The Application and Comparison of Object Detection Algorithm in Counter Strike 2

Yifan Cai
ShanghaiTech University
caiyf@shanghaitech.edu.cn

Yu Shi
ShanghaiTech University
shiyu1@shanghaitech.edu.cn

Xihe Yu
ShanghaiTech University
yuxh@shanghaitech.edu.cn

Abstract

In recent years, real-time object detection has garnered significant attention due to its wide range of applications, including in the gaming industry. This paper implements three algorithms, YOLOv7, SSD, and Faster R-CNN for real-time object detection in the game Counter Strike 2. Due to the innovation of the application, we first independently created the dataset of CSGO. Then, by combining these algorithms, our proposed approach aims to achieve superior performance in detecting various objects in real-time game play. We conducted extensive experiments to compare the accuracy, speed, and overall performance. We innovatively proposed numerous applications, for example, by real-time target distance calculation we realize auto-aiming and for better accuracy, we combine the three output results, using K-means,voting, and confidence to finally generate a better prediction.

1 Introduction

1.1 Object detection and its implementation in games

Object detection is a crucial technique in the field of computer vision, which aims to accurately identify and localize multiple target objects of interest from images or videos. Our code supports the input of picture, video and real-time screen capture. This algorithm can be perfectly applied in many different fields like: auto-aiming, automatic grabbing of game items, automatic path-finding in the game, calculation of crowd density, warning of dangerous goods, and intelligent transportation e.t.c.

Our group choose three different algorithms, Faster RCNN, SSD, and yolov7, and implement these 3 algorithm on the VOC2007 dataset first (`http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html`). Then, we made our own dataset, VOC-CS2 and implement the algorithms on it. Here's the link of our dataset and model weight: `https://pan.baidu.com/s/1HENPjGOc-iJ8achmSRfOEw` (key:Rain)

1.2 Composition of our dataset

Since there is currently no ready-made CS2 image dataset available online, we first built the dataset. Our dataset annotation follows VOC format and uses the labelimg library function provided by Python. Our pictures come from real-life game visuals and 3D models of various characters. Our tags include the entire process of the game, such as bomb placement, C4 pickup, C, CT, chicks, and

bomb disposal. In order to improve training efficiency, we preprocess the photos and increase training weights for difficult to recognize photos (see Chapter 4)

2 Algorithms

2.1 Faster R-CNN

FRCNN(Faster region convolution neutral network), including four mainly part: 1.Conv layers: the conv layers is mainly used for the feature extraction.In this part we use the Resnet50 for the backbone network. 2 Region Proposal Network: used to generate the region proposals, judge the negative or the positive by softmax, and the fix the bndbox. 3.Roi pooling: use the feature and the proposals get in 1 and 2, the then transform the data to the full connectivity layer determines target class. 4.Classification: use the data get in 1 and 2, draw the bnd box.

The conv layers mainly include three layers: conv, pooling and relu, each conv have kernel size is three, padding is one,stride is one; Each pooling is kernel size is two, padding is zero, stride is two.It is this setting that causes the conv layer in Conv layers not to change the input and output matrix sizes. A matrix of size MxN is fixed to $(M/16)$x$(N/16)$ by Conv layers. The feature maps generated by Conv layers can all correspond to the original map.

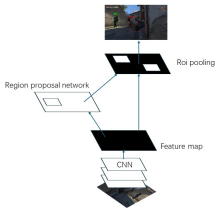


Figure 1: The main processes of Frcnn

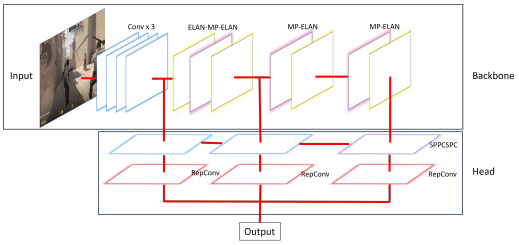


Figure 2: The main processes of YOLOv7

2.2 YOLOv7

YOLO, whose full name is You Only Look Once means that only one CNN operation is needed to recognize the category and location of the object in the figure. Therefore it is also known as one-stage, Region-free method. The YOLO framework has three main components: Backbone,Neck and Head. Backbone mainly extracts the basic features of the image and feeds them through the neck to the head. Neck collects the feature maps and creates a feature pyramid. Finally, the head consists of the output layer with the final detection.

YOLOv7 has introduced several architectural changes to improve speed and accuracy by proposing the E-ELAN network architecture. the E-ELAN is the computational block in the YOLOv7 backbone. The shortest and longest path of the gradient is controlled by an efficient long-range attention network, allowing deeper networks to learn and converge more efficiently. The E-ELAN proposed by the authors uses expand, shuffle, and merge cardinality to achieve improved learning ability of the network without destroying the original gradient path.

2.3 SSD: Single Shot MultiBox Detector

VGG16 (Visual Geometry Group)is a deep convolutional neural network architecture and is constructed by stacking multiple convolutional and pooling layers, followed by fully connected layers for classification. It is mainly used for feature extraction from photos.(see picture 3)

SSD first resizes the original image, generating a square image of 300x300 dimensions through grayscale computation. Then, the layers preceding the Conv5-3 layer of VGG16 (before the green vertical line) are utilized for feature extraction. Subsequent to this, multiple convolutional layers are applied in succession,the sizes are shown in chart. Post six convolutional operations, distinct feature maps are obtained, and for each point on these maps, Default Boxes of six different sizes are constructed. Finally, the algorithm combines these Default boxes, employing Non-maximum

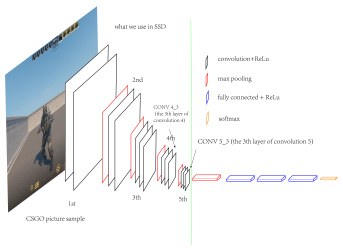


Figure 3: The flow chart of VGG16

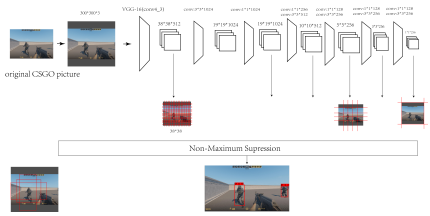
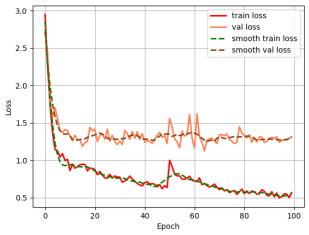


Figure 4: The flow chart of SSD

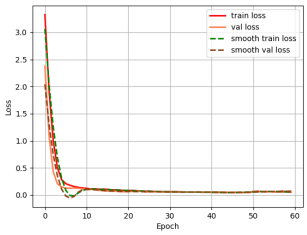
Suppression (NMS) to suppress overlapping or incorrect Default boxes, yielding the final Default boxes set (i.e., detection results). (See picture 4).

3 Comparisons of Performance between Algorithms

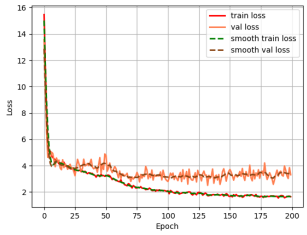
From the loss diagrams of the three algorithms, yolov7 has the best convergence and frcnn converges poorly and has a large loss. Here is the change in loss for the three algorithms as the epoch increases:



Loss of Frcnn



Loss of YOLOv7



Loss of SSD

YOLOv7 has fast processing speed and high accuracy, but may have some limitations in the detection of small and dense targets; Faster R-CNN has high accuracy and versatility, but has high computational resource requirements and slow processing speed; SSD supports multi-scale detection, capable of handling objects of different sizes. It typically performs better in terms of accuracy, especially in detecting small objects. However, it tends to be relatively slower in speed.

MAP Comparison

Algorithm	# of Epoch	Converged mAP
Frcnn	25	∼0.85
YOLOv7	30	∼0.92
SSD	125	∼0.81

4 Implement of the Algorithms

4.1 Innovation in the pre-training stage

Handling picture: To train our model faster, we preprocess the photos we use for training. Firstly, we tried to add a neural network for feature extraction to achieve contrast and brightness adjustment. Due to the unclear effect, we subsequently used the method of adding max pooling to process the images (as game characters always have large pixel blocks and bright colors). (Figure 5 will be shown in the last page)

More samples: We also find that some pictures have low accuracy due to the dimness of the game scene or the scarcity of the that feature. We reverse, pull, and crop the photos with larger loss

and add back into the dataset to increase the number of training samples.(Figure 6 will be shown in the last page)

4.2 Optimization of algorithm

In terms of the pooling layer, the original neural network(Fast R-CNN) uses a pooling layer that is maximum pooling, which can lead to some loss of information (Our samples mainly based on map: inferno and mirage, there are some small important features), we switched to mean pooling which can better represent the features of a particular region, improving the accuracy by nearly 1 percent.

4.3 Average bounding box

We found that although the three algorithms have their own advantages and disadvantages, but the pre-selected boxes will always have some errors, so we chose to weighted average of the three algorithms of the pre-selected boxes to get a better pre-selected box, first of all, we will pre-select the box of the four-dimensional data down to two-dimensional, and after that, using the k-means algorithm in accordance with the center of the pre-selected boxes to get the center of the averaged pre-selected boxes, and we in accordance with the pre-selected box of each of the prior probability of each pre-checked box to determine the size of its influence on the center, and eliminated the pre-checked boxes with a confidence probability of less than 0.7.

Here are the pre-checked box results after applying the individual neural network algorithms, and a weighted average of the three models:

ssd output

Yolov7 output

average output

4.4 In-Game Implementation: distance calculation & mouse control

Besides combing and optimizing the algorithms, we also make two in-game implementation base on the prediction from our algorithms. We implement the video detecting and real-time screen catching. Both way can detect the object and visualize the bounding box.

First implementation is that, input a picture or video of the game, the program can show the bounding box in the game. Then, base on the player's team (CT or T), the program can automatically identify the opponent's team. Calculating the distance between the center of the screen to all the bounding boxes, the program will visualize several lines between them and show the distance beside.

The second implementation is that we can get the information of the game from the screen instantly, and the program can take control of your mouse, then choose the closest object, move your mouse to the object. So that we can achieve the goal of automatic aiming. (Figure 7 will be shown in the LAST PAGE)
Here is the link for display: `https://www.bilibili.com/video/BV1oT4y147je/`

4.5 Estimation of the distance to Detected Objects

In practical applications, another important functionality is the estimation of the distance to the target object, which can be achieved through the bounding box size. For example, in the prediction box used in Chapter 4.4, the left-side T has coordinates [350 663 984 873], which means a height of 634 and a width of 210. The area of the bounding box will be inversely proportional to the distance. Specifically, in CS2, due to the recoil and damage reduction of firearms, we will not select T and CT that are too small, as well as mouse movements. (Figure 8 will be shown in the LAST PAGE)

Work assignment

Yifan Cai : SSD, Innovation in the pre-training stage, Presentation.

Yu Shi : YOLOv7, In-Game Implementation: real-game-screen capturing, distance calculation & mouse control, Presentation.

Xihe Yu : FRCNN, Average bounding box, code arrangment.

References

[1] Chien-Yao Wang, Alexey Bochkovskiy & Hong-Yuan Mark Liao (2023) YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*,
`https://github.com/WongKinYiu/yolov7`

[2] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Doll & Kaiming He (2018) Detectron.
`https://github.com/facebookresearch/detectron`

[3] Wei Liu & Dragomir Anguelov &Dumitru Erhan. e.t.c.
`https://github.com/amdegroot/ssd.pytorch`

[4] Bubbliiing, yolov7-pytorch & ssd-pytorch & FrRNN-pytorch.
`https://github.com/bubbliiiing/yolov7-pytorch`
`https://github.com/bubbliiiing/ssd-pytorch`
`https://github.com/bubbliiiing/faster-rcnn-pytorch`

[5] ssd-blog
`https://blog.csdn.net/weixin_44791964/article/details/104981486`

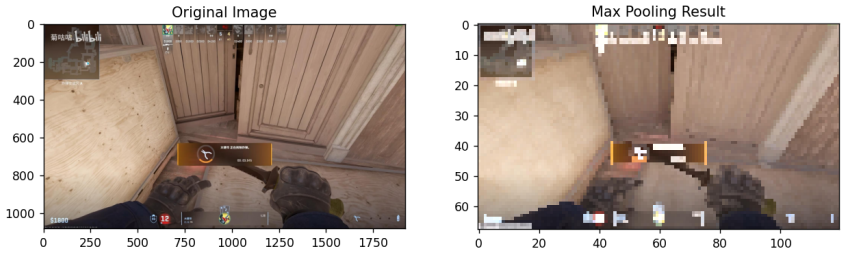


Figure 5: Max pooling

Figure 6: Flipped, noisy, brightened, Mosaic Image

Figure 7: Distance calculating

Figure 8: Distant detection