Weekly Work Report

Zhang, Liqiang

VISION@OUC

July 22, 2018

1 Research problem

It's my honour to take part in underwater robot picking contest with classmates. My task is to collocation environment for YOLOv3. Before that, I never had experience and knowledge about this, so I'm a little scared about this work. But it's hard to begin with everything, therefore, I have access to some basic information on the Internet.

2 Research approach

My initial approach is to search online tutorials such as CSDN blog and YOLO official network. There are usually very detailed steps on these websites.

3 Research progress

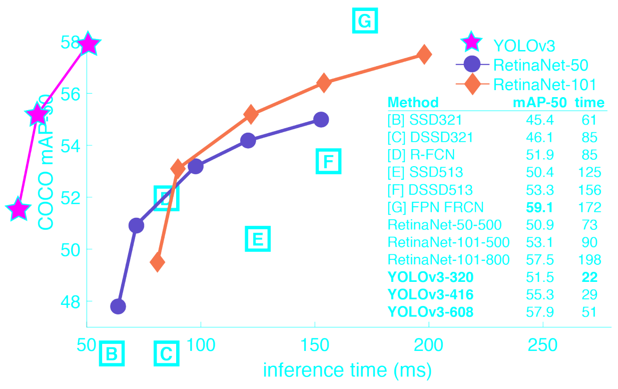
This is the first time I have carried out this research, and I have some deep learning and programming foundations, and I hope to learn more knowledge in the process of completing this study.

4 Progress in this week

4.1 YOLOv3

YOLO is you only look once, which is a state-of-the-art, real-time object detection system. It official data show that it processes images at 30 FPS and has a mAP of 57.9% on COCO test-devs On a Pascal Titan X.

Compared with other detectors, YOLOv3 is extremely fast and accurate. In mAP measured at .5 IOU YOLOv3 is on par with Focal Loss but about 4x faster. Moreover, we can easily trade off between speed and accuracy simply by changing the size of the model, no retraining required. Fig. 1 shows the inference time on YOLOv3 training.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 1: Inference time (ms) on YOLOv3

4.2 Building the test environment for YOLOv3 [2]

On the YOLO's official website, it is clear how to download the program, use the following code, I can clone the algorithm program into the local folder.

```
git clone https://github.com/pjreddie/darknet
cd darknet
```

Because I use GPU to train VOC data set. So I must modify the Makefile with the following code to make the algorithm use GPU. Using GPU to train the model can save a lot of time. When these files are set up, using the command of "make", YOLOv3 was built on the computer.

```
gedit Makefile
GPU=1
CUDNN=1
...
NVCC=/usr/local/cuda-8.0/bin/nvcc
```

I can run a demo on the YOLOv3 just being built on the computer, but before that I must download a weigh from following code.

```
wget https://pjreddie.com/media/files/yolov3.weights
```

Now, I can run the detector.

```
./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg
```

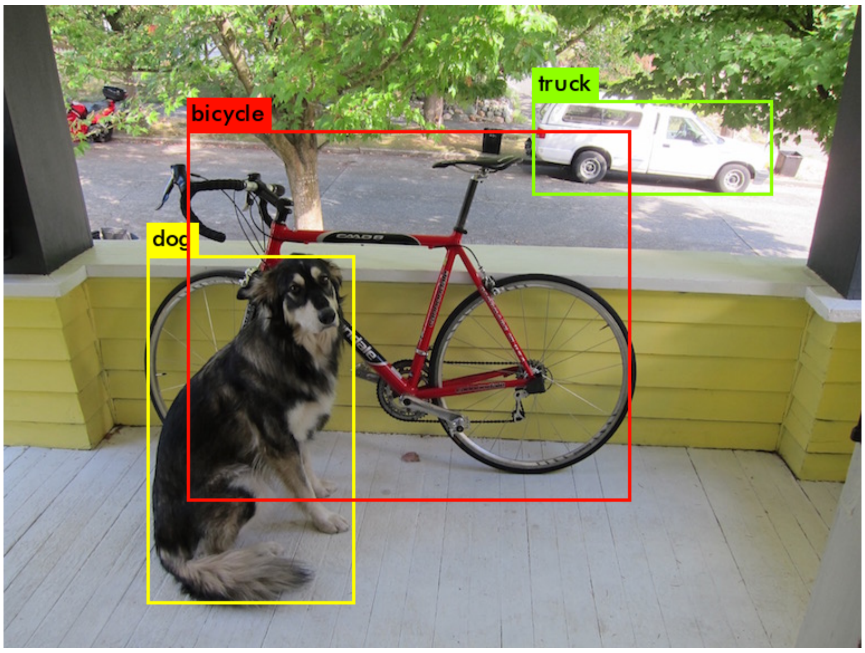


Figure 2: The demo.

The predictions as shown in the Fig .2. This is the first time I can run such a picture, so I feel very excited. Although these are some basic operations, I also explored it one by one and before that I have not a reserve of knowledge in this area. Teacher Yu tell me that I should train the VOC data set when I run a demo successfully. On the YOLO's official website, there are the introduction for using YOLOv3 to train the VOC data set.

Firstly, to train YOLO I will need all of the VOC data from 2007 to 2012. To get all the data, make a directory to store it all and from that directory run:

```
wget https://pjreddie.com/media/files/VOCtrainval_11-May-2012.tar
wget https://pjreddie.com/media/files/VOCtrainval_06-Nov-2007.tar
wget https://pjreddie.com/media/files/VOCtest_06-Nov-2007.tar
tar xf VOCtrainval_11-May-2012.tar
tar xf VOCtrainval_06-Nov-2007.tar
tar xf VOCtest_06-No-2007.tar
```

Now I need to generate the label files that Darknet uses. Darknet wants a .txt file for each image with a line for each ground truth object in the image. To generate these file we will run the voc_label.py script in Darknet's scripts, I can download it from following code.

```
wget https://pjreddie.com/media/files/voc_label.py
python voc_label.py
```

After a few minutes, this script will generate all of the requisite files.:

```
ls
2007_test.txt    VOCdevkit
2007_train.txt   voc_label.py
2007_val.txt     VOCtest_06-Nov-2007.tar
2012_train.txt   VOCtrainval_06-Nov-2007.tar
2012_val.txt     VOCtrainval_11-May-2012.tar
```

The text files like 2007_train.txt list the image files for that year and image set. Darknet needs one text file with all of the images you want to train on. In this example, let's train with everything except the 2007 test set so that we can test our model. Run:

```
cat 2007_train.txt 2007_val.txt 2012_*.txt > train.txt
```

Now go to my Darknet directory. I have to change the cfg/voc.data config file to point to the data:

```
  1 classes= 20
  2 train  = <path-to-voc>/train.txt
  3 valid  = <path-to-voc>2007_test.txt
  4 names = data/voc.names
  5 backup = backup
```

After getting the weight from the darknet53 model, I can begin to train the model by running the command:

```
wget https://pjreddie.com/media/files/darknet53.conv.74
./darknet detector train cfg/voc.data cfg/yolov3-voc.cfg darknet53.conv.74
```

4.3 Problems

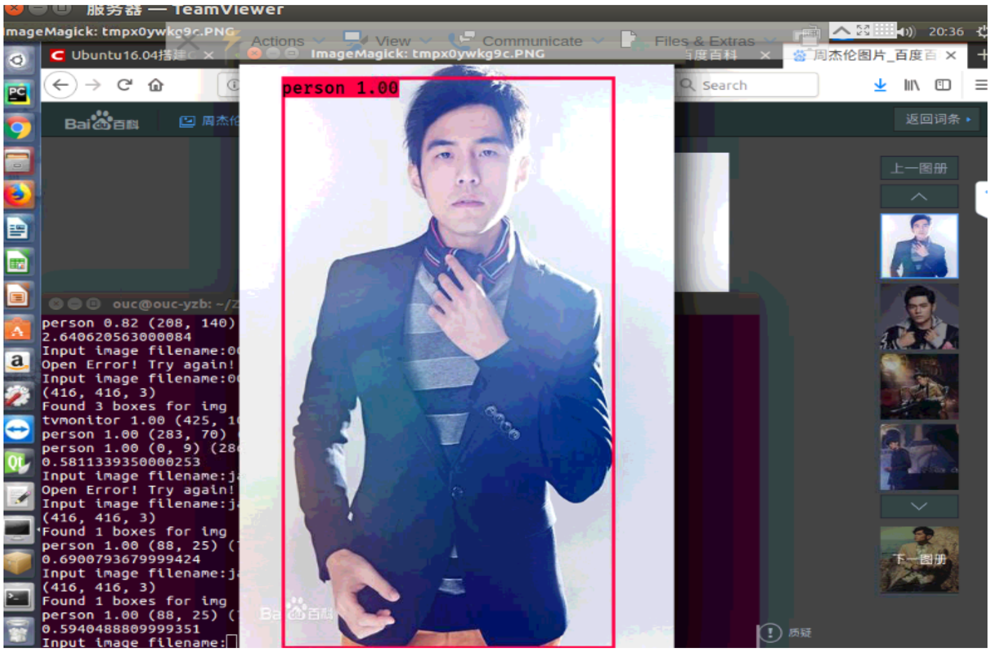
Although I operated step by step according to official website commands, I still encountered some problems that could not be solved by me. During my training, the class given by the terminal have been unstable, sometimes very small, and sometimes close to 1. The value of class should be closed to 1, but the value I got is very volatile. It did not converge after training for an afternoon. I inquire my senior Zhangshaoyong, he tell me that I can use the keras-YOLO3 which can test the video.

5 Keras-YOLO3

The environmental requirements of Keras-YOLO3 is very harsh, it need a conda environment. During establishing the virtual environment for Keras-YOLO3, I have a series of problems. It lack document or project. I set up an environment from From seven in the morning until nine in the evening. I was almost crazy during building the virtual environment, fortunately, I finally succeeded in establishing the environment, and successfully detected the portrait of my idol Jay Chou as Fig.3.

5.1 Training our own data

The problem I am facing now is how to transform the data used in the competition into VOC data format. The true value table given by the competition is rather chaotic. It is difficult to extract XML files from it. A program on the Internet does not apply to our TXT file. Finally, we find the help of



```
person 0.82 (208, 140)
2.640620563000084
Input image filename:0
Open Error! Try again!
Input image filename:0
(416, 416, 3)
Found 3 boxes for img
tvmonitor 1.00 (425, 1
person 1.00 (283, 70)
person 1.00 (0, 9) (28
0.5811339350000253
Input image filename:j
Open Error! Try again!
Input image filename:j
(416, 416, 3)
Found 1 boxes for img
person 1.00 (88, 25) (
0.6900793679999424
Input image filename:j
(416, 416, 3)
Found 1 boxes for img
person 1.00 (88, 25) (
0.5940488809999351
Input image filename:
```

Figure 3: Jay Chou.

Yuan Hao and have successfully extracted the XML files from the true value table. Now I have converted our data into VOC format.

6 Deep learning [1]

6.1 How to Effectively Improve the Neural Network Foundation

We know that there are three datasets in deep learning: training datasets, dev datasets and test datasets. Actually we should guarantee that the origin of dev datasets and test datasets are from one. The factors that we should consider are various, such as layers, hidden units, learning rates and activation functions. Deep learning is a typical iteration process.

6.2 Bias and Variance

By verifying the error we can get whether there is a high variance. Fig. 4 shows us the straightforward differences. We can use deeper neural network, add some train sets or choose other alogrithm to decrease the bias.

6.3 Regularization

In this subsection, Professor Wu show us L2 regularization as Eq. 1, This is a way to solve high variance probelms. If there is an over-fitting, ie high variance, then regularization regularization needs to be solved. Although expanding the number of training samples is also a way to reduce high variance, the cost of obtaining more training samples is often too high and difficult. Therefore, a more feasible and effective way is to use regularization.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \parallel w \parallel^2 \tag{1}$$

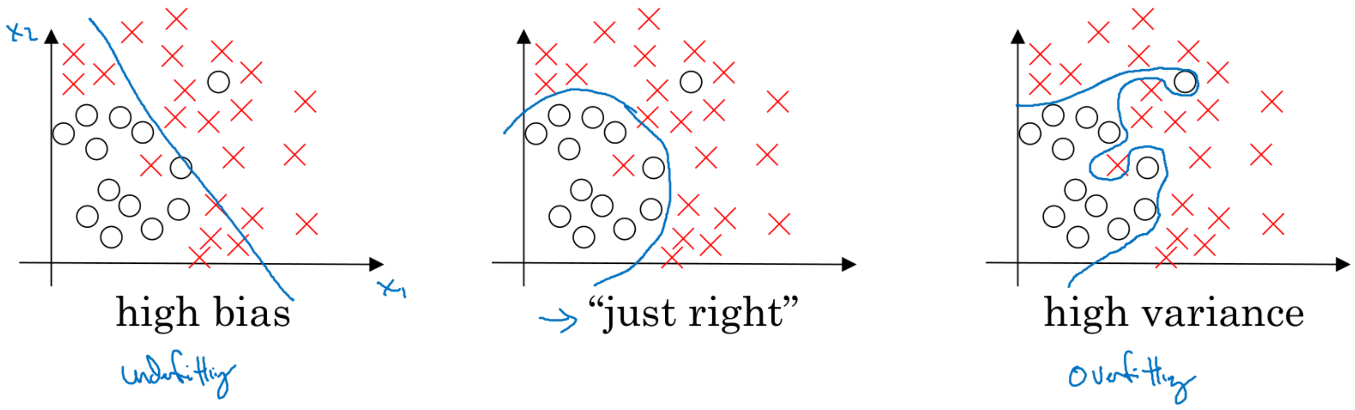


Figure 4: Underfitting and overfitting.

76880076733 The L1 regularization can be seen as Eq. 2

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \parallel w \parallel_1 \tag{2}$$

Compared with L2 regularization, the l obtained by L1 regularization is more sparse, that is, many w are zero. The advantage is that it saves storage space because most of the w is zero. However, in fact L1 regularization is not superior to L2 regularization in solving high variance. Moreover, L1 is more complicated in terms of differential derivation. Therefore, general L2 regularization is more commonly used.

6.4 Dropout Regularization

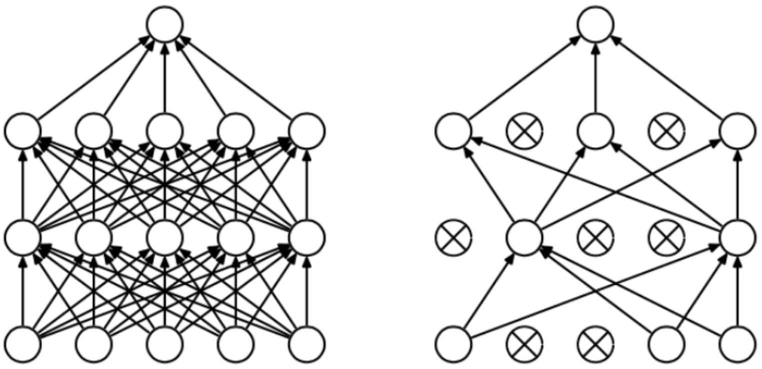


Figure 5: Jay Chou.

The function of dropout is to delete the unit of the neural network randomly. Actually we hope there are less training sets. Assume that for the first layer of neurons, set the probability of keeping the neuron proportional keep prob = 0.8, that is, 20% of the neurons in the layer stop working. d^l s the dropout vector, and d^l is set to a random vector, where 80% of the elements are 1, and 20% of the elements are 0.

For m samples, a single iteration of training randomly deletes a certain number of neurons in the hidden layer; then, the weights and constants b are updated forward and backward on the remaining neurons after deletion; In one iteration, the previously deleted neurons are restored, a certain number of neurons are randomly deleted, and w and b are updated in the forward and reverse directions.

6.5 Normalizing Inputs

Normalized input is the normalization of the training data set, that is, the original data is subtracted from its mean μ, and then divided by its variance σ^2 as Fig. 6.

The result of this is that the cost function and the relationship between w and b may be a very slender oval bowl. When the gradient descent algorithm is applied to it, since the values of $w1$ and $w2$ are very different, only a small learning factor α can be selected to avoid J oscillation. Once α is large, oscillation will inevitably occur, and J will no longer monotonically decrease.

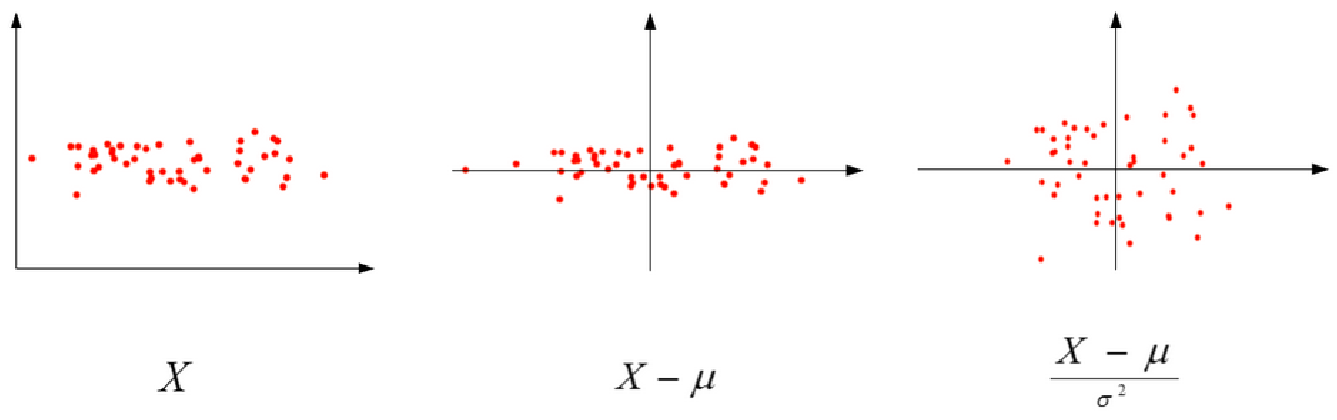


Figure 6: Normalization.

6.6 Mini-batch Gradient Descent

m is the number of samples. If m is large, for example, on the order of a million, the training speed tends to be slow, because all samples are summed and matrixed for each iteration. This process calls gradient descent algorithm Batch Gradient Descent.The implementation of Mini-batches Gradient Descent is to divide the total training samples into T subsets (mini-batches), then perform neural network training for each mini-batch, including Forward Propagation, Compute Cost Function, Backward Propagation, and cycle to T mini-batch are trained.

The Batch gradient descent will be closer to the global minimum, but because of the use of all m samples, the speed of each advance is somewhat slow. Stachastic gradient descent is very fast every time, but the route twists and turns, there is a large oscillation, and finally will fluctuate around the minimum value, it is difficult to really reach the minimum. Moreover, vectorization cannot be used to improve the speed of calculation in numerical processing.

7 Plan

Objective:	Training contest data and testing the test set.
Deadline:	2018.07.29.

2018.07.23—2018.07.25 Training contest data.

2018.07.26—2018.07.29 Testing the test set. lot lot lot of things.

References

[1] Neural Network & Deep Learning. http://mooc.study.163.com/learn/2001281002?tid=2001392029#/learn/announce. 4

[2] YOLO. https://pjreddie.com/darknet/yolo/. 1