文章编号　1004-924X(2023)04-0543-09

基于ZYNQ的 Yolo v3-SPP实时目标检测系统

张丽丽，陈　真*，刘雨轩，屈乐乐
(沈阳航空航天大学 电子信息工程学院，辽宁 沈阳 110000)

摘要：基于卷积神经网络的目标检测算法发展迅速，随着计算复杂度增加，对设备的性能及功耗要求越来越高。为了使目标检测算法能够部署在嵌入式设备上，本文采用软硬件协同设计方法，使用FPGA对算法进行硬件加速，提出了ZYNQ平台下的Yolo v3-SPP目标检测系统。本文将该系统部署在XCZU15EG芯片上，并对系统所需的功耗、硬件资源及性能进行了分析。首先对要部署的网络模型进行优化，并在Pascal VOC 2007数据集上进行训练，最后使用Vitis AI工具对训练后的模型进行量化、编译，使其适用于ZYNQ端的部署。为了选取最佳的配置方案，探究了各配置对硬件资源及系统性能的影响，从系统功耗(W)、检测速度(FPS)、各类别平均精度的平均值(mAP)、输出误差等方面对系统进行了分析。结果表明：在300 M时钟频率下，输入图片大小为(416，416)时，针对Yolo V3-SPP和Yolo V3-Tiny网络结构，检测速度分别为38.44 FPS和177FPS，mAP分别为80.35%和68.55%，片上芯片功耗为21.583 W，整板功耗23.02 W。满足嵌入式设备部署神经网络模型的低功耗、实时性、高检测精度等要求。

关　键　词：目标检测；硬件加速；ZYNQ；Yolo v3-SPP；Yolo v3-Tiny
中图分类号：TP394.1；TH691.9　　**文献标识码**：A　　**doi**：10.37188/OPE.20233104.0543

Yolo v3-SPP real-time target detection system based on ZYNQ

ZHANG Lili，CHEN Zhen*，LIU Yuxuan，QU Lele

(*College of Electronic and Information Engineering*，*Shenyang Aerospace University*，*Shenyang* 110000，*China*)
* *Corresponding author*，*E-mail*：*chenzhen*_1996@*qq.com*

Abstract：The target detection algorithm based on the convolutional neural network is developing rapidly，and with the increase in computational complexity，requirements for device performance and power consumption are increasing. To enable the target detection algorithm to be deployed on embedded devices，this study proposes a Yolo v3-SPP target detection system based on the ZYNQ platform by using a hardware and software co-design approach and hardware acceleration of the algorithm through FPGA. The system is deployed on the XCZU15EG chip，and the required power consumption，hardware resources，and performance of the system are analyzed. The network model to be deployed is first optimized and trained on the Pascal VOC 2007 dataset，and finally，the trained model is quantified and compiled using the Vitis AI tool to make it suitable for deployment on the ZYNQ platform. To select the best configuration scheme，the impact of each configuration on hardware resources and system performance is explored. The system power consumption (W)，detection speed (FPS)，mean value of average precision (mAP) for

收稿日期：2022-06-02；**修订日期**：2022-07-14.
基金项目：国家自然科学基金资助项目(No. 61671310)；辽宁省兴辽英才计划项目基金资助项目(No. XLYC1907134)；辽宁省教育厅项目资助(No. LJKZ0174)

each category, output error, etc. are also analyzed. The experimental results show that the detection speed is 38.44 FPS and 177 FPS for Yolo V3-SPP and Yolo V3-Tiny network structures, respectively, with mAPs of 80.35% and 68.55%, on-chip power consumption of 21.583 W, and board power consumption of 23.02 W at 300 M clock frequency and input image size of (416,416). This shows that the proposed target detection system meets the requirements of embedded devices for deploying neural network models with low power consumption, real-time, and high detection accuracy.

Key words: object detection; hardware acceleration; ZYNQ; Yolo v3-SPP; Yolo v3-Tiny

1 引　言

目标检测是计算机视觉领域的一个重要研究方向,近年来,基于卷积神经网络的目标检测算法取得了巨大突破,在自动驾驶、人脸识别、行人检测等领域都获得了广泛应用,文献[1]针对Yolo v3提出使用Generalized Intersection over Union(GIOU)计算损失、密集连接等方法,实现了2.11%的性能提升,文献[2]指出传统的目标检测算法对于小尺寸目标的检测效果较差,针对此问题,文献[3]通过不同通道特征图的叠加,引入空间注意力机制,增强了模型对于小目标的检测能力,文献[4]使用Mish激活函数替换ReLu激活函数,使用Complete Intersection over Union(CIoU)计算损失,实现1.88%的性能提升,然而随着检测性能的提升,目标检测算法计算冗杂、网络参数繁多、计算复杂度大幅增加,使得其只能在高性能计算机上运行,在实时性要求更高的应用场景中,传统的中央处理器的计算架构无法满足实时计算的需求,需要硬件加速器进行加速计算,降低延时,主流的解决方法之一是采用现场可编程逻辑门阵列(Field Programmable Gate Array, FPGA)提高运算速度,FPGA具有高实时性、低功耗以及并行处理等特点,使其能够完成多种情形的工作,具有较好的实用性和灵活性,适用于目标检测算法的各种应用场景。

文献[5]中,Wei等人实现了Yolo网络的加速,并将网络中的Leaky ReLu激活函数替换为ReLu激活函数以减少资源消耗,在ZYNQ7035中实现了19 FPS的性能;文献[6]提出了一种轻量级Yolo v2的实现方法,该方法通过使用二值化的特征提取网络减少了计算量与内存消耗,并使用支持向量机回归对物体进行分类,在XCZU9EG器件上实现了40.81 FPS的性能,该设计通过降低算法的复杂度成功地提高了速度;文献[7]对Yolo网络进行了优化,针对优化后的网络,使用AXI4总线封装了应用程序接口,并且使用ReLu激活函数替换Leaky ReLu激活函数,参数模型全部存储在片上存储器中以减少外部存储器的访问,实现了19.6 FPS的性能;文献[8]采用流水线架构,所部署的神经网络中的每一层均映射到专门的硬件模块,在Virtex XC7VX486T器件上实现了109 FPS的性能,该方案需要相当大的片上存储空间,对于一些中低端芯片,部署难度大。相对于Yolo v2而言,Yolo v3在检测精度方面有了巨大提升,同时也带来了更多的计算量,于是部分工作中采用Yolo v3-Tiny网络进行部署,文献[9]中通过将Yolo v3-Tiny网络中的特征图映射为矩阵并且将归一化层与卷积层合并以降低计算复杂度,在XCZU7EV器件上实现了8.3 FPS的性能;文献[10]使用Yolo v3-Tiny网络在XCZU9EG器件上实现了104 FPS的性能,但其未提到图片大小、可检测物体种类数

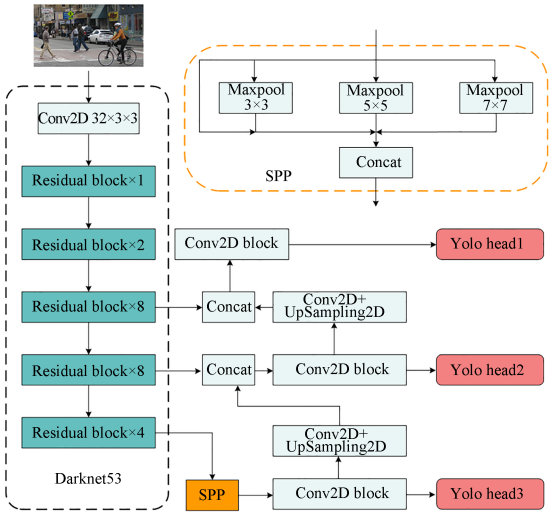


图1　Yolo v3-SPP网络结构

Fig. 1　Yolo v3-SPP network structure

以及资源消耗，以上工作需要针对网络进行特定的优化，灵活性低，对开发者硬件知识储备要求较高，对于非专业硬件开发人员而言上手难度大，且开发周期较长，难以适应快速迭代神经网络模型，因此需要一种普适性强，开发周期短，开发流程简洁的神经网络模型加速方法。

针对上述需求，为达到嵌入式设备部署神经网络时所需的低功耗、高检测准确度、实时性以及方便移植的目的，本文提出了一种目标检测网络模型在ZYNQ平台上的实现方法，该方法采用软硬件协同设计，使用ZYNQ芯片中的FPGA部分对算法进行硬件加速处理，实现了两个目标检测模型，分别是改进后的 Yolo v3-Tiny 与 Yolo v3-SPP。首先优化 Yolo v3-Tiny 与 Yolo v3-SPP 模型结构使其适用于ZYNQ端的部署，并对其进行训练，然后将训练好的模型进行量化，再对量化后的模型根据构建的硬件信息进行编译，得到可以在ZYNQ端执行的模型文件，最后编写程序调用该模型文件，达到硬件加速的目的。

2　目标检测算法介绍及优化

Yolo v1-v3[11-13]是一种被广泛使用的 one stage 目标检测架构，Yolo v3 网络模型在保证检测精度的同时兼顾了检测速度，由于该模型参数量较大，在使用ZYNQ芯片中的FPGA部分对算法进行加速时，无法直接将模型的全部参数存储于FPGA上有限的片上存储器中，且其参数类型为 float32，不适于FPGA等硬件设备进行计算加速，因此需要对其进行量化、编译等操作以使其适用于FPGA的部署，而 Yolo v3 模型在量化、编译之后会有较大的精度损失，为了弥补这个损失，本文采用 Yolo v3-SPP 网络模型进行部署，该模型相较于 Yolo v3 模型，主要区别在于特征提取网络之后加入了 Spatial Pyramid Pooling (SPP)模块，在该结构中分别使用大小为 1×1，5×5，9×9，13×13 的池化核进行最大池化处理，再将不同尺度的特征图在第一维度进行 concatenate 操作，得到 SPP 结构的输出。

SPP模块通过使用不同大小的池化核进行池化操作，增加了特征提取网络提取全局信息的能力，可以提高特征图的表达能力，提高模型的检测性能，而上述 SPP 结构无法直接应用于ZYNQ端的部署，因为该结构中包含大小为 9×9 与 13×13 的池化核，由 Xilinx 公司的产品指南[14]可知，FPGA 上支持的最大池化核尺寸为 8×8，一种解决方案是保持 SPP 结构不变，将模型的中间结果从FPGA传回至CPU处理，待CPU计算完成后再将结果传至FPGA完成之后的计算，这样的缺点是会造成额外的数据搬移，此时数据在CPU与FPGA之间的传输会成为系统整体性能的瓶颈，另一种方法是更改网络结构，使其适用于FPGA的部署从而避免不必要的数据搬移，提升系统性能。本文选择更改 SPP 结构，然后对模型进行训练，使用训练完成的模型进行部署，整体网络结构如图 1 所示，SPP 结构对运行速度的影响及模型性能测试见本文 4.2 节。

3　系统设计

3.1　系统结构设计

ZYNQ为Xilinx公司推出的一种异构加速平台，该平台由两部分构成，为 Processing System (PS)端与 Programmable Logic(PL)端，即FPGA部分。本文所用芯片属于 ZYNQ UltraScale+ MPSoCs 系列芯片，型号为 XCZU15EG-ffvb1156-2-i，所用编译环境为 Vivado 2021.2，Petalinux 2021.2，Vitis 2021.2 以及 Vitis AI 1.4.0。系统采用软硬协同实现目标检测功能，系统框图如图 2 所示，其中 PS 端挂载了一张 SD 卡，卡中烧录有 Linux 操作系统，系统中包含有 opencv、numpy 等常用 python 库，还包含 VART

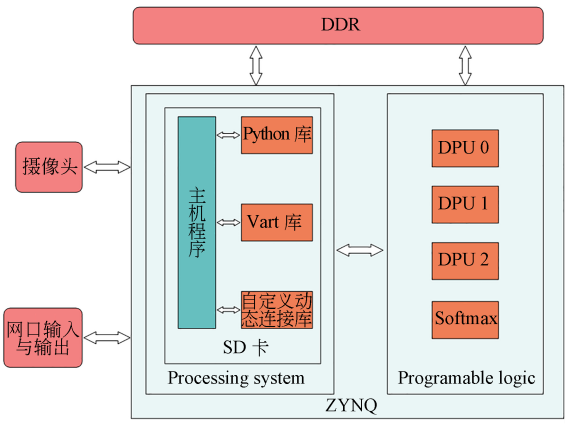


图 2　系统框图

Fig. 2　System block diagram

以及自定义动态链接库，摄像头负责获取图像数据，获取到的图像数据传入主机程序中进行处理，处理完成的数据通过AXI总线传输至PL端的DPU IP核中进行运算，PL端再将运算后的结果返回至PS端，PS端再对结果进行后处理，得到模型的输出结果，另外PS端还负责结果的储存与输出。

3.2 DPU IP核的设计

Deeplearning Process Unity（DPU）是Xilinx公司开发的一款IP核。该IP核在ZYQN中的PL端实现，可由Vitis AI编译器生成的指令驱动。在使用该IP核时需要对其进行配置，部分可配置参数如下，Arch，该参数用于配置DPU架构并行度，如B512，B4096，数字512或4 096代表每秒最大计算量（Ops）；RAM Usage，该参数用于配置Block RAM（BRAM）的使用模式，可配置为低BRAM使用与高BRAM使用；Number of DPU Cores，该参数用于配置一个DPU IP核中的内核个数，可配置为1～4个；DSP Usage，该参数用于配置DSP的使用模式，可配置为低DSP使用与高DSP使用；URAM Use per DPU，该参数用于配置Ultra RAM（URAM）的使用模式，可配置DPU中每个内核使用的URAM数量，各参数的配置以及对系统性能影响的测试见4.1节。

3.3 ZYNQ端可执行模型文件生成

ZYNQ端的可执行模型文件，即xmodel文件，基于Xilinx公司的Vitis AI工具生成，该工具中包含有Vitis AI量化器与Vitis AI编译器，其中，Vitis AI量化器负责将浮点型数据量化为定点数据，在量化过程中，需要输入校准图像数据；Vitis AI编译器负责根据DPU的配置参数将经Vitis AI量化器量化后的模型文件编译为xmodel文件。当DPU不支持模型中的某个运算时，Vitis AI编译器仍可编译该模型。在这种情况下，模型被分成几个部分，每个部分称为子图，DPU上无法执行的子图将被放在CPU上执行，此时，DPU执行完子图后需要将结果回传至CPU中进行相应子图的计算，DPU需要等待CPU计算完成，才可以进行下一子图的计算，这样会增加数据传输以及等待时间，影响计算效率。

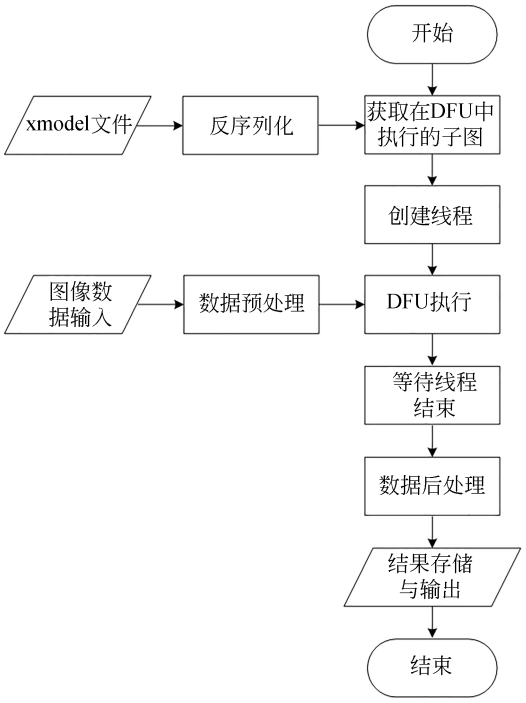


图3　主程序结构

Fig. 3　Main program structure

3.4 ZYNQ端执行程序设计

执行程序基于Python语言编写，采用多线程设计，程序结构如图3所示，通过ctypes库实现对c/c++的兼容以及动态链接库的调用，程序执行时首先加载xmodel文件并对xmodel文件进行反序列化，得到所有子图，然后获取各子图执行的位置，取出需要在DPU中执行的子图；然后通过opencv库中的函数接口获取摄像头输出的图片数据，对图片数据进行预处理，预处理包括数据归一化、数据缩放以及数据类型转换，数据缩放倍数由模型编译时确定。再将预处理后的数据送入DPU或CPU中完成对应子图的计算，得到网络输出结果，结果经处理后可以存储在SD卡中或传输给下一级。

4 性能测试与分析

4.1 DPU性能测试

为探究DPU IP核的内核数、架构并行度以及系统时钟频率对系统性能的影响，本文针对7种DPU配置方案进行了测试，测试使用改进后的Yolo v3-SPP模型，每种方案的配置方式如表1所示，其中方案1、方案3架构并行度相同，时钟

频率相同，内核数不同，用于探究时内核数对系统性能的影响；方案 2、方案 3 架构并行度相同，内核数相同，时钟频率不同，用于探究时钟频率对系统性能的影响；方案 4、方案 5 架构并行度相同，内核数相同，时钟频率不同，均使用相同数量的 URAM 代替部分 BRAM，用于探究使用 BRAM 情况下时钟频率对系统性能的影响；方案 6、方案 7 内核数相同，时钟频率相同，架构并行度不同，用于探究架构平行度对系统性能的影响。除上述参数外，每种方案的激活函数开启 Leaky-ReLU，ReLU，ReLU6 支持，开启 Softmax 支持、Depthwise Conv 支持、ElementWise Multiply 支持、AveragePool 支持，开启 channel augmentation，配置为低 RAM 使用率，高 DSP 使用率。

针对以上 7 种方案使用 1 000 张图片对系统进行性能测试，测试结果包括各类型资源占用率（%）、系统功耗（W）以及运行速度（FPS），结果如图 4 所示，对比方案 1 和方案 3，在架构并行度为 B4096，时钟频率为 300 MHz 的情况下，增加一个 DPU 内核，FPS 提升 51.68%，同时会增加约 96.68% 的功耗，各类别资源占用平均增加 17.25%；对比方案 2 和方案 3，在架构并行度为 B4096，内核数为 2 的情况下，时钟频率增加 50%，FPS 可提升 31.30%，同时会增加约 31.71% 的功耗，各类别资源占用平均减少 0.33%；对比方案 6 和方案 7，在内核数为 4，时钟频率为 300 MHz 的情况下，增加 1.25 倍的架构并行度，FPS 可提升 44.84%，同时会增加约 29.95% 的功耗，各类别资源占用平均增加 10.89%；对比方案 4 和方案 5，在架构并行度为 B4096，内核数为 3 且使用 URAM 的情况下，时钟频率增加 50%，FPS 可提升 35.59%，同时会增加约 22.33% 的功耗，各类别资源占用平均增加 5.34%；综上所述，考虑到板卡功耗，资源占用以及系统性能等因素，本文使用方案 5 进行部署。

表 1　DPU 配置方式

Tab. 1　DPU configuration method

方案	Arch	Number of Cores	Ultra RAM/Cores	Clock /MHz
1	B4096	1	0	300
2	B4096	2	0	200
3	B4096	2	0	300
4	B4096	3	28	200
5	B4096	3	28	300
6	B1152	4	0	300
7	B512	4	0	300

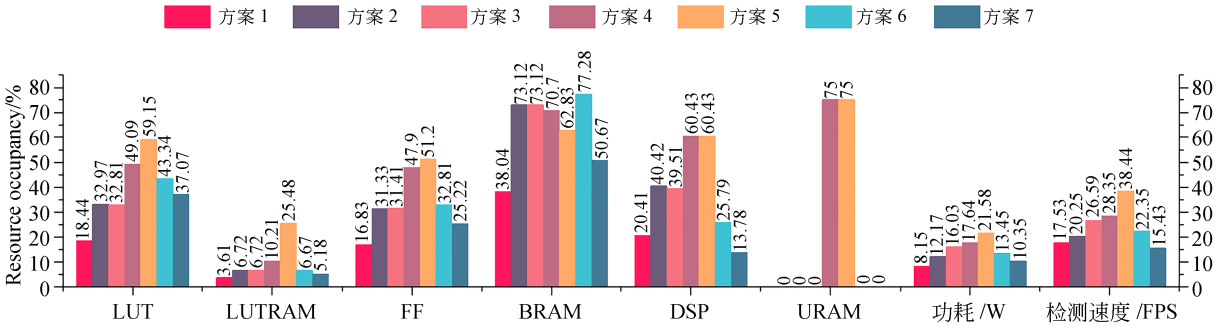


图 4　不同方案资源占用率及性能分析

Fig. 4　Analysis of resource occupancy of different schemes

4.2　SPP 结构性能测试

本文采用更改 SPP 结构的方式使得 Yolo v3-SPP 模型适用于 ZYNQ 端的部署，将原 SPP 结构的池化核大小由 1×1，5×5，9×9，13×13 改为 1×1，3×3，5×5，7×7，为探究更改后的 SPP 结构与原 SPP 结构对模型性能以及部署后模型运行速度的影响，使用不同的 SPP 结构进行了测试，分别对包含二者的模型采用相同的训练方案进行训练，使用 Pascal VOC 2007 测试集对二者进行性能测试，测试指标为 mAP，阈值取 0.5；另外在 ZYNQ 端使用 1 000 张图片分别对各模型进行推断速度的测试，结果如表 2 所示，其中 SPP-o

表示使用原SPP结构，SPP-n表示使用更改后的SPP结构，可知，更改后的SPP结构模型性能提升效果与原SPP结构相当；表中还列出了每个DPU内核推理一张图片所需的平均时间(avg)以及3个DPU内核推理一张图片所需的平均时间(total avg)，该时间由每个DPU内核所需的平均时间计算得到，从表中可以看出，增加了SPP结构以后，ZYNQ端的DPU内核平均推断速度由76.225 ms分别增加到76.391 ms与76.331 ms，平均推断耗时分别增加了0.166 ms与0.106 ms，增加的SPP结构对系统推断速度影响较小，因此，本文采用包含SPP-n结构的模型进行后续测试与部署。

表2　SPP结构对运行速度的影响

Tab. 2　Influence of SPP structure on running speed

SPP结构	内核	Avg/ms	Total avg/ms	mAP/%
NA	1	76.941	76.225	79.25
	2	74.056		
	3	77.679		
SPP-o	1	76.658	76.391	83.61
	2	74.866		
	3	77.651		
SPP-n	1	77.232	76.331	83.55
	2	74.228		
	3	77.534		

4.3　模型性能测试

为探究各模型之间的性能差异，本文分别在PC端与ZYNQ端共部署了6个神经网络模型，分别是部署在PC端的Yolo v3-SPP，Yolo v3，Yolo v3-Tiny以及量化后的Yolo v3-SPP模型，部署在ZYNQ端的Yolo v3-SPP与Yolo v3-Tiny模型，针对各类别的AP值对各模型进行了测试，阈值取0.5，使用Pascal VOC 2007数据集进行测试，共20个物体类别，测试结果如图5所示，图中展示出了不同模型的各类别AP值以及不同模型的结果在该类别处的差值；其中，Diff 1为部署在ZYNQ端的Yolo v3-SPP模型与部署在PC端的Yolo v3模型所得结果的差值，Diff 2为部署在PC端的Yolo v3-SPP模型与Yolo v3模型所得结果的差值，Diff 3为部署在PC端的Yolo v3-SPP模型量化后与量化前所得结果的差值，整体来看，模型经量化后其性能并未发生明显下降，符合量化要求。

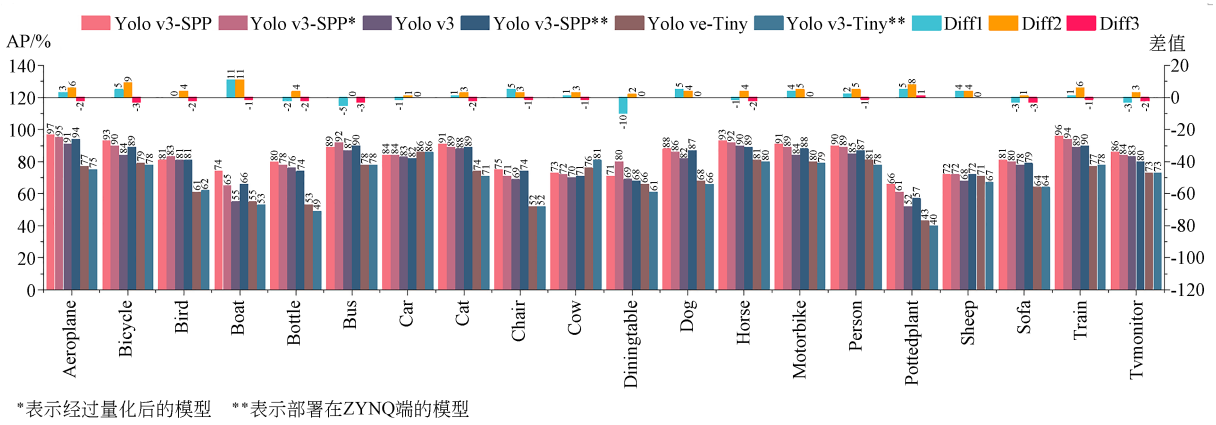


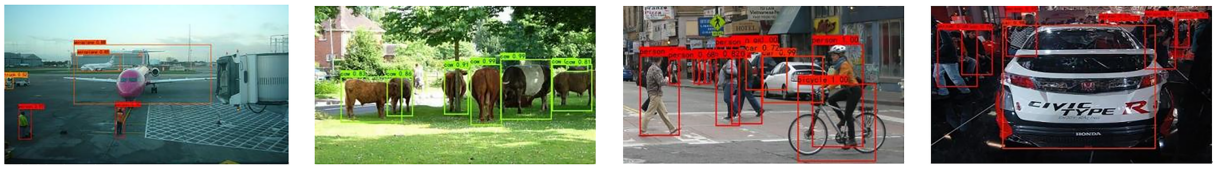
图5　各模型性能对比及差值

Fig. 5　Analysis of model performance after quantification

部署在PC端的Yolo v3-SPP模型的mAP为83.55%，量化后的Yolo v3-SPP模型的mAP为82.62%，部署在PC端的Yolo v3模型的mAP为79.25%，Yolo v3-Tiny模型的mAP为69.75%；部署在ZYNQ端的Yolo v3-SPP模型的mAP为80.35%，部署在ZYNQ端的Yolo v3-Tiny模型的mAP为68.55%，部署在ZYNQ端的Yolov3-SPP模型的mAP相较于PC端有3.2%的下降，但仍高于PC端的Yolo v3模型，整体来看，ZYNQ端的模型检测性

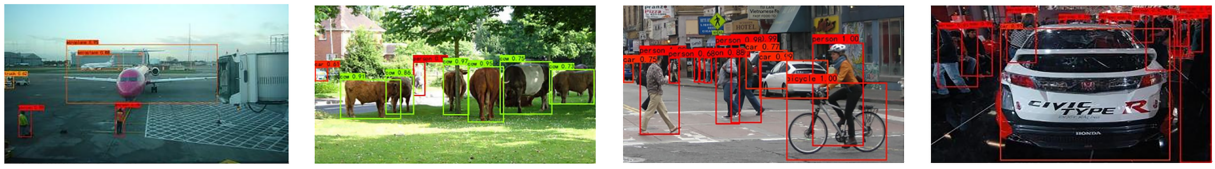
能发生了少许下降，在可接受范围之内，符合部署要求。

图 6 为ZYNQ端运行的Yolo v3-SPP模型输出结果与PC端运行的Yolo v3-SPP模型输出对比，其中(a)为ZYNQ端执行模型时输出的结果，(b)为PC端执行模型时输出的结果，在简单场景下，ZYNQ端的检测结果与PC端结果无明显区别，如图片A，复杂场景下，ZYNQ端对图片中小尺寸目标的检测能力略差于PC端，如图片B、图片C、图片D。



A　　B　　C　　D

(a) ZYNQ端输出结果
(a) ZYNQ terminal output result



A　　B　　C　　D

(b) PC端输出结果
(b) PC terminal output result

图 6　运行结果对比

Fig. 6　Comparison of running results

本文实现的Yolo v3-SPP模型可检测类别个数为80，模型的最终输出包含有3个输出特征层，每个输出特征层有255个维度，其计算方式为(80＋4＋1)×3，其中80为各类别对应概率，模型每个输出特征层中的每个特征点存在3个先验框，故进行乘3操作，每个先验框含有4个调整参数，另外还有1个参数表示该先验框内是否含有物体，图7展示了使用该模型对同一张图片进行推断时，每个输出特征层上ZYNQ端的输出结果、PC端的输出结果以及二者的差值，横坐标表示维度，纵坐标表示输出值的大小，其中黄色折线表示ZYNQ端的输出结果，蓝色折线表示PC端的输出结果，红色折线表示二者的差值；可以看出，体现二者差值的红色折线稳定在0值附近，表明ZYNQ端的输出结果与PC端的输出结果基本保持一致，满足实际部署要求(彩图见期刊电子版)。

另外，将本文的实验结果与其他文献进行了比较，主要比较指标为检测速度(FPS)、功耗

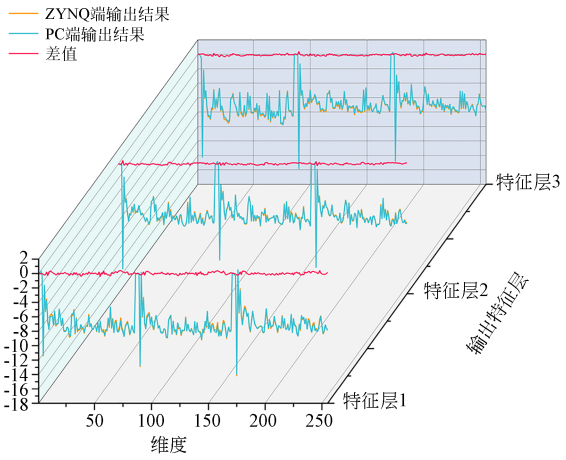


图 7　特征层输出对比

Fig. 7　Feature layer output comparison

(W)以及mAP(阈值取0.5)，测试使用Pascal VOC 2007数据集；结果如表3所示，可知，对于Yolo v3-Tiny模型，本文实现了最高的FPS，为177FPS，分别是文献[15]、文献[16]的57.65倍和51倍，mAP仅次于GPU和文献[9]，为68.55%，实现的Yolov3-SPP模型达到了除GPU

外了最高的mAP，为80.35%，较文献[15]、文献[9]、文献[6]分别有21.95%，5.35%和12.75%的提升，检测速度仅次于GPU和文献[6]，为38.44 FPS。

表3　与其他硬件平台的性能对比

Tab. 3　Performance comparison with other hardware platforms

平台	GPU			文献[15]	文献[9]	文献[6]	文献[16]	本文	
年份	2021			2021	2021	2018	2020	2021	
数据集	PascalVOC 2007								
器件	GTX1660Ti			Zedboard	Ultra96V2	XCZU9EG	XCZU9EG	XCZU15EG	
网络类型	Yolo v3-SPP	Yolo v3-Tiny	Yolo v3	Yolo v3-Tiny	Yolo v3-Tiny	Lightweight Yolo v2	Yolo v2	Yolo v3-Tiny	Yolo v3-SPP
数据类型	float32			fixed-16	fixed-8	binary	fixed-16	fixed-8	
图像大小	(416,416)			(416,416)	(416,416)	(224,224)	(416,416)	(416,416)	
FPS	48	258	52	3.07	8.3	40.81	3.47	**177**	**38.44**
功耗(W)	118			24.32	4.26	4.5	11.8	23.02	
mAP(%)	83.55	69.75	79.25	58.4	75	67.6	\	**68.55**	**80.35**

5　结　论

本文通过对Yolo v3-SPP网络结构进行优化，使其目标检测性能进一步提升的同时适用于ZYNQ端的部署，弥补了ZYNQ端部署模型时带来的性能损失，使用Vitis AI工具对预训练网络模型进行量化、编译，生成板卡运行所需的模型文件，采用多线程思想编写程序，充分利用板卡资源，完成了Yolo v3-Tiny和Yolo v3-SPP网络模型在ZYNQ端的部署，然后对系统进行了性能测试，测试结果表明，本文所实现的系统在保证检测速度和网络性能的前提下完成了模型在ZYNQ端的部署，对于Pascal VOC 2007数据集，图片输入大小为(416,416)，可检测类别数为80，分别实现了177 FPS和38.44 FPS的性能，mAP分别为68.55%和80.35%，优于以往部署在嵌入式设备的目标检测系统的设计，本文所提出的设计有望促进边缘设备实现实时目标检测。

参考文献：

[1] 唐悦，吴戈，朴燕. 改进的GDT-YOLOV3目标检测算法[J]. 液晶与显示，2020，35(8)：852-860.
TANG Y，WU G，PIAO Y. Improved algorithm of GDT-YOLOV3 image target detection[J]. *Chinese Journal of Liquid Crystals and Displays*，2020，35(8)：852-860.(in Chinese)

[2] 范丽丽，赵宏伟，赵浩宇，等. 基于深度卷积神经网络的目标检测研究综述[J]. 光学 精密工程，2020，28(5)：1152-1164.
FAN L L，ZHAO H W，ZHAO H Y，*et al*. Survey of target detection based on deep convolutional neural networks[J]. *Opt. Precision Eng.*，2020，28(5)：1152-1164.(in Chinese)

[3] 鞠默然，罗海波，刘广琦，等. 采用空间注意力机制的红外弱小目标检测网络[J]. 光学 精密工程，2021，29(4)：843-853.
JU M R，LUO H B，LIU G Q，*et al*. Infrared dim and small target detection network based on spatial attention mechanism[J]. *Opt. Precision Eng.*，2021，29(4)：843-853.(in Chinese)

[4] 王宸，张秀峰，刘超，等. 改进YOLOv3的轮毂焊缝缺陷检测[J]. 光学 精密工程，2021，29(8)：1942-1954.
WANG CH，ZHANG X F，LIU CH，*et al*. Detection method of wheel hub weld defects based on the improved YOLOv3[J]. *Opt. Precision Eng.*，2021，29(8)：1942-1954.(in Chinese)

[5] WEI G J，HOU Y Z，CUI Q M，*et al*. YOLO acceleration using FPGA architecture[C]. 2018 *IEEE/CIC International Conference on Communications in China* (*ICCC*). *August* 16-18，2018，*Beijing*，*China*. IEEE，2019：734-735.

[6] NAKAHARA H，YONEKAWA H，FUJII T，*et al*. A lightweight YOLOv2：a binarized CNN with

A parallel support vector regression for an FPGA [C]. *Proceedings of the* 2018 *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. February* 25 - 27, 2018, *Monterey*, *CALIFORNIA*, *USA*. *New York*: *ACM*, 2018: 31-40.

[7] LI Z G, WANG J T. An improved algorithm for deep learning YOLO network based on Xilinx ZYNQ FPGA [C]. 2020 *International Conference on Culture-oriented Science & Technology* (*ICCST*). *October* 28-31, 2020, *Beijing*, *China*. IEEE, 2020: 447-451.

[8] NGUYEN D T, NGUYEN T N, KIM H, *et al*. A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection [J]. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, 2019, 27(8): 1861-1873.

[9] ADIONO T, PUTRA A, SUTISNA N, *et al*. Low latency YOLOv3-tiny accelerator for low-cost FPGA using general matrix multiplication principle [C]. *IEEE Access*. *October* 15, 2021, IEEE, 2021: 141890-141913.

[10] OH S, YOU J H, KIM Y K. Implementation of compressed YOLOv3-tiny on FPGA-SoC [C]. 2020 *IEEE International Conference on Consumer Electronics-Asia* (*ICCE-Asia*). *November* 1-3, 2020, *Seoul*, *Korea* (*South*). IEEE, 2020: 1-4.

[11] REDMON J, DIVVALA S, GIRSHICK R, *et al*. You only look once: unified, real-time object detection [C]. 2016 *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*). *June* 27-30, 2016, *Las Vegas*, *NV*, *USA*. IEEE, 2016: 779-788.

[12] REDMON J, FARHADI A. YOLO9000: better, faster, stronger [C]. 2017 *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*). *July* 21-26, 2017, *Honolulu*, *HI*, *USA*. IEEE, 2017: 6517-6525.

[13] REDMON J, FARHADI A. YOLOv3: an incremental improvement [EB/OL]. 2018: *arXiv*: 1804.02767. https://arxiv.org/abs/1804.02767

[14] DPUCZDX8G for zynq ultraScale+ MpSoCs product guide PG338 (v3.4) [EB/OL]. Xilinx, [2022-01-20]. https://docs.xilinx.com/r/en-US/pg338-dpu? tocId=3xsG16y_QFTWvAJKHbisEw

[15] ZHANG H B, JIANG J Q, FU Y H, *et al*. Yolov3-tiny object detection SoC based on FPGA platform [C]. 2021 6*th International Conference on Integrated Circuits and Microsystems* (*ICICM*). *October* 22-24, 2021, *Nanjing*, *China*. IEEE, 2021: 291-294.

[16] ZHANG S, CAO J, ZHANG Q, *et al*. An FPGA-based reconfigurable CNN accelerator for YOLO [C]. 2020 *IEEE* 3*rd International Conference on Electronics Technology* (*ICET*). IEEE, 2020: 74-78.

作者简介:

张丽丽(1979—),女,黑龙江省讷河人,博士,副教授,硕士生导师,2002年、2005年、2012年于吉林大学分别获得学士、硕士、博士学位,主要从事FPGA系统设计及深度学习算法的研究。E-mail: 20052727@sau.edu.cn

通讯作者:

陈　真(1996—),男,河南商丘人,硕士研究生,2020年于沈阳航空航天大学获得学士学位,主要从事深度学习以及FPGA的算法研究。E-mail: chenzhen_1996@qq.com