# Yolo环境搭建

## 参考代码地址

Pytorch版本的yolo3代码下载地址：<https://github.com/ultralytics/yolov3>

## **安装依赖包**

conda install numpy opencv matplotlib tqdm

conda install pytorch torchvision -c pytorch

## **下载预训练的权重文件**

直接执行weights目录下的download\_yolov3\_weights.sh，下载的权重文件会保存在weights/weights目录下，这个脚本只下载darknet框架的权重文件，可以通过model.py中的convert转换为pytorch框架的权重文件

python3 -c "from models import \*; convert('cfg/yolov3-spp.cfg', 'weights/yolov3-spp.weights')"

# 目标检测

使用预训练的权重文件预测图片，默认输入的图片为项目data/samples目录下的图片，输出在output目录下，默认输出的一张被标记的图片，如果要输出图片中标记的box的坐标以及分类信息，在检测的时候需要将参数save\_txt设置为true。

YOLOv3:

python3 detect.py --cfg cfg/yolov3.cfg --weights weights/yolov3.weights

python3 detect.py --cfg cfg/yolov3.cfg --weights weights/yolov3.pt

YOLOv3-tiny:

python3 detect.py --cfg cfg/yolov3-tiny.cfg --weights weights/yolov3-tiny.weights

python3 detect.py --cfg cfg/yolov3-tiny.cfg --weights weights/yolov3-tiny.pt

YOLOv3-SPP:

python3 detect.py --cfg cfg/yolov3-spp.cfg --weights weights/yolov3-spp.weights

python3 detect.py --cfg cfg/yolov3-spp.cfg --weights weights/yolov3-spp.pt

# 模型训练

## 模型的输入与输出

**模型的输入**:416\*416的图片信息

**模型的输出**：一个或多个检测到的box坐标分类信息（x,y,w,h,class,classconf）

x,y:表示中心点x,y坐标信息,都是相对于416×416的比例，值为0~1

w,h:表示box的宽和高，都是相对于416×416的比例，值为0~1

Class：表示box对应的分类信息

Classconf：表示box对应的分类概率

## 训练数据准备

1. my\_coco\_160.data

classes=80 #分类数  
train=./data/coco\_10img.txt #训练集图片信息  
valid=./data/coco\_10img.txt #验证集图片信息  
names=data/coco.names #分类信息  
backup=backup/  
eval=coco

1. coco\_10img.txt

../coco/images/train2014/COCO\_train2014\_000000000009.jpg  
../coco/images/train2014/COCO\_train2014\_000000000025.jpg  
../coco/images/train2014/COCO\_train2014\_000000000030.jpg  
../coco/images/train2014/COCO\_train2014\_000000000034.jpg  
../coco/images/train2014/COCO\_train2014\_000000000036.jpg  
../coco/images/train2014/COCO\_train2014\_000000000049.jpg  
../coco/images/train2014/COCO\_train2014\_000000000061.jpg  
../coco/images/train2014/COCO\_train2014\_000000000064.jpg  
../coco/images/train2014/COCO\_train2014\_000000000071.jpg  
../coco/images/train2014/COCO\_train2014\_000000000072.jpg

在../coco/images/tain2014/目录下每张图片对应个txt文件，其文件名与图片文件名一致，内容与3.1章节的模型输出文本内容一致。

## 开始训练

python3 train.py --data-cfg data/my\_coco\_160.data

训练完成之后会在当前目录生成pytorch版本的权重文件best.pt和last.pt

训练结果说明

# ONNX模型转换

我们参考的代码，支持将模型转换为onnx模型，只需要在model.py文件中讲EXPORT\_ONNX常量改为true，执行检测代码就可以在weights目录下生成export.onnx模型文件。

python3 detect.py --cfg cfg/yolov3.cfg --weights weights/yolov3.pt

# WEB端运行yolov3

使用onnxjs组件，在web界面上传图片进行目标检测，效果图如下：

1. 加载模型

/\*模型改变效应事件\*/

modelChangeHandler = (value: any) =>{

const { dispatch } = this.props

const myOnnxSession = new InferenceSession()

try {

dispatch({type:'onnx/getModel',payload:{modelName:value}})

.then((response) => {

response.arrayBuffer().then((modelfile) => {

myOnnxSession.loadModel(modelfile)

this.setState({...this.state,curSession:myOnnxSession,curModel:value})

})

})

} catch (e){

throw new Error('Error: Backend not supported. ');

}

}

1. 输入图片预处理，将图片按比例缩放到416\*416，并置灰（像素值除以255（0~1之间））

/\* 图片数据预处理 \*/

preprocess(ctx: CanvasRenderingContext2D): Tensor {

const { canvasSize:{x,y,w,h} } = this.state

const imageData = ctx.getImageData(0, 0, ctx.canvas.width, ctx.canvas.height);

const { data, width, height } = imageData;

// data processing

const dataTensor = ndarray(new Float32Array(data), [width, height, 4]);

const dataProcessedTensor = ndarray(new Float32Array(width \* height \* 3), [1, 3, width, height]);

ops.assign(dataProcessedTensor.pick(0, 0, null, null), dataTensor.pick(null, null, 0));

ops.assign(dataProcessedTensor.pick(0, 1, null, null), dataTensor.pick(null, null, 1));

ops.assign(dataProcessedTensor.pick(0, 2, null, null), dataTensor.pick(null, null, 2));

const tensor = new Tensor(new Float32Array(width\* height\* 3), 'float32', [1, 3, width, height]);

(tensor.data as Float32Array).set(dataProcessedTensor.data);

const two = new Tensor([255.0], 'float32');

const newTensor = yoloTransforms.div(tensor, two)

return newTensor;

}

1. 目标检测

/\* 运行模型 \*/

runModel = (ctx:any) => {

const { curSession } = this.state

const data = this.preprocess(ctx)

curSession.run([data]).then((result) => {

const outputdata = result.values()

const scores = outputdata.next().value

const boxs = outputdata.next().value

this.postprocessyolov3(scores,boxs)

})

}

1. Box选择处理

postprocessyolov3(scores: Tensor, boxs: Tensor) {

const max = yoloTransforms.max2(scores,1)

const max\_score = max[0]

const zipped = []

for(let i=0;i<max[0].size;i++){

if(max[0].data[i]>0.5){

const x = boxs.data[i\*4+0]

const y = boxs.data[i\*4+1]

const w = boxs.data[i\*4+2]

const h = boxs.data[i\*4+3]

const x1 = x - w/2

const x2 = x + w/3

const y1 = y - h/2

const y2 = y + h/2

zipped.push([max[0].data[i],max[1].data[i],[x1,y1,x2,y2]])

}

}

const sortedBoxes = zipped.sort((a: number[], b: number[]) => b[0] - a[0])

const selectedBoxes: any[] = [];

sortedBoxes.forEach((box: any[]) => {

let add = true;

for (let i=0; i < selectedBoxes.length; i++) {

const curIou = this.box\_iou(box[2], selectedBoxes[i][2]);

if (curIou > 0.5) {

add = false;

break;

}

}

if (add) {

selectedBoxes.push(box);

}

})

selectedBoxes.forEach((box) => {

const x1y1x2y2 = this.scale\_box(box[2])

this.drawRect(x1y1x2y2[0], x1y1x2y2[1], (x1y1x2y2[2]-x1y1x2y2[0]), (x1y1x2y2[3]-x1y1x2y2[1]),`${classNames[box[1]]} Confidence: ${Math.round(box[0] \* 100)}%`);

})

}

BOX的IOU计算

box\_intersection = (a: number[], b: number[]) => {

const w = Math.min(a[2], b[2]) - Math.max(a[0], b[0]);

const h = Math.min(a[3], b[3]) - Math.max(a[1], b[1]);

if (w < 0 || h < 0) {

return 0;

}

return w \* h;

}

box\_union = (a: number[], b: number[]) => {

const i = this.box\_intersection(a, b);

return (a[3] - a[1]) \* (a[2] - a[0]) + (b[3] - b[1]) \* (b[2] - b[0]) - i;

}

box\_iou = (a: number[], b: number[]) => {

return this.box\_intersection(a, b) / this.box\_union(a, b);

}

# 参考资料

<https://github.com/ultralytics/yolov3>

https://github.com/Microsoft/onnxjs  
https://github.com/onnx/onnx  
https://github.com/onnx/tutorials/blob/master/tutorials/PytorchTensorflowMnist.ipynb  
https://github.com/onnx/tutorials/blob/master/tutorials/PytorchOnnxExport.ipynb  
https://heartbeat.fritz.ai/building-an-image-recognition-app-using-onnx-js-c7147f4f291b

[https://microsoft.github.io/onnxjs-demo/#/](https://microsoft.github.io/onnxjs-demo/" \l "/)

腾讯优图

<http://bbs.cvmart.net/articles/251/teng-xun-you-tu-kai-yuan-yolo-xi-lie-dai-ma-han-yolov3-yi-ji-ge-zhong-backbone>