计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(4):163−169 [doi: 10.15888/j.cnki.csa.006866]

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

面向嵌入式设备的深度学习物体检测优化算法①

戴雷燕, 冯　杰, 董　慧, 杨小利

(浙江理工大学 信息学院, 杭州 310018)
通讯作者: 冯　杰, E-mail: arlose@zstu.edu.cn

摘　要: 随着深度神经网络研究地不断深入, 物体检测的精度和速率都在不断提升, 但是随着网络层的加深, 模型体积不断增大, 计算代价也越来越高, 无法满足神经网络直接在嵌入式设备上实现快速前向推理的需求. 为了解决这个问题, 本文针对嵌入式设备进行深度学习物体检测优化算法研究. 首先, 选择合适的物体检测算法框架和神经网络架构; 然后在此基础上针对特定检测场景下采集的图片进行训练和模型剪枝; 最后, 对移植到嵌入式设备上的模型剪枝后的物体检测模型进行汇编指令优化. 综合优化后, 与原有网络模型相比, 模型体积减小 9.96%, 速度加快 8.82 倍.

关键词: 深度学习; 物体检测; 剪枝; 汇编优化; 嵌入式设备

引用格式: 戴雷燕,冯杰,董慧,杨小利.面向嵌入式设备的深度学习物体检测优化算法.计算机系统应用,2019,28(4):163–169. http://www.c-s-a.org.cn/1003-3254/6866.html

Deep Learning Object Detection Optimization Algorithm for Embedded Devices

DAI Lei-Yan, FENG Jie, DONG Hui, YANG Xiao-Li

(School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: Along with the deep research on neural network, the object detection precision and speed are improved. But, computational cost is higher and higher with the deepening of network layer and increasing model volume, it cannot meet the needs that the neural network realizes fast forward reasoning directly in the embedded devices. In order to solve this problem, we study deep learning object detection optimization algorithm for embedded devices in this study. First, we choose the appropriate object detection algorithm and neural network frame structure. Then, the training and model pruning are carried out for the images collected under the specific detection scenario. Finally, the assembly instruction is optimized for the pruned object detection model transplanted to the embedded device. Compared with the original network model, the proposed model volume is reduced by 9.96% and the speed is accelerated by 8.82 times after comprehensive optimization.

Key words: deep learning; object detection; pruning; assembly optimization; embedded device

1　引言

物体检测是计算机视觉领域的一个重要问题, 物体检测算法从传统的人工特征 (如梯度方向直方图, SIFT 角点) 和经典的统计机器学习方法 (如支持向量机, 朴素贝叶斯)[1–5]发展到现今的基于卷积神经网络的物体检测算法, 物体检测效果令人惊叹. 2014 年, R.Girshick 等人将卷积神经网络应用到物体检测中, 提出了基于区域的卷积神经网络 (R-CNN) 算法[6], 使得检测效果远远优于当时的传统物体检测算法. 随后的 Fast R-CNN[7]和 Faster R-CNN[8]等都在检测准确率和

① 基金项目: 国家自然科学基金青年基金项目 (61501402)
Foundation item: Young Scientists Fund of National Natural Science Foundation of China (61501402)
收稿时间: 2018-10-29; 修改时间: 2018-11-19; 采用时间: 2018-11-23; csa 在线出版时间: 2019-03-28

定位精度上不断突破, 而 2015 年 Joseph Redmon 等提出的 YOLO[9]算法和 2016 年 Wei Liu 等提出的 SSD[10]算法都是在算法速度上进行突破. 在物体检测算法不断突破的同时, 神经网络架构也在不断更新.

最早的卷积神经网络 LeNet[11]是 Yann LeCun 在 1998 年设计并提出的, 它一开始是针对识别手写数字问题, 网络结构较为简单, 但是包含了卷积神经网络的基本单元, 是以后各种模型的始祖. 2012 年, 在 ImageNet 比赛上大放异彩的 AlexNet[12]虽然只比 LeNe 多了几层, 但效果远远优于最早的 LeNet, AlexNet 将激活函数由 Sigmoid 换成了 ReLU, 解决了因为网络层更深时的梯度弥散问题, 还加入了 dropout 层, 有效解决过拟合问题. 2014 年出现的 VGG[13]相较于 AlexNet 而言网络层越来越深, 整个网络层全部使用 3×3 大小的小卷积核和 2×2 的最大池化核, 使得整个网络结构简洁, 具有更好的泛化性. Network in Network[14]模型的亮点在于 1×1 卷积核和全局平均池化层 (Global Average Pooling) 的应用, 使得整个网络模型的参数规模锐减, 有利于网络的加深, 训练时间也得到提升. 2014 年的 GoogLeNet[15]中使用了基础卷积块 Inception, 最后的全连接层借用 NiN 模型思想换成 Global Average Pooling, 整个模型的参数量大大减小, 效果更好. 2015 年的 ResNet[16]中包含了 Residual 单元, 允许原始输入信息直接传递到后面的网络层中, 解决了因为网络层过深而梯度消失, 无法反向传播的问题. 2017 年谷歌公司提出的 MobileNets 使用深度分离卷积[17], 进一步减小计算量, 压缩模型, 可以满足嵌入式设备的需求.

随着硬件计算能力的快速发展, 深度学习技术已经应用到很多便携式嵌入设备中, 比如各种刷脸支付、相册智能分类、图片风格迁移等, 但是目前一大挑战是如何将庞大深度模型移植到资源有限的嵌入式设备上[18].

2 优化流程

本文针对嵌入式设备进行的深度学习物体检测优化算法研究主要在 PC 端和嵌入式平台 ARM 上完成. 首先, 准备好训练数据, 选择合适的物体检测框架和神经网络架构进行构建; 然后在此网络结构下, 对训练数据进行模型训练, 得出原始物体检测模型; 接着对原始物体检测模型进行模型剪枝, 以上过程都是在 PC 端上完成. 将 PC 端上完成模型剪枝的物体检测模型移植到嵌入式设备 ARM 平台上, 针对该物体检测模型的各个网络层进行运算代价计算, 根据计算结果, 对运算代价较高的网络层进行汇编优化. 汇编优化后的物体检测模型在 ARM 平台上对冰箱内采集到的视频内容进行前向推理, 在 ARM 平台上显示检测结果, 上述过程在 ARM 平台上完成, 图 1 为整个物体检测优化算法的流程图.

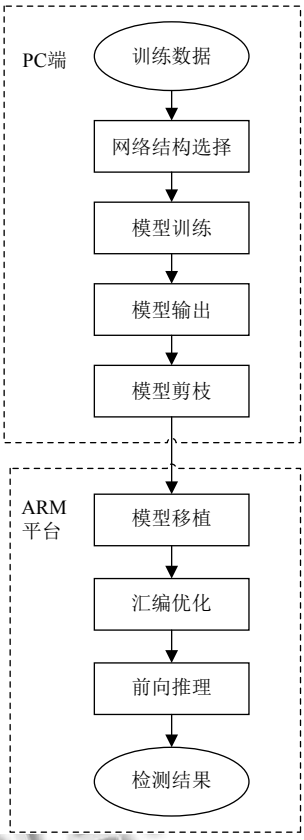

图 1 优化算法流程图

2.1 网络结构选择

本文整体网络架构选用 MobileNets, MobileNets 中的深度可分解卷积是将标准卷积分解成一个深度卷积和一个点卷积.

假设输入 F 的维度为 $D_F \times D_F \times M$, 其中 D_F 为输入 F 的宽和高, M 为输入层的通道数. 标准卷积时, 卷积 K 的参数量表示为 $D_K \times D_K \times M \times N$, 其中 D_K 为卷积 K 的维度, N 为输出通道数, 则此时的计算代价为 C:

$$C = D_K \times D_K \times M \times N \times D_F \times D_F \quad (1)$$

深度可分解卷积时, 将标准卷积 K 拆分为深度卷积 $D_K \times D_K \times 1 \times M$ 和点卷积 $1 \times 1 \times M \times N$, 计算代价为 C':

$$C' = D_K \times D_K \times M \times D_F \times D_F + M \times N \times D_F \times D_F \quad (2)$$

将深度可分解卷积的计算代价 C'与标准卷积的计

算代价 C 相除, 进行比较:

$$\frac{C'}{C}=\frac{D_K\times D_K\times M\times D_F\times D_F+M\times N\times D_F\times D_F}{D_K\times D_K\times M\times N\times D_F\times D_F}=\frac{1}{N}+\frac{1}{D_K^2} \tag{3}$$

在 ImageNet 数据集上, 深度分离卷积的参数量约是标准卷积参数量 1/7, 而准确率只下降了不到 1%[17].

深度分离卷积在网络结构上的变化如图 2 所示, 将标准卷积拆分成深层卷积和点卷积, 在每层卷积后面都有 BN 层和激励函数 ReLU.

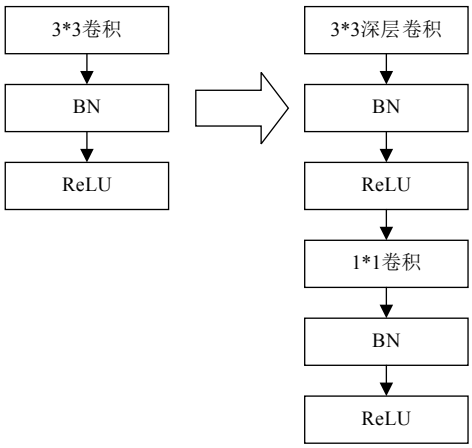

图 2 网络结构变化

在物体检测算法方面, R-CNN 系列都是物体定位和物体分类分开进行的物体检测方法, 虽然准确率很高, 但是速度较慢; 而 YOLO 和 SSD 则都是一步完成物体定位和物体分类, 其中 YOLO 的速度很快, 但是准确率不尽人意, 而 SSD 综合了 Faster R-CNN 的高准确率和 YOLO 的快速率优点, 使得物体检测方法更加适合应用在嵌入式设备上, 故选择 SSD 作为本次面向嵌入式设备的深度学习物体检测优化算法研究的物体检测框架.

2.2 模型剪枝

在 PC 端上的模型剪枝流程图如图 3 所示.

首先, 对 MobileNets-SSD 网络进行神经元重要性的评估, 然后根据评估结果将其中最不重要的神经元置零, 对部分神经元置零后的网络进行微调, 判断微调后的网络是否达到预定标准, 如果达到预定的标准, 则可以停止剪枝; 如果未到到预定的标准, 则需要重复神经元重要性评估、不重要神经元置零、微调等步骤.

本文在模型剪枝过程中采用了一种基于一阶泰勒展开的新的神经元重要性评估准则[19]. 该准则中对神经元的评估公式为:

$$|\Delta C(h_i)|=\left|C(D|W')-C(D|W)\right| \tag{4}$$

上式中, D 表示训练样本集, W 表示未修剪前的权重参数, W'表示修剪后的权重参数, $C(D|W)$ 为修剪后的代价函数, 而 $C(D|W')$ 表示的是修剪后的代价函数, 最优化的剪枝就是使得修剪后的代价函数 $C(D|W')$ 尽可能地逼近修剪前的代价函数 $C(D|W)$. 将剪枝前后的代价函数差值赋值给$|\Delta C(h_i)|$, 找出令式 (4) 中$|\Delta C(h_i)|$值最小的非零权重参数, 将该参数置零, 即完成一次剪枝. 又因为 $C(D|hi)=C(D|(w,b)_i)$, 所以式 (4) 可以变换为式 (5):

$$|\Delta C(h_i)|=|C(D,h_i=0)-C(D,h_i)| \tag{5}$$

在式 (5) 中的 $C(D,hi=0)$ 部分引入一阶泰勒展开式, 如式 (6) 所示:

$$C(D,h_i=0)=C(D,h_i)-\frac{\delta C}{\delta h_i}h_i+R_1(h_i=0) \tag{6}$$

其中, $R_1(h_i=0)$ 为高阶项, 因为 ReLU 激活层中含有高阶信息, 同时为了避免复杂的计算, 高阶项 $R_1(h_i=0)$ 可以忽略不计.

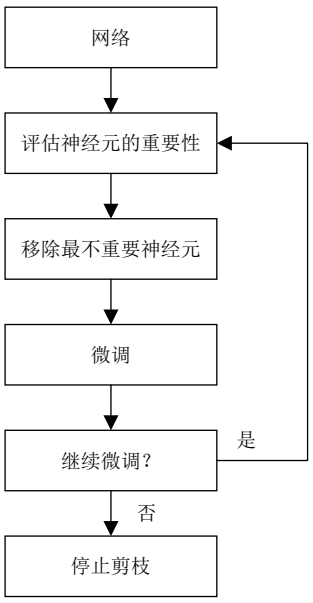

图 3 模型剪枝流程图

综合式 (5) 和式 (6) 可得式 (7):

$$\Theta_{TE}(h_i)=|\Delta C(h_i)|=\left|C(D,h_i)-\frac{\delta C}{\delta h_i}h_i-C(D,h_i)\right|=\left|\frac{\delta C}{\delta h_i}h_i\right| \tag{7}$$

将特征图看作变量的向量表示, 当有 M 个变量时, 剪枝前后的损失函数差值可表示为:

$$\Theta_{TE}(z_l^{(k)}) = \left| \frac{1}{M} \sum_m \frac{\delta C}{\delta z_{l,m}^{(k)}} z_{l,m}^{(k)} \right| \quad (8)$$

2.3 汇编优化

嵌入式设备主要由嵌入式处理器、相关支撑硬件和嵌入式软件系统组成, 而树莓派是一款基于 ARM 的微型电脑主板, 是作为学习计算机编程的首选嵌入式设备. 本文使用的是树莓派 3 B, 它有 1.2 GHz 和 64 位处理器, 满足物体检测模型汇编优化和前向推理的性能需求.

在该 ARM 平台上, 通过对表 1 模型剪枝对比中各个网络层的耗时比较, 决定针对其中卷积层 Conv_0、Conv_1、Conv_2、Conv_3、Conv_4、Conv_5、Conv_6、Conv_7、Conv_8、Conv_9、Conv_10、Conv_11、Conv_12、Conv_13、Conv_14_1 和 Conv_14_2 进行汇编优化,主要策略分为指令调整、寄存器分配和条件执行[20].

表 1 模型剪枝对比 (单位: ms)

网络层	原始	剪枝 512	剪枝 1024
Conv_0	111	83	61
Conv_1_dw	27	22	18
Conv_1	324	243	186
Conv_2_dw	21	16	14
Conv_2	313	251	197
Conv_3_dw	27	24	21
Conv_3	617	561	517
Conv_4_dw	10	8	8
Conv_4	316	253	215
Conv_5_dw	15	12	12
Conv_5	643	587	535
Conv_6_dw	6	6	6
Conv_6	208	152	111
Conv_7_dw	8	7	7
Conv_7	646	586	537
Conv_8_dw	8	7	7
Conv_8	714	639	592
Conv_9_dw	8	7	7
Conv_9	718	628	584
Conv_10_dw	8	7	7
Conv_10	783	683	604
Conv_11_dw	8	8	8
Conv_11	675	585	509
loc,conf	102	100	99
Conv_12_dw	3	3	3
Conv_12	263	258	259
Conv_13_dw	5	5	5
Conv_13	716	710	709
loc,conf	110	109	109
Conv_14_1	182	181	181
Conv_14_2	106	105	106
其他	42	41	41
总计	7773	6887	6275

(1) 指令调整: 通过展开循环对装载指令进行人工优化, 仔细安排装载指令的时间次序, 防止流水线终止.

(2) 寄存器分配: 限制局部变量的个数; 把多个局部变量存放在一个寄存器中.

(3) 条件执行: 使用 ARM 处理器特有的条件执行指令来减少判断跳转和分支等对流水线影响较大的操作.

通过以上策略最大程度地利用平台资源, 发挥处理器最大效能, 来满足物体检测模型直接在嵌入式设备上快速前向推理的需求.

3 实验分析

3.1 数据集制作

通过冰箱内的 USB 摄像头对总共 76 种蔬菜水果饮料等冰箱常见食材进行训练样本采集, 一共采集 5 万张图片, 最后挑选出 45 488 张作为训练样本, 如图 4 所示.

图 4 训练样本

图 5 数据标注

利用网上标注系统对挑选出的训练样本进行数据标注, 网上标注系统的界面如图 5 所示, 可以对物体进行种类选择和位置定位.

3.2 训练优化

训练时采用 MobileNets 作为本次实验的基础网络架构, 其中总共有 35 层卷积层, 在 Conv_11、Conv_13、Conv_14_2、Conv_15_2、Conv_16_2 和 Conv_17_2 后进行物体检测方法 SSD 的分类器分类、位置回归和锚框的生成. 最后, 将以上 6 个卷积层生成的类别特征、位置特征和 anchor box 信息进行 concat、Softmax

和 flatten. MobileNets 的网络层与本文的 MobileNets-SSD 网络层如表 2 所示. 相比于 MobileNets, 本文的 MobileNets-SSD 多 8 层卷积层, 且最后的平均池化层、全连接层和 Softmax 变成了 conf_reshape、conf_softmax 和 conf_flatten. 对训练集进行训练后, 最终本文的 MobileNets-SSD 模型的 mAP 达到 84.7%.

表 2 MobileNets 与本文的 MobileNets-SSD

MobileNets	MobileNets-SSD
Conv/s2	Conv_0
Conv dw/s1	Conv_1_dw
Conv/s1	Conv_1
Conv dw/s2	Conv_2_dw
Conv/s1	Conv_2
Conv dw/s1	Conv_3_dw
Conv/s1	Conv_3
Conv dw/s2	Conv_4_dw
Conv/s1	Conv_4
Conv dw/s1	Conv_5_dw
Conv/s1	Conv_5
Conv dw/s2	Conv_6_dw
Conv/s1	Conv_6
Conv dw/s1	Conv_7_dw
Conv/s1	Conv_7
Conv dw/s1	Conv_8_dw
Conv/s1	Conv_8
Conv dw/s1	Conv_9_dw
Conv/s1	Conv_9
Conv dw/s1	Conv_10_dw
Conv/s1	Conv_10
Conv dw/s1	Conv_11_dw
Conv/s1	Conv_11
	Conv_11_loc,conf,priorbox
Conv dw/s2	Conv_12_dw
Conv/s1	Conv_12
Conv dw/s2	Conv_13_dw
Conv/s1	Conv_13
	Conv_13_loc,conf,priorbox
	Conv_14_1
	Conv_14_2
	Conv_14_2_loc,conf,priorbox
	Conv_15_1
	Conv_15_2
	Conv_15_2_loc,conf,priorbox
	Conv_16_1
	Conv_16_2
	Conv_16_2_loc,conf,priorbox
	Conv_17_1
	Conv_17_2
	Conv_17_2_loc,conf,priorbox
Avg Pool/s1	loc,conf,priorbox
FC/s1	conf_reshape
Softmax/s1	conf_softmax
	conf_flatten

MobileNets-SSD 完成训练后进行基于一阶泰勒展开的模型剪枝. 控制裁剪的整体过程为: 1) 前向传播 2) 获取排序后的卷积窗口 3) 计算需要剪枝的卷积窗口个数 4) 裁剪. 主要函数如下所示:

(1) forward

(2) compute_rank

(3) normalize_ranks_per_layer

(4) get_prunning_plan

(5) lowest_ranking_filters

其中函数 (1) 表示模型的前向传播过程; 函数 (2) 用在梯度更新时, 其输出的计算值用于卷积窗口排序; 函数 (3) 是将每层的结果归一化; 函数 (4) 和函数 (5) 表示利用最小堆方法得到 N 个排名最低的卷积窗口.

完成一次剪枝后, 进行模型微调和迭代. 所有参数参与训练与学习, 重新训练模型 10 个迭代.

具体实验时, N 分别设置为 512 和 1024, 此时每层网络层耗时情况如表 1 模型剪枝对比所示. N 为 512 时, 物体检测模型的 mAP 为 85.1%; N 设为 1024 时, 物体检测模型的 mAP 为 82.3%. 可以看出, N 为 512 时, 物体检测模型的 mAP 有所提升; 而 N 为 1024 时, 物体检测模型的 mAP 有所下降, 说明 N 设置为 512 时, 模型剪枝后的物体检测模型复杂度更为适合本次实验训练数据集和测试数据集, 前向推理效果更好.

所以, 综合表 1 模型剪枝对比中 N 为 512 和 1024 时的耗时情况以及相对应的物体检测模型的 mAP 值, 本实验选择针对 N 为 512 的模型剪枝后的 MobileNets-SSD 物体检测模型进行汇编优化. 经过模型剪枝后的 MobileNets-SSD 模型体积由 23.1 MB 缩减为 20.8 MB, 模型体积减小 9.96%.

3.3 ARM 平台优化

由表 1 模型剪枝对比可知, 物体检测模型中卷积层 Conv_0、Conv_1、Conv_2、Conv_3、Conv_4、Conv_5、Conv_6、Conv_7、Conv_8、Conv_9、Conv_10、Conv_11、Conv_12、Conv_13、Conv_14_1 和 Conv_14_2 无论是模型剪枝前还是模型剪枝后, 耗时都比较久. 我们通过指令调整、寄存器分配和条件执行等手段对模型剪枝前和模型剪枝后的耗时多的网络层分别进行汇编优化. 主要应用到的是 NEON 技术, 它是 ARM 处理器的 128 位 SIMD 架构扩展, 旨在为消费性多媒体应用程序提供灵活、强大的加速功能. 它具有 32 个 64 位寄存器和 16 个

128 位寄存器. 寄存器的具体调用代码如下所示:

```
(1) .macro MobileNets-SSD
(2) vldl.32 {d16-d19},[BO]!
(3) vldl.32 {d0-d3},[AO]!
(4) vldl.32 {d16-d19},[BO]!
(5) vldl.32 {d4-d7},[AO]!
(6) vmla.f32 q12,q0,d16[0]
(7) vmla.f32 q12,q2,d18[0]
(8) vmla.f32 q12,q3,d20[0]
(9) vmla.f32 q12,q4,d22[0]
(10) ...
(11) vstl.32 {d24-d27},[CO]!
(12) vstl.32 {d28-d31},[CO]!
(13) .endm
```

上述代码 (1) 行, .macro 后面的字符串表示该宏的名称,在后续的调用中可以直接利用该宏来替代内部的具体实现代码. 代码 (13) 行的.end 为宏的载止位置.

代码 (2)–(5) 行为数据加载指令, 其中 A0 表示矩阵起始地址, B0 为右矩阵的起始地址, 两者均为地址寄存器. vldl,32 指令的调用可以将数据按照 32 位为一个数据单位的顺序加载到 NEON 寄存器当中, 数据的连续性能够减少指令的访存时间.

代码 (6)–(9) 行为结果矩阵第一列的计算过程, vmla.f32 为乘加指令.

代码 (10) 行表示其他几列的计算方式.

代码 (11)–(12) 行将寄存器当中的结果矩阵的值存储到内存当中, 当存储位置是连续的情况下, 指令的访存将同样是连续的, 减少数据存储所需时间.

汇编优化后的结果如表 3 汇编对比所示.

由表 3 汇编对比可知, 模型剪枝后的 MobileNets-SSD 模型前向推理速率相较于未进行模型剪枝时的速率加快了 1.16 倍, 而汇编前后模型的 mAP 无变化; 又结合表 1 模型剪枝对比和表 3 汇编对比, 经过模型剪枝和汇编优化的 MobileNets-SSD 模型前向推理速率相较于原始模型加快了 8.82 倍.

本文还选择其他两个前向推理框架进行了对比, 使用的都是剪枝之后的网络模型, 如表 4 所示. Mini-Caffe 是对 Caffe 的前向推理版本, 汇编优化比较少, Ncnn 是腾讯开发的带汇编优化的前向推理框架, 做了比较深度的汇编优化. 与它们相比, 本文最终的前向推理时间是最低的.

表 3 汇编对比

网络层	汇编 (ms)	汇编 512(ms)
Conv_0	60	43
Conv_1_dw	21	17
Conv_1	141	107
Conv_2_dw	15	11
Conv_2	38	30
Conv_3_dw	12	11
Conv_3	77	69
Conv_4_dw	8	6
Conv_4	34	27
Conv_5_dw	6	5
Conv_5	68	62
Conv_6_dw	4	4
Conv_6	22	16
Conv_7_dw	4	4
Conv_7	69	62
Conv_8_dw	4	4
Conv_8	69	63
Conv_9_dw	4	4
Conv_9	67	57
Conv_10_dw	4	4
Conv_10	68	60
Conv_11_dw	5	5
Conv_11	69	61
loc,conf	13	13
Conv_12_dw	2	2
Conv_12	26	26
Conv_13_dw	3	3
Conv_13	54	53
loc,conf	9	9
Conv_14_1	14	13
Conv_14_2	20	20
其他	11	10
总计 (ms)	1021	881

表 4 与其他框架的对比

前向推理框架	Mini-Caffe	Ncnn	本文
总耗时 (ms)	7146	1097	1021

3.4 检测结果

在 ARM 平台上的前向推理时, 测试样本检测结果如图 6 所示.

部分物体未被检测的原因有以下几点:

(1) 遮挡物过多, 未能检测到整个物体;

(2) 离 USB 摄像头太远, 暴露面积太小;

(3) 只暴露物体部分特征, 前向推理困难等.

前期的训练样本的选择对最后嵌入式设备上物体检测模型的前向推理检测结果有一定影响. 整体效果较为理想.

图 6 检测结果

4 结论与展望

本文选择了合适的物体检测框架 SSD 和神经网络架构 MobileNets, 训练出一个满足嵌入式设备需求的物体检测模型, 并通过模型剪枝对该物体检测模型进行优化, 在移植到 ARM 平台上之后又进行汇编优化, 进一步加快前向推理速率. 在准确精度和模型大小方面还可以有进一步的研究:

(1) MobileNets V2 的准确率和速率都有提高, 可以在此网络结构上进行物体检测算法优化.

(2) 因为模型中参数的存储精度为 32 位的浮点数, 可以在 ARM 平台上针对模型参数进行量化, 更大程度地压缩模型, 加快前向推理速率.

参考文献

1 胡仕玲, 顾爽, 陈启军. 基于 HOG 的物体分类方法. 华中科技大学学报 (自然科学版), 2011, 39(S2): 124–126, 130.

2 潘子昂. 基于 SIFT 算法的图像匹配研究[硕士学位论文]. 西安: 西安电子科技大学, 2012.

3 艾扬利, 赵忠芹, 杨兵. 基于新核函数的支持向量机在物体分类中的应用. 中国测试技术, 2008, 34(1): 80–83.

4 Lowd D, Domingos P. Naive Bayes models for probability estimation. Proceedings of the 22nd International Conference on Machine Learning. Bonn, Germany. 2005. 529–536.

5 邓江帆. 基于学习的目标检测及应用[硕士学位论文]. 北京: 北京邮电大学, 2017.

6 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, OH, USA. 2014. 580–587.

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

8 Ren SQ, He KM, Girshick R, *et al*. Faster R-CNN: Towards real-time object detection with region proposal networks. Proceedings of the 28th International Conference on Neural Information Processing Systems. Montreal, Canada. 2015. 91–99.

9 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. Las Vegas, NV, USA. 2016. 779–788.

10 Liu W, Anguelov D, Erhan D, *et al*. SSD: Single shot MultiBox detector. Proceedings of 14th European Conference on Computer Vision. Amsterdam, The Netherlands. 2016. 21–37.

11 LeCun Y, Bottou L, Bengio Y, *et al*. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278–2324. [doi: 10.1109/5.726791]

12 Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Proceedings of the 25th International Conference on Neural Information Processing Systems. Lake Tahoe, NV, USA. 2012. 1097–1105.

13 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv: 1409.1556, 2014.

14 Lin M, Chen Q, Yan S. Network in network. arXiv preprint arXiv: 1312.4400, 2013.

15 Szegedy C, Liu W, Jia Y Q, *et al*. Going deeper with convolutions. Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition. Boston, MA, USA. 2015. 1–9.

16 He KM, Zhang XY, Ren SQ, *et al*. Deep residual learning for image recognition. Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, NV, USA. 2016. 770–778.

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

18 黄萱昆. 基于深度学习的移动端图像识别算法[硕士学位论文]. 北京: 北京邮电大学, 2018.

19 李晓云, 周聪. 基于 ARM9TDMI 的汇编优化方法. 计算机与现代化, 2007, (2): 25–27, 31. [doi: 10.3969/j.issn.1006-2475.2007.02.009]

20 Molchanov P, Tyree S, Karras T, *et al*. Pruning convolutional neural networks for resource efficient transfer learning. arxiv preprint arXiv: 1611.06440, 2017.