计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(2):176−184 [doi: 10.15888/j.cnki.csa.008331]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 MobileNet 与 YOLOv3 的路面障碍检测轻量化算法①

齐永康

(华南师范大学 计算机学院, 广州 510631)
通信作者: 齐永康, E-mail: qyk59914@163.com

摘　要: 为了避免人们边行走边使用手机发生危险, 本文提出了实时性强的轻量级模型 (Mobile-YOLOv3) 来检测路面障碍. 我们在广州各地拍摄路障图片并标注了一个路障数据集, 使用了一个轻量级的 MobileNetv1 网络来替换 YOLOv3 的骨干网络实现轻量化, 并且应用了 4 个方法用于提高检测精度和模型的鲁棒型. 4 个方法分别为: 边框回归损失函数 CIOU、分类损失函数 Focal、预测框筛选算法 Soft-NMS、负样本训练. 实验结果证明, 该模型获得了 98.84% 的 MAP. 与 YOLOv3 对比, 该模型的规模缩减了 2.5 倍, 检测精度却提高了 7%.

关键词: 深度学习; 目标检测; 路障检测; YOLOv3; 轻量化模型

引用格式: 齐永康.基于 MobileNet 与 YOLOv3 的路面障碍检测轻量化算法.计算机系统应用,2022,31(2):176–184. http://www.c-s-a.org.cn/1003-3254/8331.html

Lightweight Algorithm of Road Obstacle Detection Based on MobileNet and YOLOv3

QI Yong-Kang

(School of Computer Science, South China Normal University, Guangzhou 510631, China)

Abstract: To avoid the danger of people using mobile phones while walking, this study proposes a lightweight model (Mobile-YOLOv3) with strong real-time performance to detect road obstacles. We photograph roadblocks and annotate a roadblock data set around Guangzhou City. Lightweight is achieved by the replacement of the backbone network of YOLOv3 with a lightweight MobileNetv1 network. In addition, we apply four methods to improve detection accuracy and model robustness, i.e., border regression loss function CIOU, classification loss function Focal, prediction box screening algorithm Soft-NMS, and negative sample training. The experimental results show that the model obtains 98.84% MAP. Compared with YOLOv3, this model has the scale reduced by 2.5 times but the detection accuracy improved by 7%.

Key words: deep learning; object detection; road obstacle detection; YOLOv3; lightweight model

1　引言

随着移动网络和智能设备的不断发展, 越来越多的年轻人喜欢边行走边玩手机. 这导致了他们的注意力分散, 进而遇到路面障碍发生危险[1].

因此, 我们设计了一个希望能在智能手机上应用的基于深度学习方法的路面障碍实时检测模型, 用来及时提醒行人在行走时遇到的障碍, 这样可以有效减少边走边使用手机带来的危险. 其中智能手机能够实时识别路面障碍的本质要求是在保证检测路障图像高精度值的前提下, 检测速度也要足够快. 对比传统的移动端路障检测方法, 本文提出的基于深度学习的 Mobile-YOLOv3 模型在提高了精度值的前提下, 也能达到实时检测的最低要求[2].

传统的目标检测方法和近年来使用深度学习来进

① 收稿时间: 2021-04-19; 修改时间: 2021-05-19; 采用时间: 2021-06-07; csa 在线出版时间: 2022-01-17

行目标检测的方法大相径庭. 传统的目标检测方法典型代表有 AdaBoost 和 Cascaded 人脸检测算法[3], HOG 和 SVM 行人检测算法[4], DPM 检测算法[5]. 传统的目标检测方法大致可分为 3 个步骤, 包括选定目标区域、特征提取和特征分类. 传统方法先将目标图像区域像素遍历一遍, 然后使用人工设计好的特征提取器来分析这些特征区域提取特征, 最后利用类别分类器对这些目标特征区域分类, 已获得预期的分类结果. 传统的目标检测方法存在较多的局限性, 例如像素窗口冗余, 以及检测结果的不稳定性. 随着复杂神经网络和深度学习的出现, 这些局限性得到了一定程度上的解决. 因此, 以深度学习为基础的目标检测技术逐渐代替了传统的目标检测方式. 以深度学习为基础的目标检测网络分为两种, one-stage 网络和 two-stage 网络. One-stage 网络以 SSD[6] , YOLO[7] 系列算法为代表, two-stage 网络常见的有 RCNN[8], Faster-RCNN[9], R-FCN[10]. Two-stage 类算法输入图像后首先会生成目标对象候选区域, 然后会将生成的目标对象候选区域输入到卷积神经网络进行分类和回归. 而 one-stage 类算法输入图像后会直接利用卷积神经网络提取特征进行分类和回归. 两种算法的流程不同导致的结果就是 one-stage 类网络的检测速度要比 two-stage 类网络快很多, 而检测的准确性要比 two-stage 类网络低.

One-stage 类网络 YOLOv3 提取特征的骨干网络 Darknet53 是为在 VOC 等目标种类多的大型数据集上训练而设计的深层卷积神经网络, 而本文应用的路面障碍检测场景目标种类并没有那么复杂, 浅层特征提取网络 MobileNetv1 已经足够适用, 使用 YOLOv3 原始的骨干网络 Darknet53 训练反而增加了参数优化空间的复杂度, 降低了训练效果. 所以在本文中, 我们基于 one-stage 网络 YOLOv3 网络和 MobileNetv1 网络提出了一个精确率较高且具有一定实时性的轻量级道路障碍检测模型:

1) 拍摄制作了所需要的路面障碍数据集, 因为路面障碍种类多样化, 且需要使用特定的手持角度拍摄, 因此无法通过网络爬虫进行搜集, 需要在现实生活中实地拍摄. 数据集图片由作者在广州市各街道, 公园等地拍摄, 路面障碍数据集分类有井盖, 减速带, 楼梯台阶 3 类.

2) 将 YOLOv3 的骨干网络 Darknet53 替换为了 MobileNetv1, 提出了轻量级网络 Mobile-YOLOv3.

3) 使用了 4 个改进训练速度和提高训练精度的方法. 首先是训练时的改进, 通过对 IOU 局限性的分析, 利用回归损失函数 CIOU 更全面地学习了真实框与预测框之间的位置及形状关系, 提升了模型的训练深度, 速度及检测精度. 之后使用类别损失函数 Focal 替换了交叉熵损失函数使得训练更关注于难分类样本, 并在一定程度上解决了正负样本不均衡的问题. 最后, 通过加入不包含目标路障的负样本背景图片进行训练, 进一步提升了模型的精度. 而对于检测时的改进, 利用 Soft-NMS 算法替换 NMS 算法, 更加科学合理地处理冗余预测框, 减少了对密集目标的漏测误删情况.

本文实地拍摄制作了一个路障数据集, 将 YOLOv3 骨干网络替换为 MobileNetv1, 设计和实现了一个道路障碍检测模型 Mobile-YOLOv3, 并使用了 4 个改进方法来提升模型的检测性能. 最终模型在路面障碍数据集上取得了 98.84% 的检测精度, 相比于 YOLO3 在路面障碍数据集上所取得的 91.98% 的检测精度, 提升了约 7%. 而且模型规模较 YOLOv3 训练出来的模型减小了约 2.5 倍, 使用 CPU 进行检测的速度提升了 1.8 倍, 使用 GPU 进行检测的速度提升了 1.3 倍.

2 相关工作

2.1 YOLOv3

YOLO (you only look once), 是 Redmon 等于 2016 年提出的一种 one-stage 目标检测算法[7]. 在 YOLO 算法出现之前, 目标检测算法通常都是将检测问题转化为分类问题, 而 YOLO 算法将目标检测问题转化成为了回归问题, 用一个卷积神经网络就可以直接从输入图像上预测目标对象所在位置和概率, 实现了从端到端 (end-to-end) 的目标检测算法. 本文使用的网络以 YOLO 系列第 3 代算法 YOLOv3 为基础构建, YOLOv3 在 YOLOv1 和 YOLOv2 的基础上主要改进了骨干网络结构, 损失函数的计算方式, 以及使用了多尺度预测融合特征的检测方法[11].

2.2 轻量级网络 MobileNet

为了让神经网络模型在小型移动和嵌入式设备上快速实时地运行, Google 公司减小了神经网络模型规模并设计出了模型 MobileNetv1. 它在保证了模型的精确性和检测速度的基础上, 极大地减小了模型的规模和计算量[12]. MobileNetv1 是专门为移动和嵌入式设备提出的高效轻量级模型, 它基于流线型架构, 使用深度可分离卷积来替代普通的卷积方式来构建轻量级神经

网络. 深度可分离卷积方式和传统的卷积方式不同的地方在于它将传统的卷积方式分为了深度卷积和点卷积, 通过这种方式能够在一定程度上达到和传统的卷积方式一样的卷积效果, 同时, 参数量会大幅减少.

3 改进的网络模型

本节从 YOLOv3 特征提取网络骨干网络的替换, CIOU 和 Focal 损失函数, 预测框筛选算法 Soft-NMS 和负样本训练 4 个方面来详细地阐述改进的网络模型.

3.1 骨干网络结构的优化

本文最终的目的是设计出一个能够落地于便携设备上的轻量级路面障碍检测算法. YOLOv3 虽然是目前最常用的 one-stage 目标检测算法之一, 在很多场景下也能够精确的检测出目标物体, 但是其网络结构过于庞大, 计算参数也十分巨大, 最终导致训练出来的网络模型规模太大, 不适用于在运算能力和内存容量不够充足的移动设备上运行.

MobileNetv1 是一个轻量级神经网络, 它基于一个可深度分离的卷积方法, 将标准卷积分解为深度卷积和卷积核大小为 1×1 的点卷积. 深度卷积将每个卷积应用于输入图像的每个通道, 而点卷积用于组合通道卷积输出[12]. 这种深度可分离卷积方法可以有效地减少计算量, 减少模型的规模.

因此, 为了结合 YOLOv3 的目标检测结构和 MobileNetv1 网络轻量化的特点, 我们将 YOLOv3 的骨干网络 Darknet53 替换为 MobileNetv1. 这样, 新的目标检测网络的参数量和运算量都将大大降低, 使得该网络可以在移动设备上运行. 本文改进后的网络结构 Mobile-YOLOv3 如图 1 所示. 其中, $y1$, $y2$, $y3$ 为 3 个不同图像尺寸的输出矩阵. DBL 块是 YOLOv3 的基本组件, 由标准卷积层 (Conv)、批标准化层 (Batch-normalization)、激活函数 Leaky_ReLU 层组成. Concat 是向量拼接操作, 用于将 MobileNetv1 的中间层和深层网络的上采样特征向量进行拼接. MPWBR 块是 MobileNetv1 的基本组件之一, 由卷积核大小为 1×1 的点卷积层、批标准化层、激活函数 ReLU 层组成. MDWBR/S1 块是 MobileNetv1 的基本组件之一, 由步长为 1 的深度卷积层、批标准化层、ReLU 层组成. MDWBR/S2 块是 MobileNetv1 的基本组件之一, 由步长为 2 的深度卷积层、批标准化层、ReLU 层组成, 用于缩放图像尺寸. MBR/S2 块由步长为 2 的标准卷积层、批标准化层、ReLU 层组成. MDWS/S2 块由零填充层 (zeropadding)、MDWBR/S2 块、MPWBR 块组成. MDWS/S1 块由 MDWBR/S1块、MPWBR 块组成.

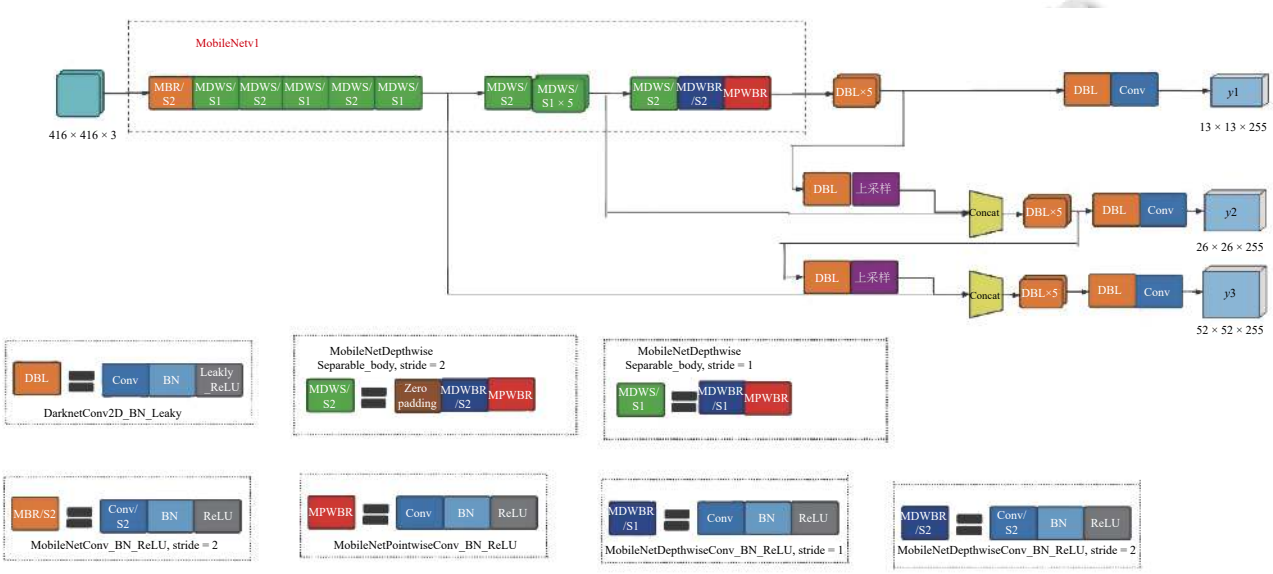


图 1 Mobile-YOLOv3 网络结构

3.2 回归损失函数 CIOU

YOLOv3 算法中使用交并比损失函数 *IOU_Loss* (intersection over union) 作为边框回归损失函数, *IOU* 是为了评价预测框对目标物体的预测效果而定义的一

个指标. 假设预测框为 A, 真实框为 B, 则对于图 2 所示的预测框 A 和 B 的 IOU_Loss 运算公式如式 (1) 所示:

$$IOU\_Loss = 1 - IOU = 1 - \frac{A \cap B}{A \cup B} \tag{1}$$

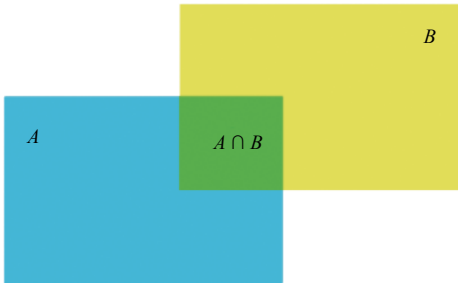


图 2 交并比 IOU

IOU 有两个缺陷, 一是当预测框和真实框不相交时, IOU 的值等于 0, 无论两个框之间的位置如何, 都不可能知道预测框和真实框之间的相对位置关系. 第二点是, 当真实框包含预测框时, 若不同的预测框大小相同, 但此时 IOU 值相等, IOU 值无法反映不同的预测框和真实框之间相对位置的不同. 针对 IOU 存在的问题, 我们使用 $CIOU$ (complete intersection over union) 函数替换了 IOU 作为本文模型的回归损失函数.

欧氏距离交并比 $DIOU$ (distance intersection over union) 被提出用来解决 IOU 存在的两个问题[13], 如图 3, 图 4 所示. $DIOU$ 添加了一个同时包含真实框和预测框的最小外接矩形 C, 并且 $DIOU$ 还考虑了两个框的重叠面积和中心点距离, $DIOU\_Loss$ 的运算公式如式 (2) 所示, $DIOU$ 引入了一个惩罚项, 用来最小化真实框和预测框之间的距离. 这样无论是当预测框和真实框不相交时, 还是真实框包含预测框时, $DIOU$ 的值都可以反映出预测框和真实框之间的相对位置关系.

$$\begin{aligned} DIOU\_Loss &= 1 - DIOU \\ &= 1 - \left(IOU - \frac{(Distance\_2)^2}{(Distance\_C)^2} \right) \end{aligned} \tag{2}$$

虽然上述方法已经解决了 IOU 存在的两个问题, 但是当真实框包含预测框时, 如图 3, 图 5 所示, 若预测框的长宽大小不一定但面积和中心点位置不变时, 也无法确定不同的预测框与真实框的相对位置. 针对这个问题, 完整交并比 $CIOU$ 和 $DIOU$ 被同步提出用来解决这个问题[13], $CIOU$ 在保留了 $DIOU$ 惩罚项的前提下, 选择添加了一个反映预测框与真实框的长宽比关系的参数项来进一步解决这个问题, $CIOU\_Loss$ 的运算公式如式 (3)–式 (5) 所示.

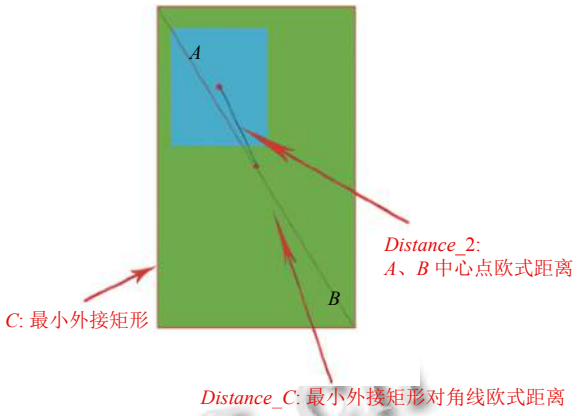


图 3 真实框包含预测框 $DIOU$ 图

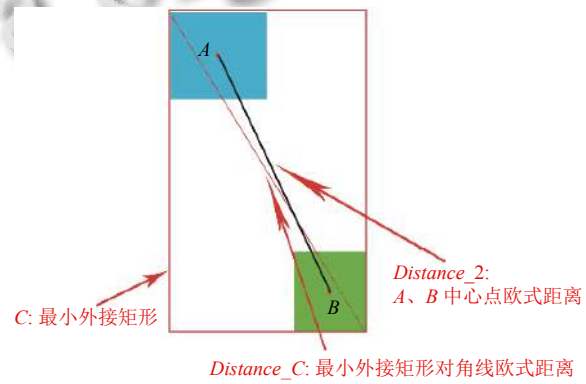


图 4 真实框与预测框不相交 $DIOU$ 图

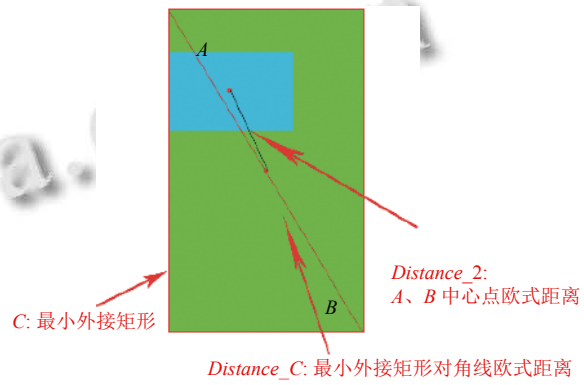


图 5 预测框被真实框包含时长宽比不一致 $DIOU$ 图

$$\begin{aligned} CIOU\_Loss &= 1 - CIOU \\ &= 1 - (IOU - \frac{(Distance\_2)^2}{(Distance\_C)^2} - \alpha v) \end{aligned} \tag{3}$$

$$\alpha = \frac{v}{1 - IOU + v} \tag{4}$$

$$v = \frac{4}{\pi} \left(\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w^{p}}{h^{p}} \right)^2 \tag{5}$$

其中, a是用来平衡比例的参数, v 用来衡量预测框和真实框之间长宽比例一致性的参数, w^{gt}, h^{gt} 是真实框的长度和宽度, w^{p}, h^{p} 是预测框的长度和宽度.

3.3 分类损失函数 Focal

在 YOLOv3 算法中, 使用的是二分类交叉熵 (binary cross entropy) 作为模型的分类损失函数. 然而, 交叉熵损失函数并没有对模型训练时的难易样本进行区分, 这导致了模型的训练效果不太理想. 因此为了解决这个问题, 我们使用 Focal 损失函数替代了 YOLOv3 中的二分类交叉熵函数作为本文模型的分类损失函数.

在二分类问题中, y 表示目标的类别标签值, 取值为{0, 1}, 0 为负样本, 1 为正样本. y'是输入样本经过激活函数后预测属于正样本的概率, 取值在 0–1 之间. 二分类交叉熵损失函数如式 (6) 所示. 若目标为正样本, 则当预测概率 y'越接近 1 时越容易分类, 损失值越小, 若目标为负样本, 则预测概率 y'越接近 0 时越容易分类, 损失值越小, 说明在二分类交叉熵损失函数中易分类样本对损失函数的数值影响不大. 而在实际训练过程中, 训练样本主要由大量易分类样本组成, 若采用交叉熵损失函数进行训练, 会导致损失函数在训练过程中迭代较慢, 且可能无法得到最优的结果.

$$Loss_{CE}=\begin{cases}-\ln y', & y=1\\ -\ln(1-y'), & y=0\end{cases} \tag{6}$$

针对这个问题, Focal 损失函数[14] 在交叉熵损失函数的基础上引入了一个权重系数, 通过控制易分类样本和难分类样本的权重, 使得模型在训练时更加专注于困难样本. Focal 损失函数如式 (7) 所示:

$$Loss_{FL}=\begin{cases}-\alpha(1-y')^{\gamma}\ln y', & y=1\\ -(1-\alpha)y'^{\gamma}\ln(1-y'), & y=0\end{cases} \tag{7}$$

Focal 损失函数首先在二分类交叉熵的基础上加了一个因子 γ, 其中 γ>0. 若目标为正样本, 则当预测概率 y'越接近 1 越容易分类时, $(1-y')^{\gamma}$ 的值会逐渐减小接近 0, Focal 损失函数的值相比二分类交叉熵损失也会大量减少. 同样若目标为负样本, 则当预测概率 y'越接近 0 越容易分类时, $(y')^{\gamma}$ 的值也会逐渐减小接近 0, Focal 损失函数的值相比于二分类交叉熵损失也会大量减少. 这样 Focal 损失函数通过添加了一个干扰因子 γ, 使得模型可以降低易分类样本的权重, 更加关注于难以分类的样本, 此外还加入了平衡因子a, 用来平衡数据集中正负样本的比例. 在本文经过对比实验得出当a=0.75, γ=2 时实验效果最优.

3.4 软化非极大值限制 Soft-NMS

YOLOv3 算法中使用非最大值抑制 NMS (non-maximum suppression) 来删除检测时的冗余框, NMS 的本质是寻找局部最大值, 去除非极大值元素, 找到最接近真实框的预测框位置, NMS 流程如算法 1 所示. NMS 算法先对所有可能包含了同一个检测对象的预测框 B 的检测得分 S 进行排序, 然后将得分最高的预测框作为目标真实框 M, 并计算该框和其他预测框的 IOU 值, 如果该值高于预先设置的阈值 N_t, 则删除此预测框并重复此过程, 直到所有预测框都被选中或被删除. NMS 处理结果如图 6 所示.

图 6 NMS 处理结果

根据 NMS 算法设计, 如果两个目标对象都处于预先设置的重叠阈值中, 则 NMS 算法可能会无法检测到所有的目标对象. 当两个目标对象的预测框靠的太近时, 如果重叠区域的 IOU 值高于预先设置的阈值, 那么检测得分低的目标对象预测框就会被删除, 这样就导致了一些目标对象的预测框被误删从而无法被检测出来. 针对误检的问题, 我们使用 Soft-NMS 函数替代了 NMS 作为本文模型的目标框筛选算法.

为了解决 NMS 会误删距离太近, 重叠区域过大的目标物体预测框的问题, Bodla 等人提出了软化非极大值限制 Soft-NMS[15]. Soft-NMS 处理流程如算法 2, 对于初始预测框列表 B 中的某一预测框 b_i 也是先计算其和检测得分最高框 M 的 IOU 值 , 但 Soft-NMS 不是像 NMS 一样当 IOU 值高于预设阈值 N_t, 在预测框列表 B 和检测得分列表 S 中直接暴力删除该预测框 b_i 及其得分 s_i. 而是将该预测框 b_i 和目标真实框 M 的 IOU 值和预测框 b_i 的检测得分 s_i 相乘重新作为该预测框 b_i 的检测得分进行检测. 通过这种方式可以降低改该预测框 b_i 的检测得分, 不会像 NMS 中直接被删除.

算法 1. NMS 处理步骤

Input:
$B = \{b_1, \cdots, b_n\}, S = \{s_1, \cdots, s_n\}, N_t$
B 是初始预测框的表
S 列表包含预测框列表 B 对应的检测得分
N_t 是 NMS 预设阈值
begin
$D \leftarrow \{\}$
While $B \neq empty$ do
$m \leftarrow$ argmax S
$M \leftarrow b_m$
$D \leftarrow D \cup M; B \leftarrow B - M$
for b_i in B do
if $IOU(M, b_i) \geqslant N_t$ then
$B \leftarrow B - b_i; S \leftarrow S - s_i$
end
end
end
return D, S
end

算法 2. Soft-NMS 处理步骤

Input:
$B = \{b_1, \cdots, b_n\}, S = \{s_1, \cdots, s_n\}, N_t$
B 是初始预测框的列表
S 列表包含预测框列表 B 对应的检测得分
N_t 是 NMS 预设阈值
begin
$D \leftarrow \{\}$
While $B \neq empty$ do
$m \leftarrow$ argmax S
$M \leftarrow b_m$
$D \leftarrow D \cup M; B \leftarrow B - M$
for b_i in B do
$s_i \leftarrow s_i f(IOU(M, b_i))$
end
end
end
return D, S
end

传统的 NMS 计算公式如式 (8) 所示:

$$S_i = \begin{cases} s_i, & IOU(M,b_i) < N_t \\ 0, & IOU(M,b_i) \geqslant N_t \end{cases} \tag{8}$$

在图像上具有连续性的高斯加权的 Soft-NMS 计算公式如式 (9) 所示:

$$S_i = \begin{cases} s_i, & IOU(M,b_i) < N_t \\ s_i \exp\left\{\dfrac{IOU(M,b_i)}{\sigma}\right\}, & IOU(M,b_i) \geqslant N_t \end{cases} \tag{9}$$

其中, σ 是 n 个预测框的计算复杂度.

3.5 增加负样本的优化

负样本是一种不包含识别目标的图像, 负样本训练的意义是为了减少模型错误检测的概率. 我们通过对原始训练集图像的研究, 发现了一个可能会导致模型误检的负样本场景, 该场景包括具有直线特征的背景如斑马线、直线等. 另外, 在对正样本进行标注的时候, 难免会有一部分路面和目标路面障碍同时被标记, 而这些路面的特征也会被模型学习到. 所以, 我们在收集负样本时侧重收集了具有直线特征的非典型路面障碍背景图像. 通过在训练数据集中添加负样本, 模型检测的错误率将会得到一定程度的缓解.

本文以原始模型训练中常见误测的对象为重点, 在广州所有地区的街道上随机收集和筛选了 500 张负样本, 再结合 1 500 张正样本, 一共得到了包含 2 000 张样本的数据集 A. 我们将 Mobile-YOLOv3 网络分别在数据集 A 与只有 2 000 张正样本的数据集 B 上进行了训练, 得到的检测精度对比如表 1 所示, 实验结果证明在数据集中添加负样本可以提升模型训练的效果.

表 1 添加负样本测试精度对比

数据集	正样本 (张)	负样本 (张)	MAP (%)
A	1 500	500	94.77
B	2 000	0	92.61

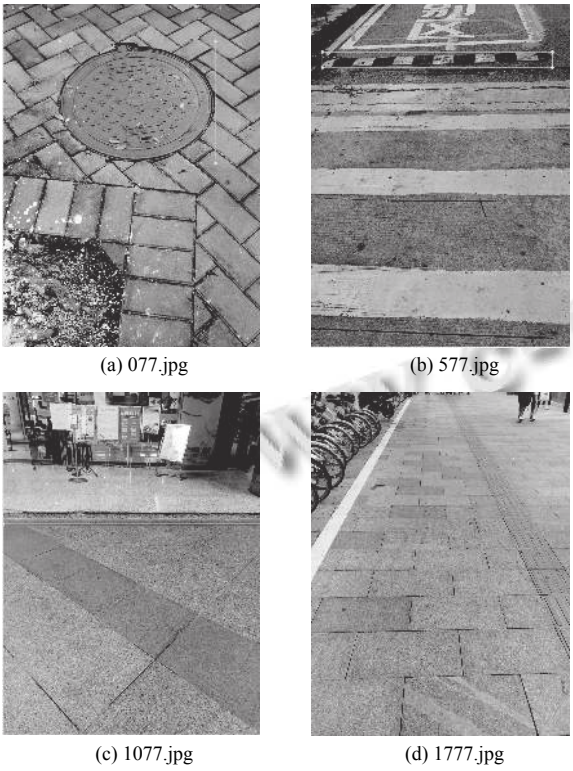
4 实验结果和分析

4.1 实验环境和数据预处理

实验的模型训练和检测的软硬件环境为 Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz, 128 GB 内存, Nvdia GeForce GTX 1080ti 显卡. 操作系统为 Ubuntu 16.04.6 LTS. 开发环境为 Python 3.6.10, TensorFlow 1.10.0, CUDA9.0, CUDNN7.6.4, IDE 平台为 Visual Studio Code, 数据标注软件为 labelImg.

由于本文应用的场景是行人在行走时低头使用手机所遇到的道路障碍, 因此对数据集的拍摄角度和形状等方面存在特殊性要求, 而现有的开源数据集没有此类型的图像. 路面障碍数据集是一个模拟行人低头使用手机角度所拍摄的照片, 我们拍摄了广州各区具有代表性的道路障碍, 这些图片可以用来代表在日常生活中许多场景可能遇到的路面障碍. 在拍摄了一系列的路障图像, 我们进行筛选并去除了难以用肉眼识别的图像之后, 总共收集了 2 000 张路面障碍数据集, 路面障碍分类有井盖, 减速带, 楼梯台阶 3 类各 500 张,

为了降低误检率, 还拍摄了不含目标障碍的负样本 500 张. 我们将路障图像批量标号, 并使用 labelImg 软件进行标注, 生成了对应的 XML 文件. 本文将数据集划分为 1 620 张训练集, 180 张验证集, 200 张测试集. 部分数据集如图 7 所示.



(a) 077.jpg (b) 577.jpg

(c) 1077.jpg (d) 1777.jpg

图 7 部分数据集示例

4.2 实验评价指标 MAP

在目标检测中, 如果预测框与真实框的交并比 IOU 大于预设的阈值, 则认为此预测框预测正确, 否则认为此预测框预测错误. 假如给定一张 A 类别的图像, 则此图的预测精度 P_A 如式 (10) 所示:

$$P_A = \frac{N(TruePositives)_A}{N(TotalObjects)_A} \tag{10}$$

即在数据集中有一张类别为 A 的图像, 其检测精度等于该图像准确预测 (true positives) 框的数量除以该图像检测生成的预测框的总数目. 则 A 类图像预测的平均精度 AP_A 如式 (11) 所示:

$$AP_A = \frac{\sum P_A}{N(TotalObjects)_A} \tag{11}$$

即一个 A 类图像预测的平均精度等于测试集上 A 类图像的预测精度值之和除以 A 类图像的总数量. 而一个数据集中会有多个类别的图像, 因此需要一个统一的概念来衡量一个网络在此数据集上训练最后得到的模型表现如何, 则均值平均精度 MAP (mean average precison) 定义如式 (12) 所示:

$$MAP = \frac{\sum AP_A}{N(Classes)} \tag{12}$$

即 MAP 等于数据集中所有类别图像预测的平均精度值之和除以该数据集中类别的总数量. MAP 值越高, 代表模型在此数据集上训练的效果越好.

4.3 模型训练过程及参数设置

在对收集到的图片进行预处理并标注生成数据集之后, 将数据集放入 Mobile-YOLOv3 模型中进行训练, 其中重要的参数设置如表 2 所示.

表 2 训练参数设置

参数名	参数值
First_Stage_Epoch	300
Second_Stage_Epoch	150
First_Stage_Lr	1e–3
Second_Stage_Lr	1e–4
BatchSize	4
ReduceLRONPlateau	Factor=0.1, patience=5
EarlyStopping	patience=10

ReduceLRONPlateau 参数设置为如果训练持续 5 个 epoch 后, 验证损失没有下降, 则将学习率调小 0.1 倍. EarlyStopping 参数设置为如果训练持续 10 个 epoch 后, 验证损失没有下降, 则提前结束训练.

本实验使用了迁移学习, 深度学习中的迁移学习是将已经训练好学习完数据集特征的模型参数迁移到新模型中, 帮助新模型进行训练的一种方法. 因为几乎所有的数据和特征基本上都是有相关关系的, 所以通过迁移学习的方式, 新模型可以直接学习已经训练好的模型参数, 从而加快模型的学习效率, 加速模型优化速度, 不需要像大多数网络一样重新开始学习.

本文训练分为两个阶段进行, 第 1 阶段采用了迁移学习的方式, 冻结了 MobileNetv1 模型的前 160 层, 并使用在 ImageNet 数据集上预训练的 MobileNetv1 模型进行迁移学习, 训练次数为 300 个 epoch. 第 2 阶段将第 1 阶段冻结的所有卷积层解封进行训练, 训练次数为 150 个 epoch. 训练过程中设置了早停和学习率自动衰减参数, 防止无效训练. Mobile-YOLOv3 的训练过程如图 8, 图 9 所示, 检测精度如图 10 所示, 其中, SpeedBump, WellLid, Stairs 为路面障碍数据集中的 3 类路障标签.

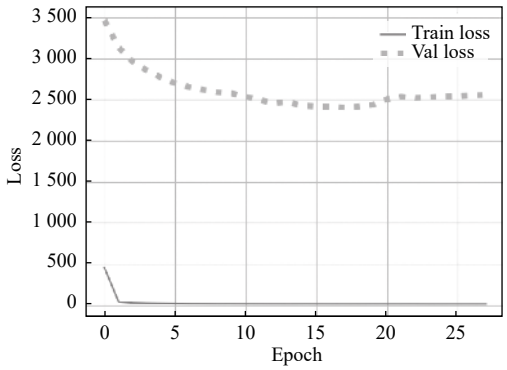


图 8 Mobile-YOLOv3 第一阶段训练图像

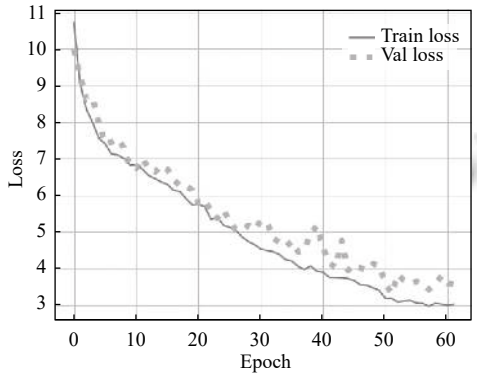


图 9 Mobile-YOLOv3 第二阶段训练图像

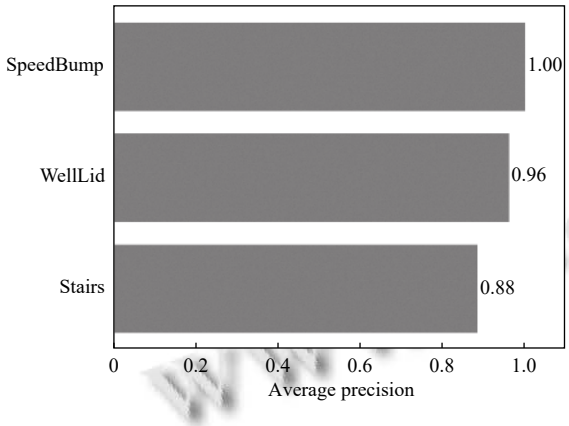


图 10 Mobile-YOLOv3 训练检测精度图

4.4 各网络性能对比

本文对 YOLOv3, YOLOv3 网络的精简版 Tiny-YOLOv3 和 Mobile-YOLOv3 网络分别进行了训练, 训练参数设置和数据集预处理均和第 4.3 节一致, 训练结果如表 3 所示. 由表 3 可以看到, 虽然 Tiny-YOLOv3 训练出来的模型规模只有 YOLOv3 原模型的 1/7 左右, 但是 *MAP* 下降了将近 11%. 对比之下, Mobile-YOLOv3 训练出来的网络模型规模为原来的 0.4 倍, 使用 CPU 检测图像的速度比 YOLOv3 提升了将近一倍, 使用 GPU 检测图像的速度也比 YOLOv3 提升了 26%, 而且 *MAP* 也提升了将近 3%, 我们考虑到可能是因为 YOLOv3 是针对大型数据集检测而设计的目标检测模型, 对于具有直线, 圆形等平面特征的小型路障数据集没有很好的效果. 实验结果证明 Mobile-YOLOv3 网络更适用于基于移动设备的路障检测问题.

表 3 各网络性能对比

Model	Weight (M)	Time_CPU (s)	Time_GPU (s)	*MAP* (%)
YOLOv3	246.9	0.693 15	0.215 6	91.98
Tiny-YOLOv3	34.8	0.196 2	0.140 8	81.15
Mobile-YOLOv3	97.1	0.391 1	0.170 5	**94.77**

4.5 消融实验

本文将 Mobile-YOLOv3 模型的边框回归损失函数替换为了 CIOU, 分类损失函数替换为了 Focal, 目标框筛选算法替换为了 Soft-NMS, 数据集添加了负样本进行训练. 作为对照组分别将 Mobile-YOLOv3 的边框回归损失函数, 分类损失函数, 目标检测框筛选算法替换为 YOLOv3 原来的算法, 数据集添加负样本训练, 最后训练得到的结果如表 4 所示. 由表 4 可以看出, 本文改进的 3 种优化算法都可以提升 Mobile-YOLOv3 模型对路面障碍数据集的检测速度和精度, 最终使用了 3 种优化算法和增加了负样本训练机制的模型 Mobile-YOLOv3 检测精度为 98.84%, *MAP* 相比原始 Mobile-YOLOv3 模型的 94.77% 提升了约 4.1%. 改进后的 Mobile-YOLOv3 模型相比于 YOLOv3 模型, 均值平均精度 *MAP* 提升了约 7%, 模型规模也缩小了约 2.5 倍, 使用 CPU 检测图像的速度提升了 80%, 使用 GPU 检测图像的速度也提升了 30%.

5 总结

本文自行设计标注了一个包含各类道路障碍特征的数据集, 用于模型效果的性能评估. 我们在基于 YOLOv3 的基础上分别对其骨干网络和边框回归损失函数, 分类损失函数, 预测框筛选算法进行了改进, 并在数据集中添加了负样本训练, 提出了一种针对行人使用便携移动设备走路时遇到路面障碍的目标检测算法. 实验结果表明, 本文提出的 Mobile-YOLOv3 道路障碍检测轻量化算法对日常生活场景经常遇到的具有楼梯, 井盖, 减速带等几何特征的道路障碍的检测, 具有良好的实时性和精确度, 且模型性能相比

于 YOLOv3 网络, 模型规模缩减了约 2.5 倍, 平均均值精度 *MAP* 提升了约 7%, 识别的精度值达 98.84%.

表 4 各优化方法消融实验

Network	CIOU	Focal	Soft-NMS	Time_CPU (s)	Time_GPU (s)	*MAP* (%)
Mobile-YOLOv3	N	Y	Y	0.384 1	0.171 2	97.71
Mobile-YOLOv3	Y	N	Y	0.379 5	0.176 6	97.22
Mobile-YOLOv3	Y	Y	N	0.383 7	0.178 0	96.54
Mobile-YOLOv3	Y	Y	Y	0.379 8	0.171 0	**98.84**

参考文献

1 吴欧, 刘庆敏, 赵江磊, 等. 杭州市中学生步行时使用电子设备行为调查. 中国学校卫生, 2017, 38(10): 1489–1492.

2 谷宇. 基于智能手机的步行安全警报系统 [硕士学位论文]. 南京: 南京大学, 2018.

3 Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Kauai: IEEE, 2001. I-511–I-518 [doi: 10.1109/cvpr.2001.990517.]

4 Dalal N, Triggs B. Histograms of oriented gradients for human detection. Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Diego: IEEE, 2005. 886–893. [doi: 10.1109/cvpr.2005.177]

5 Felzenszwalb P, McAllester D, Ramanan D. A discriminatively trained, multiscale, deformable part model. Proceedings of 2008 IEEE Conference on Computer Vision and Pattern Recognition. Anchorage: IEEE, 2008. 1–8. [doi: 10.1109/cvpr.2008.4587597]

6 Liu W, Anguelov D, Erhan D, *et al*. SSD: Single shot MultiBox detector. Proceedings of the 14th European Conference on Computer Vision. Amsterdam: Springer, 2016. 21–37. [doi: 10.1007/978-3-319-46448-0_2]

7 Redmon J, Divvala S, Girshick R, *et al*. You only look once: Unified, real-time object detection. Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas: IEEE, 2016. 779–788. [doi: 10.1109/cvpr.2016.91]

8 Girshick R, Donahue J, Darrell T, *et al*. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition. Columbus: IEEE, 2014. 580–587.

9 Girshick R. Fast R-CNN. Proceedings of 2015 IEEE International Conference on Computer Vision. Santiago: IEEE, 2015. 1440–1448.

10 Dai JF, Li Y, He KM, *et al*. R-FCN: Object detection via region-based fully convolutional networks. arXiv: 1605.06409, 2016.

11 Redmon J, Farhadi A. YOLOv3: An incremental improvement. arXiv: 1804.02767, 2018.

12 Howard AG, Zhu ML, Chen B, *et al*. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv: 1704.04861, 2017.

13 Zheng ZH, Wang P, Liu W, *et al*. Distance-IoU loss: Faster and better learning for bounding box regression. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 34(7): 12993–13000. [doi: 10.1609/aaai.v34i07.6999]

14 Lin TY, Goyal P, Girshick R, *et al*. Focal loss for dense object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, 42(2): 318–327. [doi: 10.1109/TPAMI.2018.2858826]

15 Bodla N, Singh B, Chellappa R, *et al*. Improving object detection with one line of code. arXiv: 1704.04503, 2017.