在 OpenVINO™ 精度检查器工具中添加 DIoU-NMS 指标，以获取 YOLO v4 的正确 mAP

白皮书

2021 年 1 月

文档编号：634498-0.8

目录

修订记录

日期	修订版	说明
2021 年 1 月	0.8	初始版本。

§

1.0 简介

Yolov4 模型于 2020 年中期推出，它的问世对深度学习目标检测领域产生了深远影响。为取得成功，Yolov4 集成了许多一流的技术，其中包括改进的非极大值抑制 (NMS) 算法，该算法使用距离交并比 (DIoU) 而非交并比 (IoU)。DIoU 总结了边界框回归中的两个几何因素（即重叠区域和中心点距离），从而加快了收敛速度并提升了性能。

在本白皮书中，DIoU-NMS 函数将被添加到 OpenVINO™ 精度检查器工具中，以用于计算 yolov4 的正确预期精度。尽管 DIoU-NMS 可帮助大幅提升精度，但它也可以与训练后优化工具包 (pot) 一起使用，以生成优化的 INT8 模型。

§

2.0 设置环境

- 系统环境
  - 设置 OpenVINO™ 2021.2
- Ubuntu 18.04 或 Ubuntu 20.04。

§

3.0 添加 Diou-nms 的步骤

3.1. 安装精度检查器工具

按照 openvinotookit_open_model_zoo 指南进行安装。

```
1. cd /opt/intel/openvino_2021/deployment_tools/open_model_zoo/to
   ols/accuracy_checker
2. sudo apt-get install python3 python3-dev python3- setuptools python3-pip

3. python3 setup.py install
```

3.2. 编辑 “/opt/intel/openvino_2021/deployment_tools/open_model_zoo/tools/accuracy_checker/accuracy_checker/postprocessor/nms.py”

将以下代码添加到文件中：

```
1.  #add DIOU-NMS support
2.  class DIOU_NMS(Postprocessor):
3.      __provider  = 'diou_nms'
4.
5.      prediction_types = (DetectionPrediction, ActionDetectionPr
    ediction)
6.      annotation_types = (DetectionAnnotation, ActionDetectionPr
    ediction)
7.
8.      @classmethod
9.      def parameters(cls):
10.             parameters = super().parameters()
11.             parameters.update({
12.                 'overlap':
13.                     NumberField( min_value=0, optional=True, de
    fault=0.5,
14.                     description="Overlap threshold for merging
    detections."
15.                 ),
16.                 'include_boundaries': BoolField( optional=True,
17.                     default=True, description="S
    hows if boundaries are included."
18.                 ),
19.                 'keep_top_k': NumberField(min_value=0, optional
    =True, description="Keep top K."),
20.                 'use_min_area': BoolField(
21.                     optional=True, default=False,
```

```
                    description="Use minimum area of two boundi
ng boxes as base area to calculate overlap"
                )
            })
            return parameters

        def configure(self):
            self.overlap = self.get_value_from_config('overlap'
)
            self.include_boundaries = self.get_value_from_confi
g('include_boundaries')
            self.keep_top_k = self.get_value_from_config('keep_
top_k')
            self.use_min_area = self.get_value_from_config('use
_min_area')

        def process_image(self, annotations, predictions):
            for prediction in predictions:
                scores = get_scores(prediction)
                keep = self.diou_nms(
                    prediction.x_mins, prediction.y_mins, predi
ction.x_maxs, prediction.y_maxs, scores,
                    self.overlap, self.include_boundaries, self
.keep_top_k, self.use_min_area
                )
                prediction.remove([box for box in range(len(pre
diction.x_mins)) if box not in keep])

            return annotations, predictions

        @staticmethod
        def diou_nms(x1, y1, x2, y2, scores, thresh, include_bo
undaries=True, keep_top_k=None, use_min_area=False):
            """
            Pure Python NMS baseline.
            """
            b = 1 if include_boundaries else 0

            areas = (x2 - x1 + b) * (y2 - y1 + b)
            order = scores.argsort()[::-1]

            if keep_top_k:
                order = order[:keep_top_k]

            keep = []

            while order.size > 0:
                i = order[0]
                keep.append(i)

                xx1 = np.maximum(x1[i], x1[order[1:]])
                yy1 = np.maximum(y1[i], y1[order[1:]])
                xx2 = np.minimum(x2[i], x2[order[1:]])
```

```
                yy2 = np.minimum(y2[i], y2[order[1:]])

                w = np.maximum(0.0, xx2 - xx1 + b)
                h = np.maximum(0.0, yy2 - yy1 + b)
                intersection = w * h

                cw = np.maximum(x2[i], x2[order[1:]]) - np.minimum(x1[i], x1[order[1:]])
                ch = np.maximum(y2[i], y2[order[1:]]) - np.minimum(y1[i], y1[order[1:]])
                c_area = cw**2+ch**2+1e-16
                rh02 = ((x2[order[1:]]+x1[order[1:]])-(x2[i]+x1[i]))**2/4+((y2[order[1:]]+y1[order[1:]])-(y2[i]+y1[i]))**2/4

                if use_min_area:
                    base_area = np.minimum(areas[i], areas[order[1:]])
                else:
                    base_area = (areas[i] + areas[order[1:]] - intersection)

                overlap =
                    np.divide( intersectio
                    n, base_area,
                    out=np.zeros_like(intersection, dtype=float),
                    where=base_area != 0
                ) - pow(rh02/c_area,0.6)
                order = order[np.where(overlap <= thresh)[0] + 1] # pylint: disable=W0143

            return keep
```

§

4.0 用 DIoU-NMS 检查 yolo-v4-tf 模型的精度

4.1. 下载和转换 yolo-v4-tf 模型

```
1. cd ~/Documents/
2. python3 /opt/intel/openvino_2021/deployment_tools/tools/model_
   downloader/downloader.py --name yolo-v4*
3. python3 /opt/intel/openvino_2021/deployment_tools/tools/model_
   downloader/converter.py --name yolo-v4*
```

4.2. 准备数据集和 yml 文件

4.2.1. 下载数据集

```
1. wget http://images.cocodataset.org/zips/val2017.zip
2. wget http://images.cocodataset.org/annotations/annotations_tra inval2017.zip

3. unzip -
   d annotations_trainval2017/ annotations_trainval2017.zip
4. unzip -d annotations_trainval2017/annotations/ val2017.zip
```

4.2.2. 复制和编辑“accuracy-check.yml”

复制“accuracy-check.yml”

```
1. sudo mv /opt/intel/openvino_2021/deployment_tools/open_mode
   l_zoo/models/public/yolo-v4-tf/accuracy-check.yml ~/Documents/
```

编辑“accuracy-check.yml”

1. 标注出第 2-60 行
2. 在第 65 行添加以下内容

```
1. - framework: dlsdk
2.       tags:
3.           - FP32
4.       model: yolo-v4-tf.xml
5.       weights: yolo-v4-tf.bin
```

3. 从以下位置开始编辑第 94 行

   ~~- type: nms~~

   一直到

```
    - type: diou_nms
```

4.3. 运行精度检查器工具，计算 yolo-v4-tf 的统计精度

如果上述设置和修改正确，请使用以下命令获取精度结果。

```
1. accuracy_check -c accuracy-check.yml -m public/yolo-v4- tf/FP32/ --
   definitions /opt/intel/openvino_2021/deployment_tools/open_mod
   el_zoo/tools/accuracy_checker/dataset_definitions.yml - s
   annotations_trainval2017/annotations/ -td CPU
```

mAP 结果：

yolo-v4-tf	DIoU-NMS	IoU-NMS
	71.23%	71.17%

结果证实 DIoU 的 mAP 比 IoU 更准确。尽管使用训练后优化工具包 (pot) 将模型转换为 INT8 时只有微小的差异，但是这些差异可能会影响转换后的 INT8 模型的性能。因此，实现面向 OpenVINO 工具™（精度检查器和 pot）的 DIoU-NMS 函数非常重要。

§

5.0 参考资料

参考文档	文档编号/位置
OpenVINO™	*https://docs.openvinotoolkit.org/*
OpenVINO Open Model Zoo	https://github.com/openvinotoolkit/open_model_zoo

§