计算机工程
Computer Engineering
ISSN 1000-3428,CN 31-1289/TP

《计算机工程》网络首发论文

题目：基于 NCS2 神经计算棒的车辆检测方法
作者：江枭宇，李忠兵，张军豪，彭娇，文婷
DOI：10.19678/j.issn.1000-3428.0056214
网络首发日期：2020-03-26
引用格式：江枭宇，李忠兵，张军豪，彭娇，文婷．基于 NCS2 神经计算棒的车辆检测方法．计算机工程. https://doi.org/10.19678/j.issn.1000-3428.0056214

网络首发时间：2020-03-26 16:50:55
网络首发地址：http://kns.cnki.net/kcms/detail/31.1289.tp.20200326.1307.004.html

基于 NCS2 神经计算棒的车辆检测方法

江枭宇，李忠兵，张军豪，彭娇，文婷
（西南石油大学 电气信息学院，成都 610500）

摘　要：目前深度学习检测车辆的大部分方法已经具有很高的准确率，但其实时性依赖于性能卓越的 PC 机或 GPU 设备，限制了在性能相对较低的嵌入式设备中的实际应用。因此，本文设计了一套基于树莓派 3b+的嵌入式车辆检测系统，采用 NCS2 神经计算棒（一个微型的深度学习硬件驱动器），为低性能设备提供深度学习加速功能。并提出了一种基于深度可分离卷积的 Tiny-YOLO 网络模型，用于实时检测车辆，以提高嵌入式设备目标检测准确度。实验证明，本文提出的算法实时性相比原始 Tiny-YOLO 提高了一倍，将经过重训练后的网络模型部署到拥有 NCS2 神经棒的树莓派 3b+时，可达到每秒 12 帧的处理速度，同时改进后的算法在 MS COCO、VOC2007 车辆数据集上平均检测准确率分别提高了 1.12%、0.23%。

关键词：车辆检测；深度可分离卷积；深度学习；Tiny-YOLO；嵌入式设备

开放科学标识码(OSID):

Vehicle detection method based on NCS2 neural compute stick

Jiang Xiaoyu, Li Zhongbing, Zhang Junhao, Peng Jiao, Wen Ting.
(School of Electrical Engineering and Information, Southwest Petroleum University, Chengdu 610500, China)

【Abstract】At present, most methods of deep learning to detect vehicles have high accuracy, but the real-time performance depends on PC or GPU devices with excellent performance, which limits its practical applications in embedded devices with relatively low performance. Therefore, an embedded vehicle detection system based on raspberry PI 3b+ is designed in this paper, which uses NCS2 neural computing rod (a miniature deep learning hardware driver) to provide deep learning acceleration for low-performance devices. A Tiny-YOLO network model based on depth separable convolution is proposed for real-time vehicle detection to improve the accuracy of target detection of embedded devices. Real-time experiments show that the real-time performance of the proposed algorithm is twice that of the original Tiny-YOLO algorithm. After the retrained network model is deployed to the raspberry PI 3b+ with NCS2 neural stick, the processing speed can reach 12 frames per second. At the same time, the improved algorithm improves the average detection accuracy by 1.12% and 0.23% respectively in MS COCO and VOC207 vehicle data sets.

【Key words】Vehicle detection; Depth separable convolution; Deep learning; Tiny-YOLO; Embedded device;
DOI:10.19678/j.issn.1000-3428.0056214.

1　概述

环境感知是无人驾驶路径规划的基础，在对周边环境形成认知模型时，通过摄像头采集并实时获取周边车辆信息，是其中的关键。在车辆检测方面，经典的车辆检测方法使用梯度直方图（HOG）[1] 或者尺度不变特征（SIFT）[2] 作为特征，然后使用支持向量机（SVM）[3] 或自适应分类器（Adaboost[4] 、 Gradient Boosting[5]）进行识别，缺乏高层语义信息。近年来，深度学习在车辆检测领域的作用越来越大。与经典方法不同，深度学习不需要人为设置特征而是通过反向传播(Back Propagation)[6] 自适应获取特征，对高层语义信息有更好的描述能力。基于深度学习的车辆检测方法可以划分为两个类别，one stage 和 two stage。two stage 方法是先通过预选框（Region Proposal）[7-8] 找到位置，针对选出的位置再进行识别，常用算法有 R-CNN[9]、Fast R-CNN[10]、Faster R-CNN[7] 等。而 one stage 直接端到端，一次性识别出位置和类别，比如 YOLO[11-12]、SSD[13] 等。基于深度学习的方法对车辆检测精度都能达到满意的效果，但是因运算能力的原因，无法在实际应用中落地。

基金项目：四川省大学生创新创业训练计划项目（201810615094）、产学合作协同育人项目（201801006095）
作者简介：江枭宇(1996—)，男，本科，研究领域为计算机视觉，E-mail:ceroo1005@gmail.com；李忠兵，博士，讲师；张军豪，硕士研究生；彭娇，硕士研究生；文婷，本科。

为解决运算能力问题，Joseph Redmon[14] 对 YOLO 算法进行精简，提出 Tiny-Yolo，但是其模型相对于嵌入式来说依旧具有重量。已有研究者采用计算棒来对硬件实现加速计算[15]。本文将车辆检测算法 Tiny-YOLO 中的传统卷积使用深度可分离卷积（depthwise separable convolutions）[16] 替换，并部署到具有 NCS2 神经计算棒的树莓派上，最终达到每秒 12 帧数的理想效果。

2 YOLOV3 原理

YOLOV3 网络结构主要由基础网络 Darknet53 紧跟全卷积层[17] 组成，基础网络 Darknet53 包含 53 层卷积层，并且由残差结构[18] 构建，可使训练网络的难度大大减少，提高算法效率。

将输入 416×416 图像经过基础网络 Drknet53、全卷积层，输出得到 13×13、26×26、52×52 三个特征图，每个特征图又划分多个网络域，每个网络域输出尺寸为 1×1×(B×(5+C))，1×1 是最后一层卷积核大小，B 代表每个网络域可以预测的边界框数量，而每个边界数量则是一种检测目标，每个目标有 5+C 个属性，分别描述每个边界框的t_x, t_y, t_w, t_h，objectness 分数以及 C 类置信度。

由于训练会带来不稳定的梯度，YOLOV3 在数据集样本中先使用 K-means 聚类算法[19] 生成了 3 个不同尺度的 3 组先验框，预测框则是基于这 9 个先验框进行微调。设P_x, P_y为先验框在特征图上的中心点坐标，P_w, P_h是先验框在特征图上的宽和高，G_x, G_y, G_w, G_h是 ground truth 在特征图上的 4 个坐标，它们之间的偏移关系t_x, t_y, t_w, t_h如下

$$\begin{aligned} t_x &= G_x - P_x \\ t_y &= G_y - P_y \\ t_w &= \log\left(G_w / P_w\right) \\ t_h &= \log\left(G_h / P_h\right) \end{aligned} \tag{1}$$

中心点坐标的偏移直接作差即可，对于框的宽和高，需要把尺度缩放到对数空间，易于使用梯度下降做训练。先验框和 ground truth 之间的偏移求出后，可用于修正先验框和预测框之间的关系，如图 1 所示。

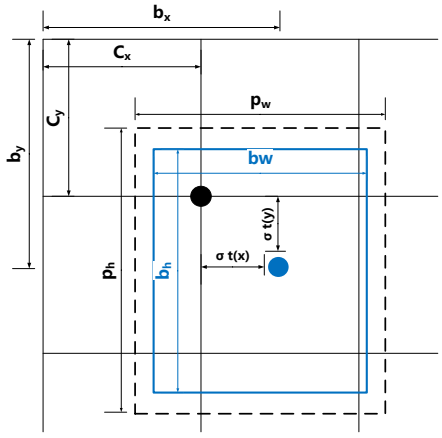


图 1 先验框和预测框之间的偏移关系

Fig.1 The offset between prior box and prediction box

蓝色实线表示预测框，蓝色实点表示预测框的中心点，黑色虚线表示先验框，黑色实点表示预测中心点所在网络域的左上角坐标(C_x, C_y)。(C_x, C_y)是由P_x, P_y来确定的，对于预测框的b_x, b_y, b_w, b_h使用式（1）中算出的偏移t_x, t_y, t_w, t_h预测即可，具体公式如下

$$\begin{aligned} b_x &= \sigma\left(t_x\right) + C_x \\ b_y &= \sigma\left(t_y\right) + C_y \\ b_w &= P_w e^{t_w} \\ b_h &= P_h e^{t_h} \end{aligned} \tag{2}$$

式中σ是 sigmoid 函数，网络域的尺寸是 1×1，使用 sigmoid 函数将t_x, t_y缩放到 0～1 之间，可以有效确保目标中心处于网络域中，防止偏移过度。而t_w, t_h之前使用了对数空间，使用指数回到G_w / P_w或者G_h / P_h再乘以真实的P_w或P_h得到真实的宽和高。

objectness 分数通过以下公式计算

$$C_{Object} = P_{Object} \times IOU_{pred}^{truth} \tag{3}$$

当存在目标时 P_{Object}=1，否则为 0；IOU 则是预测框和 ground truth 的面积交并比。

在原始 YOLOV3 中最佳效果是 B 取 3，C 取 80，表示一个网络域需要预测 3 个边界框并且有 80 类别数量。

3 改进的 Tiny-YOLO

Tiny 版本的 YOLO 删除了原始版本中反复出现加深网络的残差结构，节省内存，提高速度，并且只有两个尺寸的输出，分别是 13×13 和 26×26。但其结构中使用的标准卷积仍消耗了大量的运算，而 MobileNet[20] 中的深度可分离卷积可使运算大幅度减少。

3.1 深度可分离卷积

深度可分离卷积把标准卷积分解成深度卷积(depthwise convolution)和逐点卷积(pointwise convolution)，以减少复杂度节省算力，更适合嵌入式设备，其分解示意图如图 2 所示。

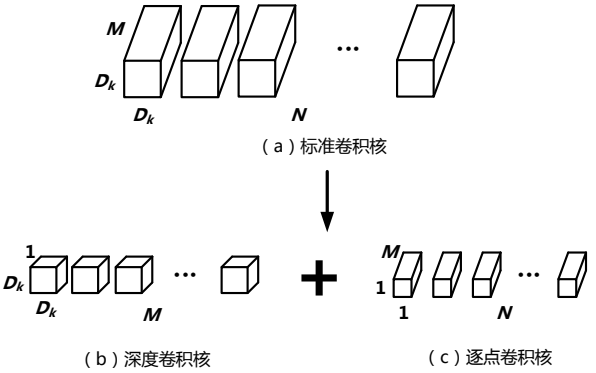


图 2 深度可分离卷积分解图

Fig.2 The structure of deep separable convolution

卷积运算时输入为 $F \in R^{D_f \times D_f \times M}$，其中 $D_f \times D_f$ 是输入图像或特征图大小，M 为图像或特征图通道数，采用的卷积核为 $K \in R^{D_k \times D_k \times M \times N}$，其中 $D_k \times D_k$ 为卷积核尺寸,M 和 N 分别对应卷积核通道数和卷积核个数，输出 $G \in R^{D_g \times D_g \times N}$，其中 $D_g \times D_g$ 是输出特征图的大小，N 是输出特征图通道数。标准卷积计算公式如下

$$\mathbf{G}_{k,l,n} = \sum_{i,j,m} \mathbf{K}_{i,j,m,n} \cdot \mathbf{F}_{k+i-1,l+j-1,m} \tag{4}$$

其计算量为 $O_{(conv)} = D_K \cdot D_K \cdot M \cdot N \cdot D_f \cdot D_f$

深度可分离卷积由两部分组成：深度卷积和逐点卷积。深度卷积是在输入特征图的每个通道上应用单个滤波器进行滤波。深度卷积操作时输入为 $F \in R^{D_f \times D_f \times M}$，卷积核 $\hat{K} \in R^{D_k \times D_k \times 1 \times M}$，其计算公式如下

$$\hat{\mathbf{G}}_{k,l,m} = \sum_{i,j} \hat{\mathbf{K}}_{i,j,m} \cdot \mathbf{F}_{k+i-1,l+j-1,m} \tag{5}$$

其卷积计算量为 $D_K \cdot D_K \cdot M \cdot D_f \cdot D_f$。

相比于标准卷积，深度卷积非常有效，但是它仅仅过滤输入通道，还需要将他们组合以创建新功能，因此需要通过 1x1 卷积这一层来创建线性组合，生成新特征。所以深度可分离卷积总的卷积运算量为

$$O'_{(conv)} = D_K \cdot D_K \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f$$

深度可分离卷积计算量与标准卷积计算量之比为

$$\frac{O'_{(conv)}}{O_{(conv)}} = \frac{D_K \cdot D_K \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f}{D_K \cdot D_K \cdot M \cdot N \cdot D_f \cdot D_f} = \frac{1}{N} + \frac{1}{D_K^2} \tag{6}$$

由此可见，当 N>1，D_k 不变时，深度可分离卷积的运算量将比标准卷积明显降低。

3.2 改进的 Tiny-YOLO 网络结构

为了进一步降低 Tiny-YOLO 的运算量，本文引入深度可分离卷积（图 3 中“S Conv 3×3”模块），替代原始 Tiny-YOLO 特征提取网络结构中所有大小为 3×3 的标准卷积层（图 3 中“Conv 3×3”模块，一共 9 层，具体信息在表 1 中）。为了防止由池化引起的低级特征丢失，本文删除了原始 Tiny-YOLO 中所有的池化层（图 3 中“Maxpool”模块），并通过全卷积连接。原始和改进的 Tiny-YOLO 网络结构如图 3 所示。

表 1 原始网络中 3x3 卷积信息

Tab.1 3 x 3 convolution information in the original network

网络层	滤波器大小	滤波器个数	输入尺寸	输出尺寸
1	3x3	16	416x416x3	416x416x16
2	3x3	32	208x208x16	208x208x32
3	3x3	64	104x104x32	104x104x64
4	3x3	128	52x52x64	52x52x128
5	3x3	256	26x26x128	26x26x256
6	3x3	512	13x13x256	13x13x512
7	3x3	1024	13x13x512	13x13x1024
8	3x3	512	13x13x256	13x13x512
9	3x3	256	13x13x384	13x13x256

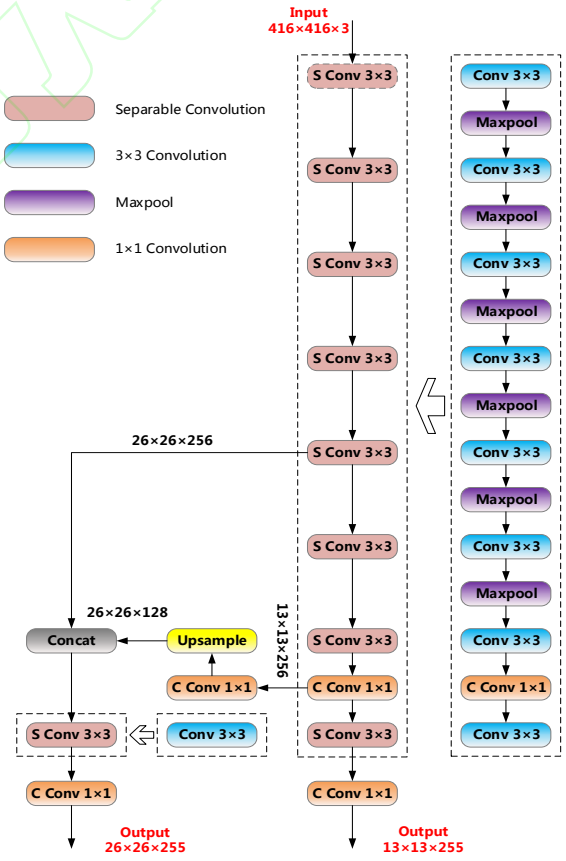


图 3 原始及改进的 Tiny-YOLO 网络结构

Fig.3 Original and improved Tiny-YOLO network structure

4 实验分析

本文在含有车辆图片的 MS COCO 数据作为实验数据，从中筛选出 821 张车辆图片，图片中车辆角度随机且像素高度大于 400。车辆尺度不一，光线明亮且背景丰富，划分 700 张作为训练集和 121 张作为测试集，并增加 VOC 数据集做测试。

本文的实验环境 1）操作系统：ubuntu 16.04，2）深度学习框架：tensorflow，3）下位机设备：树莓派 raspberry 3b+、计算棒 NCS2，4）上位机设备：E5 2680 + GTX1066。

4.1 开发流程

改进算法训练和测试图片的输入尺度与原始算法一致，通过深度可分离卷积网络提取车辆特征，再通过两个不同尺度的特征图来预测。在上位机上采用 tensorflow 深度学习框架对改进算法进行训练，获得模型文件后，用 OpenVINO 模型优化器将 tensorflow 模型转换为 NCS2 支持的 IR 文件，最后部署到具有 NCS2 计算棒的树莓派上，具体流程图如图 4 所示。

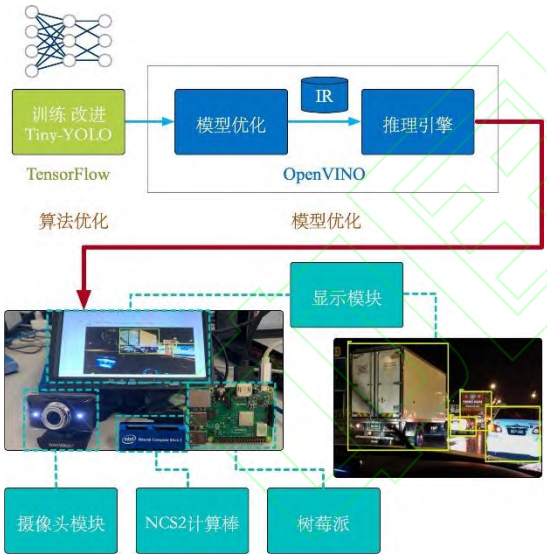
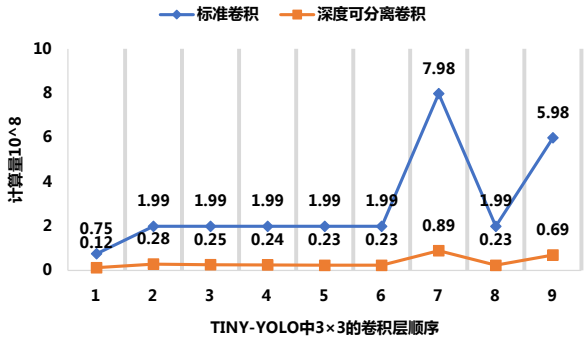


图 4 开发流程图

Fig.4 Development flow chart

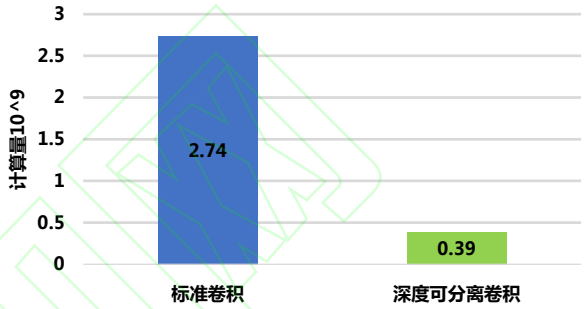
4.2 计算量对比实验

在神经网络计算中，由于乘法次数远大于加法次数，而一次乘法计算时间远大于一次加法时间，整体上加法运算时间的总和可以忽略。本文以一次乘法计算记为一次计算量，原始 Tiny-YOLO 网络结构中 9 层各自计算量在做卷积结构替换前后的结果如图 5 所示。



(a) 替换结构前后计算量对比

(a) comparison of calculation amount before and after replacing structure



(b) 总计算量

(b) total calculation

图 5（a）为替换结构部分计算量，(b)为算法总计算量

Fig.5 (a) is the calculation amount of the replacement structure part, and (b) is the total calculation amount of the algorithm

由实验可得，标准卷积被深度可分离卷积替换后，计算量大幅度减少，并且总的计算量从 2.74×10^11 减少到 0.39×10^11，节省了约 86%的运算量。

4.3 准确度和速度对比实验

在对模型进行评价时，使用平均准确率（Mean Average Precision，MAP）指标，具体公式为：

$$m_{AP} = \int_0^1 P(R)\mathrm{d}R \tag{7}$$

其中，P 为准确率，R 为召回率，P(R)为不同召回率上的平均准确率。

在对实时性评价时，使用每秒传输帧数（Frames Per Second，FPS），值越大，动画越流畅。

表 2 本算法和原始算法的对比

Tab.2 Comparison between this algorithm and the original algorithm

方法	精度（MAP）		模型大小 (MB)	速度 (FPS)
	VOC 数据集	COOC 数据集		
Tiny-YOLO	55.66%	33.10%	34.6	2
本算法	**55.89%**	**34.22%**	22.4	4
NCS2	54.12%	**34.21%**	**13.2**	**12**

表 2 是本算法和原始算法的比较。从实验数据可以看出，改进后的算法在 MS COCO、VOC2007 车辆数据集上的平均检测准确率比起原始算法分别提高了 1.12%、0.23%，而算法速度是原始算法的 2 倍。经 OpenVINO 模型优化器后的模型在保持精度相当的情况下，在采用 NCS2 计算棒的树莓派上速度能达到 12FPS，说明改进后的算法在嵌入式设备上比起原始算法精度更高速度更快。

4.4 不同场景的效果对比

在 VOC 测试数据中对两种算法进行了结果对比，如图 6 所示，对每一组图片，左图是原始 Tiny-YOLO 算法，右图是本文算法。

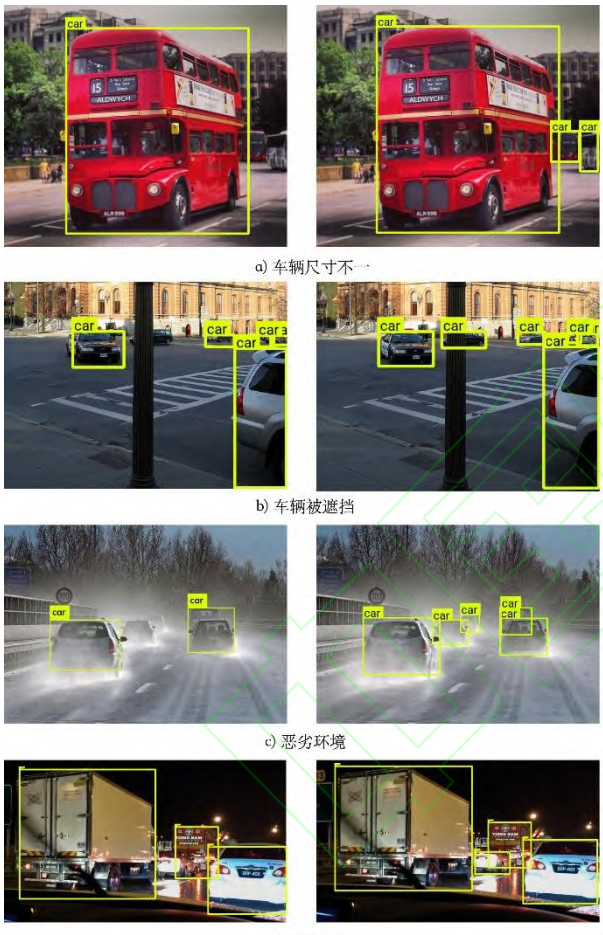


图 6 效果对比

Fig.6 Effect contrast

从实验结果可以看出，在车辆尺寸不一时原始算法丢失了小目标，在遮挡的情况下原始算法对遮挡的车辆无法捕获，在恶劣和夜间环境下原始算法易受环境和光线干扰。实验说明本文提出的删除丢失信息后的池化层，并用全卷积的改进 Tiny-YOLO 算法检测效果优于原始算法。

5 结束语

文本引入了深度可分离卷积来代替原始的 Tiny-YOLO 中的标准卷积块，并部署到资源紧张的嵌入式设备中，有效降低了计算量，同时采用神经计算棒提高算力，使得最终车辆检测结果比起原始算法速度更快精度更高，可作为无人驾驶感知环境的重要依据，并且在边缘计算中意义重大。下一步将对已有的模型做裁剪和量化压缩，进一步降低模型大小从而提高速度，以便在更加常用的嵌入式处理器（如 STM32）中得到广泛应用。

参考文献：

[1] Dalal, N, Triggs, B. Histograms of Oriented Gradients for Human Detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05); IEEE, 2005, Vol. 1, 886–893.

[2] David G.Lowe. Object Recognition from Local Scale-Invariant Features. In Proceedings of the Seventh IEEE International Conference on Computer Vision; IEEE: 1999, vol. 2, 1150–1157.

[3] Sánchez A, V. David. Advanced Support Vector Machines and Kernel Methods. Neurocomputing, 2003, 55 (1–2), 5–20.

[4] Artue Ferreira, Mario Figueiredo. Boosting Algorithms: A Review of Methods, Theory, and Applications. In Ensemble Machine Learning, 2012, 35–85.

[5] Friedman, Jerome. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics. 2000

[6] David E, Rumelhart, Geoffrey E, Hinton, Ronald J. Williams. Learning Representations by Back-Propagating Errors. 1986, 533–536

[7] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497 [cs] 2015.

[8] Tao Kong, Anbang Yao, Yurong Chen, Fuchun Sun. HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); IEEE, 2016, 845–853.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. arXiv:1311.2524

[cs] 2013.

[10] Ross Girshick. Fast R-CNN. arXiv:1504.08083 [cs] 2015.

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once Unified, Real-Time Object Dete. arXiv:1506.02640 [cs] 2015.

[12] Joseph Redmon, Ali Farhadi. YOLO9000 Better, Faster, Stronger. arXiv:1 1612.08242 [cs] 2016.

[13] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy. SSD: Single Shot MultiBox Detector. arXiv:1512.02325 [cs] 2016.

[14] Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv:1 1 1804.02767 [cs] 2018.

[15] ZHANG Yangshuo, MIAO Zhuang, WANG Jiabao, LI Yang. Pedestrian detection method based on Movidius neural computing stick. Journal of Computer Applications[j], 2019, 39 (08), 2230–2234. 张洋硕; 苗壮; 王家宝; 李阳. 基于Movidius神经计算棒的行人检测方法. 计算机应用 2019, 39 (08), 2230–2234.

[16] Francois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR);IEEE,2017 1800–1807.

[17] Jonathan Long, Evan Shelhamer, Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation; IEEE, 2016, 640–651

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs] 2015.

[19] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schroedl. Constrained K-Means Clustering with Background Knowledge. Proceedings of 18th International Conference on Machine Learning[j], 2001, 577-584

[20] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861 [cs] 2017.