文章编号:1671-4598(2019)07-0015-05 DOI:10.16526/j.cnki.11—4762/tp.2019.07.004 中图分类号:TP389.1 文献标识码:A

基于深度学习的实时图像目标检测系统设计

李 林[1],张盛兵[1],吴 鹃[2]
(1.西北工业大学 计算机学院,西安 710072;2.西安职业技术学院 动漫软件学院,西安 710077)

摘要:针对图像目标检测的嵌入式实时应用需求,采用合并计算层的方法对基于MobileNet和单发多框检测器(SSD)的深度学习目标检测算法进行了优化,并采用软硬件结合的设计方法,基于ZYNQ可扩展处理平台设计了实时图像目标检测系统;在系统中,根据优化后的算法设计了一款多处理器核的深度学习算法加速器,并采用PYTHON语言设计了系统的软件;经过多个实验测试,深度学习目标检测系统处理速度可以达到45FPS,是深度学习软件框架在CPU上运行速度的4.9倍,在GPU上的1.7倍,完全满足实时图像目标检测的需求。

关键词:深度学习;图像目标检测;实时;算法加速器

Design of Real-time Image Object Detection System Based on Deep Learning

Li Lin[1], Zhang Shengbing[1], Wu Juan[2]
(1. School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072, China;
2. School of Animation and Software, Xi'an Vocational and Technical College, Xi'an 710077, China)

Abstract: Aiming at the requirements of the embedded real-time application of image object detection, the deep learning object detection algorithm based on MobileNet and Single Shot Multi-Box Detector (SSD) is optimized by the method of combining computational layers, and the real-time image object detection system is designed by using software and hardware combination method based on ZYNQ scalable processing platform. In the system, a multi-processor core deep learning algorithm accelerator is designed according to the optimized algorithm, and the software of the system is designed by PYTHON language. After several experiments, the processing speed of deep learning object detection system can reach 45 FPS, which is 4.9X faster than deep learning framework running on CPU and 1.7X faster than on GPU. It fully meets the requirements of real-time image object detection.

Keywords: deep learning; image object detection; real-time; algorithm accelerator

0 引言

图像目标检测作为当前计算机视觉和深度学习领域的研究热点,主要解决图像中目标的类别和位置信息的获取问题。早期的图像目标检测算法主要基于手工特征提取和分类器实现,如HOG+SVM方法[1]和DPM[2]算法等,不仅缺乏有效的图像表征方法,而且计算复杂。自2012年Krizhevsky等人提出的深度卷积神经网络AlexNet[3]在ImageNet大规模视觉识别挑战赛中夺冠以来,掀起了深度学习技术在图像识别应用领域的研究热潮。众多研究者提出了诸如R-CNN[4]、Faster R-CNN[5]、YOLO[6]和SSD[7]等优秀的深度学习目标检测算法,较大程度地提高了图像目标检测的效率和识别准确率。

在当前研究中,为了构建深度学习图像目标识别系统,算法的实现分为软件和硬件两类。大部分研究者采用软件编程或基于当前流行的Caffe[8]或TensorFlow[9]等深度学习框架在CPU或GPU上实现[3-9]。也有一些研究者采用FPGA或ASIC来设计专用的硬件算法加速器的实现方式[10-11]。采用软件编程方式存在着计算效率低、占用资源多以及功耗高等缺点。基于ASIC实现虽然可以获得良好的计算效率和功耗,但是存在灵活性差和成本高的问题。而FPGA以其丰富的片上资源和可重构的特性,比较适合用于实现硬件算法加速器,但是当前研究中并没有完全发挥出算法和FPGA的计算潜能。

为了适应图像目标检测的嵌入式实时应用需求,充分挖掘目标检测算法及FPGA的并行计算特性,本文首先对基于SSD[7]与MobileNet[12]的深度学习目标检测算法进行了计算优化;然后在基于ZYNQ可扩展平台上搭建了图像目标检测系统,并给出了了软硬件的详细设计;最后通过多个实验表明,系统可有效实现图像目标检测的功能,对于VGA分辨率(640 * 480)的视频图像处理速度可以达到45FPS,满足嵌入式实时应用的要求。

1 基于SSD和MobileNet的深度学习目标检测算法及优化

目前主流的深度学习目标检测算法包括以Faster R-CNN[5]为代表的基于区域选择的目标检测算法,和以YOLO[6]和SSD[7]为代表的基于回归学习的目标检测算法。后者相对于前者具有更好的计算效率,只需前馈网络一次计算即可得到检测结果。同时,SSD[7]算法相对于YOLO[6]算法具有较高的识别准确率。因此,本文中将采用基于SSD[7]

收稿日期:2019-01-18; **修回日期**:2019-02-18。

作者简介:李 林(1982-),男,陕西华阴人,博士研究生,高级工程师,主要从事微处理器体系结构及集成电路设计方向的研究。

框架的深度学习目标检测算法进行设计。

1.1 基于 MobileNet[12]的 SSD 深度学习目标检测算法

SSD[7]算法模型通过一个基础的深度卷积神经网络来提取图像的特征，然后在多尺度特征图上的提取局部特征，并将得到的特征用于预测结果。基于 MobileNet[12]的 SSD[7]深度学习算法模型如图 1 所示。

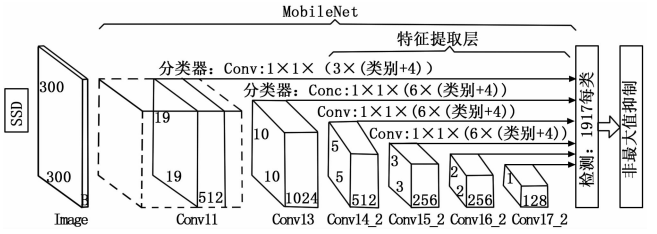

图 1 基于 MobileNet 的 SSD 深度学习算法模型

为了实现实时处理，本研究采用了适用于嵌入式移动端的轻量级的 MobileNet[12]代替文献［7］中的 VGGNet 作为 SSD[7]算法的基础卷积神经网络。MobileNet[12]采用了如图 2（b）所示的深度可分离卷积代替标准卷积，很大程度减少了深度卷积网络的参数和计算量。

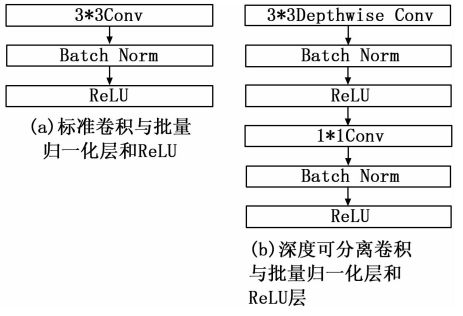

图 2 深度可分离卷积与标准卷积的对比[7]

从图 1 算法模型和图 2 中 MobileNet[12]的计算结构可以看出，特征提取算法是由卷积层、批量归一化层和非线性激活函数层。下面将根据这些层的计算抽象出算法加速器所支持的基本运算并进行优化。

1.1.1 卷积层

卷积层由多个局部滤波器（卷积核）组成，主要用于从输入特征图中提取不同的局部特征。当输入特征图的尺寸为 $W\times H\times C_{in}$，C_{out} 个通道的卷积核表示为 $K_x\times K_y\times C_{in}\times C_{out}$，则位于输出特征图 f_o（$f_o\in C_{out}$）的（x，y）位置上的神经元 N 的计算公式如式（1）所示。

$$N_{x,y}^{f_o}=\sum_{f_i=0}^{C_{in}-1}\sum_{j=0}^{K_y-1}\sum_{i=0}^{K_x-1}W_{i,j}^{(f_i,f_o)}*N_{x*S_x+i,y*S_y+j}^{f_i}+B^{(f_i,f_o)} \tag{1}$$

式中，W 和 B 为参数分别表示权重和偏置，N 表示神经元，S_x 和 S_y 表示卷积运算在 x 和 y 方向上的步长。深度可分离卷积的卷积核通道数 C_{out} 的值为 1，输出特征图由对应的输入特征图进行卷积运算得到。

1.1.2 批量归一化层

批量归一化层使得整个网络模型更加稳定，并且加快了深度卷积网络训练和收敛的速度，以特征图为一批时，它的计算如式（2）所示。

$$N_{x,y}^{f_o}=\frac{N_{x,y}^{f_i}-\frac{mean}{scalefactor}}{\sqrt{\frac{variance}{scalefactor}+\varepsilon}} \tag{2}$$

式中，mean，variance，scalefactor 和 ε 均为学习得到的参数，mean 为与特征图同维度的均值向量、variance 为与特征图同维度的方差向量，scalefactor 为一维缩放因子，ε 为一个很小的常数，通常取 0.00001。

1.1.3 缩放层

缩放层对归一化后的神经元 N 进行比例缩放和位移，它的计算如式（3）所示：

$$N_{x,y}^{f_o}=\alpha N_{x,y}^{f_i}+\beta \tag{3}$$

式中，α 和 β 为与特征图同维度的向量参数。在模型中并不体现该层，由于本文采用深度学习框架 Caffe[8]进行算法模型的训练和参数的获取，在该框架中将实际的批量归一化计算分为式（2）和式（3）两步来实现。

1.1.4 非线性激活函数层

为了使深度神经网络具有非线性的学习及表达能力，在其中加入了非线性激活函数层。在基于 MobileNet[12]的 SSD[7]算法中采用了非线性整流函数（ReLU）作为非线性激活函数，它的计算公式如式（4）所示。

$$ReLU(N_i)=\begin{cases}N_i, N_i>0\\ 0, N_i\leqslant 0\end{cases} \tag{4}$$

式中，N_i 表示输入神经元。

1.2 算法优化

本文在设计目标检测系统时，侧重于算法推理阶段的计算，训练及参数获取将采用深度学习框架 Caffe[8]完成。根据对算法各层的分析，从图 2、式（2）和式（3）可以看出实际归一化层和缩放层位于卷积层之后，它们的计算都是针对特征图内单个神经元进行的。因此，通过对训练获取的参数进行预处理，即可将它们合并到卷积层进行计算。具体过程为，设定卷积层输出神经元为 N_{conv}，批量归一化层输出神经元为 N_{BN}，缩放层输出神经元为 N_{scale}，首先根据式（2），令：

$$P_{BN}\_a=\frac{1}{\sqrt{\frac{variance}{scalefactor}+\varepsilon}} \tag{5}$$

$$P_{BN}\_b=(-\frac{\frac{mean}{scalefactor}}{\sqrt{\frac{variance}{scalefactor}+\varepsilon}}). \tag{6}$$

则根据式（2）、式（5）和式（6），式（3）可以变为：

$$N_{scale}=\alpha N_{BN}+\beta=\alpha(P_{BN}\_a*N_{conv}+P_{BN}\_b)+\beta=\alpha P_{BN}\_a*N_{conv}+\alpha P_{BN}\_b+\beta. \tag{7}$$

将式（2）带入到式（7），可以得到：

$$N_{scale}=(\sum_{f_i=0}^{C_{in}-1}\sum_{j=0}^{K_x-1}\sum_{i=0}^{K_y-1}\frac{\alpha P_{BN\_a}}{K_x*K_y*C_{in}}W_{i,j}^{(f_i,f_o)}*N_{x*S_x+i,y*S_y+j}^{f_i})+\alpha P_{BN\_a}+B^{(f_i,f_o)}+\alpha P_{BN\_b}+\beta \tag{8}$$

式中，α，β，P_{BN_a}，P_{BN_b} 为与卷积运算输出特征图同维且对应的参数向量。将批量归一化层和缩放层的参数与卷积层的权重参数进行合并，即可实现三层合一，不仅减少了参数而且减少了计算量。

另外由于算法加速器部分采用了 16 位有符号定点数来表示神经元和参数，因此对于非线性激活函数 ReLU 通过判断神经元的符号位即可实现，为了计算的便利性将其设计在算法加速器的计算单元（PE）内，减少了分层计算时将特征图存储后再取出计算的时间。

2　实时图像目标检测系统结构及软硬件设计

2.1　深度学习实时图像目标检测系统体系结构及软硬件划分

实时图像目标检测嵌入式系统应具有图像数据的采集、处理及输出的功能，基于 Xilinx 公司的 ZYNQ 可扩展开发平台设计了如图 3 所示的图像目标检测系统。

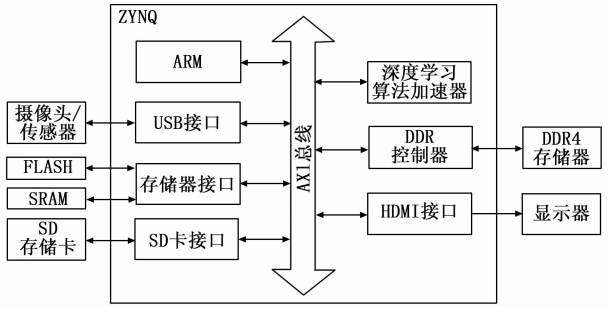

图 3　基于 ZYNQ 的图像目标检测系统结构框图

根据 ZYNQ 系统的结构和对算法可加速部分的分析进行软硬件的划分。优化后的深度学习目标检测算法的大量计算集中在卷积网络计算，占据了整个算法计算量的 90% 以上，因此将该部分采用 ZYNQ 的可编程逻辑端（PL 端）以硬件算法加速器的方式实现。算法的 PriorBox 计算、非最大值抑制计算和 Softmax 输出因包含开方、指数等计算不便采用硬件实现，因此将采用软件方式计算。系统的图像采集、缓存及输出显示操作将采用软件方式在 ZYNQ 的可编程系统端（PS 端）实现。

在如图 3 所示的系统结构中，实时的图像采集采用了一款 30W 像素的 CMOS 摄像头，捕获画面分辨率为 640×480，帧速率 30/60 FPS 可选，输出为 YUV 格式，通过 USB 接口将捕获到的视频图像缓存于 DDR4 存储器中。

图像的预处理通过 ARM 处理器通过软件方式完成。主要实现将采集到的 YUV 图像转换为 RGB 格式，并进行图像目标检测前的调整大小和去均值操作。

图像的目标检测由深度学习硬件算法加速器完成。算法加速器的参数及操作指令存储于片外的 FLASH 存储器中，在系统启动时通过 AXI 总线发送至 PL 端 FPGA 的 BRAM 中，进行图像目标检测时，将预处理后的图像通过 AXI 总线发送至 FPGA 的用于存储输入特征图的 BRAM 中进行目标检测计算，计算完成后将计算结果通过 AXI 总线发送至 PS 端缓存中。由 PS 端完成算法的其余计算。

图像目标检测结果的输出显示由软件实现，将 DDR4 缓存中的目标检测结果经过标注后通过开发板的 HDMI 接口在显示器上显示。

2.2　软件设计

ZYNQ 的可编程系统端（PS 端）的 ARM 处理器运行在 Linux 操作系统之上，Linux 操作系统的镜像文件存储在外部的 SD 存储卡中。PS 端的软件编程采用了 Linux 下的 PYTHON 语言在 Jupyter Notebook 环境下实现。在本系统中，软件用于控制整个系统的工作流程，主要实现对采集图像的预处理，调用可编程逻辑端的深度学习算法加速器计算，算法的其余计算，以及计算结果在采集图像上标注后输出显示，系统工作的流程如图 4 所示。

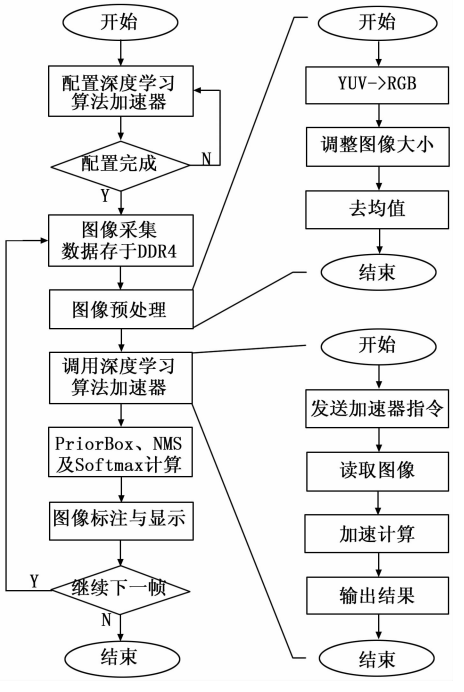

图 4　系统工作流程图

目标检测系统的软件部分还包括了参数预处理软件和深度学习指令编译器软件。由于这两部分是在本系统外一次完成，在此不再赘述。

2.3　硬件设计

2.3.1　体系结构设计

根据算法分析及系统软硬件的划分，算法加速器主要完成算法优化后的卷积计算，具有并行度高，计算量大的特点，在此基础上设计了如图 5 所示的算法加速器体系

结构。

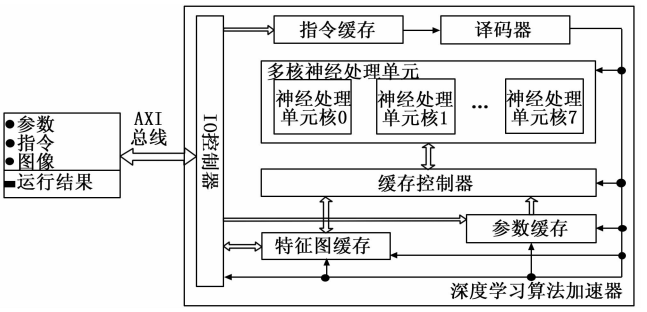

图 5 深度学习算法加速器体系结构图

算法加速器结构包含了：I/O 控制器、指令缓存、译码器、多核神经处理单元、缓存控制器、特征图缓存（包含输入特征图缓存、输出特征图缓存和临时缓存）和参数缓存。

算法加速器工作时，首先接收系统发送的参数存储于参数缓存，接着在图像预处理完成后接收系统发送的算法加速器指令并读取图像数据存于特征图缓存中，在缓存控制器的控制下从特征图缓存和参数缓存中读取卷积运算的数据和参数经过数据建立组织后发送至多核神经处理单元进行卷积运算。多核神经处理单元包含八个处理核，按照特征图并行的方式进行计算，神经处理单元内为二维的计算单元，以二维并行的方式完成特征图内神经元的计算。神经处理单元完成计算后将输出特征图经过缓存控制器存储于特征图缓存中，以便于下一层计算。直至一帧图像计算完成后将卷积运算结果经 I/O 控制器发送至 PS 端，用于后续的 PriorBox、非最大值抑制及 Softmax 输出计算。

2.3.2 神经处理单元核和计算单元的设计

神经处理单元核的设计思路来源于图像卷积运算时的二维滑动窗口过程，采用了二维计算单元的组织方式，并且支持局部的神经元数据传输，以此来减少多次读取特征图的代价。设计的神经处理单元如图 6 所示。

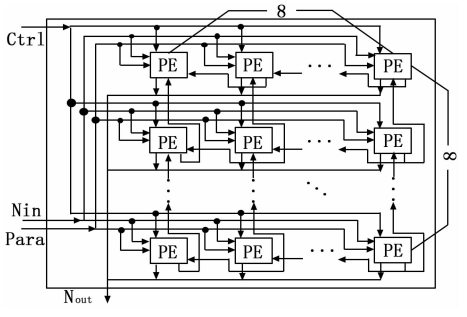

图 6 神经处理单元结构

在每个神经处理单元核内包含 8 * 8 个计算单元，可同时计算一个特征图中的 64 个神经元的卷积运算。计算单元采用局部寄存器设计支持神经元数据的行与列传输，基于乘累加器设计了卷积运算单元，通过判断符号位实现了 ReLU 的计算，所设计的计算单元如图 7 所示。

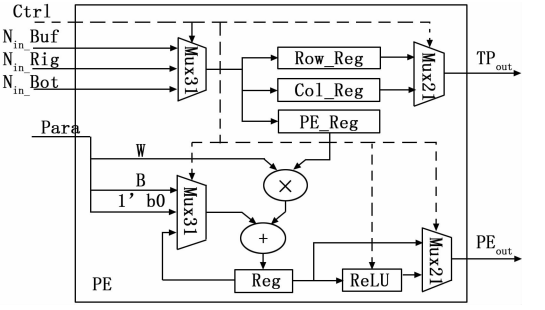

图 7 计算单元结构

采用 Vivado2018.1 开发环境基于 Xilinx 的 ZCU104 评估板进行算法加速器的硬件实现，占用可编程逻辑端的资源情况如表 1 所示。

3 实验结果与分析

3.1 训练

深度学习目标检测算法模型基于深度学习框架 Caffe 以离线方式进行训练，训练过程中采用的硬件环境包括：

表 1 算法加速器主要资源占用情况

资源	使用情况	百分比
Slice Registers	28,152/460,800	6%
Slice LUTs	82,578/230,400	35%
RAMB36E1/FIFO36E1	4/312	1%
RAMB18E1/FIFO18E1	512/624	82%
DSP48E1s	512/1,728	29%

Intel Core i7 6700HQ CPU 和 NVIDIA 的 GTX960 GPU。训练数据集采用了在 Microsoft 的 COCO[13] 数据集上进行预训练，然后在 Pascal VOC0712[14] 数据集上微调训练，可检测不包括背景的 20 种物体，获得的平均精度均值（mAP）为 0.727。在获取参数并进行预处理后对系统进行测试。

3.2 测试

系统测试环境为 Xilinx 的 ZCU104 评估开发板。开发板的 PS 端运行频率为 500MHz，PL 端运行频率为 200MHz。采用 VOC0712[14] 训练验证标准数据集进行静态图片的系统测试测得平均精度均值（mAP）为 0.721。

在单幅图片的目标检测时，本文开发系统对比基于 Caffe[8] 的深度学习框架在 3.1 节所述 CPU 和 GPU 上的运行速度如表 2 所示。

表 2 本文与 Caffe 在 CPU/GPU 上运行速度对比

测试基准	Caffe on CPU	Caffe on GPU	本文
VOC0712 单幅图片	109.76ms	37.89ms	22.15ms

对摄像头实时采集图像进行目标检测时，近景目标检测效果如图 8 所示，远景检测效果如图 9 所示，连续帧的目标检测效果如图 10 所示。

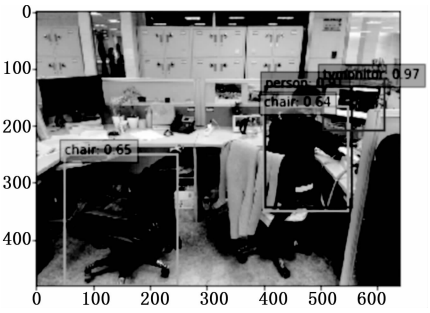

图 8　近景效果图

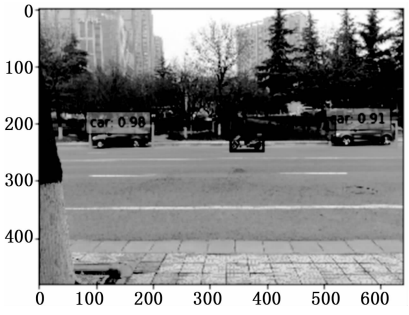

图 9　远景效果图

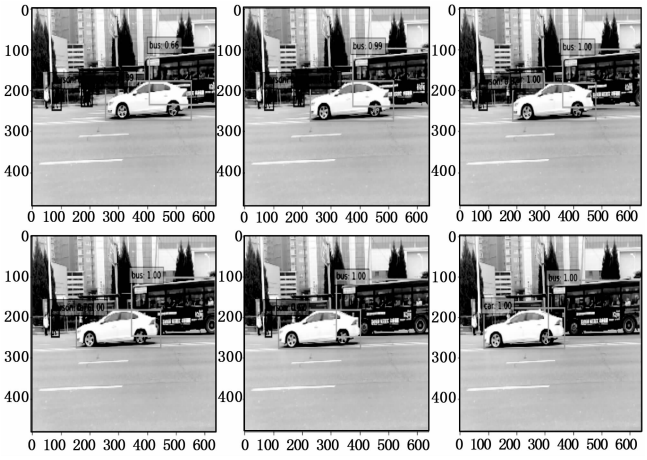

图 10　连续帧检测效果图

3.3　测试结果分析

根据测试结果可以看出，在标准测试集上进行测试时，本文设计系统的测试精确率与深度学习软件框架 Caffe[8] 的测试结果误差不超过 1%。由表 2 可得，本系统的处理速度是 CPU 的 4.9 倍，是 GPU 的 1.7 倍，检测速度达到 45FPS 完全满足实时处理的需求。由图 8、图 9 和图 10 可以看出本系统完全满足对实时采集数据的目标识别，但是对于重叠物和小目标的识别还需改进。

4　结束语

本文针对图像目标检测的嵌入式实时应用，在对算法优化的基础上，采用软硬件结合的方式，基于 ZYNQ 可扩展处理平台设计了一种基于深度学习的实时目标检测系统。经过多项测试，该系统处理速度可以达到 45FPS，完全满足嵌入式实时图像目标检测的应用需求。

参考文献：

[1] Dalal N, Triggs B. Dalal N, Triggs B. Histograms of Oriented Gradients for Human Detection [A]. IEEE International Conference on Computer Vision & Pattern Recognition [C]. IEEE Computer Society, 2005: 886 - 893.

[2] Felzenszwalb P F, Mcallester D A, Ramanan D. A Discriminatively Trained, Multiscale, Deformable Part Model [A]. IEEE Computer Society Conference on Computer Vision and Pattern Recognition [C]. IEEE. 2008: 1 - 8.

[3] Krizhevsky A, Sutskever I, and Hinton G. ImageNet Classification with Deep Convolutional Neural Networks [A]. International Conference on Neural Information Processing Systems [C]. Curran Associates Inc. 2012: 1097 - 1105.

[4] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation [A]. IEEE Conference on Computer Vision and Pattern Recognition [C]. IEEE. 2014: 580 - 587.

[5] Ren S, He K, Girshick R, et al. Faster R—CNN: Towards Real—Time Object Detection with Region Proposal Networks [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 39 (6): 1137 - 1149.

[6] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real—Time Object Detection [A]. IEEE Conference on Computer Vision and Pattern Recognition [C]. IEEE. 2016: 779 - 788.

[7] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot Multibox Detector [A]. European Conference on Computer Vision [C]. Springer International Publishing. 2016: 21 - 37.

[8] Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional Architecture for Fast Feature Embedding [A]. ACM International Conference on Multimedia [C]. ACM. 2014: 675 - 678.

[9] Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large—Scale Machine Learning on Heterogeneous Distributed Systems [J]. arXiv preprint. 2016, arXiv: 1603: 04467.

[10] Chen T, Du Z, Sun N, et al. DianNao: a small—footprint high—throughput accelerator for ubiquitous machine—learning [J]. Acm Sigplan Notices, 2014, 49 (4): 269 - 284.

[11] Farabet C, Martini B, Corda B, et al. NeuFlow: A runtime reconfigurable dataflow processor for vision [A]. IEEE Conference on Computer Vision and Pattern Recognition Workshops [C]. IEEE Computer Society, 2011.

[12] Howard A G, Zhu M, Chen B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [J]. arXiv preprint. 2017, arXiv: 1704. 04861.

[13] Lin T Y, Maire M, Belonqie S, et al. Microsoft COCO: Common Objects in Context [A]. European Conference on Computer Vision [C]. Springer. 2014: 740 - 755.

[14] Everingham M, Eslami S, Van G, et al. The Pascal Visual Object Classes Challenge: A Retrospective [J]. International Journal of Computer Vision, 2015, 111 (1): 98 - 136.