Training tinyYoloV2 based on OHEM

IOP Conf. Series: Journal of Physics: Conf. Series **1074** (2018) 012085 doi:10.1088/1742-6596/1074/1/012085

Training tinyYoloV2 based on OHEM

Jieming Chen[1, 2], Yueli Hu[1, 2, 4], Siqi Liu[1, 2], Yiwei Wu[3] and Shuai Yin[1, 2]

[1] Shanghai Key Laboratory of Power Station Automation Technology, Shanghai 200072, China
[2] School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China
[3] School of Management, Shanghai University, Shanghai 200072, China

[4] E-mail: huyueli@shu.edu.cn

Abstract. We have used tinyYOLOV2 Neural Network to realize object detection and the aim is to detect specific three classes of objects (socks, slippers, wires) to prevent cleaning robots from being entangled by them. This work describes our methods of training neural network. Considering lack of related data sets, we have adopted various ways to process data sets and added special online hard example mining (OHEM) method in process of training, in order to achieve reliable performance of object detection during the use of cleaning robots. We have also compared results of different values of hyper-parameters and also compared results of TinyYoloV2 and YOLO. At last, 75% mAP in our specific test sets which are the same complicate as realistic situations is achieved. We summarize experience and methods of training network under the circumstance of limited datasets, because it is most likely that there are few data sets which could fit demands of specific projects.

1. Introduction

Nowadays, state of the art object detection networks have reached high standard in accuracy and there has been rising interest in building small and efficient neural networks. Therefore, many detection programs, such as face detection and automatic drive, appear and many projects need to object many specific things. For example, cleaning robots need to detect obstacles which would entangle them, such as slippers and socks. However, it is likely that there are not enough data sets to do these tasks. This work describes how to deal with data sets and train the network to achieve acceptable mAP in our test sets.

The remainder of this work is organized as follows. Section 2 introduces an overview of related works. Section 3 describes the network we use. Section 4 details methods of enlarging data sets and improving stability of performance of detection. Section 5 describes the method of online hard example mining used in training procedure. Section 6 compares results of tinyYolov2, and Yolov2.Section 7 presents conclusions drawn from this work.

2. Related work

Deep neural networks have been used in different applications [1, 2], with object detection being one of the key areas where CNN has significant progress. In the aspect of object detection, some studies using CNN [3-5] are first introduced and use more than one networks to replace feature extraction and candidate regions gradually. Then, end-to-end networks, one-stage detectors appear, which realize these functions together and object detection.

One-Stage Detectors [6-9] are more convenient to be trained and could get satisfied detection results. These small and efficient networks make it possible to realize object detection in embedded systems. The design of size of kernels and number of channels of CNN in [3] is based on [4], and [4] describes a fire module and evenly-spaced downsampling. These methods are used in [10] to compress SqueezeNet to less than 0.5MB. The architecture of YOLOv2 takes the idea of [4] and the structure of CNN in this work is based on [2] to realize our own detection tasks.

Having an excellent effect of detection is necessary for a product, so how to train networks is also a key problem. [10] introduces online hard example mining (OHEM) method and uses it in Fast RCNN. [6] uses another method to consider the balance between loss of positive samples and loss of negative samples. A method of mining hard samples in [11] is similar to our method, which selects 70% of their hard samples, but our method is different from [11], because of small data sets and we take the idea of [4].

Considering small data sets, we train tinyYolo [3] and prepare data sets in our ways to realize detection functions in cleaning robots.

3. Network framework

3.1. TinyYoloV2

The architecture of TinyYoloV2 is used in cleaning robots which run in an embedded platform. Considering the situation of embedded devices [12], we choose TinyYolov2 which is a real time object detection system, and could meet the demand of speed and accuracy [3].

3.2. Modification of network

Tabel 1 displays the structure of our network. In order to detect specific three types of objects, we modify number of filters in last layer. We also modify numbers of filters in last three layers to prevent overfitting, because data sets about wire, socks and slippers are limited and these data sets also need to fit perspective of cleaning robots are fewer. The size of input of the network is 608×608. We enlarge the size of input in order to improve accuracy of small objects. By comparing results of training with different size of input, it is obvious that much larger size of input could achieve better performance of small objects.

Table 1. Structure of network.

Type	Filters	Size/Stride	Output
Convolutional	16	3×3	608×608
Maxpool	16	2×2/2	304×304
Convolutional	32	3×3	304×304
Maxpool	32	2×2/2	152×152
Convolutional	64	3×3	152×152
Maxpool	64	2×2/2	76×76
Convolutional	128	3×3	76×76
Maxpool	128	2×2/2	38×38
Convolutional	256	3×3	38×38
Maxpool	256	2×2/2	19×19
Convolutional	128	3×3	19×19
Maxpool	128	2×2	19×19
Convolutional	3	3×3	19×19

4. Processing of Data Sets

We have labeled tens of thousands of pictures by ourselves to get reliable data sets, but it is not enough for training TinyYolo, so we take several measures to enlarge our data sets.

4.1. Using Composite image

The first method is cropping a picture, which is a normal way. This method do improve diversity of scale of positive samples a lot. We usually crop five sections: top, bottom, left, right, middle from one picture.

The next method provides a large number of negative samples, and obviously decreases the error detection ratio by cooperating with OHEM. This method is to crop sections of foregrounds and then make composite image using these sections and pictures from COCO data sets. Compared with using pure background pictures without containing positive samples, in the condition of only tens of thousands of data sets, this method could prevent the network from being forgotten features of positive samples during the period of training. Figure 1 and figure 2 display examples of composite image.

Figure 1. A example of composite image.

Figure 2. A Example of composite image.

In order to avoid the case that the net learns artificial features from composition image, we keep margin randomly when cropping positive samples. Random length from the ground truth solves the artificial feature.

4.2. Other methods

In order to improve reliability of detection, we take following methods.

When dealing with realistic tests, we find that noise on pictures would influence the detection performance a lot, so we imitate similar noise and add it into the data sets to train the network and do make the precision of pictures with noise very close to it without noise. When we find some specific error detections, we would supply related pictures as negative samples. We also take many pictures at home, which are consistent with actual scene of the use of cleaning robots. These methods are all used to guarantee the stability of detection in practice.

5. OHEM used in TinyYolo

We use the idea of online hard example mining, and realize specific hard example mining in the darknet framework.

5.1. A subsectionPrinciple of hard example mining

During backward propagation, total loss affects all trainable parameters. Loss of sum is comprised of loss of negative samples and that of positive samples.

$$loss_{sum} = loss_{negative} + loss_{positive} \tag{1}$$

It is intuitive that the number of negative samples is larger than that of positive samples. Due to our limited datasets and using of composite images, positive samples is much less than negative samples. In order to prevent loss of positive samples from being submerged by loss of negative samples, limiting negative loss is necessary, so we implement online hard example mining which is helpful for feature extraction of positive samples.

Another benefit of OHEM is to reduce error detection. The ratio of simple samples is much greater than that of hard samples. Although loss of single simple sample is tiny, sum of all simple samples is ineligible. Therefore, loss of hard samples would be submerged sum of simple samples. Nevertheless, loss of a simple sample is close to zero, and it is unnecessary to pay more attention to these simple samples and neglect hard samples. It is reasonable to adjust trainable parameters for considerable loss of hard examples so we ignore parts of simple negative samples and focus on hard samples.

5.2. Implementation details

There are several steps to realize it. First, outputs of last layer are decided as positive or negative samples by IOU. Then, we sort negative samples by their confidence score which shows the degree of error detection. We also need to introduce a hyper-parameter to control number of negative samples. We only choose a part of negative samples whose confidence are highest, and the rest would be set zero, which means that they are not involved in the back propagation and they are neglected. Therefore, the network can focus on hard examples.

[11] takes similar methods both on negative samples and positive samples, but we only focus on negative samples, because the amount and diversity of positive samples are limited.

There is a trick during the training process. When training in first several epochs, it is unnecessary to use hard example mining, because at that time, the confidence of most negative samples are close, so it is unreasonable to only choose a part of negative samples. However, after the network converges to some degree, hard example mining could work very well.

5.3. Setting of related hyper-parameter

Using this method needs to introduce a hyper-parameter which controls the number of negative samples, and the number of negative samples is related with the number of positive samples.

At the same time, some loss is set zero, so the overall loss of negative samples is reduced. We need to enlarge the coefficient which multiplies the overall loss of negative samples as offset.

If there are one positive examples per image, the number of negative samples should be three. In our experiments, we change this value and get the table 2. It is apparent that if number is set too high or too low, the performance will drop. This test also demonstrates effectiveness of OHEM.

Tabel 2. Number of negative samples and mAP.

Network	number of negative samples	mAP
tinyYolov2	40	0.654994909
tinyYolov2	50	0.749867629
tinyYolov2	55	0.752487618
tinyYolov2	60	0.737685243
tinyYolov2	80	0.686661452

After using composite image and hard example mining and processing data sets properly, mAP increases dramatically from 0.29 to at least 0.55, which means the importance of correct data sets and effect of our methods.

6. Experiments

6.1. Comparison between Yolov2 and TinyYolov2

We train the Yolo and TinyYolo to compare the performances of them. We choose a very difficult test set which contains socks and slippers. The sock class is hard to converge, because its feature is unfixed, however the slipper class is relatively easy to learn its feature.

Table 3 displays that Yolo could better represent complex features of sock, but it is prone to overfitting and needs more data to train it. After epoch is 75000, it starts overfitting. TinyYolo is unable to represent complex features, although it is easier to train.

Tabel 3. Comparison between Yolo and TinyYolo.

Network	epochs	sock	slipper	mAP
Yolo	75000	0.620723998	0.92345295	0.772088474
Yolo	125000	0.358115315	0.849287014	0.603701164
Yolo	320000	0.284304371	0.810337733	0.547321052
TinyYolo	100000	0.434512523	0.632413596	0.533463059
TinyYolo	125000	0.456407373	0.647313289	0.551860331
TinyYolo	320000	0.425410948	0.744666779	0.585038864

Actually, in normal situations, using TinyYolo is enough for our cleaning robots, its mAP could achieve 0.7498. Figure 3, 4, 5, 6 show the effect of detection.

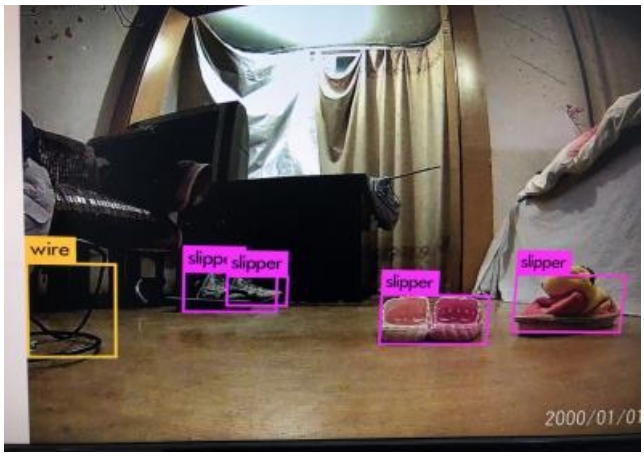

Figure 3. Detection of socks.

Figure 4. Detection of slippers.

Figure 5. Detection of slippers.

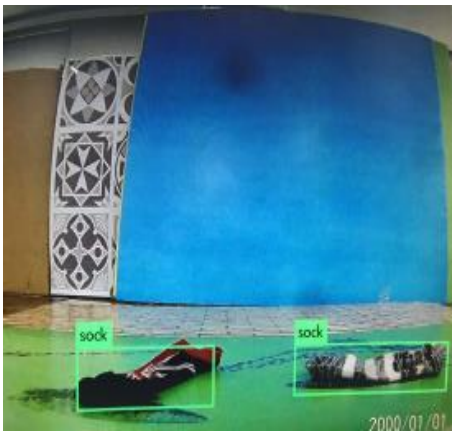

Figure 6. Detection of socks.

7. Conclusions

This work focuses on using end-to-end CNN to realize a specific detection task under the circumstance of a limited data set containing twenty thousand pictures. At this situation, we suggest some methods to train nets. Using the method of composite image may be better than using pure background pictures and other methods to deal with pictures could enhance robustness of detection results. Then, we describe details, a trick and results of using our hard example mining in tinyYolo. At last, we compare and analyze results of Yolo and TinyYolo, and show the outcomes of our network.

References

[1] Shafiee M J, Chywl B, Li F and Wong A Fast Yolo: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video 2017 *arXiv* **1709** 05943
[2] Girshick R Fast R-CNN 2015 *Comput. Sci*
[3] Ren S, Girshick R, Girshick R and Sun J 2015 Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks *IEEE Trans. Pattern Anal. Mach Intell* **39** 1137-49
[4] Krizhevsky A, Sutskever I and Hinton G 2012 Imagenet Classification with Deep Convolutional Neural Networks *NIPS*
[5] Redmon J and Farhadi 2018 A YOLOv3:An Incremental Improvement *arXiv* **1804** 02767
[6] Redmon J and Farhadi A 2016 YOLO9000: Better, Faster, Stronger *arXiv* **1612**
[7] Iandola F and Keutzer K 2017 Keynote: Small Neural Nets are Beautiful: Enabling Embedded Systems with Small Deep-neural-network Architectures *In Proceedings of the 12fth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis Companion. ACM* p 1
[8] Wong A, Shafiee M J, Li F and Chwyl B 2018 Tiny ssd: A Tiny Single-shot Detection Deep Convolutional Neural Network for Real-time Embedded Object Detection *arXiv* **1802** 06488
[9] Xiang J and Zhu G 2017 Joint Face Detection and Facial Expression Recognition with Mtcnn *Int. Conf. on Inf. Sci. and Ctrl. Eng. IEEE Comput. Societ* pp 424-427
[10] Shrivastava A, Gupta A and Girshick R 2016 Training Region-based Object Detectors with Online Hard Example Mining *In Proc. of the IEEE Conf. on Comput. Vision and Pattern Recog* pp 761–769
[11] Lin TY, Goyal P, Girshick R, He K and Dollar P 2017 Focal Loss for Dense Object Detection *arXiv* **1708** 02002
[12] Iandola F N, Han S, Moskewicz M W, Ashraf K, Dally W J and Keutzer K 2016 SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and <0.5MB Model Size *arXiv* **1612** 01051