DOI:10.11918/202111015

PYNQ 框架的高精度异构无预选框检测模型实现

张瑞琰[1,2],姜秀杰[1],安军社[1],崔天舒[1]

(1. 复杂航天系统电子信息技术重点实验室(中国科学院国家空间科学中心),北京 100190; 2. 中国科学院大学,北京 100049)

摘 要: 由于深度卷积网络的参数量及计算量过大,多尺度目标检测网络难以快速高精度地部署在许多资源及功耗受限的平台上。为解决此问题,本文基于 Python productivity for ZYNQ(PYNQ)框架实现了无预选框检测模型 CTiny 的 IP 核设计及异构系统架构部署。首先,提出在卷积核中分段量化整体缩放系数的方式,使得预训练的高精度算法低损地部署于可编程门阵列(field programmable gate array,FPGA)上;其次,基于 PYNQ 框架实现了 CTiny 模型的系统搭建,包含 ResNet 主干网络、反卷积网络和分支检测网络;最后,将图片预处理及后处理等耗时计算从串行的 ARM 端移入并行的 FPGA 中,进一步缩减了总处理时长。实验结果表明:在 PYNQ-Z2 开发板上部署 CTiny 模型后,本文所提量化方式在公开光学遥感数据集 NWPU VHR-10 的平均检测精度达到 81.60%,相较于截断量化提升了 14.27%,实现了部署精简无预选框检测网络的精度低损耗的需求,且后处理的处理时长由 ARM 端的 9.228 s 缩减为了 FPGA 端的 0.008 s,提高了检测模型的速度。

关键词: 目标检测;Python productivity for ZYNQ;光学遥感图像;无预选框;整体缩放系数

中图分类号: TP391.4 **文献标志码:** A **文章编号:** 0367-6234(2022)05-0024-10

Realization of high-precision heterogeneous anchor-free detection model based on PYNQ framework

ZHANG Ruiyan[1,2], JIANG Xiujie[1], AN Junshe[1], CUI Tianshu[1]

(1. Key Laboratory of Electronics and Information Technology for Space Systems (National Space Science Center, Chinese Academy of Sciences), Beijing 100190, China; 2. University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Due to the large number of parameters and large amount of calculation of deep convolutional networks, it is difficult to quickly and accurately deploy multi-scale target detection networks on many platforms with limited resources and power consumption. To solve this problem, based on the Python productivity for ZYNQ (PYNQ) framework, this paper realizes the IP core design and heterogeneous system architecture deployment of CTiny model, which is an anchor-free object detection model. First, a method of segmental quantization of the overall scaling factors in the convolution kernel was proposed, so that the pre-trained high-precision algorithm could be deployed on the field programmable gate array (FPGA) with low loss. Then, the system of the CTiny model was constructed based on the PYNQ framework, including ResNet backbone network, deconvolution network, and branch detection network. Finally, the time-consuming calculation such as picture preprocessing and post-processing was moved from serial ARM to parallel FPGA, further reducing the total processing time. Experimental results show that after deploying the CTiny model on the PYNQ-Z2 development board, the proposed quantization method achieved a mean average precision of 81.60% in the public optical remote sensing dataset NWPU VHR-10, which increased by 14.27% than truncated quantization. It has realized the requirement of deploying a tiny anchor-free object detection network with low loss. In addition, the processing time of post-processing was reduced from 9.228 s on the ARM side to 0.008 s on the FPGA side, which improved the speed of the detection model.

Keywords: object detection; Python productivity for ZYNQ; optical remote sensing image; anchor-free; overall scaling factor

光学遥感目标检测是光学遥感图像处理领域中的一项基础性研究工作[1-2]。在光学遥感目标检测中,目标尺度的巨大差异一直是阻碍模型精度提升的一大因素。例如大片的操场和道路旁的小汽车,操场的尺度是汽车的几十倍甚至几百倍,同时兼顾二者的特征提取较为困难。随着多层神经网络提取的特征不断丰富,多尺度目标的检测精度已经获得了大幅提升。而这些精度的提升依赖于以浮点型运算为主的高计算量卷积神经网络。为了实现将检测模型部署于边缘端(如搭载图像处理功能的卫星终端)的任务,研究人员需提前量化浮点型算法以适

收稿日期: 2021-11-04
基金项目: 中国科学院复杂航天系统电子信息技术重点实验室自主部署基金(Y42613A32S)
作者简介: 张瑞琰(1995—),女,博士研究生;
姜秀杰(1965—),女,研究员,博士生导师
通信作者: 姜秀杰,jiangxj@nssc.ac.cn

应可编程门阵列（field programmable gate array，FPGA）、专用集成电路（application specific integrated circuit，ASIC）等平台。量化是将参数值（Weight）和激活值（Activation）从浮点型数据转化为定点型的过程。Krishnamoorthi[3] 给出了关于量化及反量化的具体方式和步骤。

现阶段，算法在 FPGA 端的推断部署过程为：首先，在 CPU/GPU 端训练算法并量化；之后，导出必要的参数值；然后，在 FPGA 端设计模型的 IP 核并搭建系统框架；随后，在 CPU 端设计交互窗口，控制调用 FPGA 端程序；最后，输入图片并输出结果。对于量化的研究多集中于在 CPU/GPU 端的量化算法的研究。然而，由于 CPU/GPU 和 FPGA 平台资源情况不同，在 FPGA 端实际部署时会存在精度下降的现象，主要有以下 3 点原因：

首先，在大量量化算法的文献中，模型量化时的缩放系数常保持浮点类型[4]，而 FPGA 属于全定点计算。因而对缩放系数的量化时，若将超过规定量化范围的数据直接截断[5] 会造成大量精度损失。

其次，不同平台适用的数据类型及精度不同。现阶段算法训练及推断中常用 PyTorch 框架。由于该框架没有低比特定点数据类型，须用浮点数进行伪量化，会造成 CPU 模拟量化结果与 FPGA 不同。

最后，网络架构设计的差异。在训练时，卷积层和激活层是无关联的两个独立层。但 FPGA 要求卷积运算输出定点值结果，即激活值需要量化为下层卷积层可直接参与卷积运算的定点数。

以上 3 点原因为平台转换过程中的主要不匹配因素。基于此，本文提出在硬件层面实现每层可控的系数调节卷积层的方法，即分段量化整体缩放系数，以充分利用定点数的表达范围，实现在 FPGA 端高精度地部署检测网络。

由于待检测图像的尺寸偏大，检测模型的计算量庞大，而边缘端设备资源稀少，整幅图像难以完整进行一次卷积运算，因而在边缘端需要切割图片并分块进行卷积计算。而这些步骤无疑造成了检测速度减慢。再加上检测模型存在图像预处理和后处理步骤，这些步骤在硬件部署中常被分配在 CPU 端完成。CPU 端主要负责计算量少的部分，而对于基于候选框的算法[6] 来说，其后处理含有复杂的非极大值抑制（non-maximum suppression，NMS）操作，会导致检测时间拉长。因此，本文使用无预选框模型 CenterNet[7] 的检测流程来设计部署算法。这种无预选框模型的后处理过程摒弃了 NMS 操作，更适合于实际部署。而为了提高检测速度，本文考虑将检测流程中的图片预处理及后处理等耗时计算过程从串行的 CPU 端（本文采用 ARM 处理器）移入并行的 FPGA 中，从而缩减了总处理时长。为快速便捷地实现异构系统搭建，本文采用了 Python productivity for ZYNQ（PYNQ）框架，即为 ZYNQ 系列硬件平台提供 Python 接口的一种软件开发框架。ZYNQ 的硬件本质为 ARM 与 FPGA 的异构系统。基于该框架，研究人员可通过 Python 轻松访问 FPGA，利于算法的快速开发应用。

综上所述，为在边缘端快速高精度地部署检测模型，本文提出了在卷积核中分段量化整体缩放系数的方法，并充分考虑了检测模型中预处理和后处理的并行化因素，缩短了检测时间。实验证明本文所提方法具备检测精度高、部署简单、推断时间快等特点。

1　CTiny 网络结构及 PYNQ 设计概述

1.1　CTiny 模型结构介绍

本文的检测模型基于 CenterNet 检测框架而得，该检测模型将目标视作点，适合于多尺度目标检测。本文采用的 CTiny 模型的主干网络为 ResNet_9[8]，其中的卷积运算和反卷积运算均采用分离卷积，ResNet 中的 Residual 模块的详细流程见图 1。

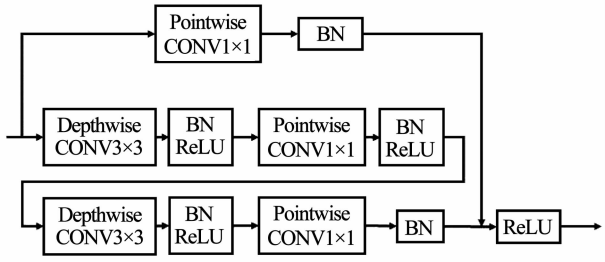


图 1　Residual 模块的详细流程

Fig. 1　Detailed process of Residual module

为实现卷积层的高复用性，将卷积核大小统一为 3×3、1×1 和 4×4。网络中的卷积模块按照功能分为普通分离卷积（Depthwise Convolution3×3，DW）、分离反卷积（Depthwise Deconvolution4×4，DDW）以及点卷积（Pointwise Convolution1×1，PW）。

1.2　基于 PYNQ 的子模块设计

由于 CTiny 网络结构较复杂，以及输入图片的分辨率较大，将网络所有层放在 FPGA 上处理并不现实。因此，本文需根据资源使用情况及模块复用性，对网络进行模块划分。这里共划分为 5 个模块见表 1，其中 ADD 代表 ResNet 网络中的直连层相加操作，MP 指最大池化操作。本网络的整体架构见图 2。

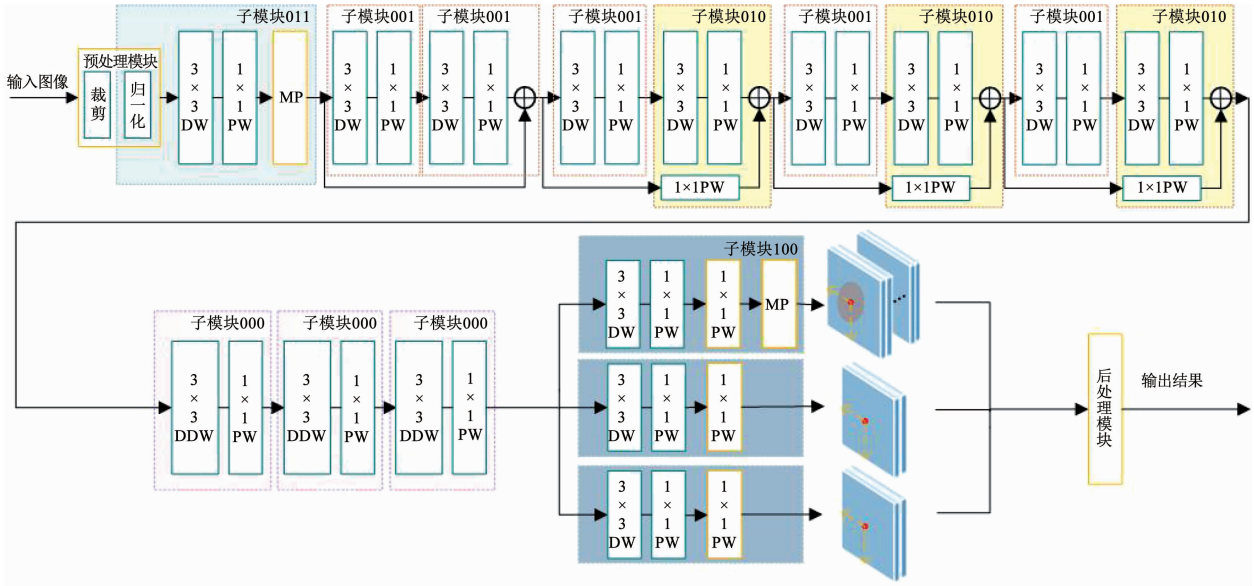


图 2 CTiny 整体网络架构

Fig. 2 Overall network architecture of CTiny

表 1 子模块划分

Tab. 1 Submodule division

序号	000	001	010	011	100
所含模块	DDW + PW	DW + PW	DW + PW + PW + ADD	DW + PW + MP	DW + PW + PW + MP

鉴于板上资源的限制，每个子模块的处理流程为：首先，从 AXI 总线读取特征图和参数值并传入处理寄存器，由于图片尺寸较大，读取待处理数据的过程需重复多次；然后，进行 DW 或 DDW 操作，循环处理每个通道的特征图，并通过 AXI 总线传入总缓存；待到所有通道处理完毕，再从总缓存中传回特征图缓存值，并将所有通道的值累加以完成一次 PW 运算；最后，将处理完毕的数据通过 AXI 总线传回 ARM 端。至此完成了一次完整的子模块处理流程。本文的 CTiny 模型在 FPGA 上的总体网络模型设计见图 3，其中 FM 代表特征图。

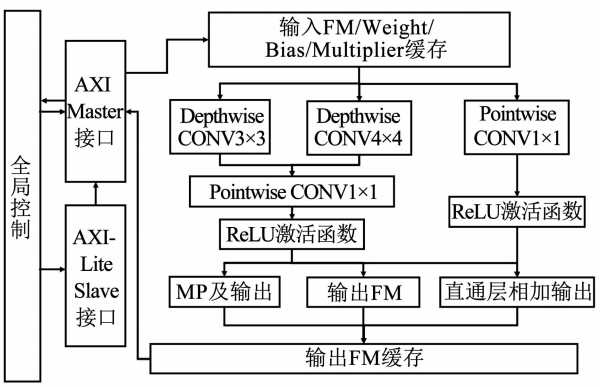


图 3 基于 FPGA 的总体网络模型设计

Fig. 3 Overall network model design based on FPGA

2 基于 PYNQ 的 CTiny 设计关键要点

2.1 分段量化整体缩放系数设计

本文所提的分段量化整体缩放系数的方法主要应用于卷积层内。Weight 量化中每个卷积核独享一个量化缩放系数，Activation 量化则是每层中所有卷积核共享一个量化缩放系数。

设第 i 层激活值浮点数为 A_i，第 $i+1$ 层的激活值为 A_{i+1}。相对应地，参与 FPGA 中 MAC 运算的量化激活值为 $\overline{A}_i$ 及 $\overline{A}_{i+1}$，这里采用对称量化，激活值的量化比特数为 N_A，该过程如下：

$$A_i = \overline{A}_i \times S_i^A \tag{1}$$

式中 $S_i^A = \dfrac{|A_i|_{\max}}{2^{N_A-1}-1}$ 为量化系数。

设参数值的量化比特数为 N_w，量化系数为 $S_{i,j}^W$，其中下标 i 指第 i 层卷积，下标 j 代表卷积的第 j 个通道。则该层中的不同通道的卷积量化过程如式(2)所示，其中 H 和 W 为卷积核的高和宽。

$$\overline{A}_{i+1,j} \times S_{i+1}^A = \sum_H \sum_W (\overline{A}_{i,j,h,w} \times S_i^A) \times (\overline{W}_{i,j,h,w} \times S_{i,j}^W) + b_i \tag{2}$$

再将式(2)传入批量归一化层(batch normalization，BN)，γ 为 BN 层的缩放系数，β 为偏移，μ 为批量数据的均值，σ 为标准差。将 BN 层融入卷积层的过程见式(3)，可保证一次卷积运算直接得到下一层的量化输入值。

$$\overline{A}_{i+1,j} = M \times \left(\sum_H \sum_W \overline{W}_{i,j,h,w} \overline{A}_{i,j,h,w} + B \right) \tag{3}$$

式中：$M = \dfrac{S_i^A S_{i,j}^W}{S_{i+1}^A} \times \dfrac{\gamma}{\sqrt{\sigma^2+\varepsilon}}$ 为该卷积层的整体缩放

系数（即为图 3 中的 Multiplier），$B = S_i^A S_{i,j}^W \left(\beta \frac{\sqrt{\sigma^2+\varepsilon}}{\gamma} - \mu + b_i\right)$ 为整体偏移系数（即为图 3 中的 Bias）。

这里先叠加 B 的原因是可以提前一步确定计算结果是否比零小，比零小的数据不再进行乘 M 的计算，更节约计算成本。本文以有符号的 16 bit 的定点数表示两种系数，其中 M 为远小于 1 的小数，因此需要将其量化为 $[-2^{15}, 2^{15}-1]$。而为了不消耗 MAC 资源，采用左/右移操作来实现两种缩放系数的反量化。

本文所用检测网络的卷积层包含卷积运算和反卷积运算，其中卷积层进行图片尺度的缩小，而反卷积层实现图片尺度的扩大，最终输出高分辨率热点图，输出尺寸为原始图片的 1/4。图 4（a）和（b）分别为各层卷积激活值的整体缩放系数及整体偏移系数的最大值统计图。横坐标为各卷积层的索引值，纵坐标分别为整体缩放系数值和整体偏移系数值。不同层的整体缩放系数最大相差 1 000 倍，而整体偏移系数相差 4 000 倍。Zhang 等[5]将二者每层缩放固定值但不影响最终效果，原因是该工作的检测目标呈现尺度相似的特点，且检测网络的卷积层较为规整，不包含 ResNet 的直连层。但面对本文这种背景复杂、目标尺度差异大的检测图像，包含直连层的检测网络才能得到较好的检测精度。为了使本设计具有更强的易迁移性和通用性，提出分段量化整体缩放系数。

由上述可知，M 的值极小而 B 的值极大。部署时需要将其通过线性运算分布于 $[-2^{15}, 2^{15}-1]$。为了使缩放系数有较大的表达范围，本文将分段量化方式细分为分块量化和逐层量化。分块量化需要设置块数为 R，而逐层量化则为每一层卷积设置不同量化系数，R 为层数。设某一段每个卷积层整体缩放系数为 M，其定点数为 $\overline{M}$，$\overline{M}$ 的右移位数为 K_M，偏移系数 B 的左移位数为 K_B，这样 B 可获得较大分布范围，对于检测大尺度目标效果显著。为了经过四舍五入后与浮点数更接近，需叠加一个常量 G，且 $G = S_{\text{left}}(1, K_M - 1)$。这里算法的参数值存储需要配合该硬件的改进。则该段的计算过程为

$$Y = S_{\text{right}}(\overline{M} \times [X + S_{\text{left}}(B, K_B)] + G, K_M)$$

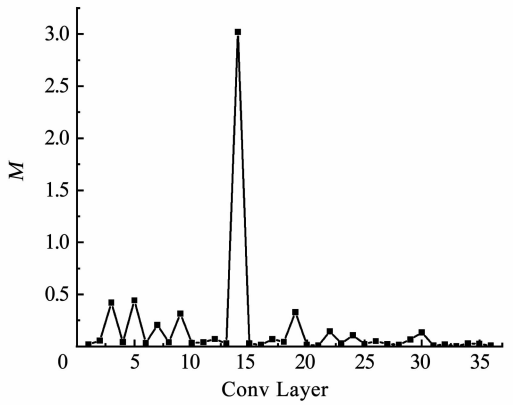
$$\text{s.t. } \min \sum_R (|\text{round}(S_{\text{left}}(M, K_M)) - \overline{M}|) \quad (4)$$

其中 $S_{\text{left}}(\cdot)$ 为左移运算，$S_{\text{right}}(\cdot)$ 为右移运算。

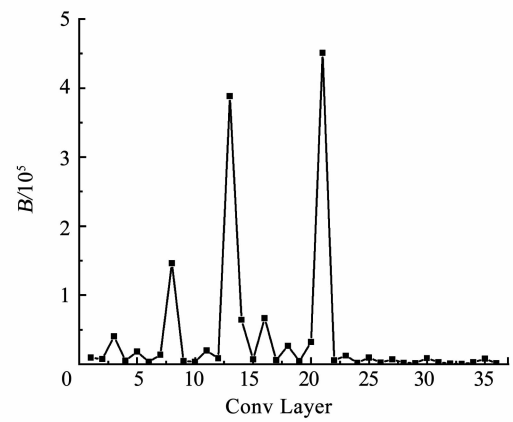
2.2　CTiny 检测模型的系统搭建

2.2.1　卷积层设计

PYNQ-Z2 开发板的资源无法处理整幅图片，因此需将图像分割为片上能处理的最小处理单元[9-10]。因 FPGA 中计算资源的限制，需在输出通道数、特征图长度及宽度 3 个维度进行循环展开，分块进行卷积运算[11-12]。本文主要设计方式与 Zhang 等[5]保持一致，将特征图在宽高维度作为循环展开的外层循环，通道维度作为内层循环，由此可实现特征图在通道维度的并行计算。



(a) 各卷积层整体缩放系数值 M 的最大值



(b) 各卷积层整体偏移系数值 B 的最大值

图 4　各卷积层的 M 和 B 的统计图

Fig. 4　Statistical graphs of M and B of each convolutional layer

这里主要介绍 DW 及 PW 的分块处理流程。DW 的处理流程见图 5。本文将 Weight 通过 AXI 总线传入片上缓存区域，并转换为所需的三维结构。设所需处理的特征图为 $X \in \mathbb{R}^{C \times H \times W}$，将其切割为片上能处理的最小处理单元 $F_{\text{in}} \in \mathbb{R}^{C_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}}$，其中 C_{in} 为输入通道数，H_{in}、W_{in} 为片上所需处理图片的高和宽，并满足 $C = N_0 \times C_{\text{in}}$，$H = N_1 \times H_{\text{in}}$，$W = N_2 \times W_{\text{in}}$。因此一整幅图片的卷积操作需要重复 F_{in} 操作的次数为 $N_0 \times N_1 \times N_2$。这里的卷积运算采用经典的滑窗思想。$k$ 为卷积核尺寸，将 C_{in} 维度的卷积核按维度展开，每次可实现并行处理 C_{in} 次，因此每个时钟可并行计算 $C_{\text{in}} \times k \times k$ 个像素点。

PW 模块的设计与 DW 类似。由于 PW 负责各通道的加权操作，不涉及宽高维度，因此对于步长为

2 的情况，本文只存储真正参与卷积运算的数值，实现提前间隔采样，避免了存储资源的浪费。

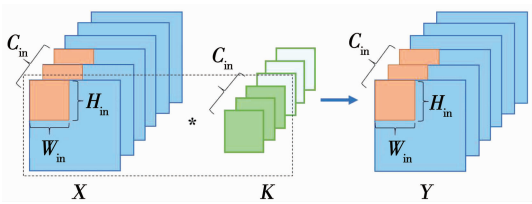


图 5 DW 卷积分块处理流程

Fig. 5 Block processing flow of DW convolution

2.2.2 非对称填充及下采样拼接

在进行卷积操作之前，需要从总线中读取所需要的数据，并组合成为片上能处理的最小单元，将该过程定为 Load_IMG。为了提高资源利用率，Load_IMG 需令特征图的缓存满足所有子模块的需求，即包含两次下采样卷积（步长为 2 以及最大池化操作）、一次下采样卷积、无下采样卷积和反卷积。

这里主要考虑两次下采样卷积的情况。本文提出一种左右非对称填充方法，即在图像左上侧补充 3 个像素点，右下侧补充 1 个像素点。该过程的详细介绍见图 6。设图中输入图片大小为 8 ×8，如阴影所示，左边填充 3，右边填充 1 后，得到 12 ×12 的图片。进行第一次卷积操作后，得到 6 ×6 的图片大小（浅蓝色像素点）。之后进行最大池化操作（图中的 MAX_POOL），便可得到预期的 2 ×2 的图片。而对比于通常意义上的左右对称填充（需要左右各填充 3 个像素点），可以减小缓存的图片尺寸，从而降低计算资源。

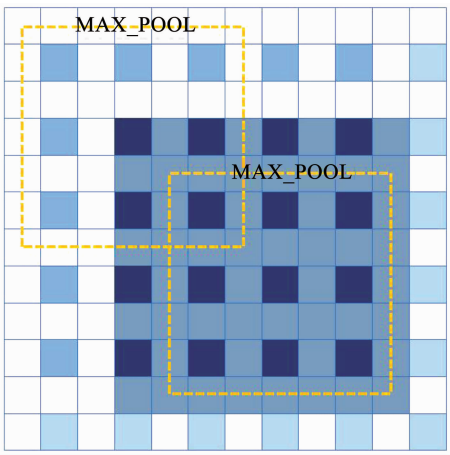


图 6 非对称填充方法

Fig. 6 Asymmetric padding method

为方便操作，本文采用下采样拼接的方法解决 DW 中卷积步长为 2 的情况。该步骤将 4 个片上处理单元视为一组，设置输出坐标点的分块位置，每输入完一组图片（4 张）便可得到原始图片大小，便于后续的存储和读取。

2.3 图片预处理及后处理的 FPGA 设计

2.3.1 图片预处理的 FPGA 设计

本文采用的输入图像为 8 bit 存储的 RGB 三通道数据。常见的检测算法需要先对图片进行数值归一化，得到正负皆有的正态分布数据。而在部署过程中，该步骤存在一定的困难：其一，FPGA 中的特征图为无符号的 8 bit 定点数，若将归一化后的原始图像通过均匀量化转为[0, 255]之间，会产生放缩和偏移量，在硬件设计中无多余模块进行输入数据的反量化处理，而开辟新的模块会造成资源浪费；其二，两次转换会造成较大的精度损失。因此，本文考虑在 Load_IMG 步骤中融入归一化操作。

设原始输入图像为 X_0，数据集均值为 $\mu_{\text{data}} \in \mathbb{R}^{3\times1}$，标准差为 $\sigma_{\text{data}} \in \mathbb{R}^{3\times1}$。输入图片的正态化过程定义如下：

$$X = \left(\frac{X_0}{255} - \mu_{\text{data}}\right) \times \frac{1}{\sigma_{\text{data}}} = \frac{X_0}{255 \times \sigma_{\text{data}}} - \frac{\mu_{\text{data}}}{\sigma_{\text{data}}} \tag{5}$$

数据归一化后经过卷积层和 BN 层后的结果如式(6)所示：

$$A_{i+1,j} = \frac{\gamma}{\sqrt{\sigma^2+\varepsilon}} \times \left[S_{i,j}^{W} \sum_{H}\sum_{W} \overline{W}_{i,j,h,w} \left(\frac{\overline{x}_{i,j,h,w}}{255 \cdot \sigma_{\text{data}}} - \frac{\mu_{\text{data}}}{\sigma_{\text{data}}} \right) \right] + \frac{\gamma}{\sqrt{\sigma^2+\varepsilon}} \times S_i^A S_{i,j}^W \left(\beta \frac{\sqrt{\sigma^2+\varepsilon}}{\gamma} - \mu + b_i \right) \tag{6}$$

与式(3)对照可得 $S_i^A = \frac{1}{255 \cdot \sigma_{\text{data}}}$，简化式(6)可得

$$A_{i+1,j} = \frac{\gamma S_{i,j}^W S_i^A}{\sqrt{\sigma^2+\varepsilon}} \times \sum_{H}\sum_{W} \overline{W}_{i,j,h,w} \overline{x}_{i,j,h,w} - \frac{\gamma S_{i,j}^W S_i^A}{\sqrt{\sigma^2+\varepsilon}} \times 255 \times \mu_{\text{data}} \times \sum_{H}\sum_{W} \overline{W}_{i,j,h,w} + \frac{\gamma S_{i,j}^W S_i^A}{\sqrt{\sigma^2+\varepsilon}} \times S_i^A S_{i,j}^W \left(\beta \frac{\sqrt{\sigma^2+\varepsilon}}{\gamma} - \mu + b_i \right) \tag{7}$$

$$I = -255 \cdot \mu_{\text{data}} \cdot \sum_{H}\sum_{W} \overline{W}_{i,j,h,w} \tag{8}$$

其中 I 是与输入数据无关的常数，其只与通道有关，因此可以提前计算出输入 3 个通道的数值，并将其视为整体偏移系数的一部分。

归一化处理时，载入图像 Load_IMG 过程中需要对原始图像进行补零操作。因为算法层面是在归一化之后补零，而本文所提硬件方法是在归一化之前补零。当按照上述的输入层归一化融合操作后，硬件层面补充的零值相当于实际运算中多余偏移值 P。因此本设计将补充值按照不同通道分别补充 $P = 255 \times \mu_{\text{data}}$，代替以往的零值。该过程见图 7，其中 j 代表第 j 通道，B_j 为原始的整体偏移系数。

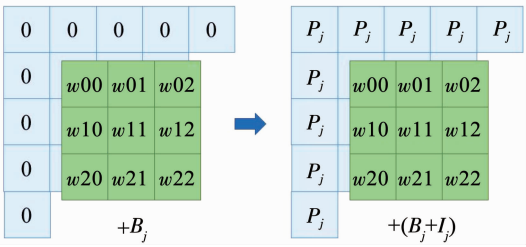


图 7　输入层融合归一化操作

Fig. 7　Input layer integrating normalization operations

2.3.2　后处理的 FPGA 设计

CTiny 网络输出结果为高分辨率的热点图，并通过最大池化的方法找到一定区域内的最大值，以此确定目标的位置。最大池化操作在串行的 ARM 端运行极为缓慢，因此本文考虑将算法中的最大池化操作和 Sigmoid 操作调换顺序，在 FPGA 端处理 MP，以此可大大降低预测时间。因图像处理结果输出值大于 0 等同于 Sigmoid 操作输出值大于 0.5，所以 FPGA 端的 MP 操作可提前筛选掉分类概率小于 0.5 的目标。如图 8 所示，PS 指 ARM 端，PL 指 FPGA 端。将 MP 操作提前后，Sigmoid 操作可在后续处理过程中省去。因后续已经确定目标出现位置，只需将宽、高和偏移量提取出即可，此举可极大地节约检测时间。

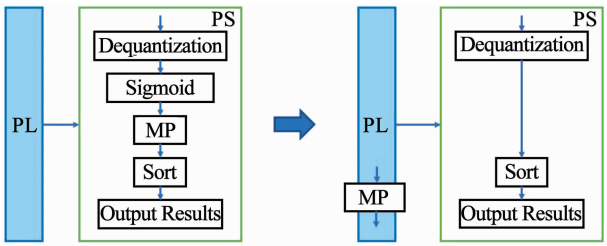


图 8　最大池化的 PL 融合过程

Fig. 8　Process of integrating the Max Pooling into PL

3　实验设计与结果分析

3.1　实验平台及算法介绍

本文所用开发平台为 PYNQ-Z2 开发板。该开发板采用 PYNQ 框架，基于 Xilinx ZYNQ-7000 的 FPGA 平台，支持用户在 ARM 端（PS 端）进行 Python 编程及调试。用户设计的电路模块运行于 FPGA 端（PL 端），可作为 IP 核供 PS 端调用。PS 端采用 ARM Cortex-A9，运行频率为 650 MHz，PL 端工作频率为 100 MHz，FPGA 型号为 xc7z020clg400。设置 AXI 传输带宽为 128 位，片上处理中的 H_{in} 和 W_{in} 均为 36，并行通道处理数 C_{in} 为 16。参数值为 8 位传入，实际计算时只考虑 6 位。鉴于硬件部署时间周期过长，本文首先采用高层次综合（high level synthesis，HLS）的方式快速构建设计架构[13]，并由软件端将 C 语言直接转换成适合的硬件描述语言，以构建无预选框检测模型的 IP 核；其次，在 Vivado 软件搭建异构系统整体框架；最后，在 ARM 端通过 Python 调用 numpy、time、PIL 等软件库，通过编程控制检测系统的输入输出及内存分配。

CTiny 输出的检测结果为定位热点图、回归热点图和中心偏移热点图。定位热点图的处理较为复杂，需要首先在 ARM 端进行各像素点的 Sigmoid 函数处理，之后对各点排序，在前 K 个目标点（文中 K 取 64）中挑出预测值大于 0.5 的点，获得其二维坐标索引；之后，根据索引值在回归热点图和偏移热点图中分别得到坐标索引对应的宽高值和偏移值；最后，经过目标中心点叠加偏移和宽高值后，可输出最终预测目标框的数值。

按照前文的模块划分，简化后的 CTiny 网络的详细结构见表 2。最后三层均为检测分支，特征图大小不发生改变。检测分支网络的中间输出通道数为 64。

表 2　CTiny 网络结构

Tab. 2　Structure of CTiny network

序号	模块	步长	输入图像尺寸	输出图像尺寸	输入通道数	输出通道数
1	DW + PW + MP	4	512	128	3	32
2	DW + PW	1	128	128	32	32
3	DW + PW	1	128	128	32	32
4	DW + PW	2	128	64	32	64
5	DW + PW + PW + ADD	1	64	64	64	64
6	DW + PW	2	64	32	64	128
7	DW + PW + PW + ADD	1	32	32	128	128
8	DW + PW	2	32	16	128	256
9	DW + PW + PW + ADD	1	16	16	256	256
10	DDW + PW	2	16	32	256	128
11	DDW + PW	2	32	64	128	64
12	DDW + PW	2	64	128	64	32
13	DW + PW + PW + MP	1	128	128	32	10
14	DW + PW + PW	1	128	128	32	2
15	DW + PW + PW	1	128	128	32	2

3.2　数据集及检测指标

本实验中数据集采用公开的光学遥感数据集 NWPU VHR-10[14]。NWPU VHR-10 有标注信息的图像 650 张，共包含 10 类目标，分别为：飞机、舰船、油罐、棒球场、网球场、篮球场、田径场、港口、桥梁和车辆。本文只采用包含目标的图像进行训练。对数据集进行裁剪处理，裁剪后图像尺寸为 640 × 640，两张相邻图像的重叠像素点为 140。经线性变换后，输入网络的图像尺寸为 512 × 512。将裁剪后数据集的 20% 作为测试集。从各类中随机挑出两张作为可视化测试图片，用以检测输出检测标记框是

否准确。

本文采用的检测指标为所有类别的平均检测精度的均值(mean average precision,mAP)。首先,本文中检测算法设置一张图中检出目标数最多为 64;其次,在 64 个候选目标框中挑选其中与真实目标框的交并比大于 0.5 的框,并认为这些预测框检测到了目标。交并比(intersection over union,IoU)是两个候选框之间的交集与并集的比值,其公式如式(9)所示:

$$I_{ou}=\frac{P_R\cap G_T}{P_R\cup G_T} \tag{9}$$

式中 P_R 为预测目标框,G_T 为真实目标框。然后,计算预测目标中每类的平均精度(AP),即 Precision-Recall 曲线下的面积。其中准确率(Precision)是指被预测为正样本中的真实正样本的比例,而召回率(Recall)指被预测为正样本占所有真实正样本的比例。最后,通过求取 AP 的平均值得到本算法的 mAP 值,mAP 值越高,检测性能越好。

3.3 实验结果

3.3.1 检测精度

为验证本文所提方法的有效性,本文评估了全浮点数检测网络 RICAOD[15] 和 Yolo-v2[16],以及定点数检测网络 CTiny、CTiny_cut、CTiny_block 和 CTiny_piece 基于 NWPU VHR-10 数据集的平均检测精度,并详细对比了各类别的平均精度。表 3 为各模型参数对比,表 4 为精度对比结果。表 3 中的 32 bit 指浮点数,其余为量化的定点数位宽。其中的 RICAOD 基于平台 NVIDIA TITAN X GPU,Yolo-v2 和 CTiny 基于 NVIDIA GTX 1080 GPU,而 FPGA 平台均为 Xilinx ZYNQ。表中 RICAOD 和 Yolo-v2 网络作为对比模型,二者均为基于预选框的检测模型,生成不同尺度、不同长宽比的候选框以匹配目标,反映了现阶段基于 NWPU VHR-10 数据集的基本检测水平。RICAOD 的主干网络为 AlexNet 的改进版,由于采用了全连接层而导致参数量较大。Yolo-v2 的主干网络为 DarkNet-19。从表 4 可以看出,与两个浮点型检测模型的效果对比,无预选框检测模型 CTiny 基本可以满足检测精度需求,虽然主干网络只有 9 层完整卷积(共 18 层分离卷积),参数值和激活值也分别量化为了 6 bit 和 8 bit,但其平均检测精度达到了 89.08%。

表 3 不同模型的参数对比

Tab. 3 Parameter comparison of different models

参数	RICAOD	Yolo – v2	CTiny	CTiny_cut	CTiny_block	CTiny_piece
平台	GPU	GPU	GPU	FPGA	FPGA	FPGA
主干网络	AlexNet	Darknet-19	DW-ResNet-9	DW-ResNet-9	DW-ResNet-9	DW-ResNet-9
参数位宽/bit	32	32	6	6	6	6
激活值位宽/bit	32	32	8	8	8	8
M 位宽/bit			32	16	16	16
B 位宽/bit	32	32	32	16	16	16

表 4 VHR-10 数据集的平不同模型在 NWPU 均检测精度对比

Tab. 4 Comparison of mean average precisions of different models on NWPU VHR-10 dataset

模型	飞机	舰船	油罐	棒球场	网球场	篮球场	田径场	港口	桥梁	车辆	mAP
RICAOD	0.997 0	0.908 0	0.906 1	0.929 1	0.902 9	0.803 1	0.908 1	0.802 9	0.685 3	0.871 4	0.871 2
Yolo-v2	0.999 2	0.909 1	0.817 2	0.983 0	0.907 7	0.909 1	0.991 4	0.897 8	0.886 9	0.815 4	0.911 7
CTiny	0.991 4	0.933 6	0.920 1	0.990 5	0.963 6	0.742 3	0.873 8	0.906 2	0.792 3	0.794 3	0.890 8
CTiny_cut	0.791 5	0.743 5	0.627 6	0.904 7	0.886 6	0.615 6	0.602 9	0.704 5	0.395 1	0.461 2	0.673 3
CTiny_block	0.923 3	0.817 4	0.767 4	0.973 0	0.948 5	0.620 1	0.784 9	0.794 9	0.584 2	0.683 8	0.789 7
CTiny_piece	0.972 8	0.859 6	0.844 6	0.968 7	0.957 2	0.655 0	0.716 8	0.818 3	0.664 2	0.702 0	0.816 0

在 FPGA 端实现时,需将卷积层、归一化层、激活层合并为一层,参数的表征形式从浮点数变为全定点数,这会导致模型的检测精度大幅降低,如表 4 中 CTiny 和 CTiny_cut 两列数据所示,侧面反映出同一模型在两个平台转换的不匹配。经前文分析,可通过 *M* 和 *B* 的量化方式提升模型性能。因此,采用

直接截断、分块量化和逐层量化的方式处理 M 和 B，结果如表 4 中 CTiny_cut、CTiny_block 和 CTiny_piece 所示，其中截断操作采取整体缩放系数值扩大 2^{14} 的结果。通过结果可以看出，直接截断的方法 CTiny_cut 效果最差，分析是因为大量削减了数值中的大数，而这些大数往往是决定检测效果的关键数据。CTiny_block 取折中的方法，分析每层量化系数和偏置的数值，统计它们的分布范围，最终将其划分在 4 个区域块中。CTiny_block 综合考虑了所有层的量化系数和偏置的影响，有效提高了检测效果。而针对每层系数均做处理的 CTiny_piece 则进一步提高了该小型定点网络的检测性能，其平均检测精度相比 CTiny_cut 和 CTiny_block 分别提高了 14.27% 和 2.63%，证明了本文所提高放缩的分段量化整体缩放系数的有效性。

3.3.2　前向推断时间

考虑到不同型号的 FPGA 开发板对同一个检测模型的执行性能相差很大，尤其是在部署较深网络时，硬件的时钟频率、资源大小等均会影响硬件的布局布线和运行时间。因此，本文采取在同一块开发板上运行 Yolo-v2 和 CTiny。二者实验的操作流程基本一致，均采用在 PL 端进行一次完整的卷积运算（一次普通卷积或两次分离卷积）之后将数据传回 PS 端，然后进行下一次卷积运算。结果显示，本设计在较低的时钟频率上，依旧可达到较快的检测速率。

从表 4 的实验结果可以看出在 FPGA 端提前进行 MP 处理对时间提升的有效性。其中，图像预处理时间指图片从 SD 卡中读取并载入内存中的总时长。后处理时间是指 FPGA 输出的结果经 PS 端计算得出目标框位置的总时长。图像处理时间指图片从输入 PL 端到输出的总用时，而 FPGA 时间指 FPGA 中的所有层的总时间，二者相减即可得到 PS 端的寄存器赋值寻址等操作所用时长。CTiny_piece 和 CTiny_block 均为在 PL 端实现 MP 操作，而 CTiny_piece（woMP）的 Sigmoid 函数和 MP 操作均在 PS 端实现。由表 5 可以看出 CTiny_piece（woMP）的后处理时间长达 9.228 s，远远不能达到实际需求；而将 Sigmoid 操作省去，在 FPGA 端提前处理 MP 则将后处理时间控制在 0.24 s，极大地减少了后处理时间，而 FPGA 端时间仅增加了 0.008 s，证明了本设计的有效性。

表 5　基于 PYNQ-Z2 开发板的前向推断时间

Tab. 5　Inference time based on PYNQ-Z2 development board

模型	分辨率	频率/Hz	卷积层数	参数位宽/bit	激活值位宽/bit	预处理时间/s	图像处理时间/s（FPGA 处理时间/s）	后处理时间/s
Yolo-v2	416	150	31	32	32	0.250	1.130	1.090
CTiny_block	512	100	36	6	8	0.110	0.517（0.373）	0.240
CTiny_piece	512	100	36	6	8	0.110	0.521（0.376）	0.240
CTiny_piece（woMP）	512	100	36	6	8	0.110	0.513（0.368）	9.228

由于 Yolo 模型是基于预选框的检测模型，其后处理中的 NMS 操作涉及到大量数据的循环比较操作，从表 5 中可看出 FPGA 的耗费时间较长。而本文的无预选框模型不需 NMS 操作，耗费时间的操作为 MP 及索引操作，只需一次排序，因此采用本文的 MP 提前实现的操作后，可显著缩减后处理时间。

同时表 5 比较了 Yolo-v2 的处理时间，可以看出片上资源过少对检测时间的限制较大。Yolo-v2 的 FPGA 端处理时间已经达到了 1.13 s。因为 FPGA 板资源过少，不得不进行模块划分。频繁与 ARM 端通信以及数据的复制、组合、移动、读取、写入等操作会大量增加处理时间。

3.3.3　FPGA 资源占用率

CTiny 网络在 FPGA 部署的资源占用情况见表 6。为了充分利用计算资源，本文设计的复合卷积 IP 核对 DSP 的利用率达到了 100%。因本文设计中大量采用了 HLS 中的循环展开操作，因此对查找表（LUT）的使用率较高，达到了 86% 左右。因为分层量化更为复杂，故 CTiny_piece 的资源占用率相较 CTiny_block 有大幅增多，运行功率也增加了 0.016 W。

3.3.4　可视化检测效果图

为更直观地验证检测效果，图 9 是每个类别的可视化检测结果。结果显示，本设计完成了基于 PYNQ 框架的 CTiny 网络实现，并达到了较高的检测精度。

表 6 FPGA 资源耗费

Tab. 6 FPGA resource consumption

模型	BRAM_18K	DSP48E	FF	LUT	LUTRAM	MMCM	Power/W
CTiny_cut	84.50 (60.36%)	220 (100%)	57 722 (54.25%)	42 656 (80.18%)	3 627 (20.84%)	1 (25%)	2.860
CTiny_block	84.50 (60.36%)	220 (100%)	50 183 (47.16%)	43 006 (80.84%)	3 629 (20.86%)	1 (25%)	2.835
CTiny_piece	84.50 (60.36%)	220 (100%)	58 764 (55.23%)	45 902 (86.28%)	3 627 (20.84%)	1 (25%)	2.851

图 9 基于 PYNQ 的 CTiny 的可视化检测效果图

Fig. 9 Visual detection results of CTiny based on PYNQ

4 结 论

本文基于 PYNQ 框架实现了无预选框检测模型 CTiny 的 IP 核设计及异构系统架构部署,主要结论如下:

1)本文提供了可分离卷积层的 FPGA 实现,并提出了分段量化整体缩放系数的方式,可供 CPU/GPU 端训练的算法高效低损地部署于 FPGA 上,提高了部署后的检测精度。

2)设计了一套基于 PYNQ 框架的无预选框检测网络 CTiny 的异构系统架构,可实现卷积运算、反卷积运算、批量归一化处理、激活函数、ResNet 直连层等功能。

3)本文充分考虑图片预处理及后处理的并行性,将两个过程中的耗时计算从串行的 ARM 端移入并行的 FPGA 中,有效缩短了图像的总检测时长。

实验结果反映出本文所提设计的有效性,为预训练的检测算法在 FPGA 上部署提供了实验基础,具备较为广泛的应用前景。

未来将结合现阶段的模型部署方式,兼从内存访问、带宽等多方面对系统做整体优化和改进,并使用资源更为丰富的 FPGA 开发板,以提高模型的总体检测速率。

参考文献

[1] LU Xiaoyan, ZHONG Yanfei, ZHENG Zhuo. A novel global-aware deep network for road detection of very high resolution remote sensing imagery [C]//2020 IEEE International Geoscience and Remote Sensing Symposium. Waikoloa: IEEE, 2020: 2579. DOI: 10.1109/IGARSS39084.2020.9323155

[2] DONG Zhipeng, WANG Mi, WANG Yanli, et al. Object detection in high resolution remote sensing imagery based on convolutional neural networks with suitable object scale features [J]. IEEE Transactions on Geoscience and Remote Sensing, 2020, 58(3): 2104. DOI: 10.1109/TGRS.2019.2953119

[3] KRISHNAMOORTHI R. Quantizing deep convolutional networks for efficient inference: A whitepaper [Z]. arXiv: 1806.08342. 2018 [2021-07-06]. https://arxiv.org/pdf/1806.08342

[4] JACOB B, KLIGYS S, CHEN Bo, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference [C]// Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018: 2704. DOI: 10.1109/CVPR.2018.00286

[5] ZHANG Xiaofan, HAO Cong, LU Haoming, et al. SkyNet: A champion model for DAC-SDC on low power object detection [Z]. arXiv:1906.10327. 2019 [2021-07-06]. https://arxiv.org/abs/1906.10327

[6] REDMON J, FARHADI A. YOLO9000: Better, faster, stronger [C]// Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu: IEEE, 2017: 6517. DOI: 10.1109/CVPR.2017.690

[7] ZHOU Xingyi, WANG Dequan, KRÄHENBÜHL P. Objects as points [Z]. arXiv: 1904.07850, 2019 [2021-07-06]. https://arxiv.org/abs/1904.07850

[8] HE Kaiming, ZHANG Xiangyu, REN Shaoqing, et al. Deep residual learning for image recognition [C]//Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016: 770. DOI: 10.1109/CVPR.2016.90

[9] MELONI P, DERIU G, CONTI F, et al. A high-efficiency runtime reconfigurable IP for CNN acceleration on a mid-range all-programmable SoC [C]//Proceedings of 2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig). Cancun: IEEE, 2016: 1. DOI: 10.1109/ReConFig.2016.7857144

[10] LIU Ye, WANG Yin, CHANG Liang, et al. A fast and efficient FPGA-based level set hardware accelerator for image segmentation [C]//Proceedings of 2020 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA). Nanjing: IEEE, 2020: 61. DOI: 10.1109/ICTA50426.2020.9331957

[11] MA Yufei, CAO Yu, VRUDHULA S, et al. Optimizing the convolution operation to accelerate deep neural networks on FPGA [J]. IEEE Transactions on Very Large Scale Integration Systems, 2018, 26(7): 1354. DOI: 10.1109/TVLSI.2018.2815603

[12] LI Huimin, FAN Xitian, JIAO Li, et al. A high performance FPGA based accelerator for large-scale convolutional neural networks[C]// Proceedings of the 26th International Conference on Field Programmable Logic and Applications. Lausanne: IEEE, 2016: 1. DOI: 10.1109/FPL.2016.7577308

[13] 魏楚亮, 陈儒林, 高谦, 等. 基于高层次融合的卷积神经网络 FPGA 硬件加速[J]. 光学精密工程, 2020, 28(5): 1212
WEI Chuliang, CHEN Rulin, GAO Qian, et al. FPGA-based hardware acceleration for CNNs developed using high-level synthesis [J]. Optics and Precision Engineering, 2020, 28(5): 1212

[14] CHENG Gong, ZHOU Peicheng, HAN Junwei. Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images[J]. IEEE Transactions on Geoscience and Remote Sensing, 2016, 54(12): 7405. DOI: 10.1109/TGRS.2016.2601622

[15] LI Ke, CHENG Gong, BU Shuhui, et al. Rotation-insensitive and context-augmented object detection in remote sensing images [J]. IEEE Transactions on Geoscience and Remote Sensing, 2018, 56(4): 2337. DOI: 10.1109/TGRS.2017.2778300

[16] 陈辰, 柴志雷, 夏珺. 基于 Zynq7000 FPGA 异构平台的 YOLOv2 加速器设计与实现[J]. 计算机科学与探索, 2019, 13(10): 1677
CHEN Chen, CHAI Zhilei, XIA Jun. Design and implementation of YOLOv2 accelerator based on Zynq7000 FPGA heterogeneous platform [J]. Journal of Frontiers of Computer Science and Technology, 2019, 13(10): 1677

（编辑　苗秀芝）

（上接第 23 页）

[8] CHEN Jianxin, ZHU Lizhong. Heterogeneous UV-fenton catalytic degradation of dyestuff in water with hydroxyl-Fe pillared bentonite [J]. Catalysis Today, 2007, 126(3/4): 463. DOI: 10.1016/j.cattod.2007.06.022

[9] FENG Jiyun, HU Xijun, YUE P L. Novel bentonite clay-based Fe-nanocomposite as a heterogeneous catalyst for photo-fenton discoloration and mineralization of orange II [J]. Environmental Science & Technology, 2004, 38(1): 269. DOI: 10.1021/es034515c

[10] WU Junxue, CAGNETTA G, WANG Bin, et al. Efficient degradation of carbamazepine by organo-montmorillonite supported $nCoFe_2O_4$-activated peroxymonosulfate process [J]. Chemical Engineering Journal, 2019, 368: 824. DOI: 10.1016/j.cej.2019.02.137

[11] CHEN Qiuqiang, WU Pingxiao, LI Yuanyuan, et al. Heterogeneous photo-Fenton photodegradation of reactive brilliant orange X-GN over iron-pillared montmorillonite under visible irradiation[J]. Journal of Hazardous Materials, 2009, 168(2/3): 901. DOI: 10.1016/j.jhazmat.2009.02.107

[12] JIA Hanzhong, ZHAO Song, SHI Yafang, et al. Mechanisms for light-driven evolution of environmentally persistent free radicals and photolytic degradation of PAHs on Fe(III)-montmorillonite surface [J]. Journal of Hazardous Materials, 2019, 362: 92. DOI: 10.1016/j.jhazmat.2018.09.019

[13] XU Haodan, WANG Da, MA Jun, et al. A superior active and stable spinel sulfide for catalytic peroxymonosulfate oxidation of bisphenol S[J]. Applied Catalysis B: Environmental, 2018, 238: 557. DOI: 10.1016/j.apcatb.2018.07.058

[14] GHAUCH A, BAALBAKI A, AMASHA M, et al. Contribution of persulfate in UV-254nm activated systems for complete degradation of chloramphenicol antibiotic in water[J]. Chemical Engineering Journal, 2017, 317: 1012. DOI: 10.1016/j.cej.2017.02.133

[15] WANG Panpan, LIU Xiaolin, QIU Wei, et al. Catalytic degradation of micropollutant by peroxymonosulfate activation through Fe(III)/Fe(II) cycle confined in the nanoscale interlayer of Fe(III)-saturated montmorillonite[J]. Water Research, 2020, 182: 116030. DOI: 10.1016/j.watres.2020.116030

（编辑　苗秀芝）