

♦ 数据库课程设计

Q 搜索

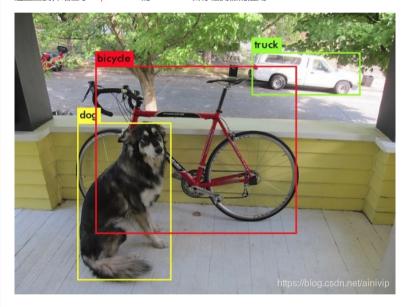
登录/注册 会员中心 🞁 足迹 动态



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<sup>Q</sup> 的 YOLOV3 目标检测器的应用.



YOLOV3<sup>Q</sup> 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.

### 目录

- 1. YOLO 工作原理
- 2. YOLO 采用 OpenCV 的原因
- 3. 基于 OpenCV 和 Darknet 的 YOLOV...
- 4. 采用 YOLOV3 进行目标检测(C++/Pyt...
  - 4.1. Step1 下载模型
  - 4.2. Step2 初始化参数
  - 4.3. Step3 加载模型和类别名
  - 4.4. Step4 读取输入
  - 4.5. Step5 处理每一帧
  - 4.6. YOLOV3 完整测试代码(Python)

### 4.7. YOLOV3 完整测试代码(C++)

### 分类专栏 C. JAVA学习实践 付费 数据分析 3篇 € 爬虫 10篇 Python 94篇 MACKER'home 1篇 後 我写的文章 4篇 百宝箱 5篇 **精彩的文档** 1篇 6 编程生涯 6篇 C 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

13篇

CentOS

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	СРИ/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 -0 ./yolov3.cfg
wget https://github.com/pjreddie/darknet/blob/master/data/coco.names?raw=true -0 ./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:

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

#### C++:

```
1 // 参数初始化
2 float confThreshold = 0.5; // Confidence threshold
3 float nmsThreshold = 0.4; // Non-maximum suppression threshold
4 int inpWidth = 416; // Width of network's input image
5 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:

### C++:

```
1  // Load names of classes
2  string classesFile = "coco.names";
3  ifstream ifs(classesFile.c_str());
4  string line;
5  while (getline(ifs, line)) classes.push_back(line);
6
7  // Give the configuration and weight files for the model
8  String modelConfiguration = "yolov3.cfg";
9  String modelWeights = "yolov3.weights";
10
11  // Load the network
12  Net net = readNetFromDarknet(modelConfiguration, modelWeights);
13  net.setPreferableBackend(DNN_BACKEND_OPENCV);
14  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.

```
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)
8
        outputFile = args.image[:-4]+'_yolo_out_py.jpg'
9
    elif (args.video):
10
       # Open the video file
11
       if not os.path.isfile(args.video):
12
           print("Input video file ", args.video, " doesn't exist")
13
           sys.exit(1)
14
        cap = cv.VideoCapture(args.video)
15
       outputFile = args.video[:-4]+'_yolo_out_py.avi'
16
   else:
17
       # Webcam input
18
       cap = cv.VideoCapture(0)
19
20
    # Get the video writer initialized to save the output video
21
    if (not args.image):
22
       vid writer = cv.VideoWriter(
23
            outputFile,
24
            cv.VideoWriter fourcc('M','J','P','G'),
25
26
            (round(cap.get(cv.CAP_PROP_FRAME_WIDTH)),
27
             round(cap.get(cv.CAP_PROP_FRAME_HEIGHT))))
```

#### C++:

```
1 outputFile = "yolo_out_cpp.avi";
   if (parser.has("image"))
3 {
4
       // Open the image file
       str = parser.get<String>("image");
       ifstream ifile(str);
       if (!ifile) throw("error");
        cap.open(str);
9
       str.replace(str.end()-4, str.end(), "_yolo_out.jpg");
10
        outputFile = str;
11
12
   else if (parser.has("video"))
13
14
       // Open the video file
15
       str = parser.get<String>("video");
16
       ifstream ifile(str);
17
       if (!ifile) throw("error");
18
        cap.open(str);
19
       str.replace(str.end()-4, str.end(), "_yolo_out.avi");
20
        outputFile = str;
21 }
22
    // Open the webcaom
23
   else cap.open(parser.get<int>("device"));
24
25
    // Get the video writer initialized to save the output video
26
   if (!parser.has("image"))
27
28
        video.open(outputFile,
29
                  VideoWriter::fourcc('M','J','P','G'),
30
31
                  Size(cap.get(CAP_PROP_FRAME_WIDTH),
32
                       cap.get(CAP_PROP_FRAME_HEIGHT)));
33 }
```

### 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
3
       hasFrame, frame = cap.read()
4
5
       # Stop the program if reached end of video
6
       if not hasFrame:
           print("Done processing !!!")
           print("Output file is stored as ", outputFile)
9
           cv.waitKey(3000)
10
           break
11
12
       # Create a 4D blob from a frame.
13
        blob = cv.dnn.blobFromImage(frame,
14
                                   1/255,
15
                                   (inpWidth, inpHeight),
16
                                   [0,0,0],
17
                                   1,
18
                                   crop=False)
19
20
       # Sets the input to the network
21
        net.setInput(blob)
22
23
        # Runs the forward pass to get output of the output layers
24
        outs = net.forward(getOutputsNames(net))
25
26
        # Remove the bounding boxes with Low confidence
27
        postprocess(frame, outs)
28
29
       # Put efficiency information.
30
        # The function getPerfProfile returns the overall time for inference(t)
31
        # and the timings for each of the layers(in layersTimes)
32
        t, _ = net.getPerfProfile()
33
       label = 'Inference time: %.2f ms' % (t * 1000.0 / cv.getTickFrequency())
34
        cv.putText(frame, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255))
35
36
       # Write the frame with the detection boxes
37
        if (args.image):
38
           cv.imwrite(outputFile, frame.astype(np.uint8));
39
        else:
40
            vid_writer.write(frame.astype(np.uint8))
```

### C++:

```
1  // Process frames.
2  while (waitKey(1) < 0)
3  {</pre>
```

```
// get frame from the video
5
        cap >> frame;
        // Stop the program if reached end of video
        if (frame.empty()) {
8
           cout << "Done processing !!!" << endl;</pre>
9
           cout << "Output file is stored as " << outputFile << endl;</pre>
10
           waitKey(3000);
11
           break;
12
13
        // Create a 4D blob from a frame.
14
        blobFromImage(frame, blob, 1/255.0, cvSize(inpWidth, inpHeight), Scalar(0,0,0), true, false);
15
16
        //Sets the input to the network
17
        net.setInput(blob);
18
19
        // Runs the forward pass to get output of the output layers
20
        vector<Mat> outs;
21
        net.forward(outs, getOutputsNames(net));
22
23
        // Remove the bounding boxes with low confidence
24
        postprocess(frame, outs);
25
26
        // Put efficiency information. The function getPerfProfile returns the
27
        // overall time for inference(t) and the timings for each of the layers(in layersTimes)
28
        vector<double> layersTimes;
29
        double freq = getTickFrequency() / 1000;
30
        double t = net.getPerfProfile(layersTimes) / freq;
31
        string label = format("Inference time for a frame : %.2f ms", t);
32
        putText(frame, label, Point(0, 15), FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0, 0, 255));
33
34
        // Write the frame with the detection boxes
35
        Mat detectedFrame;
36
        frame.convertTo(detectedFrame, CV 8U);
37
        if (parser.has("image")) imwrite(outputFile, detectedFrame);
38
        else video.write(detectedFrame);
39 }
```

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)).

```
# 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()]
```

```
1 // Get the names of the output layers
    vector<String> getOutputsNames(const Net& net)
3
4
        static vector<String> names;
5
        if (names.empty())
6
7
            // Get the indices of the output layers,
8
           // i.e. the layers with unconnected outputs
            vector<int> outLayers = net.getUnconnectedOutLayers();
10
11
           //get the names of all the layers in the network
            vector<String> layersNames = net.getLayerNames();
12
13
14
           // Get the names of the output layers in names
15
           names.resize(outLayers.size());
16
            for (size_t i = 0; i < outLayers.size(); ++i)</pre>
17
           names[i] = layersNames[outLayers[i] - 1];
18
19
        return names;
20 }
```

#### 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.

```
1 | # Remove the bounding boxes with low confidence using nms
2
    def postprocess(frame, outs):
        frameHeight = frame.shape[0]
        frameWidth = frame.shape[1]
        classIds = []
        confidences = []
8
9
        # Scan through all the bounding boxes output from the network and
10
        # keep only the ones with high confidence scores.
11
        # Assign the box's class label as the class with the highest score.
12
        classIds = []
13
        confidences = []
14
        boxes = []
15
        for out in outs:
16
            for detection in out:
17
                scores = detection[5:]
18
                classId = np.argmax(scores)
                confidence = scores[classId]
19
20
                if confidence > confThreshold:
21
                    center_x = int(detection[0] * frameWidth)
22
                    center_y = int(detection[1] * frameHeight)
23
                    width = int(detection[2] * frameWidth)
24
                   height = int(detection[3] * frameHeight)
```

```
25
                    left = int(center x - width / 2)
26
                    top = int(center y - height / 2)
27
                    classIds.append(classId)
28
                    confidences.append(float(confidence))
29
                    boxes.append([left, top, width, height])
30
        # Perform nms to eliminate redundant overlapping boxes with
31
32
        # lower confidences.
33
        indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)
34
        for i in indices:
35
           i = i[0]
36
            box = boxes[i]
37
            left = box[0]
38
            top = box[1]
39
            width = box[2]
40
            height = box[3]
41
            drawPred(classIds[i], confidences[i], left, top, left + width, top + height)
```

#### C++:

```
1 // Remove the bounding boxes with low confidence using nms
    void postprocess(Mat& frame, const vector<Mat>& outs)
3
4
        vector<int> classIds;
5
        vector<float> confidences;
        vector<Rect> boxes;
8
        for (size_t i = 0; i < outs.size(); ++i)</pre>
9
10
            // Scan through all the bounding boxes output from the network
11
            // and keep only the ones with high confidence scores.
12
            // Assign the box's class label as the class
13
            // with the highest score for the box.
14
            float* data = (float*)outs[i].data;
15
            for (int j = 0; j < outs[i].rows; ++j, data += outs[i].cols)</pre>
16
17
                Mat scores = outs[i].row(j).colRange(5, outs[i].cols);
18
                Point classIdPoint;
19
                double confidence;
20
                // Get the value and location of the maximum score
21
                minMaxLoc(scores, 0, &confidence, 0, &classIdPoint);
22
                if (confidence > confThreshold)
23
24
                    int centerX = (int)(data[0] * frame.cols);
25
                    int centerY = (int)(data[1] * frame.rows);
26
                    int width = (int)(data[2] * frame.cols);
27
                    int height = (int)(data[3] * frame.rows);
28
                    int left = centerX - width / 2;
29
                    int top = centerY - height / 2;
30
31
                    classIds.push_back(classIdPoint.x);
32
                    confidences.push_back((float)confidence);
33
                    boxes.push_back(Rect(left, top, width, height));
34
35
36
37
38
        // Perform nms to eliminate redundant overlapping boxes with
39
        // lower confidences
40
        vector<int> indices;
```

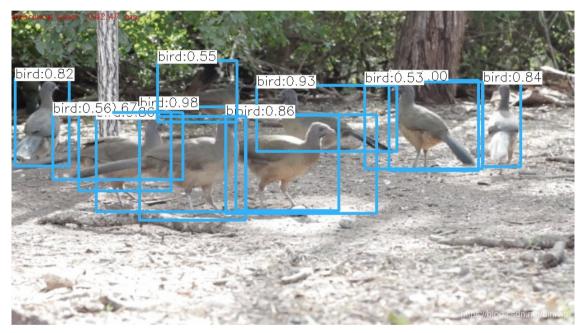
NMS 是由 nmsThreshold 参数控制的.

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.

```
# 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))
```

```
label = '%.2f' % conf
        # Get the label for the class name and its confidence
8
       if classes:
9
           assert(classId < len(classes))</pre>
10
           label = '%s:%s' % (classes[classId], label)
11
12
        #Display the label at the top of the bounding box
13
        labelSize, baseLine = cv.getTextSize(
14
           label, cv.FONT_HERSHEY_SIMPLEX, 0.5, 1)
15
        top = max(top, labelSize[1])
16
        cv.putText(frame, label, (left, top),
17
                  cv.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255))
```

### C++:

```
1 // Draw the predicted bounding box
    void drawPred(int classId, float conf, int left, int top, int right, int bottom, Mat& frame)
3
        //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
8
        string label = format("%.2f", conf);
9
        if (!classes.empty())
10
        {
11
            CV_Assert(classId < (int)classes.size());</pre>
12
            label = classes[classId] + ":" + label;
13
14
15
        //Display the label at the top of the bounding box
16
        int baseLine;
17
        Size labelSize = getTextSize(label, FONT_HERSHEY_SIMPLEX, 0.5, 1, &baseLine);
18
        top = max(top, labelSize.height);
19
        putText(frame, label, Point(left, top), FONT_HERSHEY_SIMPLEX, 0.5, Scalar(255,255,255));
20 }
```

### 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
```

```
# 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
```

```
10 | nmsThreshold = 0.4 #Non-maximum suppression threshold
11 inpWidth = 416
                     #Width of network's input image
12 inpHeight = 416 #Height of network's input image
14 parser = argparse.ArgumentParser(description='Object Detection using YOLO in OPENCV')
15 parser.add_argument('--image', help='Path to image file.')
16 parser.add argument('--video', help='Path to video file.')
17 args = parser.parse args()
19 # Load names of classes
20 classesFile = "coco.names";
21 classes = None
22 with open(classesFile, 'rt') as f:
       classes = f.read().rstrip('\n').split('\n')
25 # Give the configuration and weight files for the model and load the network using them.
26 modelConfiguration = "yolov3.cfg";
27 modelWeights = "yolov3.weights";
28
29  net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
30 net.setPreferableBackend(cv.dnn.DNN BACKEND OPENCV)
31 net.setPreferableTarget(cv.dnn.DNN TARGET CPU)
33 # Get the names of the output layers
34 def getOutputsNames(net):
35
       # Get the names of all the layers in the network
36
       layersNames = net.getLayerNames()
37
       # Get the names of the output layers,
38
       # i.e. the layers with unconnected outputs
39
        return [layersNames[i[0] - 1] for i in net.getUnconnectedOutLayers()]
40
41 # Draw the predicted bounding box
42 def drawPred(classId, conf, left, top, right, bottom):
43
       # Draw a bounding box.
44
        cv.rectangle(frame, (left, top), (right, bottom), (255, 178, 50), 3)
45
46
        label = '%.2f' % conf
47
48
        # Get the label for the class name and its confidence
49
        if classes:
50
            assert(classId < len(classes))</pre>
51
            label = '%s:%s' % (classes[classId], label)
52
53
        #Display the label at the top of the bounding box
54
        labelSize, baseLine = cv.getTextSize(label, cv.FONT HERSHEY SIMPLEX, 0.5, 1)
55
        top = max(top, labelSize[1])
56
        cv.rectangle(frame, (left, top - round(1.5*labelSize[1])), (left + round(1.5*labelSize[0]), top + baseLine),
57
        cv.putText(frame, label, (left, top), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,0), 1)
58
59 # Remove the bounding boxes with low confidence using nms
60 def postprocess(frame, outs):
61
        frameHeight = frame.shape[0]
62
        frameWidth = frame.shape[1]
63
64
        classIds = []
65
        confidences = []
66
        boxes = []
67
        # Scan through all the bounding boxes output from the network and
68
        # keep only the ones with high confidence scores.
        # Assign the box's class label as the class with the highest score.
        classIds = []
```

```
71
        confidences = []
72
        boxes = []
73
         for out in outs:
74
             for detection in out:
75
                 scores = detection[5:]
76
                classId = np.argmax(scores)
77
                confidence = scores[classId]
78
                if confidence > confThreshold:
79
                    center x = int(detection[0] * frameWidth)
80
                     center y = int(detection[1] * frameHeight)
81
                    width = int(detection[2] * frameWidth)
82
                    height = int(detection[3] * frameHeight)
83
                    left = int(center_x - width / 2)
84
                    top = int(center_y - height / 2)
85
                     classIds.append(classId)
 86
                    confidences.append(float(confidence))
87
                    boxes.append([left, top, width, height])
88
89
         # Perform nms to eliminate redundant overlapping boxes with
90
         # Lower confidences.
         indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)
91
92
         for i in indices:
93
            i = i[0]
94
             box = boxes[i]
95
            left = box[0]
96
            top = box[1]
97
             width = box[2]
98
             height = box[3]
99
             drawPred(classIds[i], confidences[i], left, top, left + width, top + height)
100
101
     # Process inputs
102 winName = 'Deep learning object detection in OpenCV'
    cv.namedWindow(winName, cv.WINDOW_NORMAL)
104
105 outputFile = "yolo_out_py.avi"
106 if (args.image):
107
        # Open the image file
108
        if not os.path.isfile(args.image):
109
             print("Input image file ", args.image, " doesn't exist")
110
             sys.exit(1)
        cap = cv.VideoCapture(args.image)
111
112
        outputFile = args.image[:-4]+'_yolo_out_py.jpg'
113 elif (args.video):
114
        # Open the video file
115
        if not os.path.isfile(args.video):
116
            print("Input video file ", args.video, " doesn't exist")
117
             sys.exit(1)
        cap = cv.VideoCapture(args.video)
118
119
        outputFile = args.video[:-4]+'_yolo_out_py.avi'
120 else:
121
        # Webcam input
122
        cap = cv.VideoCapture(0)
123
124
     # Get the video writer initialized to save the output video
125 if (not args.image):
126
        vid_writer = cv.VideoWriter(outputFile, cv.VideoWriter_fourcc('M','J','P','G'), 30, (round(cap.get(cv.CAP_PRC
127
128 while cv.waitKey(1) < 0:
129
130
         # get frame from the video
131
        hasFrame, frame = cap.read()
```

```
132
133
         # Stop the program if reached end of video
134
        if not hasFrame:
135
            print("Done processing !!!")
136
            print("Output file is stored as ", outputFile)
137
            cv.waitKey(3000)
138
            # Release device
139
             cap.release()
140
            break
141
142
        # Create a 4D blob from a frame.
143
        blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1, crop=False)
144
145
        # Sets the input to the network
146
        net.setInput(blob)
147
         # Runs the forward pass to get output of the output layers
148
         outs = net.forward(getOutputsNames(net))
149
         # Remove the bounding boxes with low confidence
150
        postprocess(frame, outs)
151
152
        # Put efficiency information.
153
        # The function getPerfProfile returns the overall time for inference(t)
154
         # and the timings for each of the layers(in layersTimes)
155
        t, = net.getPerfProfile()
        label = 'Inference time: %.2f ms' % (t * 1000.0 / cv.getTickFrequency())
156
157
        cv.putText(frame, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255))
158
159
        # Write the frame with the detection boxes
160
        if (args.image):
161
             cv.imwrite(outputFile, frame.astype(np.uint8));
162
        else:
163
             vid writer.write(frame.astype(np.uint8))
164
        cv.imshow(winName, frame)
```

### 4.7. YOLOV3 完整测试代码(C++)

```
object_detection_yolo.cpp
```

用法:

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

```
14 ;
15 using namespace cv;
16 using namespace dnn;
17 using namespace std;
19 // Initialize the parameters
20 | float confThreshold = 0.5; // Confidence threshold
21 | float nmsThreshold = 0.4; // Non-maximum suppression threshold
22 int inpWidth = 416; // Width of network's input image
23 int inpHeight = 416; // Height of network's input image
24 | vector<string> classes;
25
26 // Remove the bounding boxes with Low confidence using nms
27 void postprocess(Mat& frame, const vector<Mat>& out);
29 // Draw the predicted bounding box
30 | void drawPred(int classId, float conf, int left, int top, int right, int bottom, Mat& frame);
31
32 // Get the names of the output layers
33 vector<String> getOutputsNames(const Net& net);
34
35 int main(int argc, char** argv)
36 {
37
        CommandLineParser parser(argc, argv, keys);
38
        parser.about("Use this script to run object detection using YOLO3 in OpenCV.");
39
       if (parser.has("help"))
40
41
            parser.printMessage();
42
           return 0;
43
44
        // Load names of classes
45
        string classesFile = "coco.names";
46
        ifstream ifs(classesFile.c_str());
47
        string line;
48
        while (getline(ifs, line)) classes.push_back(line);
49
50
        // Give the configuration and weight files for the model
51
        String modelConfiguration = "yolov3.cfg";
52
        String modelWeights = "yolov3.weights";
53
54
        // Load the network
55
        Net net = readNetFromDarknet(modelConfiguration, modelWeights);
56
        net.setPreferableBackend(DNN_BACKEND_OPENCV);
57
        net.setPreferableTarget(DNN_TARGET_CPU);
58
59
        // Open a video file or an image file or a camera stream.
60
        string str, outputFile;
61
        VideoCapture cap;
62
        VideoWriter video;
63
        Mat frame, blob;
64
65
       try {
66
            outputFile = "yolo out cpp.avi";
67
            if (parser.has("image"))
68
69
               // Open the image file
70
               str = parser.get<String>("image");
71
               ifstream ifile(str);
72
               if (!ifile) throw("error");
73
                cap.open(str);
74
                str.replace(str.end()-4, str.end(), "_yolo_out_cpp.jpg");
```

```
75
                outputFile = str;
76
77
             else if (parser.has("video"))
78
79
                // Open the video file
80
                str = parser.get<String>("video");
81
                ifstream ifile(str);
82
                if (!ifile) throw("error");
83
                cap.open(str);
84
                str.replace(str.end()-4, str.end(), " yolo out cpp.avi");
85
                outputFile = str;
86
87
             // Open the webcaom
88
            else cap.open(parser.get<int>("device"));
89
90
        }
91
        catch(...) {
92
            cout << "Could not open the input image/video stream" << endl;</pre>
93
            return 0;
94
95
96
        // Get the video writer initialized to save the output video
97
        if (!parser.has("image")) {
98
             video.open(outputFile, VideoWriter::fourcc('M','J','P','G'), 28, Size(cap.get(CAP PROP FRAME WIDTH), cap.
99
100
101
        // Create a window
102
        static const string kWinName = "Deep learning object detection in OpenCV";
103
        namedWindow(kWinName, WINDOW NORMAL);
104
105
        // Process frames.
106
        while (waitKey(1) < 0)
107
108
            // get frame from the video
109
            cap >> frame;
110
111
            // Stop the program if reached end of video
112
            if (frame.empty()) {
113
                cout << "Done processing !!!" << endl;</pre>
114
                cout << "Output file is stored as " << outputFile << endl;</pre>
115
                waitKey(3000);
116
                break;
117
118
            // Create a 4D blob from a frame.
119
            blobFromImage(frame, blob, 1/255.0, cvSize(inpWidth, inpHeight), Scalar(0,0,0), true, false);
            //Sets the input to the network
120
121
             net.setInput(blob);
122
            // Runs the forward pass to get output of the output layers
123
             vector<Mat> outs;
124
             net.forward(outs, getOutputsNames(net));
125
126
             // Remove the bounding boxes with low confidence
127
             postprocess(frame, outs);
128
129
            // Put efficiency information.
130
             // The function getPerfProfile returns the overall time for inference(t)
131
            // and the timings for each of the Layers(in LayersTimes)
132
             vector<double> layersTimes;
133
             double freq = getTickFrequency() / 1000;
134
             double t = net.getPerfProfile(layersTimes) / freq;
135
             string label = format("Inference time for a frame : %.2f ms", t);
```

```
136
             putText(frame, label, Point(0, 15), FONT HERSHEY SIMPLEX, 0.5, Scalar(0, 0, 255));
137
138
             // Write the frame with the detection boxes
139
             Mat detectedFrame:
140
             frame.convertTo(detectedFrame, CV 8U);
             if (parser.has("image")) imwrite(outputFile, detectedFrame);
141
142
             else video.write(detectedFrame);
143
144
             imshow(kWinName, frame);
145
146
         cap.release();
147
        if (!parser.has("image")) video.release();
148
149
        return 0;
150 }
151
152
      // Remove the bounding boxes with low confidence using nms
    void postprocess(Mat& frame, const vector<Mat>& outs)
153
154
155
         vector<int> classIds;
156
         vector<float> confidences;
157
         vector<Rect> boxes;
158
159
         for (size t i = 0; i < outs.size(); ++i)</pre>
160
161
             // Scan through all the bounding boxes output from the network and
162
             // keep only the ones with high confidence scores.
163
             // Assign the box's class label as the class
164
             // with the highest score for the box.
165
             float* data = (float*)outs[i].data;
166
             for (int j = 0; j < outs[i].rows; ++j, data += outs[i].cols)</pre>
167
168
                 Mat scores = outs[i].row(j).colRange(5, outs[i].cols);
169
                 Point classIdPoint;
170
                 double confidence;
171
                 // Get the value and location of the maximum score
172
                 minMaxLoc(scores, 0, &confidence, 0, &classIdPoint);
173
                 if (confidence > confThreshold)
174
175
                     int centerX = (int)(data[0] * frame.cols);
                     int centerY = (int)(data[1] * frame.rows);
176
177
                     int width = (int)(data[2] * frame.cols);
178
                     int height = (int)(data[3] * frame.rows);
179
                     int left = centerX - width / 2;
180
                     int top = centerY - height / 2;
181
182
                     classIds.push_back(classIdPoint.x);
183
                     confidences.push_back((float)confidence);
184
                     boxes.push_back(Rect(left, top, width, height));
185
186
187
188
189
         // Perform nms to eliminate redundant overlapping boxes with
190
         // lower confidences
191
         vector<int> indices;
192
         NMSBoxes(boxes, confidences, confThreshold, nmsThreshold, indices);
193
         for (size_t i = 0; i < indices.size(); ++i)</pre>
194
195
             int idx = indices[i];
196
             Rect box = boxes[idx];
```

```
197
             drawPred(classIds[idx], confidences[idx], box.x, box.y,
198
                     box.x + box.width, box.y + box.height, frame);
199
200 }
201
202 // Draw the predicted bounding box
203 void drawPred(int classId, float conf, int left, int top, int right, int bottom, Mat& frame)
204
205
        //Draw a rectangle displaying the bounding box
206
        rectangle(frame, Point(left, top), Point(right, bottom), Scalar(255, 178, 50), 3);
207
208
        //Get the label for the class name and its confidence
209
         string label = format("%.2f", conf);
210
        if (!classes.empty())
211
        {
212
            CV Assert(classId < (int)classes.size());</pre>
213
            label = classes[classId] + ":" + label;
214
215
216
        //Display the label at the top of the bounding box
217
        int baseLine:
218
         Size labelSize = getTextSize(label, FONT HERSHEY SIMPLEX, 0.5, 1, &baseLine);
219
        top = max(top, labelSize.height);
220
        rectangle(frame, Point(left, top - round(1.5*labelSize.height)), Point(left + round(1.5*labelSize.width), top
221
        putText(frame, label, Point(left, top), FONT HERSHEY SIMPLEX, 0.75, Scalar(0,0,0),1);
222 }
223
224
    // Get the names of the output layers
    vector<String> getOutputsNames(const Net& net)
226
227
        static vector<String> names;
228
        if (names.empty())
229
230
            // Get the indices of the output layers,
231
            // i.e. the layers with unconnected outputs
            vector<int> outLayers = net.getUnconnectedOutLayers();
232
233
234
            //get the names of all the layers in the network
235
            vector<String> layersNames = net.getLayerNames();
236
237
            // Get the names of the output layers in names
238
            names.resize(outLayers.size());
239
            for (size_t i = 0; i < outLayers.size(); ++i)</pre>
240
            names[i] = layersNames[outLayers[i] - 1];
241
242
        return names;
243 }
```

### 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 ① 2999

04-07

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....

# opency摄像头实时拍摄显示 data-life www pp

等级

492

收藏

码龄18年 ☑ 暂无认证

1万+

88

获赞

周排名 总排名

28万+

访问

39

评论

40

原创

2640

积分

1万+

39

粉丝

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 

 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...)

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 ① 27443

Python如何释放内存 ① 11583

pd.read csv中的usecols函数实现读取指定 列 ① 9561

在sql语句查询时如何把查询为空的数据显 示为零 ① 7069

python每天固定时间运行某程序代码 ①

### 最新评论

Failed to restart network.service: Unit n... 小五809: 恕我初入hadoop不是很懂后面大 X指代什么,都试过了全是"未知的连接"或

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...

各位同学好,今天和大家分享一下如何使用 opency 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)

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 ① 2万+

qq\_45445740的博客 ① 3803

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...

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

这里主要是对基于 YOLOV3 和 OpenCV的目标检测(PythonC++))译IPython 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

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)

vishuihang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 ① 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 1万+

blobFromImage() Mat cv::dnn::blobFromImage (InputArray image, double scalefactor = 1.0, const Size & Discount Size = Size(), const Scalar & Discount Size & Di

OpenCV笔记5: 使用OpenCV Python在视频上显示日期和时间

weixin 41788560的博客 **②** 2770

1、学习目标 (1) 在python中使用opency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...

opency-python+yoloy3实现目标检测

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

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 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 ② 2519

最近在用yolo做人脸识别<mark>的</mark>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

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 ① 381

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 ① 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<mark>的</mark>一些所学。现在课程终于结束了,即将开始下一阶段<mark>的</mark>项目开发,所以趁这个间隙...

基于opency的车牌识别解析与代码 热门推荐

LinJM-机器视觉 ① 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opency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?







©2022 CSDN 皮肤主题: 大白 设计师: CSDN官方博客 返回首页

关于我们 招贤纳士 商务合作 寻求报道 ☎ 400-660-0108 ☑ kefu@csdn.net ⑤ 在线客服 工作时间 8:30-22:00











