光电技术及应用

DOI：10.16818/j.issn1001－5868.2023122501

基于Zynq平台的低功耗人脸检测加速系统

赵　民[1,2]，徐　胜[1,2]，韩路宇[1,2]，林志贤[1,2,3]*

（1. 福州大学 物理与信息工程学院，福州 350116；2. 中国福建光电信息科学与技术创新实验室，福州 350116；
3. 福州大学 先进制造学院，福建 泉州 362200）

摘　要： 基于CPU及GPU的卷积神经网络平台存在体积大、能耗高等问题，提出了一种基于Zynq平台的卷积神经网络人脸检测加速系统。该系统采用YOLOv3-Tiny算法，并利用Wider Face人脸数据集进行训练。为提高网络效率，采用层融合技术减小网络深度，加快检测速度；同时，采用8位整数量化策略，以降低内存访问量，减少资源消耗。通过利用ZynqXC7Z035芯片上FPGA端并行计算能力，设计出可重复利用的多通道卷积计算模块，实现DSP的重复递用。实验结果显示，所设计的加速系统实现了9.5 FPS的实时推理速度，检测速度是intel i7-8700CPU的7.9倍，系统功耗仅为2.65 W，满足低功耗的性能需求。

关键词： 卷积神经网络；层融合；量化；多通道卷积；现场可编程门阵列

中图分类号： TP391　**文章编号：** 1001－5868(2024)03－0469－08

Low-power Face Detection Acceleration System Based on the Zynq Platform

ZHAO Min[1,2], XU Sheng[1,2], HAN Luyu[1,2], LIN Zhixian[1,2,3]

(1. College of Physics and Information Engineering, Fuzhou University, Fuzhou 350108, CHN;
2. Fujian Science & Technology Innovation Laboratory for Optoelectronic Information of China, Fuzhou 350108, CHN;
3. School of Advanced Manufacturing, Fuzhou University, Quanzhou 362200, CHN)

Abstract: To address the issues of large size and high power consumption in CPU- and GPU-based convolutional neural network platforms, we designed and implemented a convolutional neural network-assisted face detection acceleration system based on the Zynq platform in this study. We adopted the YOLOv3-Tiny algorithm for the proposed system and used the WIDER FACE dataset for training. To improve the network efficiency, we utilized a layer-fusion technique for reducing the network depth and accelerating detection. Moreover, we employed an 8-bit integer quantization strategy to minimize memory access and resource consumption. We designed a reusable multichannel convolution computation module by leveraging the parallel computing capability of field-programmable gate arrays (FPGAs) on the ZynqXC7Z035 chip to reuse the digital signal processor (DSP). The experimental results showed that our designed acceleration system, which could achieve a real-time inference speed of 9.5 FPS, was 7.9 times faster than intel i7-8700CPU and consumed only 2.65 W of power, satisfying the performance requirement of low power consumption.

Key words: convolutional neural network; layer fusion; quantization; multichannel convolution; FPGA

收稿日期：2023－12－25.

基金项目：国家重点研发计划项目(2021YFB3600603)；福建省自然科学基金项目(2020J01468).

* **通信作者：**林志贤

0 引言

人脸检测是计算机视觉领域的一个热点问题，其在监控系统、公共安防、考勤门禁、机器人导航等多个领域有着广泛应用，为人们的日常活动提供了便利。随着应用场景的多样化，在嵌入式系统上实现人脸检测功能的需求不断增加。然而，实时处理是人机交互的基本要求，而传统的人脸检测方法往往计算成本高、功耗大，这与实时嵌入式系统的特点不相适应。因此，引入现场可编程门阵列（FPGA）来实现低功耗、低响应时间的实时嵌入式系统，具有重要的研究价值。

早期，人脸检测方法主要基于面部的几何特征（边缘形状、纹理等）来进行识别[1]，随后发展到使用图像的统计特征来进行人脸检测[2]。然而，这些方法在处理非线性结构时表现不佳。

现阶段，卷积神经网络（Convolutional Neural Networks，CNN）在目标检测方面性能卓越，CNN可以通过使用大量数据和多层神经网络自动提取特征并进行分类，相较于传统方法有更高的准确性和效率。然而，CNN算法同样面临一些挑战，如密集型的计算空间复杂度高、对计算性能的要求严格，这都给在移动设备上的实现CNN算法带来了困难。因此，如何提高CNN算法的运行效率、降低执行时间和减少功率损耗，是CNN算法的重要研究方向。

为了提高CNN算法的实时性，可使用GPU对其进行加速，但GPU的体积大、功耗高，并不适合应用于移动设备中。FPGA作为一种半定制集成电路，具有低功耗、低延时，以及灵活性好等优点，逐渐被用于加速CNN的前向推理过程。与传统复杂可编程逻辑器件（Complex Programmable Logic Device，CPLD）相比，FPGA规模更大，门电路资源更加丰富。赛灵思公司的Zynq[3] FPGA系列芯片采用ARM+FPGA的系统架构，能够从软、硬件方面联合开发，开发效率高，可实现CNN模型快速部署和算法实现。

基于CNN的人脸检测模型的参数量大、训练速度慢、成本高，以及对设备要求高等问题，采用Zynq平台，探索在片上资源有限的FPGA上对CNN网络进行针对性优化。通过量化方法减少FPGA资源消耗，降低网络前向推理计算的复杂度，最终实现一种基于轻量级CNN算法（YOLOv3-Tiny）的嵌入式人脸检测系统，达到高性能、低功耗的技术指标。

1 目标检测算法

1.1 算法比较

在开始设计基于FPGA的卷积神经网络模型之前，需要为系统选择合适的目标检测算法。深度学习的目标检测算法可以分为两类，即一阶段（One-Stage）目标检测算法和二阶段（Two-Stage）目标检测算法。一阶段目标检测算法将检测问题视为回归问题，直接用卷积网络输出目标的坐标和类别，具有较快的检测速度，但检测精度较低，代表性算法包括YOLO，SSD，YOLOv3-Tiny等[4-5]；二阶段目标检测算法首先生成候选区域，再对所选区域进行分类和回归，检测精度高但检测速度慢，以FAST-RCNN[6]算法为典型代表。

表1展示了几种检测算法的检测效果对比，检测精度越高，检测效果越好；检测时间越短，检测速度越快。从表1可以看出，YOLOv3和YOLOv3-Tiny相比其他算法，具有更高的检测速度。虽然YOLOv3检测精度也较高，但YOLOv3-Tiny的模型结构更轻量化。考虑到实时嵌入式系统对低响应时间的需求，需要选择检测速度较快的算法。因此，YOLOv3-Tiny更适合部署在嵌入式设备上。综上，本文采用YOLOv3-Tiny算法。

表1 算法检测效果

算法类型	检测精度	检测时间/ms
RetinaNet	0.894	198
R-FCN	0.865	172
DSSD	0.782	85
SSD	0.776	61
YOLOv3	0.787	51
YOLOv3-Tiny	0.722	4

1.2 YOLOv3-Tiny算法模型

YOLOv3-Tiny网络的结构如图1所示。其中CBL模块由卷积层、批量归一化层（Batch Normalization，BN）与LeakReLU激活层组合而成。BN归一化层的作用是使每个批次的输入数据服从正态分布，LeakReLU激活层的作用是避免训练过程中的梯度消失。网络中使用了两种尺寸的卷积核，分别为3×3和1×1。3×3卷积核主要用于提取图像的特征信息，1×1卷积核主要用于增加特征的非线性，从而降低网络的参数量。

YOLOv3-Tiny网络的特征提取器由7个CBL

模块和 6 个最大池化层组成，输入数据的尺寸为 416×416×3，经过 CBL 模块与最大池化层的处理后，得到 13×13 尺度上的 YOLO1 预测层，该层使用上采样操作得到 26×26×128 尺度上的特征图。将该特征图与上层特征图进行拼接可以获得更丰富的语义信息，拼接后的特征图尺寸为 26×26×384，作为 YOLO2 预测层的输入。通过这种方式，YOLOv3-Tiny 网络可以在两个尺度上进行目标预测，分别为 13×13 和 26×26。13×13 的预测层可以检测较大的目标，而 26×26 的预测层可以检测较小的目标。

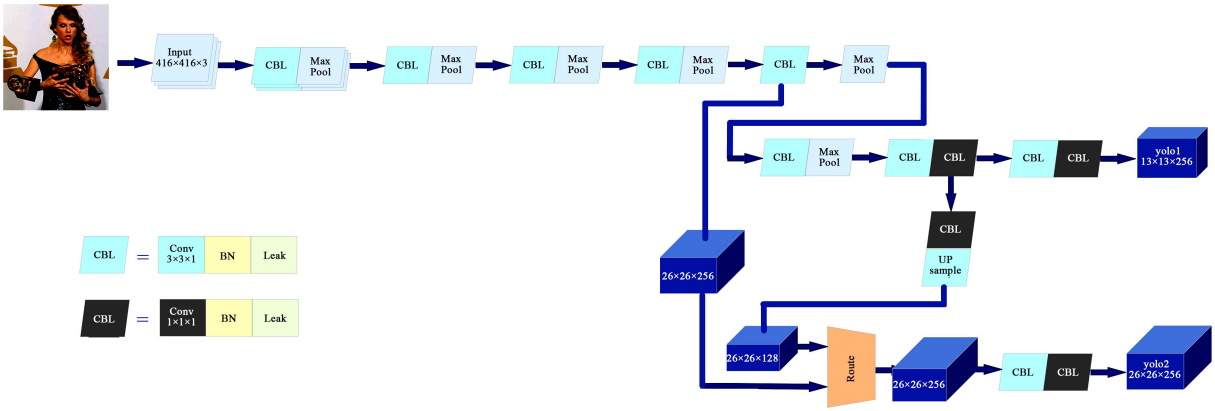


图 1　YOLOv3-Tiny 网络结构

2　优化设计

2.1　融合卷积层与 BN 层

BN 层在深度神经网络训练过程中能够加速网络的收敛过程，提高训练的稳定性，并有效防止过拟合。BN 层通过将数据归一化，可以缓解梯度消失和梯度爆炸的问题。然而，BN 层在训练过程中虽然有利，但在网络前向推理时，增加了运算层数，影响了模型的性能，同时也占用了更多的内存或显存资源。

为了提高网络的前向推理速度，本文采用了一种融合卷积层与 BN 层的策略，通过减少网络的层数来提高检测速度[7]。

对于 Batch 中的第 i 个样本，假设输入样本为 x_i，通过 BN 层后输出为 y_i。BN 层公式如下：

$$y_i=\gamma\frac{x_i-\mu}{\sqrt{\sigma^2+\varepsilon}}+\beta \tag{1}$$

式中，μ 为 Batch 中输入样本的均值，σ^2 为 Batch 内输入样本的方差，ε 为一个极小常数，用来避免除零错误。γ，β 为训练的参数，通过损失函数来优化。在神经网络的训练过程中，μ，σ^2，γ 和 β 都是变量，训练完成后，这四个参数值都会确定。因此在进行推理时，BN 层的公式可以简化为：

$$y_i=\frac{\gamma}{\sqrt{\sigma^2+\varepsilon}}x_i+\left(\beta-\frac{\gamma\mu}{\sqrt{\sigma^2+n}}\right) \tag{2}$$

令：

$$\alpha=\frac{\gamma}{\sqrt{\sigma^2+\varepsilon}} \tag{3}$$

$$b=\beta-\frac{\gamma\mu}{\sqrt{\sigma^2+n}} \tag{4}$$

联立公式(2)～(4)可得：

$$y_i=\alpha x_i+b \tag{5}$$

因此，在网络进行推理时，可以将 BN 层视为线性运算操作。

卷积层的输入输出公式如下：

$$y_i=wx_i+c \tag{6}$$

式中，w 和 c 分别表示卷积层的权重和偏置。由于式(5)、(6)都是线性表达式，可以将卷积层和 BN 层进行融合，公式如下：

$$y_i=(aw)x_i+(b+ac) \tag{7}$$

通过将卷积层与 BN 层融合，可以在降低模型参数量的同时，减小前向推理时网络的深度，提高模型推理速度。虽然在融合后参数量和浮点数变化不大，但每张输入图片的推理时间得到了相应减少。

表 2 展示了 Resnet50 网络在层融合前后的计算时间对比。在显卡为 GTX1080Ti 和 i7CPU 的实验环境下，层融合后的 Resnet50 网络在 CPU 前向计算中，平均推理时间减少了 14.48 ms，模型计算效率提升了 8.96%。在 GPU 前向计算中，推理时间减少了 3.73 ms，模型计算效率提升了 33.27%。

这些结果表明融合卷积层和 BN 层在卷积神经网络中的有效性，能够显著提高网络的检测速度，从而更好地满足对检测速度的实际需求。

表 2 Resnet50 合并卷积层与 BN 层计算时间比较

模型	CPU 计算时间/ms	GPU 计算时间/ms
Resnet50（合并前）	176.17	11.03
Resnet50（合并后）	161.69	7.3

2.2 量化设计

CNN 的优异性能往往伴随着较高的计算复杂度和巨大的内存消耗。随着网络规模的扩大，网络参数量的增加导致乘加计算量非常大，这使得在资源受限的嵌入式设备上部署神经网络难度较大。而对于一些实时性要求较高的工作场景，如自动驾驶，为了进行实时的网络推理，需要消耗计算资源以提高推理速度，否则可能造成安全隐患。

因此，当前需要研究的技术问题之一是如何在网络性能和计算成本之间找到合适的平衡点，既能保证模型精度，又能消除网络中的冗余。

模型量化是一种将神经网络中的参数（weight，bias）从高精度转换成低精度的过程，该过程可以有效降低资源消耗。训练后的模型每个层的权重参数 weight，bias 都是固定值，因此适合进行量化处理。本文采用整数量化的方法，将原本以 32 位浮点表示的参数用更低比特位宽的整数表示，从而减小模型大小、提升运算效率，可有效解决 CNN 模型部署难的问题。

浮点数转定点数的公式如下：

$$r = S(q - Z) \tag{8}$$

式中，r 表示原始的浮点数参数值；S 表示换算系数，是浮点数；q 为量化后的整数，是 int8 类型的定点数；Z 是零点，是定点数，代表浮点数的 0 值映射到定点数上的值。

换算系数 S 的计算公式如下：

$$S = \frac{r_{\max} - r_{\min}}{q_{\max} - q_{\min}} \tag{9}$$

定点数零点 Z 的计算公式如下：

$$Z = round\left(q_{\max} - \frac{r_{\max}}{S}\right) \tag{10}$$

利用 PyTorch 框架对卷积网络进行训练后，可以统计出浮点参数的最大值和最小值，因此 $r_{\max}$ 和 $r_{\min}$ 是已知的。本文采用的是 8 位整数量化方式，在确定参数的符号类型后，$q_{\min}$ 和 $q_{\max}$ 的值也可以确定，因此换算系数 S 的值可以在规定好定点数类型后通过 PyTorch 计算出来。计算得到的 S 值可以代入公式(10)中确定 Z 的值。最终将 S 和 Z 代入公式(8)即可得到浮点数对应的定点数。

对于现场可编程门阵列，相比于 32 位浮点运算，采用 8 位整数运算的功耗仅为原来的三十分之一，面积仅为原来的一百一十六分之一[6]，可显著提高计算吞吐量。通过采用 8 位位宽对 CNN 模型进行量化，是减少内存访问量从而实现更快计算的有效方法。

3 硬件加速设计

3.1 加速方案介绍

本文利用 Zynq 硬件开发平台，设计了一种基于 CNN 的目标检测加速方案。该方案将处理器端（ps）用于调度和解析预测层的输出结果，包括图像数据的调度和 CNN 检测结果的处理，将可编程逻辑资源（pl）用于执行卷积、激活、池化的计算任务。该方案的整体架构如图 2 所示。

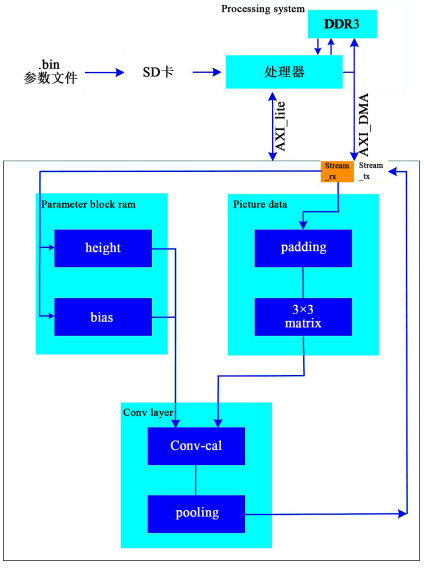


图 2 加速方案

使用 PyTorch 框架训练 CNN 模型，并将得到的参数信息保存在 bin 文件中，通过 SD 卡使 ps 端读取 bin 文件数据并将其存储在 DDR3 中，以供下一层计算使用。最后，由 ps 端调度并解析 DDR3 中 YOLO1 层与 YOLO2 层的数据，得到目标检测框的位置参数。

在数据交互方面，采用 AXI_DMA 接口 IP 实现 ps 端与 pl 端之间的大数据量传输，stream_rx 接

口负责接收从 AXI_DMA 发送过来的 stream 流数据，stream_tx 接口负责将 pl 端计算完成的数据通过 AXI_DMA IP 返回 ps 端，并存储在 DDR3 中。对于小数据量的简单通信，使用 AXI_lite 协议交互，AXI_lite 发送命令指示 pl 端在 AXI_DMA 接收数据的类型。对于权重和偏置等参数数据，在传输完成之前，要先缓存在 pl 端的 Block Ram 中等待使用。在参数数据布置完成之后，便开始向 pl 端输入图像数据，传输的 416×416 图像经过边缘填充后，再通过3×3矩阵窗口，并与参数数据进行卷积、池化操作，最后计算完成的结果通过 AXI_DMA 传输回 ps 端。

3.2　多通道卷积结构设计

为了消除 FPGA 上 DSP 资源的限制，并且进行一层卷积运算时会消耗大量乘法器，设计了一种可重复计算的卷积结构，实现 DSP 资源的循环递用。通过采用分批运算的方法，本文在有效利用板载资源的前提下，完成了卷积层的运算。多通道卷积结构如图 3 所示。

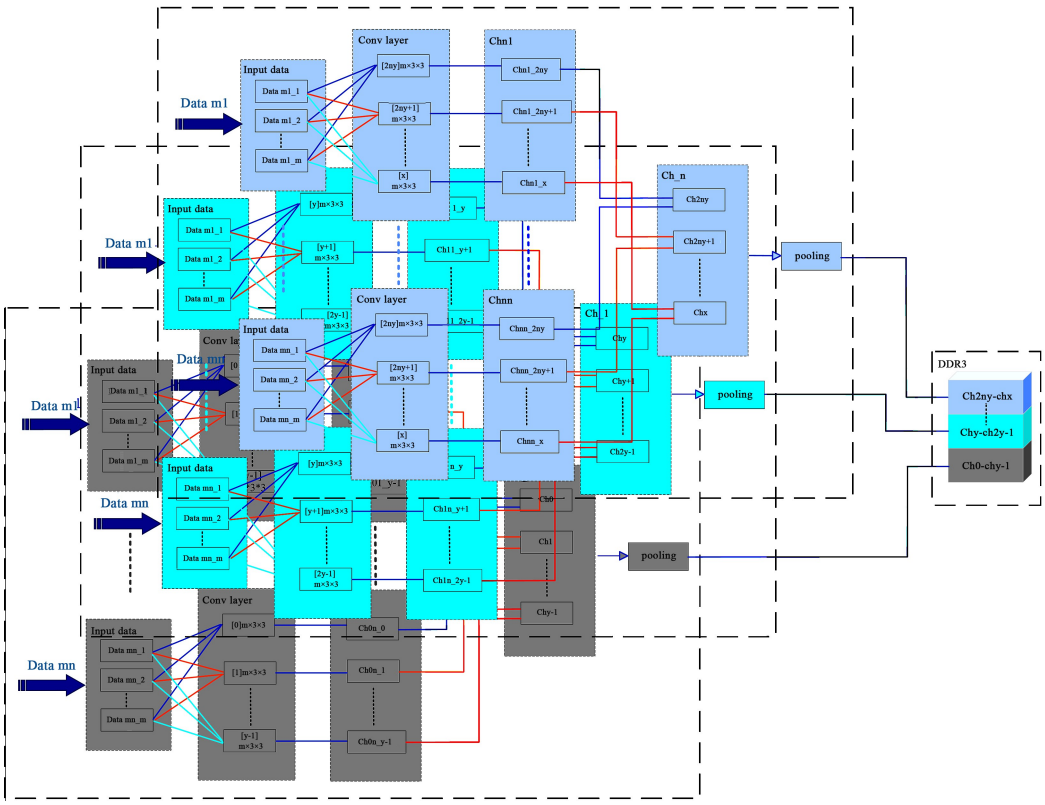


图 3　多通道卷积结构

当输入通道数量较多时，首先采取分批次写入的方式，依次对输入数据进行卷积处理。取前 m_1 个通道数据进行第一批卷积通道的计算，得到数据 ch_A，此时由于输入数据未全部参与计算，因此输出结果并不完整。本文在保持卷积通道参数数据不变的情况下，取用第二批输入通道的数据进行计算，得到数据 ch_B，重复这一操作过程直至输入数据全部计算完成。然后让各个批次的计算结果相加，最终得到输入数据在同一批卷积层计算得到的完整输出数据。

卷积核的通道数量呈 2^n 的形式出现，因此在分通道运算时也是以 2^n 的形式划分计算通道，这一设计使得卷积结构对于每一层都具有通用性，当上一层的卷积任务完成后，只需改变卷积核的权重及偏置参数，便可用于下一层的卷积计算，从而实现多通道卷积结构的重复利用。

4　实验系统搭建

4.1　数据交互接口搭建

通过处理器对控制寄存器进行一系列的配置，实现数据交互接口的搭建。将 MM2S_DMACR. RS 寄存器配置为 1，启动数据传输；将中断寄存器 MM2S_DMAC-R. IOC_irqEn 配置为 1，启动传输完成的中断。同时，通过配置地址寄存器 MM2S_

SA，控制发送数据的起始地址。配置 MM2S_LENGTH 寄存器，确定发送数据位宽。利用 Vivado 搭建 ps 处理系统，调用 DMA ip 核，便可利用 AXI_DMA 进行大数据量交互。

通过 Vivado 中的 create and package new ip 工具创建一个 AXI 外设，生成的模块包含四个 alv_reg 寄存器，四个寄存器一共有 32 个 bit 位，用于存储不同的操作信息。本文在 vitis_IDE 中通过使用 xil_out32 和 xil_in32 函数给四个寄存器赋值，实现了通过 AXI_lite 协议在 ps 与 pl 端的小数据量信息传输。

4.2 缓存模块搭建

设计 Stream_rx 模块用于接收 DMA 数据，并通过 AXI_lite 的 data_type 信号区分当前接收的数据类型，并产生 write_finish 信号告知 main_ctrl 模块数据接收完成，同时输出数据类型信号。

4.2.1 权重、偏置缓存模块

从 stream_rx 模块接收过来的偏置数据以及权重数据通过 RAM IP 进行缓存，以便在 pl 内部的卷积模块需要使用相应数据时在 RAM IP 核中快速读出；由于卷积加速模块采用 16 个卷积通道的并行计算策略，因此缓存模块也需要同时输出 16 通道的偏置及权重数据。

4.2.2 激活缓存模块

本文通过设计激活模块，存储激活函数的查找表数据，当执行激活处理时，将输入信号作为缓存数据的读地址，以输出激活处理的结果。激活函数的值由 256 个无符号整型 8 bit 数据构成，而 DMA 数据传输的位宽是 64 bit，所以激活模块一次可以缓存 8 个激活函数值。

4.2.3 卷积矩阵模块

当执行 3×3 卷积计算时，首先设计一个三行同列输出模块，用于将输入的特征图数据转换为同时输出相邻 3 行的同列数据。将该模块例化三次，进一步转换为同时输出 3×3 的卷积矩阵模块。本文使用 DSP 完成卷积操作过程中的乘法运算，利用两个权重参数同时与一个特征参数相乘，从而实现 DSP 复用。

4.2.4 累加缓存模块

为了实现分批卷积计算结果的正确输出，设计了累加模块，用于缓存分批卷积操作的中间结果，进而对这些中间结果进行累加。累加完成后的数值进入 Max_pooling 模块进行最大值池化。对于输入的一行数据 data_in_1，通过打拍方式生成数据 data_in_r1，data_in_r1 比 data_in_1 晚一个时钟周期，将两行数据进行纵向比较，得到 data_in_1 相邻两个数据之间的最大值，实现数据的最大值池化。最后通过 DMA 将缓存数据传回 ps 端的 DDR3 中。

5 实验与结果分析

5.1 模型训练

使用 Wider Face 数据集作为训练材料，该数据及包含 12 759 张训练图片和 16 127 张测试图片，通过运行 c-onvert.py 脚本文件对数据集文件进行格式转换，并将转换好的数据集进行划分得到 2007_train.txt 和 2007_val.txt 文件。采用 PyTorch 框架对改进的 YOLOv3-Tiny 模型进行训练，设置训练轮数为 300 轮，随着训练次数的不断增多，误差逐渐减小。最终得到训练好的模型参数文件，用于后续的目标检测任务。

5.2 Zynq 硬件平台上板部署

采用赛灵思公司的 Zynq7000 系列 XC7Z035 芯片作为硬件平台，该芯片内置双核 ARM-Cortex-A9CPU 内核，主频为 800 MHz。搭载 FPGA 型号为 XC7Z035-2FFG676I。

在 pl 端完成 YOLO 加速功能模块的创建后，生成 yolo_accel_top 顶层模块，将其加入到 Block Design。将相应的 bin 文件存储在 SD 卡中，在 processing system 中使 SD 卡外设，设置时钟为 200 MHz，选择 DDR3 类型为 MT41J256M16 HA_125，设置中断接口为 IRQ_F2P，高性能数据传输接口选择采用 AXI HP0 interface，通过自动连接功能，完成各接口与模块的连接。接着，执行 generate output products 操作，实现硬件产品生成。最后，启动 Vitis_IDE 创建应用工程，编写软件代码，实现 SD 卡，AXI_lite，DMA，以及中断响应的寄存器控制操作。

为了实现摄像头显示功能，将 OV5640 模块添加到 Block Design 中，该模块可以生成 24 bit 图像数据，并通过 DMA 传输到 DDR3 中。由于摄像头显示的图像尺寸为 1 280×720，而 YOLO 加速网络的输入图像数据尺寸是 416×416。因此，需要设计 Resize 模块从 1 280×720 的图像区域中选取中心的 1 248×624 区域，然后在横向上每隔两个像素点选取一个像素，再在纵向上每三行选取两行作为输出，从而得到 416×416 的图像数据。Resize 模块输

出的数据作为 YOLO 加速网络的输入数据，计算完成后通过 AXI_lite 接口将检测坐标信息传给 pl 端，实现检测框的绘制。通过 HDMI 接口，将检测后的而图像数据输出到显示屏，实时显示效果如图 4。



(a) 第 1 秒　　(b) 第 2 秒

(c) 第 3 秒

图 4　FPGA 加速器检测实例

图 4 展示了本文系统在三秒内的实时检测结果，可以看出，采用Zynq平台加速系统的检测框能够根据人脸的移动进行更新并有效跟踪，检测一帧图片的时间为 105.7 ms，相比之下，文献[8]则至少需要 355 ms，本文系统的检测速度是其 3.7 倍。intel7-8700cpu[9]的推理时间则为 828.3 ms，本文系统的检测速度更是其 7.9 倍。

表 3 为本文系统的硬件资源占用情况。

表 3　硬件资源占用情况

资源	占用资源	总资源	占用比例/%
LUT	19 396	171 900	11.28
LUTRAM	2 412	70 400	3.43
FF	24 891	343 800	7.24
BRAM	147	500	29.40
DSP	592	900	65.78

5.3　结果分析

表 4 展示了本文与其他 FPGA 平台部署卷积神经网络算法的功耗情况比较，本文部署算法的板上功耗是 2.65 W，相比其他方法具有显著的低功耗优势。

表 4　FPGA 平台部署算法的功耗情况对比

参数来源	文献[10]	文献[9]	文献[11]	文献[12]	文献[13]	本文
硬件平台	Xilinx Ultra96 v2	Ultrascale XCKU040	ZedBoard	VC707	ZCU102	XC7Z035
部署算法	YOLOv3-Tiny	YOLOv3-Tiny	YOLOv3-Tiny	YOLOv2-Tiny	YOLOv2	YOLOv3-Tiny
时钟/MHz	250	143	100	200	300	200
功耗/W	4.26	3.87	3.36	8.7	4.5	2.65

Adiono 等[10]通过将模型参数量化为 8 bit，实现了每秒 8.3 帧的检测速度和 31.5GOPS 的运算性能，但其功耗为 4.26 W，是本文的 1.61 倍。Pestana 等[9]在 LUT 和 DSP 资源利用率方面表现出较高效率，但其加速平台采用了性能较低的软核处理器，运行频率低于其他平台的硬核处理器，其功耗是本文的 1.46 倍。Yu 等[11]提出的加速框架，特点在于消除了存储模型参数和中间结果的片上内存容量的限制，从而能够在有限资源下部署 YOLOv3-Tiny 目标检测器。但其系统的检测速度只有每秒 1.88 帧，片上功率为 3.36 W，是本文的 1.27 倍。Nguyen 等[12]设计了一种专用的硬件加速系统，并采用 1～6 bit 低位数据对网络进行量化和再训练，达到了 464.67GOPS 的运算性能，但该方法需要对网络进行二次训练，增加了时间成本和功耗需求。其片上功耗为 8.7 W，相较于本文高出 2.28 倍。Nakahara 等[13]采用了二值量化方法处理算法，该方式使得加法运算可以用异或运算实现，无需设计加法器，降低了计算复杂度，但其检测精度有所损失，其片上消耗功率为 4.5 W，是本文的 1.70 倍。

6　结束语

本文基于 Zynq 平台，设计了一种采用 YOLOv3-Tiny 网络的人脸检测硬件加速系统。通过层融合技术，降低了网络深度，提高检测速度。通过 8 bit 整形量化，减少了内存访问量，降低了资源消耗。本文还设计了一种可重复利用的多通道卷积计算模块，实现了 DSP 的多次递用。通过实验测试，本文方法达到了 9.5FPS 的推理速度，其检测速度是 intel i7-8700cpu 的 7.9 倍。系统功耗仅为 2.65 W，满足低功耗的性能需求。因此，本文方法能有效解决基于深度学习的人脸检测模型参数量大、部署难的问题。

参考文献：

[1] Chandrappa D N, Ravishankar M, Ramesh Babu D R. Face detection in color images using skin color model algorithm

based on skin color information[C]// 2011 3rd Inter. Conf. on Electronics Computer Technology. IEEE, 2011: 254-258.

[2] Mathias M, Benenson R, Pedersoli M, et al. Face detection without bells and whistles[C]// Computer Vision-ECCV 2014, 2014: 720-735.

[3] Choukroun Y, Kravchik E, Yang F, et al. Low-bit quantization of neural networks for efficient inference[C]// 2019 IEEE/CVF Inter. Conf. on Computer Vision Workshop (ICCVW). IEEE, 2019: 3009-3018.

[4] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]// Computer Vision-ECCV 2016, 2016: 21-37.

[5] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]// Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, 2016: 779-788.

[6] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]// Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, 2014: 580-587.

[7] Guillemot M, Heusele C, Korichi R, et al. Breaking batch normalization for better explainability of deep neural networks through layer-wise relevance propagation [J/OL]. arXiv preprint arXiv: 2002.11018, 2020. https://ar5iv.labs.arxiv.org/html/2002.11018.

[8] 周可鑫, 左云波, 李小群, 等. 基于改进 YOLOv3-Tiny 网络的嵌入式物流商品查验系统设计与实现[J]. 计算机应用与软件, 2023, 40(9): 10-18.
Zhou Kexin, Zuo Yunbo, Li Xiaoqun, et al. Design and implementation of embeded logistic commodity inspection system based on improved Yolov3-Tiny network[J]. Computer Applications and Software, 2023, 40(9): 10-18.

[9] Pestana D, Miranda P R, Lopes J D, et al. A full featured configurable accelerator for object detection with YOLO[J]. IEEE Access, 2021, 9: 75864-75877.

[10] Adiono T, Putra A, Sutisna N, et al. Low latency YOLOv3-tiny accelerator for low-cost FPGA using general matrix multiplication principle[J]. IEEE Access, 2021, 9: 141890-141913.

[11] Yu Z, Bouganis C S. A parameterisable FPGA-tailored architecture for YOLOv3-tiny[C]// Applied Reconfigurable Computing. Architectures, Tools, and Applications, 2020: 330-344.

[12] 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 Trans. on Very Large Scale Integration (VLSI) Systems, 2019, 27(8): 1861-1873.

[13] Nakahara H, Yonekawa H, Fujii T, et al. A lightweight YOLOv2: A binarized CNN with a parallel support vector regression for an FPGA[C]// Proc. of the 2018 ACM/SIGDA Inter. Symp. on Field-Programmable Gate Arrays, 2018: 31-40.

作者简介：

赵　民(1998—)，男，硕士研究生，主要研究方向：现代光电系统；

林志贤(1975—)，男，博士，教授，主要研究方向：显示技术、平板显示驱动系统及图像处理。