Journal of Computer Applications
计算机应用,2021,41(3):669-676
ISSN 1001-9081
CODEN JYIIDU
2021-03-10
http://www.joca.cn

文章编号:1001-9081(2021)03-0669-08
DOI:10.11772/j.issn.1001-9081.2020060994

基于ZYNQ平台的YOLOv3压缩和加速

郭文旭[1],苏远歧[1*],刘跃虎[2]
(1.西安交通大学 电子与信息学部,西安 710049; 2.西安交通大学 人工智能学院,西安 710049)
(*通信作者电子邮箱yuanqisu@mail.xjtu.edu.cn)

摘 要:高精度物体检测网络急剧增加的参数和计算量使得它们很难在车辆和无人机等端侧设备上直接部署使用。针对这一问题,从网络压缩和计算加速两方面入手,提出了一种面向残差网络的新型压缩方案来实现YOLOv3的压缩,并通过ZYNQ平台对这一压缩后的网络进行加速。首先,提出了包括网络裁剪和网络量化两方面的网络压缩算法。网络裁剪方面,给出了针对残差结构的裁剪策略来将网络剪枝分为通道剪枝和残差链剪枝两个粒度,解决了通道剪枝无法应对残差连接的局限性,进一步降低了模型的参数量;网络量化方面,实现了一种基于相对熵的模拟量化方法,以通道为单位对参数进行量化,在线统计模型的参数分布与参数量化造成的信息损失,从而辅助选择最优量化策略来减少量化过程的精度损失。然后,在ZYNQ平台上设计并改进了8比特的卷积加速模块,从而优化了片上缓存结构并结合Winograd算法实现了压缩后YOLOv3的加速。实验结果表明,所提压缩算法较YOLOv3 tiny能够进一步降低模型尺寸,但检测精度提升了7个百分点;同时ZYNQ平台上的硬件加速方法获得了比其他平台更高的能耗比,从而推进了YOLOv3以及其他残差网络在ZYNQ端侧的实际部署。

关键词:物体检测;神经网络压缩;计算加速;网络剪枝;网络量化;ZYNQ平台
中图分类号:TP391.4 **文献标志码**:A

YOLOv3 compression and acceleration based on ZYNQ platform

GUO Wenxu[1], SU Yuanqi[1*], LIU Yuehu[2]
(1. *Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an Shaanxi* 710049, *China*;
2. *College of Artificial Intelligence, Xi'an Jiaotong University, Xi'an Shaanxi* 710049, *China*)

Abstract: The object detection networks with high accuracy are hard to be directly deployed on end-devices such as vehicles and drones due to their significant increase of parameters and computational cost. In order to solve the problem, by considering network compression and computation acceleration, a new compression scheme for residual networks was proposed to compress YOLOv3 (You Only Look Once v3), and this compressed network was then accelerated on ZYNQ platform. Firstly, a network compression algorithm containing both network pruning and network quantization was proposed. In the aspect of network pruning, a strategy for residual structure was introduced to divide the network pruning into two granularities: channel pruning and residual connection pruning, which overcame the limitations of the channel pruning on residual connections and further reduced the parameter number of the model. In the aspect of network quantization, a relative entropy-based simulated quantization was utilized to quantize the parameters channel by channel, and perform the online statistics of the parameter distribution and the information loss caused by the parameter quantization, so as to assist to choose the best quantization strategy to reduce the precision loss during the quantization process. Secondly, the 8-bit convolution acceleration module was designed and optimized on ZYNQ platform, which optimized the on-chip cache structure and accelerate the compressed YOLOv3 with combining the Winograd algorithm. Experimental results show that the proposed solution can achieve smaller model scale and higher accuracy (7 percent points increased) compared to YOLOv3 tiny. Meanwhile, the hardware acceleration method on ZYNQ platform achieves higher energy efficiency ratio than other platforms, thus helping the actual deployment of YOLOv3 and other residual networks on the end sides of ZYNQ.

Key words: object detection; neural network compression; computation acceleration; network pruning; network quantization; ZYNQ platform

0 引言

随着人工智能和云计算技术的快速发展,物体检测技术逐渐应用到人们生活的方方面面。例如,车辆检测在交通运输和安防[1]、自动驾驶[2]等领域有着广泛的应用。近年来卷积神经网络在物体检测领域取得了许多重大的进展,新的网络结构不断涌现,如R-CNN(Region-based Convolutional Neural Network)[3]、Faster-RCNN(Faster Region-based Convolutional Neural Network)[4]、SSD(Single Shot multibox Detection)[5]、

收稿日期:2020-07-09;**修回日期**:2020-11-12;**录用日期**:2020-11-18。 **基金项目**:国家自然科学基金资助项目(61973245)。
作者简介:郭文旭(1996—),男,陕西杨凌人,硕士研究生,主要研究方向:计算机视觉、深度学习; 苏远歧(1982—),男,江苏如皋人,讲师,博士,CCF会员,主要研究方向:计算机视觉、模式识别; 刘跃虎(1962—),男,山西河曲人,教授,博士,主要研究方向:计算机视觉、模式识别。

YOLO(You Only Look Once)[6]等。这些新型的网络不断提升物体检测的精度,但它们的参数和计算量也在急剧升高。受制于计算、访存、功耗等诸多因素,通常很难将它们直接应用到车辆和无人机等移动嵌入式环境中。而将图像上传到云端进行检测则需要高速稳定的无线网络支持,系统的可用性直接受制于通信的质量[7],而现有的网络环境很难支撑这一方案。因此,神经网络的参数量和计算量逐渐成为衡量算法性能的重要指标,它们决定了算法能否部署于端侧。

为了解决基于神经网络的物体检测技术在端设备实际应用中的计算负荷和性能问题,现有方法主要采用两类思路:一类是从网络设计开始,采用例如可分离卷积或者组卷积等技术来降低模型的参数,例如旷视的 ShuffleNet[8]和 Google 提出的 MobileNet 系列[9-11];另一类是通过对已有网络进行模型压缩,从而降低网络的参数与计算量。本文属于第二类,从模型压缩[12-25]和计算加速[26-33]两方面入手研究 YOLOv3(You Only Look Once v3)[34]在端侧设备上的部署。Cheng 等[35]对这两者进行了综述,指出模型压缩的主要手段包括网络裁剪[12-17]、量化[18-22]和模型蒸馏[23-25]等。网络裁剪[12-17]通过分辨网络中各个部分参数的重要程度调整网络结构,剔除不重要的参数。网络量化[18-22]从数据表示层面对神经网络进行压缩。神经网络的参数通常只占用浮点数范围较小的一部分,因此使用浮点数表示神经网络参数存在很大的冗余。同时与整型运算相比,浮点数运算会消耗更多的硬件资源和计算时钟。网络蒸馏[23-25]在预训的教师模型基础上使用简单的学生模型来学习老师模型中的信息,从而减少模型的参数数量。

计算加速[26-33]方面主要结合硬件资源和神经网络的计算过程来进行模型加速:一方面根据计算的过程,可以适当地使用硬件特有的协处理器和单指令多数据流指令集来增加计算吞吐量[26];另一方面可以结合现场可编程逻辑门阵列(Field Programmable Gate Array, FPGA)[27-29]和专用集成电路(Application Specific Integrated Circuit, ASIC)[30]等硬件资源结合计算过程的数据流设计相适应的计算模块,提升计算过程的数据吞吐量。

本文从神经网络压缩和加速两个方面同时入手对 YOLOv3 网络[34]在端侧的部署进行研究,所提压缩方法可以推广到一般具有残差连接的神经网络。本文的主要工作有:1)提出了一种针对残差网络的模型压缩方法;2)设计实现了与 ZYNQ 平台相适配的计算流程,实现了计算的加速。本文所提的压缩方法包括网络裁剪和量化两个方面,首先提出了一种针对残差网络的裁剪策略,将网络剪枝分为通道剪枝和残差链剪枝两个粒度,解决了通道剪枝无法裁剪残差连接的局限性。实验结果表明,该方法与通道剪枝相比能够进一步减少模型的参数和计算量。网络量化方面,本文引入并改进了一种基于相对熵 KL 散度(Kullback-Leibler Divergence)[36]的量化方法,能够在训练过程中统计模型的参数分布并在训练过程中计算量化造成的信息损失,从而优化量化区间减少精度损失。实验表明本文的压缩方法能够进一步降低模型量化过程造成的精度损失。

在计算加速方面,本文结合一种现有的 FPGA 加速模块 ZynqNet[29]设计并改进了针对 Xilinx 公司 ZYNQ 平台的 8 bit 卷积加速模块,通过使用 8 bit 整型数据减少了片上存储和计算资源的消耗;然后,依据算法流程优化了片上缓存结构和流水线,进一步提升了硬件模块的计算性能;最后,结合 Winograd[31]共享卷积计算的中间结果大量减少了计算资源的消耗。实验结果表明,本文的方案能够极大地提高了硬件计算卷积操作的性能,实现了比处理器(Central Processing Unit, CPU)和图形处理器(Graphics Processing Unit, GPU)等设备更高的平均能耗比,单位功耗下的计算能力更强。

1 相关工作

近年来卷积神经网络[1-6,34]在物体检测等领域取得了非常好的效果,解决了传统图像检测算法特征处理过程复杂、场景局限性强等问题,逐渐成为解决物体检测问题的主流方法。研究表明,一些神经网络通过极少的模型参数也能取得不错的检测精度,这表明深度神经网络普遍存在过量的参数和大量的冗余计算,随后研究人员开始关注神经网络的压缩和冗余计算的裁剪。另一方面,体系结构和硬件设备加工工艺的发展,也使得 ASIC、FPGA 等平台的计算性能不断提升,如何结合硬件平台和神经网络计算过程也是一个重要的研究课题。接下来,将从三个方面简要综述本文涉及的技术和方法。

1.1 模型剪枝

在模型剪枝方面,Han 等[13]在 2015 年首次提出将网络的连接性和权重分开训练,经过多次迭代后逐步裁剪模型尺寸。随后 Li 等[14]在 2016 年提出一种针对卷积核的神经网络剪枝方法,在每个卷积核中通过 l_1 范数度量卷积核对网络贡献度,并根据贡献度对卷积核进行裁剪。针对裁剪过程信息损失大的问题,2017 年 He 等[15]提出了一种基于 Lasso 回归的通道稀疏化剪枝策略,通过回归和优化实现了与直接裁剪相比更低的信息损失。Liu 等[16-17]提出并改进了一种基于稀疏化训练和通道剪枝的模型裁剪方法,通过在训练过程中添加稀疏惩罚系数使得最终的模型信息在通道层面更加集中,最后根据批归一化层的 γ 参数裁剪通道,在训练时极大地减少了模型的信息损失和参数冗余。

1.2 模型量化

在模型量化方面,2015 年 Han 等[18]提出一种基于共享权值的压缩方法,通过对训练好的模型的权重进行聚类,每个类别共享同一个数值,而后使用低比特数据索引权重,减少了参数体积。2016 年 Courbariaux 等[19]提出一种参数仅由-1、1 构成的二值神经网络,直接训练不可微的量化模型,在简单的识别任务中取得了不错的结果。Rastegari 等[20]提出的 XNOR-Net 将特征和卷积的权重全部二值化,通过二值逻辑进行快速推理,也获得了类似的结果。文献[21]和文献[22]的工作均提出了一种基于整数的神经网络量化和训练方法。其中文献[21]在网络训练中通过浮点数模拟整数量化过程,实现了通用的量化模型训练方法,解决了量化模型不能直接训练的问题,在减少模型体积的同时充分利用了成本更低的整数运算。

1.3 计算加速

在卷积运算加速方面,Vasilache 等[30]在 2014 年提出了一种通过离散傅里叶变换的卷积加速方法,通过快速傅里叶变换算法将特征图和权重转换为频域表示并利用乘法操作实现卷积,极大地减少了大卷积核计算的计算量。针对小卷积核计算的加速,Lavin 等[31]提出一种基于 Winograd 变换的卷积计算加速方法,通过对特征图和权重以及输出做 Winograd 变换将卷积转换为点乘操作,在增加少量加法运算的同时极大地减少了卷积运算的乘法运算,优化了计算的性能。Liu 等[32]挖掘特征稀疏性,提出了稀疏 Winograd 算法,进一步加速卷积

的运算。Ioffe 等[33]提出了批归一化的折叠方法，在模型推理时将归一化操作与卷积层融合，减少推理过程不必要的计算。

2　改进的网络压缩策略

本章以YOLOv3网络[34]为例对残差神经网络的压缩进行研究，并给出针对残差网络的压缩方法。YOLOv3使用的特征提取网络是Darknet-53，它由五个串联的残差链[37]模块组成，在特征提取的基础上进行多尺度的网格划分，结合Anchor机制直接对目标的位置和类别进行回归预测。本文研究包括网络裁剪和量化两个部分。在裁剪方面，本文提出一种针对残差网路的裁剪策略，相较传统剪枝进一步减少了YOLOv3网络的计算量。在网络量化方面，本文提出了一种基于KL散度的模拟量化策略，能够在训练过程中统计参数的分布，并在训练过程中模拟量化造成的信息损失，从而减少量化过程的精度损失。

2.1　模型剪枝

首先对网络进行稀疏化训练，如式(1)所示，在每个卷积层反向传播过程中添加稀疏化惩罚项，使得训练过程中稀疏通道不断增多，稀疏化训练过程由参数λ控制：

$$L(\boldsymbol{W})=\sum_{(I,y)}C\left(\mathrm{Net}\left(\boldsymbol{I};\ \boldsymbol{W}\right),\boldsymbol{y}\right)+\lambda\sum_{\gamma}\left\|\boldsymbol{\gamma}\right\|_1 \tag{1}$$

其中：Net代表YOLOv3神经网络；$\boldsymbol{W}$是网络的参数；$\boldsymbol{I}$是输入的图像；$\boldsymbol{y}$是针对该图像的真实物体标签；C度量网络的输出和真实标签之间的损失；$\boldsymbol{\gamma}$是神经网络中所有批归一化层的γ参数；$\|\cdot\|_1$对应l_1范数。在稀疏化训练中，由于存在稀疏化惩罚项，训练过程将会产生大量的稀疏通道，稀疏通道的数值趋近于零，批归一化中的γ系数决定了该输出通道对后续计算的重要程度。通过分位数筛选掉γ较小的通道即可完成稀疏化通道剪枝。

如图1所示，左侧为第i层卷积的输出通道，批归一化层经过稀疏化训练可以得到一组权重，其中权重相对较小的通道可以视为冗余通道。图中虚线代表对应的特征图通道将会被裁剪，卷积核中的对应通道以及对应批归一化参数可以直接从模型参数中移除。

$$\boldsymbol{F}_{oc}^{i+1}=\sum_{ic}\gamma_{ic}^{i}\left(\frac{\boldsymbol{F}_{ic}^{i}-\mu_{ic}}{\sqrt{\sigma_{ic}^{2}+\varepsilon^{2}}}\otimes\boldsymbol{W}_{ic,oc}^{i}\right) \tag{2}$$

其中：ic, oc分别是输入、输出通道的索引；$\boldsymbol{F}$是高维特征图，$\boldsymbol{W}$是卷积神经网络的参数，这里特指第i到第$i+1$层的卷积核；μ_{ic}和σ_{ic}^2分别是针对输入特征的批归一化处理的均值和方差；ε^2是一个较小的数值，防止产生分母为0的情况；γ_{ic}^{i}是批归一化的γ系数，$\otimes$是卷积操作。

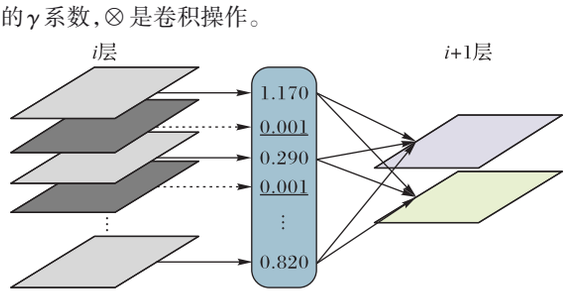


图1　稀疏化通道裁剪示意图

Fig. 1　Schematic diagram of sparse channel pruning

对于带有残差连接的神经网络，由于需要保证残差连接的结构，残差连接处的输入和输出通道需要保持一致，无法直接进行裁剪。Darknet-53中每个残差块由一个1×1的卷积层和一个3×3的卷积层组成。如图2所示，1×1和3×3卷积层之间可以直接进行裁剪，这是通道内的裁剪。如果直接对3×3的输出通道进行裁剪，可能会使得它与图2右侧残差连接的通道数不一致，因此无法直接进行。

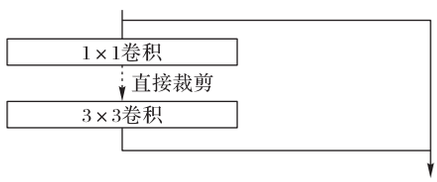


图2　残差连接中的通道剪枝

Fig. 2　Channel pruning in residual connection

虽然通道裁剪能够一定程度上减少网络的冗余参数，但残差连接中的通道仍然有许多冗余。通过本文提出的残差链剪枝，能够在通道剪枝的基础上进一步减少模型的参数量。在残差网络中，残差链可以看成是多个残差块输出的累加，因此残差链的输出通道的权重取决于每一接入残差块输出通道权重的叠加，对残差连接的剪枝需要对最终各个残差块输出的γ系数求和作为评估残差链通道权重的依据。如图3所示，本文的裁剪分为通道剪枝和残差剪枝两个粒度。

通道剪枝的对象为左侧1×1卷积的输出通道和3×3卷积的输入通道。残差链粒度包括右侧残差连接的输出通道和左侧3×3卷积的输出通道。残差剪枝需要协作各个残差块的输出通道，因此比通道剪枝更为困难。在实际裁剪过程中首先进行通道粒度的裁剪，而后在残差链粒度对每个残差链的通道数进行裁剪。在裁剪过程中，一次裁剪掉大量的通道通常会造成严重的精度损失。本文采用多次迭代进行稀疏化剪枝、剪枝后引入精调来保证精度。

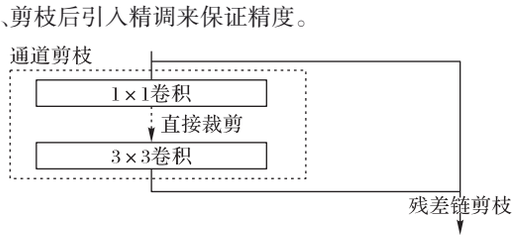


图3　通道剪枝与残差链接剪枝示意图

Fig. 3　Schematic diagram of channel pruning and residual connection pruning

2.2　模型量化

模型裁剪从结构上优化网络的冗余量，网络量化则从数据表示方面着手，文中将神经网络参数量化为8比特的整型。通过神经网络参数分布可知，相对于浮点数的表示范围，卷积神经网络的输出通常局限在非常小的范围内，因此神经网络的数据表示中也存在冗余。通过将浮点数映射到低比特的整数，一方面能够使用低成本的整数计算单元，同时减少模型计算的访存总量。对称量化的计算过程如式(3)所示：

$$w_{\mathrm{sym}}=\mathrm{round}\left(w/\Delta\right) \tag{3}$$

其中Δ为量化步长。对称量化中如果Δ是根据最大值以及量化位数确定的，会导致式(3)中Δ数值过大，使得更多的浮点数值落入同一个整数值，存在着较大的信息损失。

借助KL散度可以选择出合适的Δ值使得最终的信息失真最小。KL散度的定义如下：

$$\mathrm{KL}(P\|Q)=\mathrm{E}_{x\sim P}\left[\mathrm{lb}\left(\frac{P(x)}{Q(x)}\right)\right] \tag{4}$$

其中P和Q分别为量化前后参数的分布。

本文提出一种结合KL散度[36]和模拟量化的模型量化策略，能够在模型训练过程结合KL散度进行量化，在训练过程中将量化造成的信息损失考虑在内，减少精度损失。首先，在训练过程中统计网络每层参数的l_1范数的最大值$h_{\max}$和参数分布的直方图$\boldsymbol{h}$，其中，直方图将区间$[0,h_{\max}]$划分成2 048等份并统计落在各区间参数的个数。如果在迭代过程中遇到更大的上限$h_{\max}$，则重新统计直方图，迭代一定轮数并获得稳定的直方图作为P。

假设$\boldsymbol{b}$是上述量化的中心点，量化需要确定一个区间$[0,b_j]$其中$j\in[129,2\,048]$，根据8比特量化的需要，将这个区间划分成128个小区间（8比特量化的区间为$[-127,128]$，为了简便认为负区间也量化为128个等级，事实上仅有127个等级）。

量化区间的选择带来了失真，如果j选择过大，存在量化步长过大带来KL散度增大；但是如果j选择较小，存在丢失l_1范数值较大的参数，也会带来KL散度的增大。因此，j的选择应该是一个综合了步长和区间的最佳值。求解最优的j^*使得量化后Q的分布与P分布的KL散度最小，同时使得精度满足一定要求。计算出对应的量化区间的上限$H_{128}=(h_{\max}\times j^*)/2\,048$，式(3)中的步长则可以由这个上限确定。

在训练过程中，分别对卷积权重的各个输出通道以及卷积层输出的激活值使用上述的模拟量化算法，在训练过程中统计数值分布并在浮点数上模拟量化造成的精度损失。由于使用了KL散度量化，统计出的H_{128}值通常小于统计值的最大范围，而且随着模型的训练，统计值的范围也在不断变化，因此对于超过界限的数值不能限定量化范围，否则会使模型无法收敛，训练过程中动态更新这个上界。

3 基于ZYNQ的加速模块

3.1 硬件平台环境

ZYNQ平台是Xilinx公司推出的低功耗嵌入式平台，包括PS（Processing System）和PL（Program Logic）两个部分。PS部分主要由处理器构成，通过编写程序进行控制。PL部分由可编程逻辑电路组成，通过编写硬件逻辑实现计算和控制。PS部分和PL部分通过AXI（Advanced eXtensible Interface）总线进行交互，包括通常用作控制总线的低速总线AXI-GP和用作高速数据传输的AXI-HP总线。AXI-GP总线通常由PS端作为主设备控制外设运行，AXI-HP总线由PL端作为主设备，可以直接访问动态随机存取存储器（Dynamic Random Access Memory，DRAM）和一致性缓存，在大量数据通过一致性缓存时会极大降低PS端的运行速度。

本文设计的系统结构如图4所示，右上角为ZYNQ处理平台，与DDR2（Double Data Rate 2）和处理器相连接；中间部分上面是AXI-HP高速总线的交叉开关矩阵（Crossbar），负责处理加速模块的内存数据请求；下面是系统的时钟和复位信号控制模块，负责PL端的时钟分发和复位，系统PL端的工作频率是200 MHz；左侧上边是FPGA加速模块，主要使用ZYNQ平台的DSP48E和Block RAM实现计算加速和片上缓存。由低速总线负责卷积参数的输入和运行状态的控制。下面是低速总线的Crossbar，负责下发PS端的控制指令和参数。

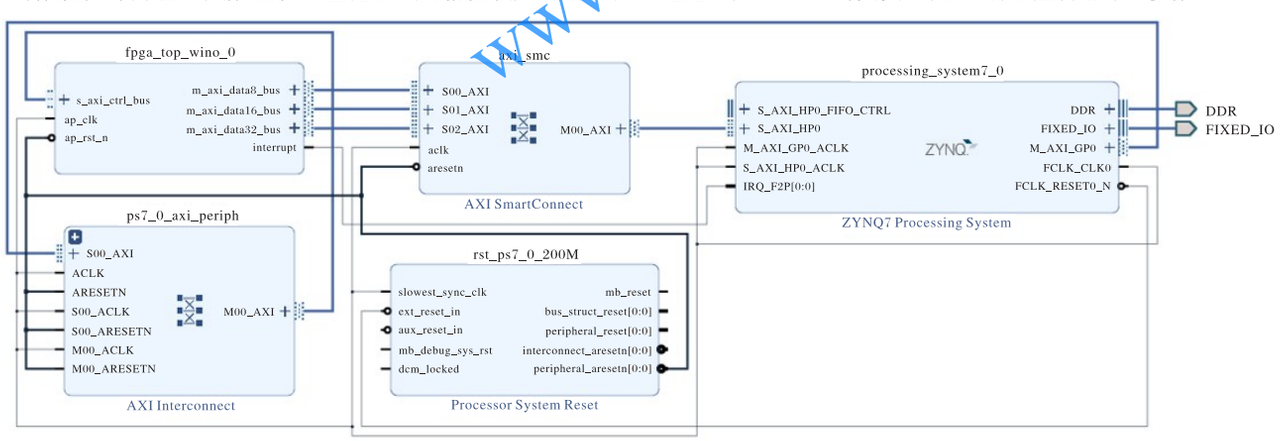


图4 系统结构

Fig. 4 System architecture

3.2 计算模块设计

计算模块的设计是在ZynqNet[29]的基础上修改的，ZynqNet是在ZC7x045平台上实现的基于GoogleNet[38]的手写数字识别的浮点计算加速模块。本文在ZynqNet[29]的基础上删除与YOLOv3[34]计算无关的全局池化（Global Pooling）、存储控制（Memory Controller）等模块、使用8 bit整数作为推理的数据类型并使用AXI-Burst来提高硬件在读写DRAM时的性能。在ZC7x020平台上重新设计并优化了卷积计算模块，优化策略主要参考文献[39]。

如图5所示，本文的设计模块包括配置面板（Configure Board）、输入缓存（Inputs Cache）、权重缓存（Weights Cache）、输出缓存（Outputs Buffer）、处理单元（Processing Element）和后处理（Post-process）等几个模块，下面分别对各个模块的划分进行介绍。

配置面板：模块存储卷积计算的各个参数，包括输入特征图的长宽、输入输出通道数、卷积核大小、卷积核移动步长、是否使用Leaky RELU[40]激活函数。

输入缓存：由16个块状随机存取存储器（Block Random Access Memory，Block RAM）并联实现，按照高度、宽度、通道的顺序存储输入特征图。16个Block RAM中每4个存储输入特征图的一行，按照输入特征图的行编号余4选择Block RAM。16个Block RAM可以同时存储输入特征图的4行。在

列上面也是按照特征图的列编号余4选取存储器。在读取特征图时允许最多对4×4个端口同时读取数据。在一个时钟周期内即可读取计算所需的特征图,减少了特征图读取的延迟。

权重缓存:由 N_PE×3×3 个 Block RAM 并联实现,按照输入通道、输出通道、卷积核高度、卷积核宽度的维度次序存储特征图,用以缓存处理单元所需要的权重系数。输出通道按照余 N_PE 的次序依次映射到各组 Block RAM 上。每个处理单元和9个 Block RAM 相关联,从而允许计算单元在一个周期内读取所需的权重数据。

输出缓存:由 N_PE 个 Block RAM 构成,保证每个处理单元计算之前能够读取输出通道的数值并在计算后将累加的结果写回存储中。

处理单元:由9个数字信号处理器(Digital Signal Processor, DSP)组成,负责对输入的特征图和权重进行乘加操作,并将最终结果累加到输出缓存中。由于本文平台资源的限制,这里选定 N_PE=16。

后处理:模块负责在写回输出特征图通道数据之前对输出进行批归一化、激活和重量化操作。在完成后将结果写回内存。

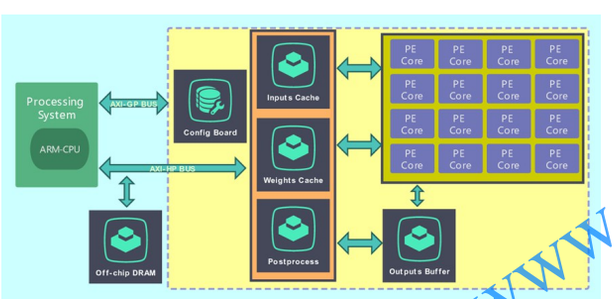


图5　加速模块结构

Fig. 5　Acceleration module architecture

3.3　算法流程

本文模块结构根据YOLOv3卷积层的计算过程实现,算法1描述了卷积模块的计算过程。从中可知,特征图的存储形式是 $\boldsymbol{F}[ih, iw, ic]$,其中 ih、iw 和 ic 分别是对特征图像高度、宽度和通道数的索引;权重的存储形式是 $\boldsymbol{W}[ic, oc, fh, fw]$,其中 ic、oc、fh 和 fw 分别是对卷积核输入通道、输出通道、高度和宽度方向进行索引。这里重新调整了输入通道的位置改变了加载数据的顺序,同时增强了数据访问的局部性,在计算过程中充分利用模块的I/O端口。

算法1　朴素卷积算法。

输入:$\boldsymbol{F}, \boldsymbol{W}, M, N, IC, OC$;

输出:$feature\_map$。

1) for $h = 0 \rightarrow M-1$ do
2)　for $w = 0 \rightarrow N-1$ do
3)　　for $ic = 0 \rightarrow IC-1$ do
4)　　$ih=h-1$, $iw=w-1$
5)　　$\boldsymbol{In}=\boldsymbol{F}[ih:ih+2, iw:iw+2, ic]$　　//读取输入特征图
6)　　for $oc=0 \rightarrow OC-1$ do
7)　　$\boldsymbol{Wt}=\boldsymbol{W}[ic, oc, 0:2, 0:2]$　　//读取卷积权重
8)　　$feature\_map[h, w, oc]$+=Conv($\boldsymbol{In}, \boldsymbol{Wt}$)　　//计算卷积
9)　　end for
10)　end for
11)　for $oc = 0 \rightarrow OC-1$ do
12)　　$feature\_map[h, w, oc]$+=Postprocess($feature\_map[h, w, oc]$)
13)　　WriteToDRAM($feature\_map[h, w, oc]$)
14)　end for

经过优化,本文实现比从ZynqNet移植的基准实现有了明显的性能提升。在验证算法计算结果正确后,本文结合Winograd算法对模块结构进行调整,进一步增强硬件性能。

如算法2所示,矩阵 $\boldsymbol{G}$、$\boldsymbol{B}$ 和 $\boldsymbol{A}$ 分别是权重、输入和点积结果的基变换矩阵[31]。借助Winograd变换F(2,3)能够在增加一些加法运算的同时将乘法操作从9个减少到4个。针对现有的设计结构,本文调整了权重缓存 $\boldsymbol{R}$ 的形状,使用 16×N_PE 个 Block RAM 存储 $\boldsymbol{GRG}^{\mathrm{T}}$ 变换后的权重。在读取输入 $\boldsymbol{S}$ 后进行 $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{SB}$ 变换,由于硬件资源的限制,将 N_PE 从16减小到8,在点乘输出 $\boldsymbol{T}$ 后进行 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{TA}$ 变换,能够在增加少量加法计算资源同时的节省3/4的输出缓存大小。

算法2　Winograd F(2,3)卷积算法。

输入:$\boldsymbol{F}, \boldsymbol{W}, M, N, IC, OC$;

输出:$feature\_map$。

1) for $ic = 0 \rightarrow IC-1$ do
2)　for $oc = 0 \rightarrow OC-1$ do
3)　　$\boldsymbol{IW}[ic, oc, 0:3, 0:3] = \boldsymbol{G}*\boldsymbol{W}[ic, oc, 0:2, 0:2]*\boldsymbol{G}^{\mathrm{T}}$　//转换权重
4)　end for
5) end for
6) for $h = 0 \rightarrow M/2$ do
7)　for $w = 0 \rightarrow N/2$ do
8)　　for $ic = 0 \rightarrow IC-1$ do
9)　　$ih = 2*h-1$, $iw=2*w-1$
10)　　$\boldsymbol{In} = \boldsymbol{B}^{\mathrm{T}} * \boldsymbol{F}[ih:ih+3, iw:iw+3, ic]*\boldsymbol{B}$　　//转换特征图
11)　　for $oc = 0 \rightarrow OC-1$ do
12)　　$\boldsymbol{Wt} = \boldsymbol{IW}[ic, oc, 0:3, 0:3]$
13)　　$feature\_map[h:h+1, w:w+1, oc]$+=$\boldsymbol{A}^{\mathrm{T}}*(\boldsymbol{In}\odot\boldsymbol{Wt})*\boldsymbol{A}$
14)　　end for
15)　end for
16)　for $oc = 0 \rightarrow OC-1$ do
17)　　Postprocess($feature\_map[2*h:2*h+1, 2*w:2*w+1, oc]$)
18)　　WriteToDRAM($feature\_map[2*h:2*h+1, 2*w:2*w+1, oc]$)
19)　end for
20) end for
21) end for

4　实验与结果分析

本文用KITTI(Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago)[41]数据集中的车辆和行人两类数据对上述的压缩算法进行了实验验证。训练数据集中包含了3 700张大小相近图片,测试数据集中包含了3 800张图片,主要包括行车和道路场景下的车辆、行人等物体的标注信息。图6中是数据集中的样例图片。在训练之前首先要按照PASCAL VOC (Pattern Analysis, Statistical modelling, And Computational Learning Visual Object Classes challenge)的标注格式对数据集的标注进行转换。

1) 模型裁剪结果对比。

模型剪枝实验首先进行稀疏化训练,训练使用1E-4的学习率和1E-5的稀疏化惩罚系数,再分别进行通道剪枝和残差链剪枝,最后进行精调。表1中对不同的裁剪方法进行了对比,由表1可知,模型裁剪能够极大地减少模型的计算量,而本文提出的方法能够进一步地减少模型计算量。这里使用平

均精度(Average Precision, AP)度量模型性能。由表 1 可知,与直接通道剪枝相比,本文的方法能够在略微降低精度的情况下进一步减小模型的尺寸和计算量。

图 6 KITTI 数据集示例数据

Fig. 6 KITTI dataset sample

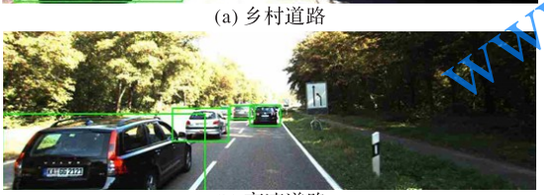
表 1 模型裁剪结果对比

Tab. 1 Comparison of model pruning results

裁剪方法	模型尺寸/MB	模型精度 AP
原始模型	230	0.74
通道剪枝 85%	61	0.70
通道剪枝 90%	47	0.68
通道剪枝+残差链剪枝 85%	21	0.68
通道剪枝+残差链剪枝 90%	18	0.66

图 7 给出一些通道剪枝+残差链剪枝 85% 模型对应的检测结果图,由图 7 可见,裁剪后的模型能够较为准确地预测出车辆的大小和位置。

(a) 乡村道路



(b) 高速道路

图 7 裁剪后模型的检测结果

Fig. 7 Detection results of pruned model

2) 模型量化结果对比。

模型量化在 Darknet-53 上分别实现了不同的量化方法,并以表 1 中通道剪枝+残差链剪枝 85% 为基准进行验证,实验结果如表 2 所示。

表 2 模型量化结果对比

Tab. 2 Comparison of model quantization results

量化方法	精度 AP
原始模型	0.68
最大最小值	0.45
KL 散度	0.58
KL 散度+模拟量化	0.62
KL 散度+每通道模拟量化	0.63

由表 2 可以看出,由于存在离群值,使得最大最小值量化过程存在较大的信息损失,精度降低严重。结合 KL 散度量化能够一定程度上减少信息损失,再结合模拟量化对模型进行微调,能够有效地减少模型的精度损失。其中,由于使用了批归一化,激活值的分布关于零点基本对称,因此使用对称量化前后的精度损失较小。最后,本文尝试了使用每通道量化的方式,对每个卷积输出通道分别进行量化,增加了模型参数数量,但可以进一步减少精度损失。

在表 3 中,将本文所提的压缩算法与经典的 YOLOv3 tiny 进行对比,这是 YOLOv3 提供的一个压缩版本,实验使用基本操作数(Basic OPerationS, BOPS)比较卷积操作的计算量,其单位为 10^9(Billion)。本文所提压缩方法获得的模型与 YOLOv3 tiny 的 BOPS 接近,但获得了超过 YOLOv3 tiny 的 7 个百分点的性能,同时模型尺寸约为 YOLOv3 tiny 的六分之一。此外,相较于其他裁剪与量化方法,本文所提方法在保持高性能的前提下也具有较高的压缩比,实验验证了本文所提压缩方法的有效性。

表 3 不同压缩算法的模型尺寸、计算量与精度对比

Tab. 3 Comparison of model size, computational cost and accuracy for different compression algorithms

模型	数据类型	模型尺寸/MB	计算量/BOPS	精度 AP
YOLOv3[34]	FP32	230	65.86	0.74
YOLOv3 tiny[34]	FP32	34	5.56	0.56
通道剪枝 85%[14-15]	FP32	61	19.25	0.70
通道剪枝 90%[14-15]	FP32	47	13.17	0.68
本文剪枝 85%	FP32	21	6.65	0.68
本文剪枝 85%+量化[36]	INT8	6	6.65	0.58
本文剪枝 85%+本文量化	INT8	6	6.65	0.63

3) 加速模块性能对比。

对本文涉及的几个加速模块的实现进行比较,包括 ARM(Advanced RISC Machine)处理器版本的实现、从 ZynqNet 移植的基准实现、本文设计和优化的没有集成 Winograd 算法的实现和集成 Winograd 算法的实现,结果如表 4 所示。

由表 4 可见,在端设备上直接使用 ARM 处理器进行推理是不现实的,借助 FPGA 进行 8 比特推理能够有效提升计算速度,在 ZynqNet 的基准实现上,本文的设计和优化能够提升模块的平均性能,有力地支持神经网络的端侧推理。表 4 中,平均性能通过迭代 200 轮进行统计,性能的度量单位是每秒十亿计算数(Giga Operations Per Second, GOPS)。

表 4 不同计算加速模块的功耗、资源利用率与性能对比

Tab. 4 Comparison of power consumption, resource utilization rate and performance for different computation acceleration modules

实现方式	计算功耗/W	资源利用率/%	性能/GOPS	时间/s
ARM 处理器	1.62	0	0.023	6 819.0
ZynqNet	1.28	70	4.600	34.6
本文(无 Winograd)	1.31	76	8.900	17.9
本文(Winograd)	1.38	70	20.400	7.8

接下来,本文结合现有的包括 CPU、GPU、FPGA 等多个平台的实现结果[42-43]对本文在 ZYNQ 平台上的计算加速算法进行评估,结果如表 5 所示,通过单位功耗实现的操作总数评估各实现的能耗比。本文实现的非 Winograd 加速模块的平均性能与 GPU 样本相近,比 FPGA 样本略高。具有 Winograd 的加速方案能够在无 Winograd 加速的基础上提升一倍多的能耗比;与其他平台相比,本文集成了 Winograd 的方案能够在更低的功耗下取得较高的计算性能。

表5　不同平台的性能、功耗以及能耗比对比

Tab. 5　Comparison of performance, energy consumption and energy consumption ratio on various platforms

平台	性能/GOPS	功耗/W	能耗比/(GOPS·W^{-1})
Xeon E5-2500[42-43]	119.0	95.00	1.25
GTX TITAN X[42-43]	1661.0	250.00	6.60
Stratix-V GSD8[42-43]	117.8	19.10	6.17
本文(无Winograd)	8.9	1.31	6.79
本文(Winograd)	20.4	1.38	14.78

5　结语

本文围绕YOLOv3网络在低功耗场景下的应用进行研究,通过模型压缩与适配端侧设备的计算加速实现了网络在ZYNQ平台的部署,达到了较高的能耗比。一方面,本文设计了针对残差网络的裁剪方法,通过引入针对残差链的剪枝和按通道的量化实现了以较低的精度损失获得较高压缩比的目的。另一方面,本文在ZynqNet的基础上设计实现了卷积加速模块,优化了片上缓存,集成了Winograd算法,极大地提高了低功耗场景下ZYNQ平台的卷积性能。与不同平台的比较表明,本文研究获得了较高的能耗比,能够加速推进YOLOv3以及其他残差网络在ZYNQ端侧的部署。

关于神经网络的压缩与部署,仍有很多值得探索的地方。例如,将本文所提压缩方案推广到具有池化等模块的网络,探讨适用更复杂结构的网络压缩方案是未来一个有价值的研究方向。另外,在加速模块部分,结合Chisel(Constructing hardware in a scala embedded language)等敏捷硬件开发工具,实现更细粒度的硬件结构控制,从而提高硬件资源的利用率、降低平均功耗也是值得考虑的一个研究点。

参考文献(References)

[1] 马旗,朱斌,张宏伟,等. 基于优化YOLOv3的低空无人机检测识别方法[J]. 激光与光电子学进展, 2019, 56(20):No. 201006. (MA Q, ZHU B, ZHANG H W, et al. Low-altitude UAV detection and recognition method based on optimized YOLOv3[J]. Laser and Optoelectronics Progress, 2019, 56(20):No. 201006.)

[2] 李云鹏,侯凌燕,王超. 基于YOLOv3的自动驾驶中运动目标检测[J]. 计算机工程与设计, 2019, 40(4):1139-1144. (LI Y P, HOU L Y, WANG C. Moving objects detection in automatic driving based on YOLOv3[J]. Computer Engineering and Design, 2019, 40(4):1139-1144.)

[3] GIRSHICK R, DONAHUE J, DARRELL T, et al. Region-based convolutional networks for accurate object detection and segmentation [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016, 38(1): 142-158.

[4] 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 and Machine Intelligence, 2017, 39(6):1137-1149.

[5] LIU W, ANGUELOV D, ERHAN D, et al. SSD: single shot multibox detector [C]// Proceedings of the 2016 European Conference on Computer Vision, LNCS 9905. Cham: Springer, 2016:21-37.

[6] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: unified, real-time object detection [C]// Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2016: 779-788.

[7] YURTSEVER E, LAMBERT J, CARBALLO A, et al. A survey of autonomous driving: common practices and emerging technologies [J]. IEEE Access, 2020, 8: 58443-58469.

[8] ZHANG X, ZHOU X, LIN M, et al. ShuffleNet: an extremely efficient convolutional neural network for mobile devices [C]// Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 6848-6856.

[9] HOWARD A G, ZHU M, CHEN B, et al. MobileNets: efficient convolutional neural networks for mobile vision applications [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1704.04861.pdf.

[10] SANDLER M, HOWARD A, ZHU M, et al. MobileNetV2: inverted residuals and linear bottlenecks [C]// Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 4510-4520.

[11] HOWARD A, SANDLER M, CHEN B, et al. Searching for MobileNetV3 [C]// Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision. Piscataway: IEEE, 2019: 1314-1324.

[12] 邹月娴,余嘉胜,陈泽晗,等. 图像分类卷积神经网络的特征选择模型压缩方法[J]. 控制理论与应用, 2017, 34(6):746-752. (ZOU Y X, YU J S, CHEN Z H, et al. Convolutional neural networks model compression based on feature selection for image classification[J]. Control Theory and Applications, 2017, 34(6): 746-752.)

[13] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural network [C]// Proceedings of the 28th International Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2015: 1135-1143.

[14] LI H, KADAV A, DURDANOVIC I, et al. Pruning filters for efficient ConvNets [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1608.08710.pdf.

[15] HE Y, ZHANG X, SUN J. Channel pruning for accelerating very deep neural networks [C]// Proceedings of the 2017 IEEE International Conference on Computer Vision. Piscataway: IEEE, 2017: 1398-1406.

[16] LIU Z, LI J, SHEN Z, et al. Learning efficient convolutional networks through network slimming [C]// Proceedings of the 2017 IEEE International Conference on Computer Vision. Piscataway: IEEE, 2017: 2755-2763.

[17] LIU Z, SUN M, ZHOU T, et al. Rethinking the value of network pruning [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1810.05270.pdf.

[18] HAN S, MAO H, DALLY W J. Deep compression: compressing deep neural networks with pruning, trained quantization, and Huffman coding [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1510.00149.pdf.

[19] COURBARIAUX M, HUBARA I, SOUDRY D, et al. Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1 [EB/OL]. [2020-09-19]. https://arxiv.org/pdf/1602.02830.pdf.

[20] RASTEGARI M, ORDONEZ V, REDMON J, et al. XNOR-Net: ImageNet classification using binary convolutional neural networks [C]// Proceedings of the 2016 European Conference on Computer Vision, LNCS 9908. Cham: Springer, 2016: 525-542.

[21] JACOB B, KLIGYS S, CHEN B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference

[C]// Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 2704-2713.

[22] WU S, LI G, CHEN F, et al. Training and inference with integers in deep neural networks [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1802.04680.pdf.

[23] SAU B B, BALASUBRAMANIAN V N. Deep model compression: distilling knowledge from noisy teachers [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1610.09650.pdf.

[24] XU Z, HSU Y C, HUANG J. Training shallow and thin networks for acceleration via knowledge distillation with conditional adversarial networks [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1709.00513.pdf.

[25] CROWLEY E J, GRAY G, STORKEY A. Moonshine: distilling with cheap convolutions [C]// Proceedings of the 32nd International Conference on Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc., 2018: 2893-2903.

[26] JIA Y, SHELHAMER E, DONAHUE J, et al. Caffe: convolutional architecture for fast feature embedding [C]// Proceedings of the 22nd ACM International Conference on Multimedia. New York: ACM, 2014: 675-678.

[27] 盛荣菊,马建伟. 人工神经网络FPGA硬件实现的研究进展[J]. 电气自动化, 2009, 31(5):53-54, 67. (SHENG R J, MA J W. Research progress of FPGA hardware implementation of artificial neural network[J]. Electrical Automation, 2009, 31(5): 53-54, 67.)

[28] 余子健,马德,严晓浪,等. 基于FPGA的卷积神经网络加速器[J]. 计算机工程, 2017, 43(1):109-114, 119. (YU Z J, MA D, YAN X L, et al. FPGA-based accelerator for convolutional neural network[J]. Computer Engineering, 2017, 43(1): 109-114, 119.)

[29] GSCHWEND D. ZynqNet: an FPGA-accelerated embedded convolutional neural network [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/2005.06892.pdf.

[30] VASILACHE N, JOHNSON J, MATHIEU M, et al. Fast convolutional nets with fbfft: a GPU performance evaluation [EB/OL]. [2020-09-19]. https://arxiv.org/pdf/1412.7580.pdf.

[31] LAVIN A, GRAY S. Fast algorithms for convolutional neural networks [C]// Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2016: 4013-4021.

[32] LIU X, POOL J, HAN S, et al. Efficient sparse-Winograd convolutional neural networks [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1802.06367.pdf.

[33] IOFFE S, SZEGEDY C. Batch normalization: accelerating deep network training by reducing internal covariate shift [C]// Proceedings of the 32nd International Conference on International Conference on Machine Learning. New York: JMLR.org, 2015: 448-456.

[34] REDMON J, FARHADI A. YOLOv3: an incremental improvement [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1804.02767.pdf.

[35] CHENG Y, WANG D, ZHOU P, et al. A survey of model compression and acceleration for deep neural networks [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1710.09282.pdf.

[36] MIGACZ S. 8-bit inference with TensorRT [EB/OL]. [2020-04-20]. http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf.

[37] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition [C]// Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2016: 770-778.

[38] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions [C]// Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2015: 1-9.

[39] XILINX. Vivado HLS optimize methodology guide (UG1270) [EB/OL]. [2020-04-20]. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_4/ug1270-vivado-hls-opt-methodology-guide.pdf.

[40] HE K, ZHANG X, REN S, et al. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification [C]// Proceedings of the 2015 IEEE International Conference on Computer Vision. Piscataway: IEEE, 2015: 1026-1034.

[41] GEIGER A, LENZ P, STILLER C, et al. Vision meets robotics: the KITTI dataset [J]. The International Journal of Robotics Research, 2013, 32(11): 1231-1237.

[42] WANG T, WANG C, ZHOU X, et al. A survey of FPGA based deep learning accelerators: challenges and opportunities [EB/OL]. [2020-06-02]. https://arxiv.org/pdf/1901.04988v1.pdf.

[43] MA Y, CAO Y, VRUDHULA S, et al. Automatic compilation of diverse CNNs onto high-performance FPGA accelerators [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(2): 424-437.

This work is partially supported by the National Natural Science Foundation of China (61973245).

GUO Wenxu, born in 1996, M. S. candidate. His research interests include computer vision, deep learning.

SU Yuanqi, born in 1982, Ph. D., lecturer. His research interest includes computer vision, pattern recognition.

LIU Yuehu, born in 1962, Ph. D., professor. His research interest includes computer vision, pattern recognition.