5.Model training

1.yolov4-tiny introduction

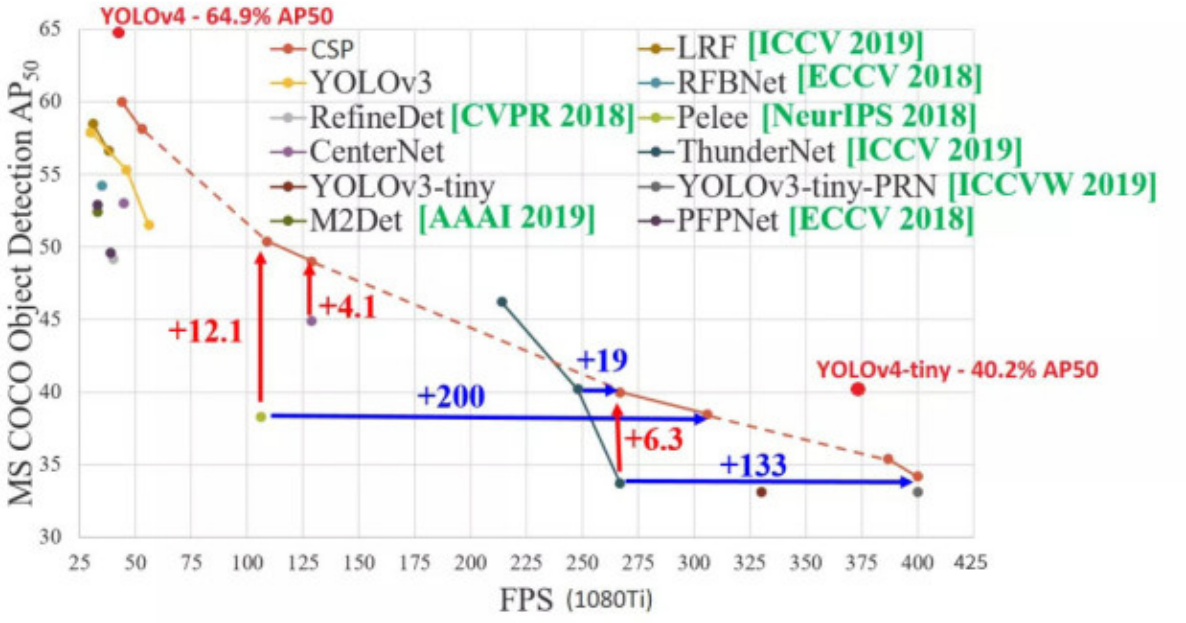
yolov4-tiny Official website：https://github.com/AlexeyAB/darknet

- Release time node

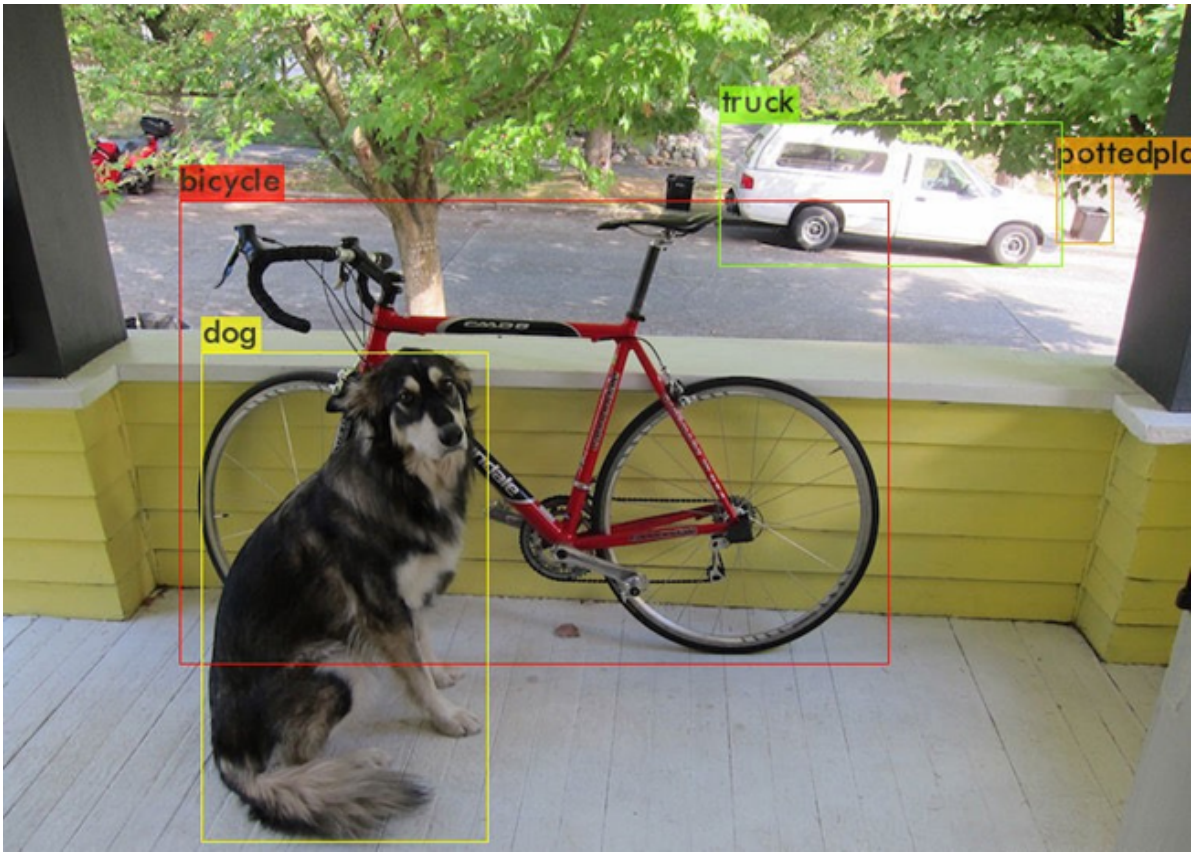
- 2020.04：YOLOv4 officially released

- 2020.06：YOLOv4-Tiny officially released

YOLOv4-Tiny performance on COCO: **40.2% AP50, 371 FPS (GTX 1080 Ti)** Whether it is AP or FPS performance, it is a huge improvement compared to YOLOv3-Tiny, Pelee, and CSP. As shown below:



- YOLOv4 test results

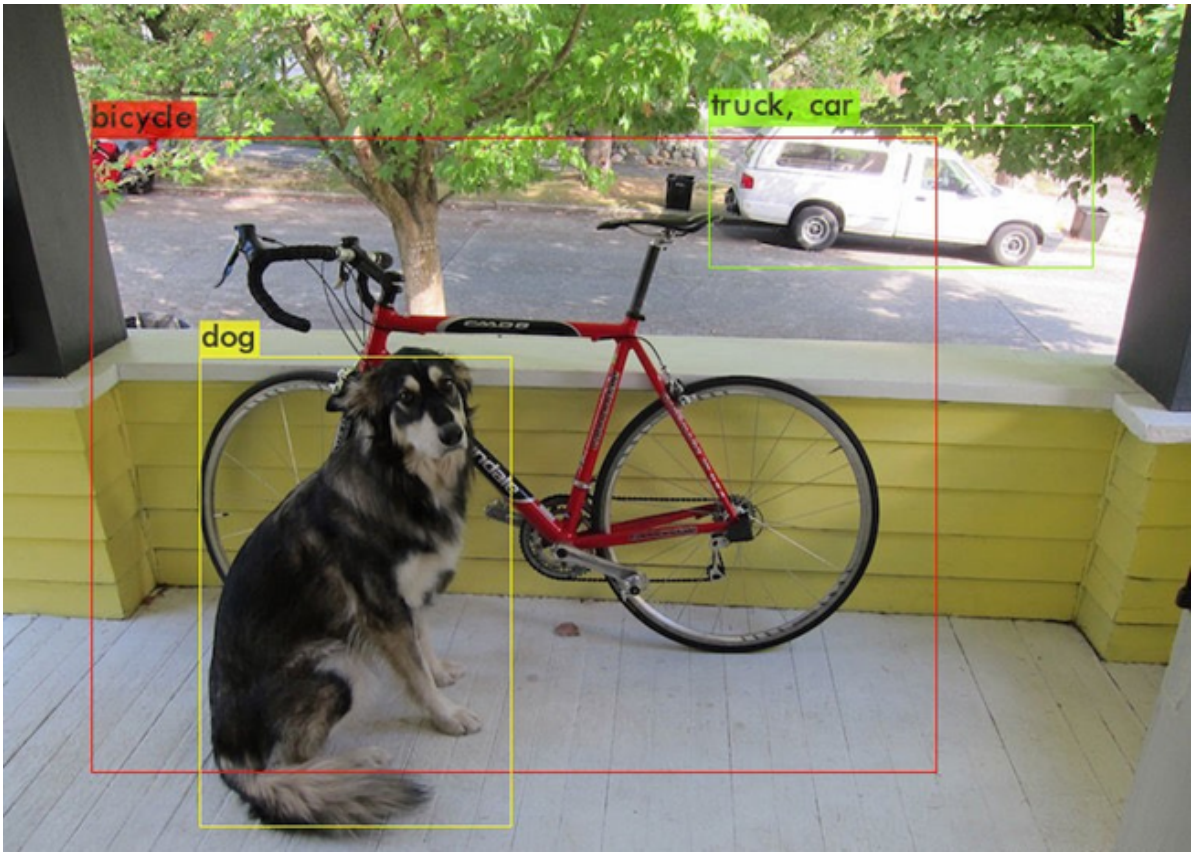



```
Done! Loaded 162 layers from weights-file
data/dog.jpg: Predicted in 27.039000   milli-seconds.
bicycle: 92%
dog: 98%
truck: 92%
pottedplant: 33%
```

Comparison of YOLOv4 and YOLOv4-Tiny detection results, source network

Link：https://blog.csdn.net/JIEJINQUANIL/article/details/106998409

- YOLOv4-Tiny test results




```
Done! Loaded 38 layers from weights-file
data/dog.jpg: Predicted in 2.609000   milli-seconds.
bicycle: 29%
dog: 72%
truck: 82%
car: 46%
```

We can see that the detection accuracy of Yolov4-tiny has declined, but in terms of time consumption, Yolov4-tiny has obvious advantages:Yolov4-tiny detection takes only 2.6 milliseconds, while Yolov4 detection takes 27 milliseconds, which is more than 10 times faster!

2.Environmental requirements

```
tensorflow-gpu==2.2.0
lxml
matplotlib
pandas
Pillow
scikit-learn
seaborn
tqdm
imgaug
```

3.Model training process

(1)Folder structure

garbage_data: store data set

garbage_data/image: target source file

garbage_data/JPEGImages: data set pictures (as many as possible)

garbage_data/texture: background picture (as many as possible)

garbage_data/train.txt: label file corresponding to the data set image

garbage_data/GetData.py: get data set code

font: store font package

img: store test images

logs: stores test logs and the final training model last1.h5.

model_data: stores pre-trained models (weight files), custom label files (corresponding to target source files), and yolo model parameter anchors.

nets and utils: some library files of yolo

In the YOLO-v2 version, the concept of anchor box was introduced, which greatly increased the performance of target detection. The essence of anchor is the reverse of the SPP (spatial pyramid pooling) idea.What SPP itself does is to resize inputs of different sizes into outputs of the same size, so the reverse of SPP is to reverse the output of the same size to get inputs of different sizes.

(2)Training steps

Training code source network, link：https://github.com/bubbliiiing/yolov4-tiny-tf2

- Make data sets

The names of the pictures and label files must correspond. The label format in the train.txt file is as follows:

```
./garbage_data/JPEGImages/0.jpg   113,163,293,298,9
# Picture path       y, x, y + w, x + h ,label
```

To create a data set, one method is to take some photos first, use an annotation tool to annotate the targets on each photo, create a new train.txt file in the garbage_data folder, and write the target information.

Another method is to put background images (as many as possible) in the garbage_data/texture folder, modify the GetData.py code as needed, and execute GetData.py to generate a data set (as many as possible).

- Add weight file

You can search and download the latest weight file on Baidu. There are good weight files yolov4_tiny_weights_coco.h5 and yolov4_tiny_weights_voc.h5 under the garbage_data file.

- Make your own classes--->garbage.txt

Note that it is best not to use Chinese tags and there should be no spaces in the folder!

```
Zip_top_can
Old_school_bag
Newspaper
Book
Toilet_paper
......
```

- Modify train.py file

Modify according to your own needs by referring to the comments.

```
# Label location
annotation_path = 'garbage_data/train.txt'
# Get the location of classes and anchor
classes_path = 'model_data/garbage.txt'
anchors_path = 'model_data/yolo_anchors.txt'
# Location of pre-trained model
weights_path = 'model_data/yolov4_tiny_weights_coco.h5'
# Get classes and anchor
class_names = get_classes(classes_path)
anchors = get_anchors(anchors_path)
# How many categories are there in total?
num_classes = len(class_names)
num_anchors = len(anchors)
# The location where the trained model is saved
log_dir = 'logs/'
# Enter the image size. If the video memory is large, 608x608 can be used.
input_shape = (416,416)
# Initial epoch value
Init_epoch = 0
# Freeze the epoch value of training
Freeze_epoch = 50
# The size of Batch_size indicates how much data is fed each time. If there is
OOM or insufficient video memory, please adjust it smaller.
batch_size = 16
# Maximum learning rate
learning_rate_base = 1e-3
# Total epoch value
Epoch = 100
```

- Start training

According to the above process, after the operation is completed, directly run the train.py file for training.

(3) Custom model detection

- Modify yolov.py file

```
class YOLO(object):
      _defaults = {
          "model_path": 'model_data/garbage.h5',
          "anchors_path": 'model_data/yolo_anchors.txt',
          "classes_path": 'model_data/garbage.txt',
          "score" : 0.5,
          "iou" : 0.3,
          "eager" : False,
          # The default is 416x416 (image size)
          "model_image_size" : (416, 416)
    }
... ...
self.font_path = 'font/Block_Simplified.TTF'
```

model_path: used for detection and trained model path (global path is required in ROS environment).

anchors_path: yolo's model parameter anchors path (the global path is required in ROS environment).

classes_path: Custom label file path (global path is required in ROS environment).

self.font_path: font package path (global path is required in ROS environment).

l Execute py file detection

predict_img.py: Image detection.

predict_video.py: Video detection.