracy of environmental perception. In addition, compared with traditional machine learning methods, deep learning has strong learning capabilities and can make better use of data sets for feature extraction, so as to effectively carry out target tracking, target orientation perception, obstacle avoidance and control judgment.

Target detection based on deep learning involves two steps [1]: In the first step, Train large amounts of tag data on the processor, the neural network learns millions of weights or parameters so that it can map the sample data to the correct response. In the second step, the newly collected data is predicted by the trained model. For automatic driving, mobile robot and other applications, inference and sensing need to be completed in real time, so it is very important to control the high throughput and response time in computing. However, the processor performance of general mobile robots is not very

X.-J. Liu et al. (Eds.): ICIRA 2021, LNAI 13013, pp. 370–377, 2021.
https://doi.org/10.1007/978-3-030-89095-7_36

high, and even many of them are not equipped with GPU. The reasoning speed of complex deep learning network on mobile robot platform is low, and it is difficult to meet the practical application requirements, especially in the application background of complex environment such as field.

2 Target Detection System

Target detection is to determine the position of the target in the input image and identify the category of the target in the input image. Target detection has always been the most challenging problem for mobile robots to perceive the environment in a complex environment due to the different appearance and posture of various objects, the interference of processor computing force and illumination, occlusion and other factors in the imaging process.

At present, target detection algorithms based on deep learning are mainly divided into two categories: first-stage target detection algorithms and second-stage target detection algorithms. Two-stage target detection algorithms include FAST R-CNN, Faster R-CNN, and R-FCN, etc. The first-stage target detection algorithms include: SSD, YOLOv3, YOLOv4, YOLOv5, etc. The first-stage target detection algorithm can directly predict the probability of the object category and the coordinates of its position without generating candidate regions, so as to complete target detection directly.The problem that the two-stage algorithm can not meet the real-time detection is solved [2–4].

YOLOv5, launched by Ultralytics in 2020, has the advantages of small size, fast speed and high precision, and is implemented in an ecologically mature PyTorch, with easy deployment and implementation. YOLOv5 includes YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x models. YOLOv5s image reasoning speed is up to 0.007 s, that is, it can process 140 frames per second, which meets the requirement of real-time detection of video images. Meanwhile, it has a smaller structure. The weight data file of YOLO5s version is 1/9 of that of YOLO4, and the size is 27 MB. It's network model structure still follows the overall layout of YOLOv4, which is mainly divided into four parts, namely Input, Backbone, Neck and Prediction [5].

In the original model of YOLOv5s, $GIoU$ loss was used as the bounding box loss. Compared with the original IoU, the optimization of $GIoU$ loss was to increase the penalty of erronet selection. After the training process, the detection effect of boxes with different proportions was better, and the principle was shown as follows [6]:

$$IoU = \frac{\left|B \cap B^{gt}\right|}{\left|B \cup B^{gt}\right|} \tag{1}$$

$$\mathrm{L}_{GIoU} = 1 - IoU + \frac{\left|C - B \cup B^{gt}\right|}{|C|} \tag{2}$$

However, $GIoU$ still has the problem of unstable target box regression, and the $GIoU$ regression strategy for the non-overlapping target detection box may degenerate into the regression strategy of IoU. The main problem is that when the overlap area is 0, $GIoU$ tends to overlap the fastest way between the detection box and the target box, and then $GIoU$ punishment mechanism gradually becomes ineffective. In order to solve this problem, the loss function should consider the overlapping area, distance from the center point and aspect ratio of the predicted box and the real box, and use a *CIoU* that is more consistent with the regression mechanism, as shown in the formula:

$$L_{CIoU} = 1 - IoU + \frac{p^2\left(b, b^{gt}\right)}{c^2} + av \tag{3}$$

$$v = \frac{4}{\pi^2}\left(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h}\right)^2 \tag{4}$$

$$a = \frac{v}{(1 - IoU) + v} \tag{5}$$

In the (3) loss function, the center point of the detection box and the target box is denoted by b and b^{gt}, and the Euclide distance is p. c is the slant distance of the smallest rectangle covering the detection box and the target box that is directly optimized and the speed is faster. In the (4) w^{gt} and h^{gt} are the width and height of the real box, w and h are the width and height of the prediction box. a is the parameter used to balance the proportions [7].

3 TensorRT

TensorRT is a high-performance inference optimizer from NVIDIA that provides low latency and high throughput deployment reasoning for deep learning applications. TensorRT can be used for inferred acceleration in large data centers, embedded platforms or autonomous driving platforms.

TensorRT optimization methods mainly have the following ways, the most important is the first two.

1) Layer Fusion or Tensor Fusion: When deploying model inference, TensorRT automatically parses the network computation diagram and looks for the optimized subgraph. The combined calculation diagram has fewer layers and takes up fewer CUDA(Compute Unified Device Architecture) cores, so the overall model structure (Fig. 1 on the right side)is smaller, faster, and more efficient, thus reducing reasoning delays [8].
2) Weight &Activation Precision Calibration: Data precision can be appropriately reduced during deployment reasoning, and lower data precision can reduce memory footprint and latency, as well as smaller model size.
3) Kernel auto-tuning.
4) Dynamic Tensor Memory.
5) Multi-stream Execution.

4 Data Preprocessing and Training

The experiment uses the data set of field environment images collected by lab members to train and test. It contains 9,800 samples and expands on the original data set, using

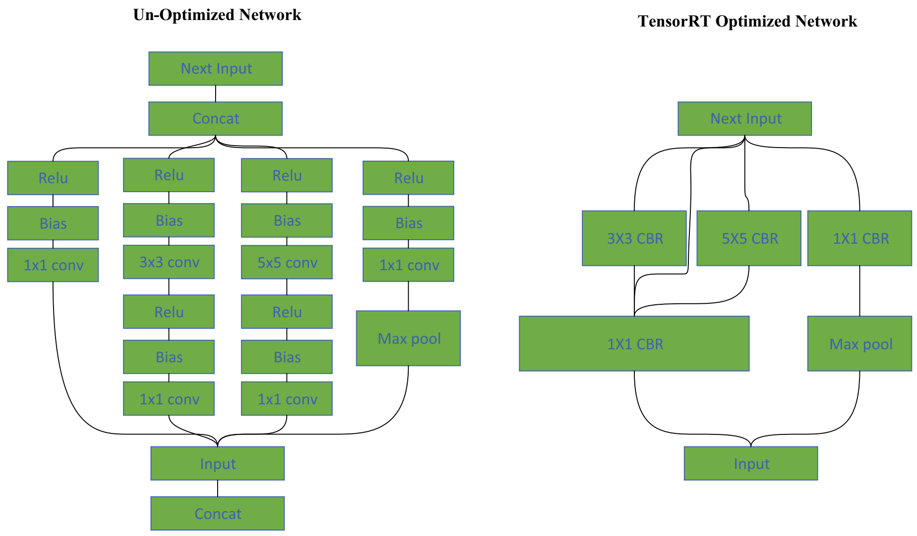

Fig. 1. Interlayer and tensor fusion (Tensorrt reduces the number of layers by combining layers horizontally or vertically (the combined structure is called CBR, meaning Convolution, Bias, and Relu layers are fused to form a single layer) Horizontal merges combine the convolution, bias, and activation layers into a single CBR structure that occupies only one CUDA core.)

both flip and rotation methods. The datasets used for training, validation, and testing included 70%, 10%, and 20% samples, respectively. The Labelimg tool is used to mark and detect the objects in the picture, including 19 categories such as people, horses, cattle, sheep, deer, etc.

As the application object is deployed on the mobile robot in the field, the mobile robot is in a complex environment with external interference and unstable movement, plus the vibration of its own movement, the ZED camera deployed by the mobile robot adopts a binocular rolling curtain camera, so collected photos can be blurry.

The fuzzy image restoration model is shown in the formula:

$$g(x, y) = h(x, y) \times f(x, y) + n(x, y) \tag{6}$$

Wherein (6), $g(x, y)$ is the observation image, $f(x, y)$ is the target image, $h(x, y)$ is the degradation function, and $n(x, y)$ is the noise function. The target is to restore $f(x, y)$ according to the observation image $g(x, y)$ and some prior or estimated information [9]. It's very difficult, and adding images to remove motion blur on mobile embedded platform will increase computing power consumption and affect real-time performance. Therefore, data enhancement was carried out on the collected samples, and 30% of the images were used for motion blur to enhance their ability to identify fuzzy targets. In addition, the motion and vibration of the mobile robot generally exist in the Z direction (the y-axis direction of the image), while the movement of the object in the environment is in the Y-direction (the x-axis direction of the image). Therefore, we create a fuzzy kernel for 30% of the data in the data set by using the fspecial function of Matlab, which convolves with the image to approximate the linear motion of the camera, the Len length is 9 pixels, and the theta values 90 and 0 are the horizontal and vertical motion angles. Figure 2, for the Raw data and motion blurred data.

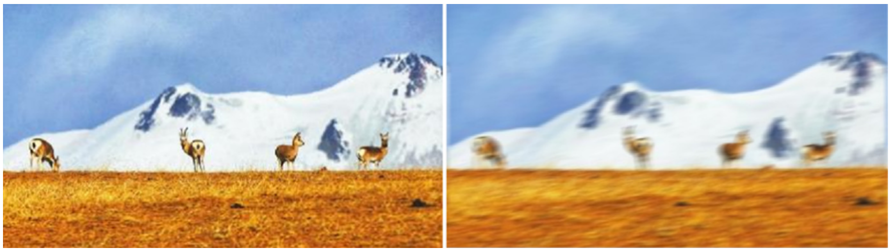

Fig. 2. Raw data and motion blurred data

Based on the PyTorch deep learning network framework, parallel GPU is used for training in this paper. Firstly, the classification model based on YOLOv5s is trained on the workstation. The batch size of each training is 32, and the Stochastic Gradient descent optimization algorithm with a momentum of 0.9 is adopted. The initial learning rate is 0.001 and the weight attenuation coefficient is 0.0005. The cross entropy loss function is used to calculate the loss. The Pt model trained by PyTorch is converted into the ONNX model and then imported into TRT, which is optimized by TensorRT. NX uses the trained classification model and TensorRT to reason and accelerate the detection of categories in the picture.

5 Optimization Test and Analysis Based on Jetson NX

5.1 Experiment Platform

The field mobile robot platform adopts YoboGO full open source quadruped bionic robot dog development platform. YoboGO is a lightweight quadruped robot with high dynamic performance, equipped with advanced gait planning, leg and foot control and environment awareness technology. In this project, the NVIDIA Jetson Xavier NX development Board is mounted on the robot dog. It is connected with the UP Board controller of the robot dog through the network cable. In terms of vision, Zed camera is used to connect with NX. At the same time, ultrasonic radar and BD/GPS positioning module are deployed on the body to improve environmental sensing.

The NVIDIA Jetson Xavier NX features 384 CUDA Cores, 48 Tensor Cores, 6 ARM CPUs and 2 NVIDIA Deep Learning Accelerator engines. Accelerated computing power of up to 21 TOPS is available in the 15W 6 core to run modern neural networks in parallel and process data from multiple high-resolution sensors. Camera adopts ZED stereo camera, adopts shutter shutter, and supports CUDA. Figure 3, for the mobile robot.

The development environments shown in Table 1:

5.2 Accelerated Testing

Motion blur is added to the data set, which can handle the recognition of blurred images collected by the mobile robot in the process of movement. The detection and recognition

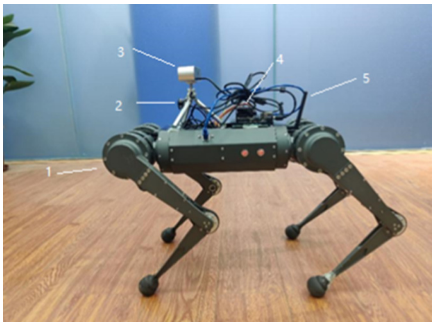

Fig. 3. Mobile robot(1.Yobogo quadruped robot 2. Ultrasonic radar 3. Zed camera 4. Jetson NX 5.BD /GPS positioning module)

Table 1. Development environment.

Hardware parameters	CPU: 6-core NVIDIA Carmel ARM®v8.2 64-bit CPU 6MB L2 + 4MB L3 RAM: 8G GPU: NVIDIA Volta™ GPU with 48 Tensor Cores GPU Memory: 8G CUDA Core: 384-core
Development environment	Operating System: Jetpack 4.4.1 (ubuntu 18.04) CUDA Version: 10.2.89 cuDNN Version: 8.0.0.180 TensorRT Version: 7.1.3.0 Programming Language: Python 3.8 Neural Network Framework: PyTorch 1.8.0

rate of motion blur images tested on NX is only about 10% lower than that of normal pictures. Figure 4, normal and motion blur images target detection results.

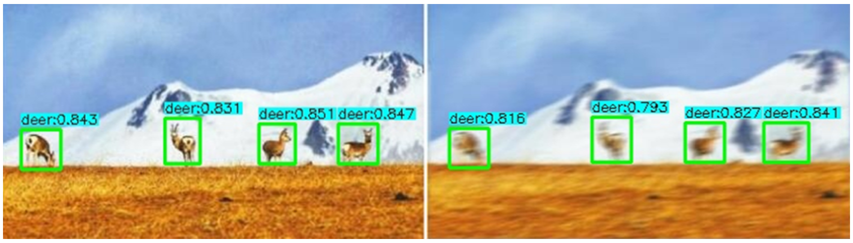

Fig. 4. Normal and motion blur images target detection results

The implementation of YOLOv5s in NX is carried out by PyTorch. In the mode of 15 W 6 core with maximum power, the GPU runs the program with full load and identifies various targets in the video stream without any object tracking. The FPS of different model image sizes with TensorRT versus without TensorRT is as Fig. 5, Use different model image size comparison, whether to use TensorRT to accelerate the comparison. When the model image size is 640 × 640, the speed of inferring a picture before acceleration is 70 ms, and the speed of inferring a picture after acceleration by TensorRT is 27 ms, which is 2.5 times higher.

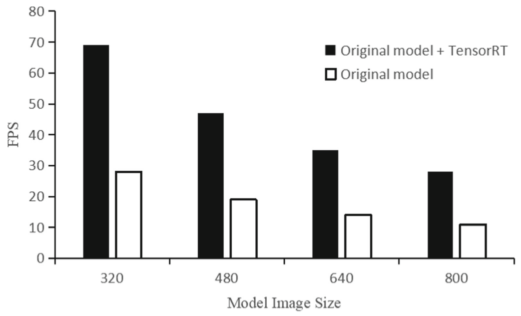

Fig. 5. Inference speed on NX

This paper uses the Pasca criterion to evaluate the model. When the intersection ratio between the ROI of the target object and the ground truth is greater than 0.3, the ROI will be marked as a positive sample by the standard. FP and FN represent false positives (false detected targets) and false positives (undetected targets), respectively. The two evaluation indexes of precision rate and recall rate can be respectively as follows:

Precision rate:

$$accurate = \frac{TP}{TP + FP} \tag{7}$$

Accuracy ratio refers to the ratio of the correct objects detected to all detected objects.

Recall rate:

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

The recall rate is the ratio of correctly detected targets to the total number of labeled targets.

As shown in Fig. 6, the accelerated and non-accelerated images of different image size models correspond to the mAP and recall comparisons, the mAP and recall rate accuracy varies by less than 1% with TensorRT acceleration.

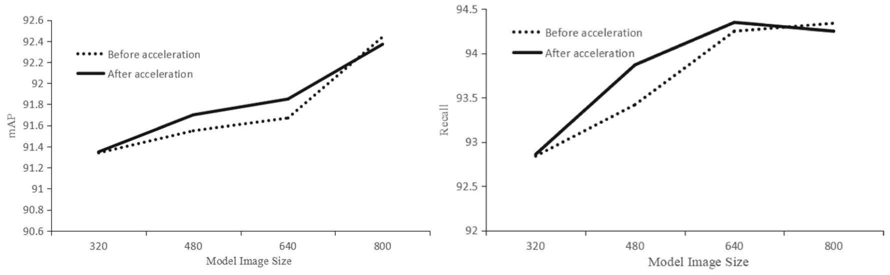

Fig. 6. Precision rated and recall rate

6 Conclusion

This paper takes the mobile robot equipped with NVIDIA Jetson NX as the research object. By building a Pytorch neural network framework on it to run YOLOv5s to perform target detection, it is found that its real-time performance is not ideal. Therefore, it is proposed to use TensorRT for reasoning optimization acceleration. First, the darknet trained model is converted into an ONNX model, and then converted into a TensorRT model, and then TensorRT is used for target detection. This method can increase the FPS by 2.5 times at 640 $\times$ 640 model without loss of accuracy. It has successfully proved that this method can make edge computing performance more excellent under the NVIDIA GPU and TensorRT acceleration, and can improve the reasoning speed while ensuring the accuracy. It can improve the environmental perception ability of mobile robots in complex environments in the wild, and provides a solution for subsequent real-time target tracking and target orientation calculation.

References

1. Du, X., Cai, Y., Wang, S., et al.: Overview of deep learning. In: 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), pp. 159–164. IEEE, (2016)
2. Ruan, J.: Design and implementation of target detection algorithm based on YOLO. Beijing University of Posts and Telecommunications, Beijing (2019). (In Chinese)
3. Tan, J.: Research on an improved YOLOv3 target recognition algorithm. Huazhong University of Science and Technology, Wuhan (2018). (In Chinese)
4. Yan, H.: Research on Static Image Target Detection Based on Deep Learning. North China Electric Power University, Beijing (2019). (In Chinese)
5. Liu, Y., et al.: Research on the use of YOLOv5 object detection algorithm in mask wearing recognition. World Sci. Res. J. **6**(11), 276–284 (2020)
6. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 658–666 (2019)
7. Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., Ren, D.: Distance-IoU loss: faster and better learning for bounding box regression. In: The AAAI Conference on Artificial Intelligence (2020)
8. NVIDIA. NVIDIA Deep learning SDK[DB/OL]. Accessed 27 Nov 2019, https://docs.nvidia.com/deeplearning/sdk/index.html
9. Jian, Z., Zhao, D., Gao, W.: Group-based sparse representation for image restoration. IEEE Trans. Image Process. **23**(8), 3336–3351 (2014)