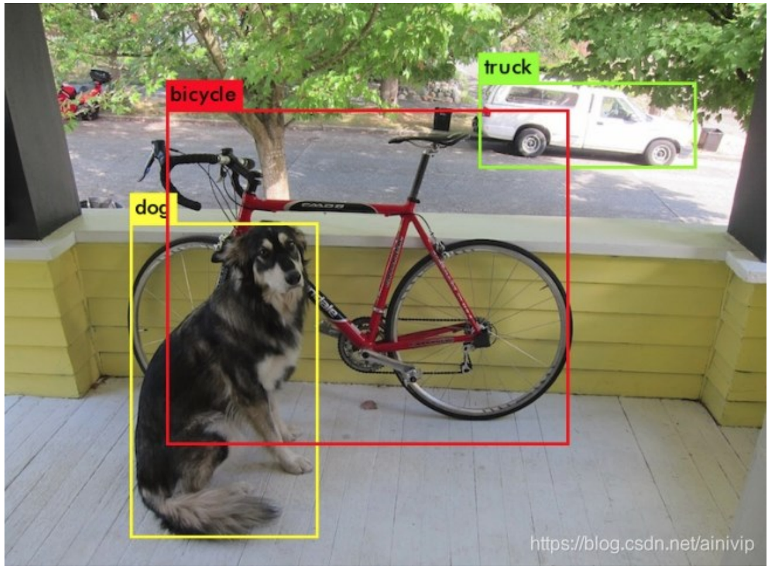
转载 data-life 于 2019-10-05 07:38:48 发布 749 收藏 8 版权
分类专栏： Python

Python 专栏收录该内容 2 订阅 94 篇文章 订阅专栏

> 原文：Deep Learning based Object Detection using YOLOv3 with OpenCV ( Python / C++ )
>
> 作者：Sunita Nayak
>
> 日期：2018-08-20

这里主要介绍基于 OpenCV 的 YOLOV3 目标检测器的应用.



YOLOV3 是 YOLO-You Only Look Once 目标检测算法的最新变形，其开源的模型能够识别图片和视频中 80 种不同的物体类别，而且最重要的是其速度非常快，并具有与 SSD(Single Shot MultiBox) 相当的精度.

> YOLOv3 Tech Report

OpenCV3.4.2 版本之后，可以很方便地在 OpenCV 应用中采用 YOLOV3 模型.

## 1. YOLO 工作原理

目标检测器可以看作**位置定位器(object locator)**和**目标识别器(object recognitizer)**的组合.

传统 CV 方法中，一般采用滑窗(sliding window) 来寻找不同位置和不同尺度的物体. 但由于其计算算代价较大，通常都会假设物体的长宽比(aspect ratio) 是固定的.

分类专栏
JAVA学习实践 付费
数据分析 3篇
爬虫 10篇
Python 94篇
HACKER'home
我为什么不要应届毕业生 1篇
我写的文章 4篇
百宝箱 5篇
精彩的文档 1篇
编程生涯 6篇
美文翩翩 2篇
翻译项目 1篇
OFFICE 7篇
Django 4篇
ORACLE 9篇
Windows 6篇
MySQL 13篇
CentOS 13篇

早期基于深度学习的目标检测算法，如，R-CNN，Fast-RCNN 等，通常 Selective Search 方法来降低算法需要测试的边界框(bounding box) 数量.

另一种方法叫作 Overfeat，其通过采用类似于滑窗的机制，以全卷积的方式对图像进行多尺度的扫描.

后面出现的方法采用 RPN(Region Proposal Network) 来确定需要测试的边界框. 通过精心的 设计，用于识别物体的提取特征，还可以被 RPN 用于提取潜在的边界框，因此节省了大量的计算量.

另一方面，YOLO 等目标检测算法，采用了完全不同的方式来处理目标检测问题，其 只需要对整张图像进行一次网络前向计算. SSD 是另一种只进行一次深度学习网络前向计算的目标检测算法， 但，YOLOV3 比 SDD 具有更快的速度和相当的精度. 再 M40，TitanX 和 1080Ti GPUs 上取得了更快的实时效果.

YOLO 对于给定图像，检测目标物体的工作原理如下：

首先，将图像划分为 13x13 个网格组成. 共169 个单元格，各单元格的尺寸取决于网络的输入尺寸. 比如，实验中，对于 416x416 的输入尺寸，则每个单元格的 尺寸为 32x32.

然后，每个单元格负责预测图像的一些框(boxes).

对于每一个边界框，网络还会预测包含物体的边界框的置信度，以及物体关于特定类别的概率.

由于大部分边界框的置信度都比较低，或者很多边界框是包含相同物体的，只保留最高置信的边界框，因此，大部分边界框都是可以被消除掉的. 这种消除边界框的方法即为 NMS(non-maximum suppression).

YOLOV3 的作者 Joseph Redmon 和 Ali Farhadi, 将精度和速度都比 YOLOV2 进行了提升. YOLOV3 可以更好的处理多尺度. 此外，还通过增大网络和添加跳跃链接(shortcut connections) 的残差网络的方式提升网络能力.

## 2. YOLO 采用 OpenCV 的原因

[1] - **更易于与 OpenCV 应用的整合**

如果已有应用已经采用了 OpenCV，则可以很方便的使用 YOLOV3，而无需担心编译新增的 Darknet 源码.

[2] - **OpenCV CPU 版本速度更快，9x倍提速**

OpenCV 中的 DNN 模块的 CPU 实现是很快的. 例如，采用 OpenMP 的 Darknet 对于单张图片的一次 CPU 推断大约耗时 2s；而 OpenCV 的实现仅仅只需 0.22s.

[3] - **Python 支持**

Darknet 是以 C 构建的，其原声不支持 Python. 而 OpenCV 是原生支持 Python 的. 尽管Darknet 也会有可用的 Python 接口.

## 3. 基于 OpenCV 和 Darknet 的 YOLOV3 速度测试

如下表：

| OS | Framework | CPU/GPU | Time(ms)/Frame |
|---|---|---|---|
| Linux 16.04 | Darknet | 12x Intel Core i7-6850K CPU @ 3.60GHz | 9370 |
| Linux 16.04 | Darknet + OpenMP | 12x Intel Core i7-6850K CPU @ 3.60GHz | 1942 |
| Linux 16.04 | OpenCV [CPU] | 12x Intel Core i7-6850K CPU @ 3.60GHz | 220 |
| Linux 16.04 | Darknet | NVIDIA GeForce 1080 Ti GPU | 23 |
| macOS | DarkNet | 2.5 GHz Intel Core i7 CPU | 7260 |
| macOS | OpenCV [CPU] | 2.5 GHz Intel Core i7 CPU | 400 |

所有测试中，网络输入均为 416x416. 不出意外的，Darknet 的 GPU 版本速度是最快的. 而且，不出意外的还有，采用 OpenMP 的 Darknet 比未采用 OpeMP 的的 Darknet 具有更好的表现，因为 OpenMP 可以支持多核CPU.

采用 OpenCV 的 DNN 的 CPU 实现，比 OpenMP 速度快了 9 倍.

**注**：在采用 OpenCV 的 DNN GPU 实现时遇到了问题. 这里只在 Intel 的 GPUs 上进行了测试，如果没有 intel GPU 则自动切换到 CPU 模型.

## 4. 采用 YOLOV3 进行目标检测(C++/Python)

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

> Github - ObjectDetection-YOLO

### 4.1. Step1 - 下载模型

```
wget https://pjreddie.com/media/files/yolov3.weights
wget https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg?raw=true -O ./yolov3.cfg
wget https://github.com/pjreddie/darknet/blob/master/data/coco.names?raw=true -O ./coco.names
```

**yolov3.weights** 文件包含了预训练的网络权重；

**yolov3.cfg** 文件包含 了网络配置；

**coco.names** 文件包含了 COCO 数据集中的 80 个不同类别名.

### 4.2. Step2 - 初始化参数

YOLOV3 算法输出边界框作为预测的检测结果. 每个预测框关联了一个置信度.

在第一阶段，所有低于置信阈值参数的 boxes 被忽略，并不进一步处理.

对于剩余的 boxes，采用 NMS 算法进行处理，以移除冗余的重叠边界框. NMS 由参数 **nmsThreshold** 来控制. 可以通过修改这些参数，来观察输出的预测 boxes 数量的变化.

接着，设置网络的输入图片的默认尺寸 - width(**inpWidth**) 和 height(**inpHeight**). 这里均设置为 416，以便于与 YOLOV3 作者开源的 Darknet C 代码进行对比. 也可以设置为 320 以得到更快的速度，设置为 608 以得到更好的精度.

**Python**：

```
# 参数初始化
confThreshold = 0.5  #Confidence threshold
nmsThreshold = 0.4   #Non-maximum suppression threshold
inpWidth = 416       #Width of network's input image
inpHeight = 416      #Height of network's input image
```

**C++**：

```
// 参数初始化
float confThreshold = 0.5; // Confidence threshold
float nmsThreshold = 0.4;  // Non-maximum suppression threshold
int inpWidth = 416;        // Width of network's input image
int inpHeight = 416;       // Height of network's input image
```

### 4.3. Step3 - 加载模型和类别名

**coco.names** 包含了模型训练时的物体类别名. 首先读取该文件.

然后，加载网络，其包含两部分：

[1] - **yolov3.weights** - 预训练的模型权重

[2] **yolov3.cfg** - 网络配置文件

这里，设置 DNN 后端为 OpenCV， 目标设置为 CPU. 也可以设置为 **cv.dnn.DNN_TARGET_OPENCL** 以在 GPU 上运行. 但要记得，当前 OpenCV 版本只支持 Intel 的 GPUs 测试，如果不是 Intel GPU，则会自动切换到 CPU 运行.

**Python**：

```python
# Load names of classes
classesFile = "coco.names";
classes = None
with open(classesFile, 'rt') as f:
    classes = f.read().rstrip('\n').split('\n')

# Give the configuration and weight files for the model
# and load the network using them.
modelConfiguration = "yolov3.cfg";
modelWeights = "yolov3.weights";

net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)
```

**C++**：

```cpp
// Load names of classes
string classesFile = "coco.names";
ifstream ifs(classesFile.c_str());
string line;
while (getline(ifs, line)) classes.push_back(line);

// Give the configuration and weight files for the model
String modelConfiguration = "yolov3.cfg";
String modelWeights = "yolov3.weights";

// Load the network
Net net = readNetFromDarknet(modelConfiguration, modelWeights);
net.setPreferableBackend(DNN_BACKEND_OPENCV);
net.setPreferableTarget(DNN_TARGET_CPU);
```

### 4.4. Step4 - 读取输入

这里从图像、视频或摄像头读取输入.

另外，也使用了 Video writer，以视频方式保存带有输出边界框的每一帧图片.

**Python**：

```python
outputFile = "yolo_out_py.avi"
if (args.image):
    # Open the image file
    if not os.path.isfile(args.image):
        print("Input image file ", args.image, " doesn't exist")
        sys.exit(1)
    cap = cv.VideoCapture(args.image)
    outputFile = args.image[:-4]+'_yolo_out_py.jpg'
elif (args.video):
    # Open the video file
    if not os.path.isfile(args.video):
        print("Input video file ", args.video, " doesn't exist")
        sys.exit(1)
    cap = cv.VideoCapture(args.video)
    outputFile = args.video[:-4]+'_yolo_out_py.avi'
else:
    # Webcam input
    cap = cv.VideoCapture(0)

# Get the video writer initialized to save the output video
if (not args.image):
    vid_writer = cv.VideoWriter(
        outputFile,
        cv.VideoWriter_fourcc('M','J','P','G'),
        30,
        (round(cap.get(cv.CAP_PROP_FRAME_WIDTH)),
         round(cap.get(cv.CAP_PROP_FRAME_HEIGHT))))
```

**C++**:

```
outputFile = "yolo_out_cpp.avi";
if (parser.has("image"))
{
    // Open the image file
    str = parser.get<String>("image");
    ifstream ifile(str);
    if (!ifile) throw("error");
    cap.open(str);
    str.replace(str.end()-4, str.end(), "_yolo_out.jpg");
    outputFile = str;
}
else if (parser.has("video"))
{
    // Open the video file
    str = parser.get<String>("video");
    ifstream ifile(str);
    if (!ifile) throw("error");
    cap.open(str);
    str.replace(str.end()-4, str.end(), "_yolo_out.avi");
    outputFile = str;
}
// Open the webcaom
else cap.open(parser.get<int>("device"));

// Get the video writer initialized to save the output video
if (!parser.has("image"))
{
    video.open(outputFile,
               VideoWriter::fourcc('M','J','P','G'),
               28,
               Size(cap.get(CAP_PROP_FRAME_WIDTH),
                    cap.get(CAP_PROP_FRAME_HEIGHT)));
}
```

### 4.5. Step5 - 处理每一帧

神经网络的输入图片需要以 **blob** 的特定格式组织.

当从输入图片或者视频流中读取了一帧图片后，其需要经过 **blobFromImage** 函数的处理，以转换为网络的 **input blob**. 在该处理过程中，图片像素值被采用 1/255 的因子缩放到 [0, 1] 范围；且在不裁剪的情况下，将图片尺寸调整为 (416, 416). **注：** 并未进行任何减均值操作，因此，函数的均值参数采用的是 [0, 0, 0]，并保持 swapRB 为默认值 1.

输入图处理后输出的 blob，被作为网络输入，进行前向计算，以得到输出的预测边界框列表. 网络输出的预测框再进行后处理，以过滤低置信度的边界框. 后面会详细介绍后处理操作. 在左上角打印每一帧图片的推断时间.

图片最终的边界框，会以图片或 video writer 的方式保存到磁盘.

**Python**：

```
while cv.waitKey(1) < 0:
    # get frame from the video
    hasFrame, frame = cap.read()

    # Stop the program if reached end of video
    if not hasFrame:
        print("Done processing !!!")
        print("Output file is stored as ", outputFile)
        cv.waitKey(3000)
        break

    # Create a 4D blob from a frame.
    blob = cv.dnn.blobFromImage(frame,
                                1/255,
                                (inpWidth, inpHeight),
                                [0,0,0],
                                1,
                                crop=False)

    # Sets the input to the network
    net.setInput(blob)

    # Runs the forward pass to get output of the output layers
    outs = net.forward(getOutputsNames(net))

    # Remove the bounding boxes with low confidence
    postprocess(frame, outs)

    # Put efficiency information.
    # The function getPerfProfile returns the overall time for inference(t)
    # and the timings for each of the layers(in layersTimes)
    t, _ = net.getPerfProfile()
    label = 'Inference time: %.2f ms' % (t * 1000.0 / cv.getTickFrequency())
    cv.putText(frame, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255))

    # Write the frame with the detection boxes
    if (args.image):
        cv.imwrite(outputFile, frame.astype(np.uint8));
    else:
        vid_writer.write(frame.astype(np.uint8))
```

**C++**：

```
// Process frames.
while (waitKey(1) < 0)
{
    // get frame from the video
    cap >> frame;
    // Stop the program if reached end of video
    if (frame.empty()) {
        cout << "Done processing !!!" << endl;
        cout << "Output file is stored as " << outputFile << endl;
        waitKey(3000);
        break;
    }
    // Create a 4D blob from a frame.
    blobFromImage(frame, blob, 1/255.0, cvSize(inpWidth, inpHeight), Scalar(0,0,0), true, false);

    //Sets the input to the network
    net.setInput(blob);

    // Runs the forward pass to get output of the output layers
    vector<Mat> outs;
    net.forward(outs, getOutputsNames(net));

    // Remove the bounding boxes with low confidence
    postprocess(frame, outs);

    // Put efficiency information. The function getPerfProfile returns the
    // overall time for inference(t) and the timings for each of the layers(in layersTimes)
    vector<double> layersTimes;
    double freq = getTickFrequency() / 1000;
    double t = net.getPerfProfile(layersTimes) / freq;
    string label = format("Inference time for a frame : %.2f ms", t);
    putText(frame, label, Point(0, 15), FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0, 0, 255));

    // Write the frame with the detection boxes
    Mat detectedFrame;
    frame.convertTo(detectedFrame, CV_8U);
    if (parser.has("image")) imwrite(outputFile, detectedFrame);
    else video.write(detectedFrame);
}
```

下面详细的对上面用到的一些函数进行说明.

4.5.1. Step5a - 获取网络输出层名

OpenCV 的 Net 类的 **forward** 函数需要知道网络的最终输出层.

由于要对整个网络进行运行，因此，需要确认网络的最后一层. 可以采用 **getUnconnectedOutLayers()** 函数来获取无连接的输出层的名字，这些层一般都是网络的输出层.

然后，运行网络的 forward 计算，以得到输出层的名字，如代码段 **net.forward(getOutputsNames(net))**.

**Python**:

```python
# Get the names of the output layers
def getOutputsNames(net):
    # Get the names of all the layers in the network
    layersNames = net.getLayerNames()
    # Get the names of the output layers,
    # i.e. the layers with unconnected outputs
    return [layersNames[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

**C++**:

```cpp
// Get the names of the output layers
vector<String> getOutputsNames(const Net& net)
{
    static vector<String> names;
    if (names.empty())
    {
        // Get the indices of the output layers,
        // i.e. the layers with unconnected outputs
        vector<int> outLayers = net.getUnconnectedOutLayers();

        //get the names of all the layers in the network
        vector<String> layersNames = net.getLayerNames();

        // Get the names of the output layers in names
        names.resize(outLayers.size());
        for (size_t i = 0; i < outLayers.size(); ++i)
        names[i] = layersNames[outLayers[i] - 1];
    }
    return names;
}
```

### 4.5.2. Step5b - 网络输出的后处理

网络输出的每个边界框表示为 类别名 + 5个元素的向量.

向量的前 4 个元素分别为: **center_x** , **center_y** , **width** 和 **height**.

第 5 个元素表示包含物体的边界框的置信度.

其余的元素是与每个类别相关的置信度(概率). 边界框被分配到对应于最高分数的类别. box 的最高分数也被叫作 **置信confidence**. 如果 box 的置信低于给定阈值，则丢弃该边界框，并不进行进一步的后处理.

置信大于或等于给定置信阈值的 boxes，会进行 NMS 进一步处理，以减少重叠 boxes 的数量.

**Python**:

```python
# Remove the bounding boxes with low confidence using nms
def postprocess(frame, outs):
    frameHeight = frame.shape[0]
    frameWidth = frame.shape[1]

    classIds = []
    confidences = []
    boxes = []
    # Scan through all the bounding boxes output from the network and
    # keep only the ones with high confidence scores.
    # Assign the box's class label as the class with the highest score.
    classIds = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            classId = np.argmax(scores)
            confidence = scores[classId]
            if confidence > confThreshold:
                center_x = int(detection[0] * frameWidth)
                center_y = int(detection[1] * frameHeight)
                width = int(detection[2] * frameWidth)
                height = int(detection[3] * frameHeight)
                left = int(center_x - width / 2)
                top = int(center_y - height / 2)
                classIds.append(classId)
                confidences.append(float(confidence))
                boxes.append([left, top, width, height])

    # Perform nms to eliminate redundant overlapping boxes with
    # lower confidences.
    indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)
    for i in indices:
        i = i[0]
        box = boxes[i]
        left = box[0]
        top = box[1]
        width = box[2]
        height = box[3]
        drawPred(classIds[i], confidences[i], left, top, left + width, top + height)
```

**C++**:

```cpp
// Remove the bounding boxes with low confidence using nms
void postprocess(Mat& frame, const vector<Mat>& outs)
{
    vector<int> classIds;
    vector<float> confidences;
    vector<Rect> boxes;

    for (size_t i = 0; i < outs.size(); ++i)
    {
        // Scan through all the bounding boxes output from the network
        // and keep only the ones with high confidence scores.
        // Assign the box's class label as the class
        // with the highest score for the box.
        float* data = (float*)outs[i].data;
        for (int j = 0; j < outs[i].rows; ++j, data += outs[i].cols)
        {
            Mat scores = outs[i].row(j).colRange(5, outs[i].cols);
            Point classIdPoint;
            double confidence;
            // Get the value and location of the maximum score
            minMaxLoc(scores, 0, &confidence, 0, &classIdPoint);
            if (confidence > confThreshold)
            {
                int centerX = (int)(data[0] * frame.cols);
                int centerY = (int)(data[1] * frame.rows);
                int width = (int)(data[2] * frame.cols);
                int height = (int)(data[3] * frame.rows);
                int left = centerX - width / 2;
                int top = centerY - height / 2;

                classIds.push_back(classIdPoint.x);
                confidences.push_back((float)confidence);
                boxes.push_back(Rect(left, top, width, height));
            }
        }
    }

    // Perform nms to eliminate redundant overlapping boxes with
    // lower confidences
    vector<int> indices;
    NMSBoxes(boxes, confidences, confThreshold, nmsThreshold, indices);
    for (size_t i = 0; i < indices.size(); ++i)
    {
        int idx = indices[i];
        Rect box = boxes[idx];
        drawPred(classIds[idx], confidences[idx], box.x, box.y,
                 box.x + box.width, box.y + box.height, frame);
    }
}
```

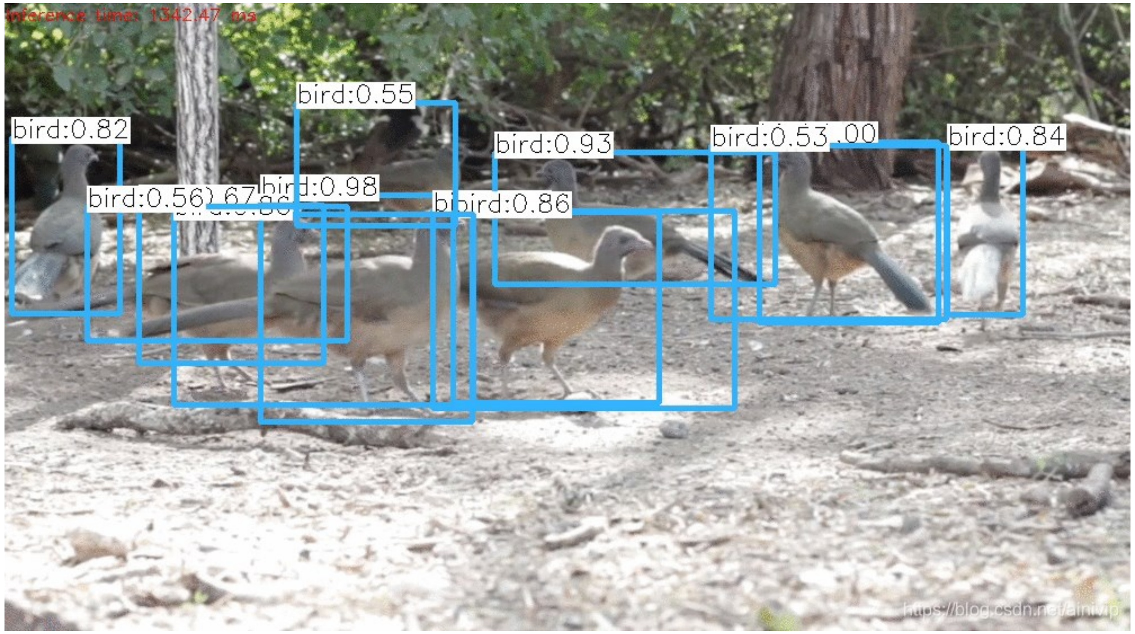
NMS 是由 **nmsThreshold** 参数控制的.

如果 **nmsThreshold** 参数过小，如 0.1，可能检测不到相同或不同类别的重叠物体.

如果 **nmsThreshold** 参数过大，如，1，则会得到同一个物体的多个框.

因此，这里采用了一个中间值 - 0.4.

不同 NMS 阈值效果如图：



https://aiuai.cn/uploads/1905/46a9eedbecd0f8b6.gif

4.5.3. Step5c - 画出预测框

最后，画出输入图片在 NMS 处理后的边界框以及对应的类别标签和置信分数.

**Python**：

```
# Draw the predicted bounding box
def drawPred(classId, conf, left, top, right, bottom):
    # Draw a bounding box.
    cv.rectangle(frame, (left, top), (right, bottom), (0, 0, 255))
```

```python

    label = '%.2f' % conf
    # Get the label for the class name and its confidence
    if classes:
        assert(classId < len(classes))
        label = '%s:%s' % (classes[classId], label)

    #Display the label at the top of the bounding box
    labelSize, baseLine = cv.getTextSize(
        label, cv.FONT_HERSHEY_SIMPLEX, 0.5, 1)
    top = max(top, labelSize[1])
    cv.putText(frame, label, (left, top),
               cv.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255))
```

**C++**:

```cpp
// Draw the predicted bounding box
void drawPred(int classId, float conf, int left, int top, int right, int bottom, Mat& frame)
{
    //Draw a rectangle displaying the bounding box
    rectangle(frame, Point(left, top), Point(right, bottom), Scalar(0, 0, 255));

    //Get the label for the class name and its confidence
    string label = format("%.2f", conf);
    if (!classes.empty())
    {
        CV_Assert(classId < (int)classes.size());
        label = classes[classId] + ":" + label;
    }

    //Display the label at the top of the bounding box
    int baseLine;
    Size labelSize = getTextSize(label, FONT_HERSHEY_SIMPLEX, 0.5, 1, &baseLine);
    top = max(top, labelSize.height);
    putText(frame, label, Point(left, top), FONT_HERSHEY_SIMPLEX, 0.5, Scalar(255,255,255));
}
```

### 4.6. YOLOV3 完整测试代码(Python)

> object_detection_yolo.py

用法:

```
python3 object_detection_yolo.py --video=run.mp4
python3 object_detection_yolo.py --image=bird.jpg
```

```python
# It is based on the OpenCV project.
import cv2 as cv
import argparse
import sys
import numpy as np
import os.path

# Initialize the parameters
confThreshold = 0.5  #Confidence threshold
```

```
nmsThreshold = 0.4   #Non-maximum suppression threshold
inpWidth = 416       #Width of network's input image
inpHeight = 416      #Height of network's input image

parser = argparse.ArgumentParser(description='Object Detection using YOLO in OPENCV')
parser.add_argument('--image', help='Path to image file.')
parser.add_argument('--video', help='Path to video file.')
args = parser.parse_args()

# Load names of classes
classesFile = "coco.names";
classes = None
with open(classesFile, 'rt') as f:
    classes = f.read().rstrip('\n').split('\n')

# Give the configuration and weight files for the model and load the network using them.
modelConfiguration = "yolov3.cfg";
modelWeights = "yolov3.weights";

net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)

# Get the names of the output layers
def getOutputsNames(net):
    # Get the names of all the layers in the network
    layersNames = net.getLayerNames()
    # Get the names of the output layers,
    # i.e. the layers with unconnected outputs
    return [layersNames[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# Draw the predicted bounding box
def drawPred(classId, conf, left, top, right, bottom):
    # Draw a bounding box.
    cv.rectangle(frame, (left, top), (right, bottom), (255, 178, 50), 3)

    label = '%.2f' % conf

    # Get the label for the class name and its confidence
    if classes:
        assert(classId < len(classes))
        label = '%s:%s' % (classes[classId], label)

    #Display the label at the top of the bounding box
    labelSize, baseLine = cv.getTextSize(label, cv.FONT_HERSHEY_SIMPLEX, 0.5, 1)
    top = max(top, labelSize[1])
    cv.rectangle(frame, (left, top - round(1.5*labelSize[1])), (left + round(1.5*labelSize[0]), top + baseLine),
    cv.putText(frame, label, (left, top), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,0), 1)

# Remove the bounding boxes with low confidence using nms
def postprocess(frame, outs):
    frameHeight = frame.shape[0]
    frameWidth = frame.shape[1]

    classIds = []
    confidences = []
    boxes = []
    # Scan through all the bounding boxes output from the network and
    # keep only the ones with high confidence scores.
    # Assign the box's class label as the class with the highest score.
    classIds = []
```

```python
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            classId = np.argmax(scores)
            confidence = scores[classId]
            if confidence > confThreshold:
                center_x = int(detection[0] * frameWidth)
                center_y = int(detection[1] * frameHeight)
                width = int(detection[2] * frameWidth)
                height = int(detection[3] * frameHeight)
                left = int(center_x - width / 2)
                top = int(center_y - height / 2)
                classIds.append(classId)
                confidences.append(float(confidence))
                boxes.append([left, top, width, height])

    # Perform nms to eliminate redundant overlapping boxes with
    # lower confidences.
    indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)
    for i in indices:
        i = i[0]
        box = boxes[i]
        left = box[0]
        top = box[1]
        width = box[2]
        height = box[3]
        drawPred(classIds[i], confidences[i], left, top, left + width, top + height)

# Process inputs
winName = 'Deep learning object detection in OpenCV'
cv.namedWindow(winName, cv.WINDOW_NORMAL)

outputFile = "yolo_out_py.avi"
if (args.image):
    # Open the image file
    if not os.path.isfile(args.image):
        print("Input image file ", args.image, " doesn't exist")
        sys.exit(1)
    cap = cv.VideoCapture(args.image)
    outputFile = args.image[:-4]+'_yolo_out_py.jpg'
elif (args.video):
    # Open the video file
    if not os.path.isfile(args.video):
        print("Input video file ", args.video, " doesn't exist")
        sys.exit(1)
    cap = cv.VideoCapture(args.video)
    outputFile = args.video[:-4]+'_yolo_out_py.avi'
else:
    # Webcam input
    cap = cv.VideoCapture(0)

# Get the video writer initialized to save the output video
if (not args.image):
    vid_writer = cv.VideoWriter(outputFile, cv.VideoWriter_fourcc('M','J','P','G'), 30, (round(cap.get(cv.CAP_PRO

while cv.waitKey(1) < 0:

    # get frame from the video
    hasFrame, frame = cap.read()

    # Stop the program if reached end of video
    if not hasFrame:
        print("Done processing !!!")
        print("Output file is stored as ", outputFile)
        cv.waitKey(3000)
        # Release device
        cap.release()
        break

    # Create a 4D blob from a frame.
    blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1, crop=False)

    # Sets the input to the network
    net.setInput(blob)
    # Runs the forward pass to get output of the output layers
    outs = net.forward(getOutputsNames(net))
    # Remove the bounding boxes with low confidence
    postprocess(frame, outs)

    # Put efficiency information.
    # The function getPerfProfile returns the overall time for inference(t)
    # and the timings for each of the layers(in layersTimes)
    t, _ = net.getPerfProfile()
    label = 'Inference time: %.2f ms' % (t * 1000.0 / cv.getTickFrequency())
    cv.putText(frame, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255))

    # Write the frame with the detection boxes
    if (args.image):
        cv.imwrite(outputFile, frame.astype(np.uint8));
    else:
        vid_writer.write(frame.astype(np.uint8))
    cv.imshow(winName, frame)
```

### 4.7. YOLOV3 完整测试代码(C++)

> object_detection_yolo.cpp

用法：

```
./object_detection_yolo.out --video=run.mp4
./object_detection_yolo.out --image=bird.jpg
```

```cpp
// It is based on the OpenCV project.
#include <fstream>
#include <sstream>
#include <iostream>

#include <opencv2/dnn.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>

const char* keys =
"{help h usage ? | | Usage examples: \n\t\t./object_detection_yolo.out --image=dog.jpg \n\t\t./object_detection_y
"{image i        |<none>| input image   }"
"{video v       |<none>| input video   }"
```

```
;
using namespace cv;
using namespace dnn;
using namespace std;

// Initialize the parameters
float confThreshold = 0.5; // Confidence threshold
float nmsThreshold = 0.4;  // Non-maximum suppression threshold
int inpWidth = 416;  // Width of network's input image
int inpHeight = 416; // Height of network's input image
vector<string> classes;

// Remove the bounding boxes with low confidence using nms
void postprocess(Mat& frame, const vector<Mat>& out);

// Draw the predicted bounding box
void drawPred(int classId, float conf, int left, int top, int right, int bottom, Mat& frame);

// Get the names of the output layers
vector<String> getOutputsNames(const Net& net);

int main(int argc, char** argv)
{
    CommandLineParser parser(argc, argv, keys);
    parser.about("Use this script to run object detection using YOLO3 in OpenCV.");
    if (parser.has("help"))
    {
        parser.printMessage();
        return 0;
    }
    // Load names of classes
    string classesFile = "coco.names";
    ifstream ifs(classesFile.c_str());
    string line;
    while (getline(ifs, line)) classes.push_back(line);

    // Give the configuration and weight files for the model
    String modelConfiguration = "yolov3.cfg";
    String modelWeights = "yolov3.weights";

    // Load the network
    Net net = readNetFromDarknet(modelConfiguration, modelWeights);
    net.setPreferableBackend(DNN_BACKEND_OPENCV);
    net.setPreferableTarget(DNN_TARGET_CPU);

    // Open a video file or an image file or a camera stream.
    string str, outputFile;
    VideoCapture cap;
    VideoWriter video;
    Mat frame, blob;

    try {
        outputFile = "yolo_out_cpp.avi";
        if (parser.has("image"))
        {
            // Open the image file
            str = parser.get<String>("image");
            ifstream ifile(str);
            if (!ifile) throw("error");
            cap.open(str);
            str.replace(str.end()-4, str.end(), "_yolo_out_cpp.jpg");
            outputFile = str;
        }
        else if (parser.has("video"))
        {
            // Open the video file
            str = parser.get<String>("video");
            ifstream ifile(str);
            if (!ifile) throw("error");
            cap.open(str);
            str.replace(str.end()-4, str.end(), "_yolo_out_cpp.avi");
            outputFile = str;
        }
        // Open the webcaom
        else cap.open(parser.get<int>("device"));

    }
    catch(...) {
        cout << "Could not open the input image/video stream" << endl;
        return 0;
    }

    // Get the video writer initialized to save the output video
    if (!parser.has("image")) {
        video.open(outputFile, VideoWriter::fourcc('M','J','P','G'), 28, Size(cap.get(CAP_PROP_FRAME_WIDTH), cap.
    }

    // Create a window
    static const string kWinName = "Deep learning object detection in OpenCV";
    namedWindow(kWinName, WINDOW_NORMAL);

    // Process frames.
    while (waitKey(1) < 0)
    {
        // get frame from the video
        cap >> frame;

        // Stop the program if reached end of video
        if (frame.empty()) {
            cout << "Done processing !!!" << endl;
            cout << "Output file is stored as " << outputFile << endl;
            waitKey(3000);
            break;
        }
        // Create a 4D blob from a frame.
        blobFromImage(frame, blob, 1/255.0, cvSize(inpWidth, inpHeight), Scalar(0,0,0), true, false);
        //Sets the input to the network
        net.setInput(blob);
        // Runs the forward pass to get output of the output layers
        vector<Mat> outs;
        net.forward(outs, getOutputsNames(net));

        // Remove the bounding boxes with low confidence
        postprocess(frame, outs);

        // Put efficiency information.
        // The function getPerfProfile returns the overall time for inference(t)
        // and the timings for each of the layers(in layersTimes)
        vector<double> layersTimes;
        double freq = getTickFrequency() / 1000;
        double t = net.getPerfProfile(layersTimes) / freq;
        string label = format("Inference time for a frame : %.2f ms", t);
```

```cpp
        putText(frame, label, Point(0, 15), FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0, 0, 255));

        // Write the frame with the detection boxes
        Mat detectedFrame;
        frame.convertTo(detectedFrame, CV_8U);
        if (parser.has("image")) imwrite(outputFile, detectedFrame);
        else video.write(detectedFrame);

        imshow(kWinName, frame);
    }
    cap.release();
    if (!parser.has("image")) video.release();

    return 0;
}

// Remove the bounding boxes with low confidence using nms
void postprocess(Mat& frame, const vector<Mat>& outs)
{
    vector<int> classIds;
    vector<float> confidences;
    vector<Rect> boxes;

    for (size_t i = 0; i < outs.size(); ++i)
    {
        // Scan through all the bounding boxes output from the network and
        // keep only the ones with high confidence scores.
        // Assign the box's class label as the class
        // with the highest score for the box.
        float* data = (float*)outs[i].data;
        for (int j = 0; j < outs[i].rows; ++j, data += outs[i].cols)
        {
            Mat scores = outs[i].row(j).colRange(5, outs[i].cols);
            Point classIdPoint;
            double confidence;
            // Get the value and location of the maximum score
            minMaxLoc(scores, 0, &confidence, 0, &classIdPoint);
            if (confidence > confThreshold)
            {
                int centerX = (int)(data[0] * frame.cols);
                int centerY = (int)(data[1] * frame.rows);
                int width = (int)(data[2] * frame.cols);
                int height = (int)(data[3] * frame.rows);
                int left = centerX - width / 2;
                int top = centerY - height / 2;

                classIds.push_back(classIdPoint.x);
                confidences.push_back((float)confidence);
                boxes.push_back(Rect(left, top, width, height));
            }
        }
    }

    // Perform nms to eliminate redundant overlapping boxes with
    // lower confidences
    vector<int> indices;
    NMSBoxes(boxes, confidences, confThreshold, nmsThreshold, indices);
    for (size_t i = 0; i < indices.size(); ++i)
    {
        int idx = indices[i];
        Rect box = boxes[idx];
        drawPred(classIds[idx], confidences[idx], box.x, box.y,
                 box.x + box.width, box.y + box.height, frame);
    }
}

// Draw the predicted bounding box
void drawPred(int classId, float conf, int left, int top, int right, int bottom, Mat& frame)
{
    //Draw a rectangle displaying the bounding box
    rectangle(frame, Point(left, top), Point(right, bottom), Scalar(255, 178, 50), 3);

    //Get the label for the class name and its confidence
    string label = format("%.2f", conf);
    if (!classes.empty())
    {
        CV_Assert(classId < (int)classes.size());
        label = classes[classId] + ":" + label;
    }

    //Display the label at the top of the bounding box
    int baseLine;
    Size labelSize = getTextSize(label, FONT_HERSHEY_SIMPLEX, 0.5, 1, &baseLine);
    top = max(top, labelSize.height);
    rectangle(frame, Point(left, top - round(1.5*labelSize.height)), Point(left + round(1.5*labelSize.width), top
    putText(frame, label, Point(left, top), FONT_HERSHEY_SIMPLEX, 0.75, Scalar(0,0,0),1);
}

// Get the names of the output layers
vector<String> getOutputsNames(const Net& net)
{
    static vector<String> names;
    if (names.empty())
    {
        // Get the indices of the output layers,
        // i.e. the layers with unconnected outputs
        vector<int> outLayers = net.getUnconnectedOutLayers();

        //get the names of all the layers in the network
        vector<String> layersNames = net.getLayerNames();

        // Get the names of the output layers in names
        names.resize(outLayers.size());
        for (size_t i = 0; i < outLayers.size(); ++i)
        names[i] = layersNames[outLayers[i] - 1];
    }
    return names;
}
```

YOLOV3 基于OpenCV DNN 的目标检测实现　　长风破浪会有时，直挂云帆济沧海　2999

原文: YOLOV3 基于OpenCV DNN 的目标检测实现 - AIUAI 这里主要是对 基于 YOLOV3 和 OpenCV的目标检测(PythonC++)[译] Python 完整实现的整理. ...

opencv摄像头实时拍摄显示　　04-07

利用opencv实现了摄像头的实时拍摄并显示

评论 1　您还未登录，请先 登录 后发表或查看评论

OpenCV+yolov3实现目标检测(C++,Python)_pan_jinquan的...　　6-13

OpenCV+yolov3实现目标检测(C++,Python) 目标检测算法主要分为两类:一类是基于Region Proposal(候选区域)的算法,如R-CNN系算法(R-CNN,Fast R-C...

data-life VIP

码龄18年　暂无认证

| 40 | 1万+ | 1万+ | 28万+ | |
|---|---|---|---|---|
| 原创 | 周排名 | 总排名 | 访问 | 等级 |
| 2640 | 39 | 88 | 39 | 492 |
| 积分 | 粉丝 | 获赞 | 评论 | 收藏 |

私信　　关注

搜博主文章

### 热门文章

Django实现html页面中 "GET /favicon.ico HTTP/1.1" 404 2181报错 27443

Python如何释放内存 11583

pd.read_csv中的usecols函数实现读取指定列 9561

在sql语句查询时如何把查询为空的数据显示为零 7069

python每天固定时间运行某程序代码 6902

### 最新评论

Failed to restart network.service: Unit n...

小五809: 怨我初入hadoop不是很懂后面大X指代什么，都试过了全是"未知的连接"可...

selenium之find_element_by_xpath定位...

m0_66463207: 为什么By.XPATH中的By出现红色曲线下划线，运行时还报错：nam...

批量创建文本文档

血泪天使的翅膀: 亲自试了，可以新建，但是出来的文件夹名乱码了，都是类似加密...

基于 YOLOV3 和 OpenCV的目标检测(P...

十里桃花~: 你好，这个python代码我的是打开摄像头来检测，但是要检测已经保存...

实现可拖拽，拉伸，吸附功能的甘特图(...

小陈的book: 你好，可以出一个x,y轴都可以拖动的教程么

### 您愿意向朋友推荐“博客详情页”吗？

强烈不推荐　不推荐　一般般　推荐　强烈推荐

### 最新文章

CentOS安装Python3

如何使用Python对时间段内的行进行分组

生物钟的研究、功能和机理——2017年度诺贝尔生理学或医学奖成果简析

| | |
|---|---|
| 2022年 3篇 | 2021年 18篇 |
| 2020年 99篇 | 2019年 54篇 |
| 2018年 7篇 | 2005年 13篇 |

【yolov3目标检测】(3) opencv+yolov3 检测交通路况,附... 5-1
各位同学好,今天和大家分享一下如何使用 opencv 调用 yolov3 模型,加载网络权重,很方便地实现 yolov3 目标检测。先放张图看效果。使用的网上找的行...

YOLO实现目标检测（利用Python和Opencv） qq_45445740的博客 3803
首先，下载相关的权重文件、配置文件和待检测图像。 链接：https://pan.baidu.com/s/1iX_g4PoeKNP9mNmITniCJg 提取码：0djr 1.关于YOLO

OpenCV+yolov3实现目标检测(C++,Python) pan_jinquan的博客 2万+
OpenCV+yolov3实现目标检测(C++,Python) 目标检测算法主要分为两类：一类是基于Region Proposal（候选区域）的算法，如R-CNN系算法（R-CNN，...

opencv-python+yolov3实现目标检测_爱CV的博客-CSDN博... 4-28
opencv-python+yolov3实现目标检测 目标检测概况 目标检测是? 目标检测,粗略来说就是:输入图片/视频,经过处理,得到:目标的位置信息(比如左上角和右...

YOLOV3 基于OpenCV DNN 的目标检测实现_AIHGF的博客 6-3
这里主要是对基于 YOLOV3 和 OpenCV的目标检测(PythonC++)[译]Python 完整实现的整理. 基于YOLOV3 和 OpenCV的目标检测(PythonC++) - AIUAI O...

基于OpenCV和YOLOv3深度学习的目标检测 CSDNwei的专栏 360
本文翻译自Deep Learning based Object Detection using YOLOv3 with OpenCV ( Python / C++ ) 基于OpenCV和YOLOv3深度学习的目标检测 本文，我...

目标检测实战必会！4种基于YOLO目标检测（Python和C++两种版本实现） yishuihanq的博客 551
目标检测实战必会！4种基于YOLO目标检测（Python和C++两种版本实现） AI算法修炼营1周前 以下文章来源于极市平台，作者CV开发者都爱看的 极市...

python目标识别运行库_目标检测和识别:Python+OpenCV+Y... 4-14
1.1 熟悉Python+OpenCV+Yolov3的目标检测和识别,以上三个都是强大的计算机视觉编程语言和库。 1.2 本代码注释清楚,小白秒懂。 1.4 原图: 1.5 效果...

OpenCV中blobFromImage函数详细解释 一点儿也不萌的萌萌 2万+
OpenCV中blobFromImage函数详细解释 在OpenCV 3.3之后的版本中，支持调用训练好的深度学习框架，其中有一些重要的函数，今天先总结一下blobFr...

blobFromImage() 函数详解 qq_33591712的博客 1万+
blobFromImage() Mat cv::dnn::blobFromImage ( InputArray image, double scalefactor = 1.0, const Size &amp; size = Size(), const Scalar &amp; mean ...

OpenCV笔记5：使用OpenCV Python在视频上显示日期和时间 weixin_41788560的博客 2770
1、学习目标 （1）在python中使用opencv将文本放在视频上 （2）使用OpenCV Python在视频上显示日期和时间 2、使用函数 cv2.putText() 3、程序 imp...

opencv-python+yolov3实现目标检测 爱CV 861
原文：https://www.cnblogs.com/hesse-summer/p/11335865.html 目标检测概况 目标检测是？ 目标检测，粗略来说就是：输入图片/视频，经过处理，得...

cv2.dnn.blobFromImage()函数用法 土豆 2万+
函数cv2.dnn.blobFromImage(image[, scalefactor[, size[, mean[, swapRB[, crop[, ddepth]]]]]]) 作用： 对图像进行预处理，包括减均值，比例缩放，裁剪，...

cv2.dnn.blobFromImage函数 baidu_38505667的博客 1万+
深度学习：OpenCV的blobFromImage如何工作 深度学习中OpnenCV的blobFromImage是对输入图像做了什么呢？ 在PyImageSearch有许多读者好奇，b...

关于CV2.dnn.readNetFromDarknet(config_path, weights_path)报错解决方式 a1103580557的博客 2519
最近在用yolo做人脸识别的时候发现使用 CV2.dnn.readNetFromDarknet(config_path, weights_path) 经常会出现错误 cv2.error: OpenCV(4.5.3) C:\Users...

深度学习：Opencv的blobFromImage是如何工作的 d3f4ult的博客 2360
文章目录说明介绍blobFromImage 的工作流程深度学习和均值减法blobFromImage 和 blobFromImages使用blobFromImage 方法深度学习结果展示总结源...

用python实现目标检测与手势识别的原理简单总结 最新发布 m0_63626366的博客 381
一、 目标检测 目标检测属于分类和回归的综合问题。 目标检测是借助于计算机和软件系统在图像/场景中，定位目标并识别出每个目标的类别的技术。 基...

Opencv+C++之身份证识别（一） weixin_34104341的博客 1050
五月份各种课程，也是最后一个学期了，所以就没有跟大家分享自己的一些所学。现在课程终于结束了，即将开始下一阶段的项目开发，所以趁这个间隙...

基于opencv的车牌识别解析与代码 热门推荐 LinJM-机器视觉 5万+
车牌识别太出名了，我也就花几天来了解下这个系统，并结合opencv进行实现。下面是一些介绍： 车辆牌照识别（License Plate Recognition，LPR）技...

“相关推荐”对你有帮助么？

非常没帮助　没帮助　一般　有帮助　非常有帮助

data-life 关注 1 1 8 专栏目录

限时特惠

举报