doi:10.3772/j.issn.1002-0470.2024.04.003

针对嵌入式设备的 YOLO 目标检测算法改进方法①

张立国② 孟子杰③ 金 梅

(燕山大学电气工程学院 秦皇岛 066000)

摘 要 针对算法在资源有限的嵌入式设备实现困难的问题,本文基于 YOLO 系列算法提出适应嵌入式设备实现的轻量化改进方法。方法具体包括:基于 YOLOv4-Tiny 算法结构,引入 GhostNet 思想改进其网络主干,大量降低网络参数量和计算量;通过加强颈部网络特征融合效果,减少模型压缩导致的精度损失;采用训练中量化的方式将网络模型参数从 32 位浮点型数据转换为适合嵌入式设备计算的 8 位定点型参数。实验结果表明,改进后的网络在检测精度满足应用要求的情况下,模型尺寸相对原算法降低 57%,在嵌入式设备上实现功耗仅 3.795 W。

关键词 目标检测;YOLOv4-Tiny;轻量化设计;嵌入式实现;加速器

目标检测[1]是计算机图像视觉领域中重要的研究方向,现如今主流的目标检测算法已经从计算复杂且鲁棒性差的传统目标检测算法转变为基于深度学习的目标检测算法[2]。基于深度学习的目标检测算法是通过深度学习的方式,利用卷积神经网络(convolutional neural networks, CNN)提取图片特征找出图片中目标并对目标进行分类和位置标注。无论是对多类目标还是单类目标进行目标检测的算法评价指标都可以总结为检测速度和检测精度,目前学者们较为看重目标检测算法的检测精度,算法检测精度的提升通常是通过更庞大的参数量和计算量实现的。但是目标检测算法在应用实现时需要依赖嵌入式设备且有些应用环境对硬件设备条件有着苛刻的要求,如搭载在无人机设备上的航空目标检测,这种应用要求用于实现算法的平台具有便携性和低功耗的特点。而许多检测精度较高的算法,由于其用于计算的网络模型尺寸过大导致在存储和计算资源有限的嵌入式设备上难以实现[3]。所以找出一种轻量化且适合嵌入式设备实现的目标检测算法对于目标检测应用具有深远的意义。

根据计算过程不同可将基于深度学习的目标检测算法分为一阶段目标检测算法和二阶段目标检测算法。其中二阶段目标检测算法的出现时间较早,以区域卷积神经网络(region-based CNN,R-CNN)[4]、快速区域卷积神经网络(fast region-CNN, fast R-CNN)[5]和更快区域卷积神经网络(faster region-CNN, faster R-CNN)[6]算法为代表;一阶段检测算法以 YOLO[7]系列和 SSD(single shot multibox detector)[8]算法为代表。二阶段目标检测算法的计算精度较高,但算法的计算速度、参数量和计算量都不适合嵌入式设备的实现。一阶段目标检测算法相对于二阶段目标检测算法,部分算法在检测精度上相近,在检测速度上也有大幅提升。所以对于资源有限的嵌入式设备,实现目标检测算法优先选择一阶段目标检测算法。

虽然一阶段目标检测算法在嵌入式设备实现上具有一定优势,但其参数量和计算量仍是嵌入式设备实现的难题。Cheng 等人[9]提出一种增强型轻量级水上目标检测网络,利用 CSPNet(cross stage partial network)结构优化 DarkNet-53 的梯度提取能力

① 国家重点研发计划(2020YFB1711001)资助项目。
② 男,1978 年生,博士,副教授;研究方向:机器视觉,故障诊断,虚拟现实;E-mail:zlgtime@163.com。
③ 通信作者,E-mail:Mengzj0711@163.com。
(收稿日期:2023-03-22)

并减少计算量。Zhang 等人[10]对 YOLO 系列算法的第 4 代 YOLOv4 算法进行轻量化处理,采用改进的 ShuffleNet V2 替换原算法主干网络,并通过裁剪的方式压缩算法模型,提出轻量化的目标检测算法用于嵌入式设备实现。Liu 等人[11]在 YOLOv4 算法基础上采用 Mobillenet V2 作为算法主干网络,并将剩余网络结构采用深度可分离卷积代替常规卷积进行计算,引入坐标注意力机制,最终在嵌入式设备上实现轻量的红外目标检测。

上述改进方式在嵌入式设备上实现存在检测精度损失大、网络模型尺寸大的问题,为解决上述问题,本文在前人研究基础上提出针对嵌入式设备的 YOLO 目标检测算法改进方式,主要内容包括:(1)基于 YOLOv4-Tiny 算法改进其主干网络,引入 GhostNet 思想,降低网络参数量和计算量;(2)针对网络参数量和计算量下降导致的精度损失严重问题,对网络模型的颈部特征增强网络进行改进,添加自浅层向深层的特征融合,增加检测精度;(3)针对嵌入式设备计算资源,采用感知训练量化的方式训练网络模型,减少因量化导致的精度损失,同时降低嵌入式设备实现的资源消耗。

1 YOLOv4-Tiny 算法与网络模型改进

1.1 YOLOv4-Tiny 算法

YOLOv4-Tiny 算法是 YOLO 系列第 4 代算法 YOLOv4 的轻量化版本。YOLOv4 算法在检测精度和检测速度方面相对 YOLOv4-Tiny 都具有一定优势,但是受其复杂重复的网络结构和庞大的参数量影响,在嵌入式设备上实现困难。YOLOv4-Tiny 算法是一种较为适合嵌入式设备移植的目标检测算法,网络参数量相比原版的 YOLOv4 算法降低了 90%,在推理速度方面有着 6 ~ 8 倍的提升。YOLOv4-Tiny 算法的网络结构如图 1 所示。

该算法的网络模型可分为 3 个部分:主干网络、颈部网络和检测头。其中主干网络用于提取输入图像的特征,共包含 3 个卷积模块和 3 个 CSP(cross stage partial)模块。卷积模块中包含卷积层、批量化归一层和激活函数,其中激活函数采用计算方式相对简单的 Leaky-ReLU 激活函数。CSP 模块是由卷积层堆叠形成,整体网络结构共包含 15 个卷积层和 3 个最大池化层。以下从算法的检测精度和检测速度 2 个方面进行分析。首先,YOLOv4-Tiny 算法为轻量化算法,在主干网络和颈部网络上都有极大改进,相对原版的 YOLOv4 算法减少了主干网络层次,降低了大部分参数量和计算量;其网络的颈部结构仅采用自深层向浅层的特征融合,结构简单但会导致特征提取效果变差;其检测层仅保留 2 个尺度的检测头,这些算法改进会导致检测精度有所损失。但是轻量化的 YOLOv4-Tiny 算法计算参数仅有约 600 万个,网络模型尺寸约 23 MB,相比同样检测精度的算法,该算法的参数量和计算量均明显减少,所以在检测速度方面十分优秀。虽然 YOLOv4-Tiny 网络模型尺寸相对其他目标检测算法小了很多,但相对于内部存储资源有限的嵌入式设备仍然偏大,所以本文对算法的网络结构进行改进,在保证检测精度满足应用要求的基础上进一步对算法的模型尺寸进行压缩。

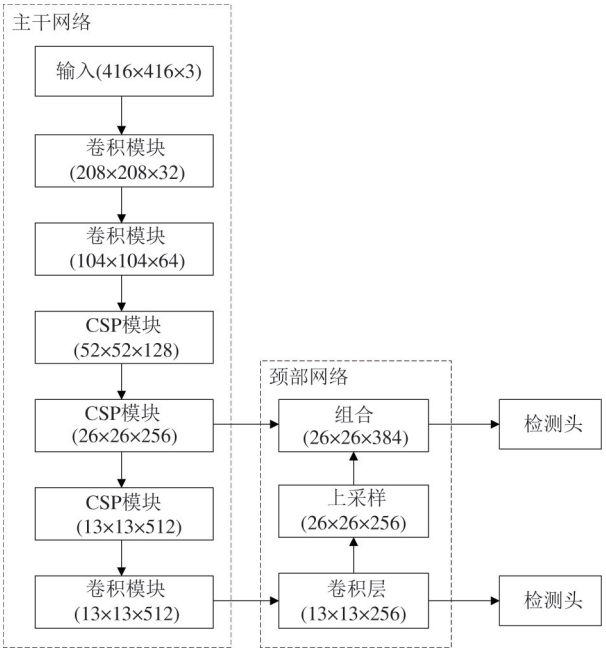

图 1 YOLOv4-Tiny 算法结构

1.2 YOLOv4-Tiny 算法网络模型改进

本文对 YOLOv4-Tiny 算法网络模型改进的目的是在保证检测精度的基础上对网络模型尺寸进行压缩,具体改进可分为 2 个方面:对主干网络的改进和

对颈部网络的改进。

1.2.1　主干网络改进

本文对 YOLOv4-Tiny 算法主干网络的改进方式是引入 GhostNet[12] 中 Ghost 卷积思想对主干网络进行压缩，并采用改进后的 GhostNet 网络代替算法原主干网络。GhostNet 是由 Ghost 卷积构成的轻量化的网络结构，其轻量化原理是减少了输入特征图在卷积层的计算量。研究发现，在常规卷积计算中，部分输出特征图之间存在相似情况，这部分相似的特征图被称为冗余数据，降低这部分冗余数据的计算复杂度就是 Ghost 卷积的轻量化方式。常规卷积和 Ghost 卷积的区别如图 2 所示，其中图 2(a) 表示常规卷积，图 2(b) 表示 Ghost 卷积。

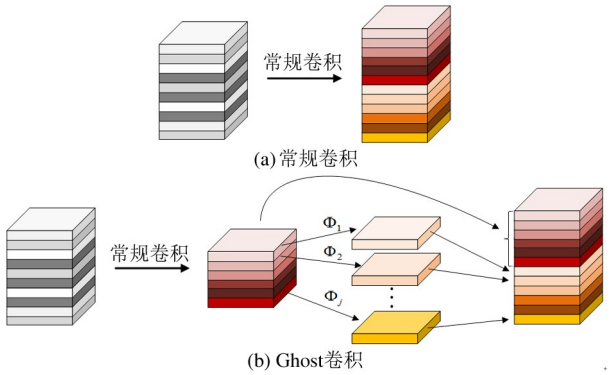

图 2　常规卷积与 Ghost 卷积

常规卷积是输入特征图直接与卷积核进行卷积计算生成全部的输出特征图。Ghost 卷积将这个过程分为 3 步：第 1 步，采用与常规卷积相同的计算方式，但只生成部分输出特征图，这部分特征图被称为本征特征图；第 2 步，对本征特征图进行相比卷积计算更简单的线性运算，生成部分输出特征图，这部分特征图被称为 Ghost 特征图；第 3 步，将本征特征图与 Ghost 特征图组合，还原输出通道数得到最后的输出。

Ghost 卷积的具体计算方式如下。

设使用 Ghost 卷积代替常规卷积生成通道数为 n 的输出特征图，首先将输出通道数分为 s 组，通过常规卷积生成通道数为 $m(m \leqslant n)$ 的本征特征图，这部分特征图的生成公式如式(1)所示。

$$Y' = X * f' + b \tag{1}$$

式中，Y' 表示本征特征图，$Y' \in R^{m\times h'\times w'}$，$m$、$h'$ 和 w' 分别表示本征特征图的通道数、高和宽；X 表示输入特征图，$X \in R^{c\times h\times w}$，$c$、$h$ 和 w 分别表示输入特征图的通道数、高和宽；f' 表示卷积核，$f' \in R^{c\times k\times k\times m}$，$c$ 表示卷积核通道数与输入特征图通道数一致，k 表示卷积核尺寸，m 表示卷积核个数；b 为偏置项；$*$ 为常规卷积计算。

为获得剩余所需特征图，需要计算 $m(s-1)$ 个线性变换，当 $s=2$ 时，对 Y' 中的特征进行简单的线性变换生成通道数为 $n/s(s-1)$ 的 Ghost 特征图，这部分特征图的生成公式如式(2)所示。

$$y_{i,j} = \Phi_{i,j}(y_i') \ \Phi_i \tag{2}$$

式中，$y_{i,j}$ 表示生成的 Ghost 特征图，i 表示该 Ghost 特征图来自于第 $i(i=1,2,3,\cdots,m)$ 通道的本征特征图，j 表示第 $j(j=1,\cdots,s)$ 通道的 Ghost 特征图；$\Phi_{i,j}$ 表示生成 Ghost 特征图 $y_{i,j}$ 的线性变换方式；y_i' 表示本征特征图 Y' 中第 i 通道的特征图。

分别对常规卷积和 Ghost 卷积的计算量和参数量进行计算，其公式如式(3)～(6)所示。

$$P_{\text{conv}} = c\times k\times k\times n \tag{3}$$

$$P_{\text{ghost}} = \frac{n}{s}\times k\times k\times c + \frac{n}{s}\times(s-1)\times d\times d \tag{4}$$

$$F_{\text{conv}} = c\times k\times k\times n\times h'\times w' \tag{5}$$

$$F_{\text{ghost}} = \frac{n}{s}\times c\times k\times k\times h'\times w' + \frac{n}{s}\times(s-1)\times d\times d\times h'\times w' \tag{6}$$

式中，P_{conv} 和 P_{ghost} 分别表示常规卷积和 Ghost 卷积的参数量；F_{conv} 和 F_{ghost} 分别表示常规卷积和 Ghost 卷积的计算量。将两者相比可知，无论参数量还是计算量 Ghost 卷积相比常规卷积均有极大程度的降低，降低倍数约 s 倍。

在 GhostNet 中存在类似 ResNet 网络结构中的瓶颈模块，如图 3 所示。图 3(a) 为步长为 1 的 Ghost 瓶颈模块，用于输入与输出特征图尺寸相同的特征提取部分；图 3(b) 为步长为 2 的 Ghost 瓶颈模块，该模块输出图像与输入图像尺寸不同，常用于网络结构中降采样的部分。本文替换后的主干网络采用上述模块进行堆叠，为进一步降低网络模型尺寸，

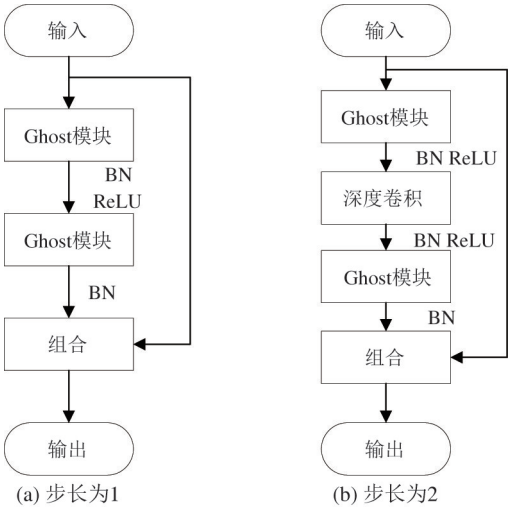

图 3　Ghost 瓶颈模块

对上述模块进行改进,采用最大池化的方式代替步长为 2 的瓶颈模块中负责调节图像尺度的深度卷积部分,改进后的瓶颈模块如图 4 所示。

改进后的网络主干如图 5 所示。其中卷积模块步长为 2,G-bneck-s1、G-bneck-s2 分别表示改进后的步长为 1 和步长为 2 的 Ghost 瓶颈模块。

1.2.2　颈部网络改进

将主干网络替换为模型尺寸更小的网络结构后会对原网络算法的检测精度造成一定损失,YOLOv4-Tiny 算法本身由于轻量化设计在检测精度方面有所降低。所以本文对 YOLOv4-Tiny 算法的颈部特征增强网络进行改进,加强其特征融合效果,

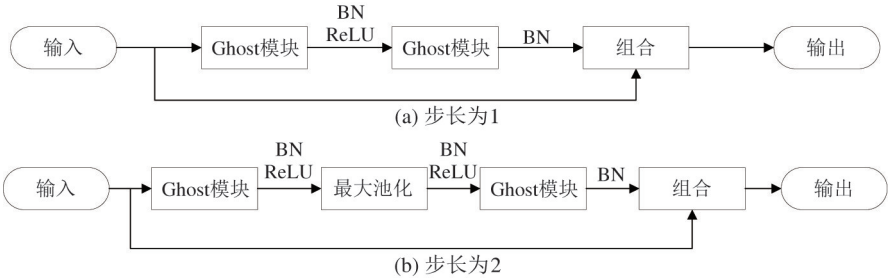

图 4　改进后瓶颈模块

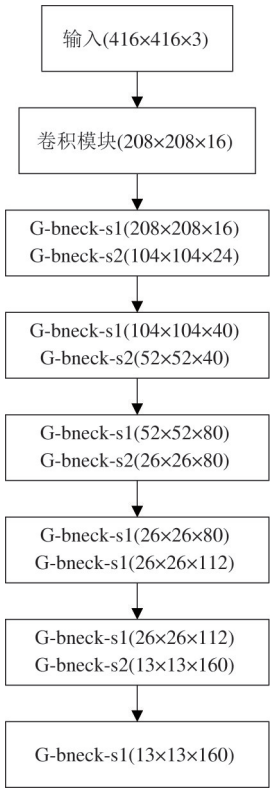

图 5　改进后主干网络结构

以提升网络检测精度。YOLOv4-Tiny 算法原本的颈部网络采用特征金字塔(feature pyramid network,FPN)结构进行特征增强,该结构中仅包含自深层向浅层的特征融合,这种操作会导致部分特征融合不完全,所提取的特征信息利用不充分。本文参考路径聚合网络(path aggregation network,PAN)结构对算法的颈部网络进行改进,改进后的颈部网络如图 6所示。

改进的颈部网络中包含上采样操作,常见的上采样操作包含插值法和转置卷积。本文改进算法方便嵌入式设备实现,上采样操作采用简单的重复插值方式完成,在嵌入式设备中仅通过行和列 2 个方向上的数据重复读取操作即可完成重复插值。重复插值的具体操作如图 7 所示。

改进后的颈部网络中增加了自浅层向深层的特征融合,对特征信息的利用程度增加,能够进一步提升算法的检测精度。虽然相对 YOLOv4-Tiny 算法增加了少量计算,但主干网络改进后对模型的压缩程度更大,所以此部分参数量和计算量的增加可以忽

略。改进后算法的整体网络结构如图 8 所示。

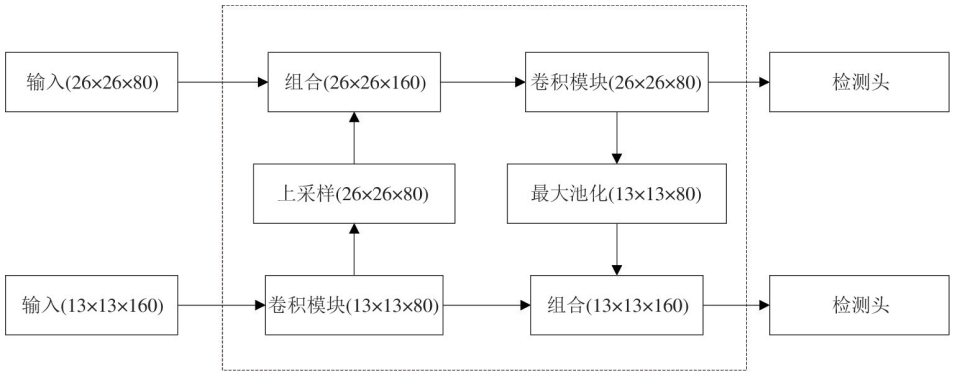

图 6　改进后的颈部网络

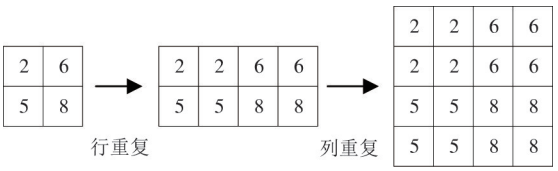

图 7　重复插值操作

2　模型量化与嵌入式设备实现

2.1　模型量化

目标检测算法的应用过程实际上就是对算法模型的前向推理过程,这一计算过程所使用的参数的数据格式多为 32 位浮点型参数,这种数据类型在嵌入式设备上进行卷积的乘法和加法运算时,对计算资源的消耗较大。同时从存储的角度看,32 位浮点型参数在存储时相对定点型参数会消耗更大的存储空间。所以本文将网络模型计算时所使用的参数量化为定点型参数。但定点型参数的数据精度较低,当替换为网络模型推理所用的数据时,会在一定程度上影响算法的检测精度。不同类型参数表达范围如表 1 所示。

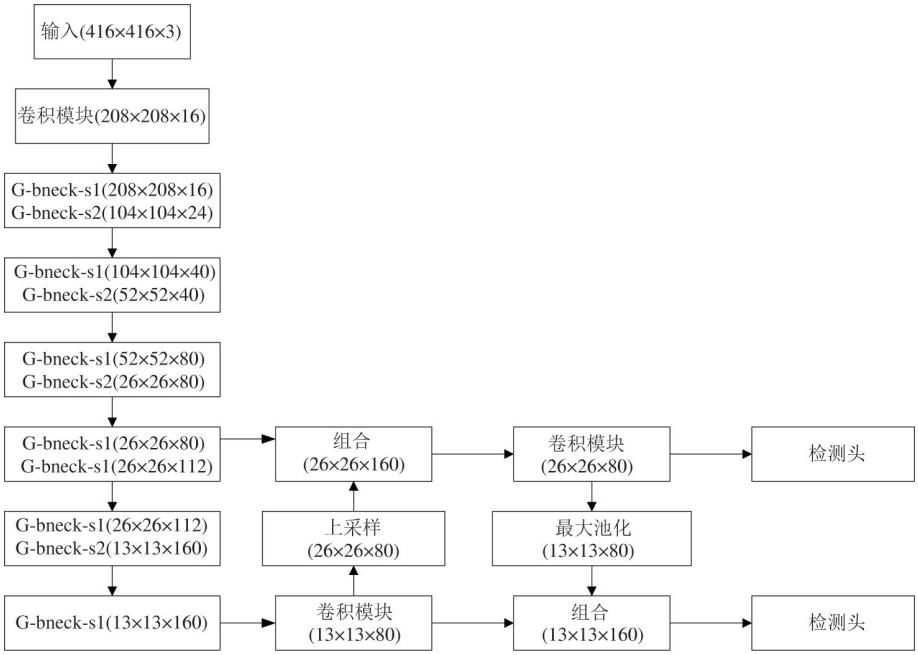

图 8　改进后算法的整体网络结构

表 1　不同类型参数表达范围

数据类型	数值区间	精度影响
32 位浮点型	$-3.4\times10^{38}\sim3.4\times10^{38}$	无
16 位定点型	−32 767 ~ 32 767	较低
8 位定点型	−127 ~ 127	较高

本文提出分段式量化方式,将数据量化[13]分为2个部分。第1部分是在模型训练过程中对网络模型的计算参数进行量化训练,这种方式可以避免在网络模型训练完成后直接进行参数量化导致的较大的算法检测精度损失;第2部分是在嵌入式设备实现过程中的参数量化,将数据量化为定点型参数后,在卷积的乘法和加法计算过程中会产生位宽超出的部分,导致增加后续计算难度,对这一部分参数采用截断式量化方式,将参数截取至量化后参数的数据位宽继续进行后续的计算操作。

本文对参数量化的第一阶段采用的训练中量化过程如图9所示。

图9　量化训练过程

首先,在原始网络模型后添加伪量化模块,将量化产生的损失代入网络训练过程。其次,进行量化训练,在训练过程中会不断调整量化所需要的参数和模型权重。最后,将模型权重和量化所需要的参数输出,再通过所选用的量化方式使用训练后的量化计算参数对权重数据进行量化计算,得到量化后的权重数据,实现对网络模型的压缩。量化计算公式如式(7)~(9)所示。

$$F = S(I - Z) \tag{7}$$

$$S = \frac{F_{\max} - F_{\min}}{I_{\max} - I_{\min}} \tag{8}$$

$$Z = round(I_{\max} - \frac{F_{\max}}{S}) \tag{9}$$

式中,F 表示量化前32位浮点型参数,I 表示量化后的定点型参数,S 表示量化计算的缩放系数,Z 表示量化后零点的位置。

当 Z 固定为0时,量化计算中仅存在一个变量缩放系数 S,将量化计算公式作为伪量化方式插入网络训练过程,获得量化后的网络计算。由于网络训练时计算结果仍需要以浮点型参数格式保存以便于进行反向传播学习,优化量化参数和权重参数,需要将量化后计算的结果进行反量化,所以在网络的最终输入后需要添加伪量化模块对计算结果进行反量化操作。当量化训练过程完成后,网络模型的各种参数仍为浮点型数据格式存储,需要对参数进行量化处理获取定点型参数,其中包括权重参数和偏置参数。

2.2　嵌入式设备实现

针对本文提出的YOLO系列算法的改进方式,进行嵌入式设备的实现,依托Xilinx公司推出的嵌入式设备ZYNQ系列开发板进行系统设计。ZYNQ系列开发板的基本架构可以分为处理系统部分(processing system,PS)、可编程逻辑部分(programming logic,PL)、AXI总线、存储以及外部接口。其中负责系统逻辑控制的PS端和系统负责计算的PL端之间通过AXI总线进行数据交互。本文依托上述架构设计基于深度学习的目标检测算法进行加速计算的卷积神经网络加速器,卷积神经网络加速器架构如图10所示。

其中PS端为设备自带的ARM处理器,负责目标检测算法网络模型各层控制参数发送和推理流程控制;PL端为卷积神经网络加速器的核心,其中控制模块负责接收PS端发送的控制指令,解析后将参数存储并发送给相应的计算模块;直接内存访问(direct memory access,DMA)负责根据PS端的配置信息,从存储空间中读取计算所需要的参数;加速计算模块中包含通过Vivado HLS工具设计的能够实现网络模型中各部分功能的计算模块,包括卷积模块、池化模块、上采样模块;中断模块是PL端计算推理过程进行完成或阶段完成后负责给PS端发送反馈信号的模块。系统的工作流程如图11所示。

3　实验结果分析

3.1　实验环境与评价指标

为验证本文改进算法在嵌入式设备上实现的有效性,进行了相关实验研究。实验环境包含软件环境和硬件设备,其中对算法有效性检测的实验环境

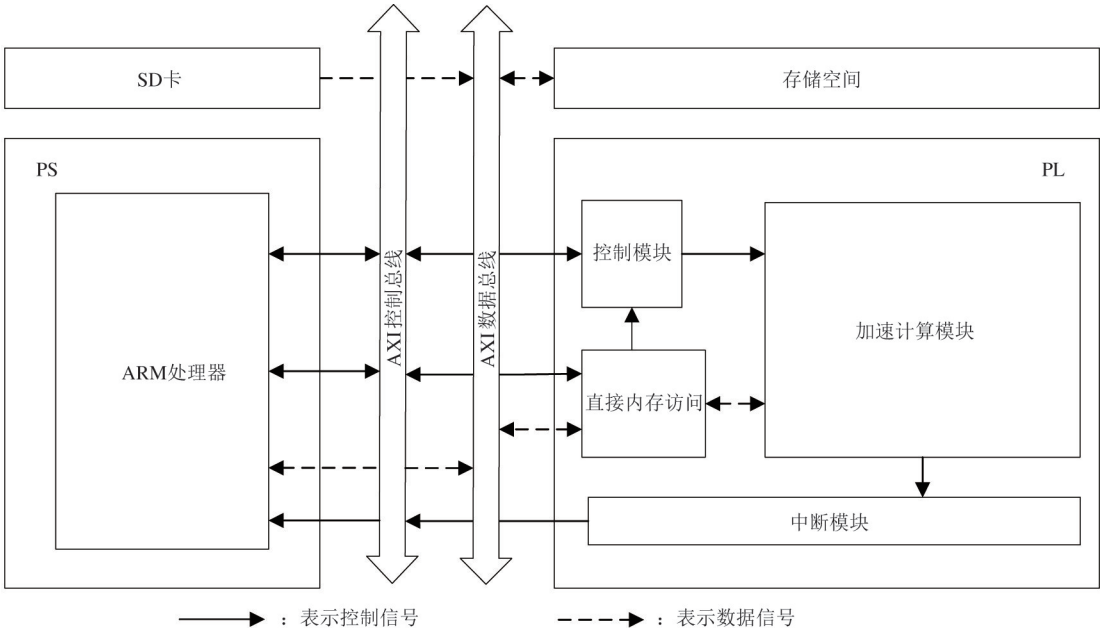

图 10　卷积神经网络加速架构

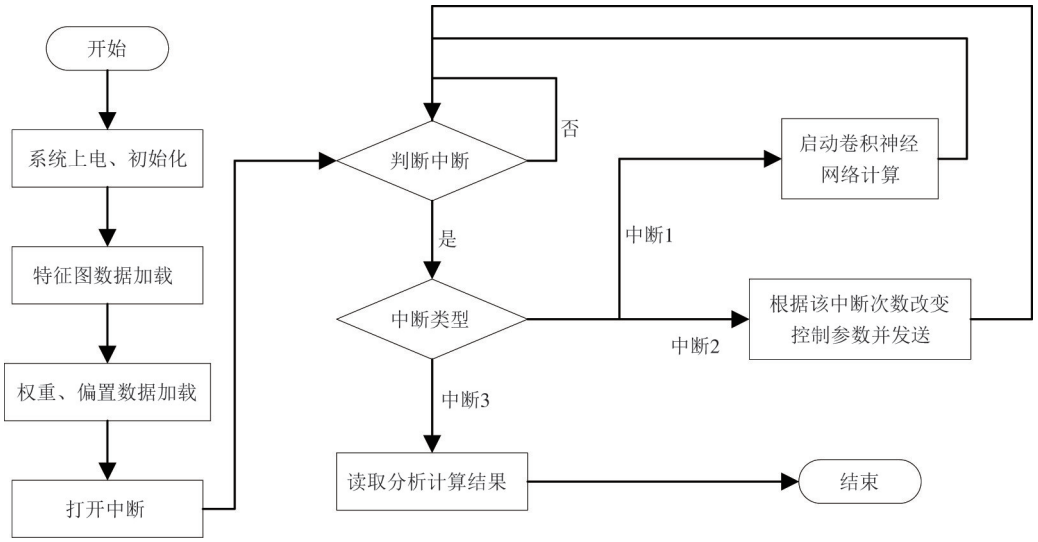

图 11　系统工作流程

为 Window 10 系统，Intel（R） Core（TM） i5-11400H 处理器，NVIDIA GeForce RTX3060 显卡，内存 16 GB。嵌入式设备选用 Xilinx 公司的 ZCU104 作为实现平台。为验证本文设计有效性，选择算法性能和嵌入式设备实现的功耗、能效比作为评价指标。

算法性能包括检测精度和检测速度，其中检测精度指标有针对单个目标的平均精度（average precision，AP）和针对多种目标的均值平均精度（mean average precision，mAP），mAP 为各类 AP 求和后的均值，AP 计算公式如式（10）~（12）所示。

$$AP = \int_0^1 P(R)\,\mathrm{d}R \tag{10}$$

$$P = \frac{TP}{TP + FP} \tag{11}$$

$$R = \frac{TP}{TP + FN} \tag{12}$$

式中，P 表示准确度，通过真正例 TP 与真正例、假正例 FP 之和的比值表示。R 表示召回率，通过真正例 TP 与真正例、假负例 FN 之和的比值表示。

本文算法改进方向是为嵌入式设备实现作准备，需要在对精度影响有限的情况下，压缩网络模型

尺寸,降低参数量和计算量,提高计算速度。因此算法性能评估主要以参数量、模型尺寸以及表示检测速度的帧率(frames per second, FPS)作为评价指标。

3.2 数据集

本文为验证改进后目标检测算法在嵌入式设备实现上的有效性,选用应用于无人机航空目标检测的数据集 VisDrone 2019 为算法训练的数据集。数据集共包含 10 个类别目标的检测,主要以行人和轿车为主。数据集中训练集、验证集和测试集的图片数分别为 8 599、548、1 580。

3.3 网络模型改进实验结果分析

为验证改进后目标检测算法的性能,证明改进后的网络更适合嵌入式设备实现,将改进后算法模型的实验结果与已有模型结果分别在参数量、模型尺寸、mAP 以及 FPS 方面进行数据对比,结果如表 2 所示。

表 2 不同算法对比

算法名称	参数量	模型尺寸/MB	mAP/%	FPS
YOLOv3	64 487 424	246.4	29.35	25.7
YOLOv4	64 040 001	244.2	31.90	22.1
YOLOv4-Tiny	5 924 454	22.6	25.78	51.2
本文算法	2 568 361	9.7	25.12	84.3

从表 2 可知,本文改进算法在参数量和模型尺寸上有明显的降低,相比于原本的 YOLOv4-Tiny 算法降低了 57%,且检测精度损失可以忽略不计。随着参数量和计算量的降低,检测速度也有大幅提升。本文中不同改进方式对算法性能的影响如表 3 所示。

表 3 不同改进方式对算法性能的影响

网络模型改进方式	参数量	模型尺寸/MB	mAP/%	FPS
YOLOv4-Tiny	5 924 454	22.6	25.78	51.2
主干网络替换	2 536 904	9.5	24.27	88.0
主干网络替换 + 特征增强优化	2 568 361	9.7	25.12	84.3

从表 3 可以看出,在仅替换主干网络不改变 YOLOv4-Tiny 算法模型其余部分的基础上进行实验,检测的精确度降低 1.51%,这部分精度损失来自于瓶颈模块,采用最大池化替换 Ghost 瓶颈模块中的深度卷积,降低了对图像中部分特征的保留,从而导致检测精度损失。为弥补主干网络替换造成的精度损失,对 YOLOv4-Tiny 的特征增强层进行优化改进,在原有的基础上增加自浅层向深层的特征融合,将二者结合完成本文改进算法的最终网络架构。实验结果表明,改进后网络参数量和计算量大幅降低,检测精度方面的损失可以忽略不计,而网络模型的检测速度明显提高,检测速度相较于原算法提升 64%。

3.4 模型压缩实验结果

本文在对网络模型实现适合嵌入式设备的轻量化改进的同时,采用模型量化的压缩方法,将网络模型参数调整为定点型数据格式。这种模型压缩方式会对算法的检测性能造成影响,不同的压缩尺寸对算法的影响也不相同,具体参数如表 4 所示。

表 4 不同尺度模型压缩对网络的影响

模型压缩尺度	模型尺寸/MB	mAP/%	FPS
未压缩	9.7	25.12	84.3
16 位定点数	4.8	25.05	95.3
8 位定点数	2.4	23.92	102.6

对 32 位浮点数进行压缩后,随着数据位宽的降低,模型尺寸也成倍缩小。算法精度随着所使用参数精度的降低而降低,但在嵌入式设备实现过程中,模型尺寸更小的网络会具有更好的推理速度,在保证检测精度满足应用条件的情况下,选择 8 位定点数进行量化完成嵌入式设备实现。

3.5 嵌入式设备实现资源利用与性能分析

3.5.1 资源利用与功耗分析

将本文改进算法在 ZCU104 开发板上实现后,资源利用情况如表 5 所示。

其中负责组合逻辑和时序逻辑功能的查找表和触发器分别占用开发板资源的 14.1% 和 9.1%。内存在模型压缩的情况下使用 54.8%,负责网络模型中卷积计算的乘法器使用占整体资源的68.2%,系

表 5 资源利用情况

	查找表	触发器	内存	乘法器
可用资源	230 400	460 800	312	1 728
使用资源	32 540	42 389	171	1 179
资源利用率	14.1%	9.1%	54.8%	68.2%

统工作的功耗为3.795 W,其中动态功耗3.100 W,静态功耗 0.696 W。

3.5.2 嵌入式设备实现与电脑端性能对比

吞吐量、功耗和能效比等指标可以证明嵌入式设备在实现目标检测算法上的优势,性能对比如表6所示。

表 6 设备性能对比

设备	数据类型	吞吐量/GOPS	功耗/W	能效比/(GOPS/W)
GPU	8 位定点数	188	154.0	1.22
ZCU104	8 位定点数	151	3.8	39.70

从表 6 中可看出,嵌入式设备实现在功耗方面相比 GPU 实现降低 97%,能效比是 GPU 的 32 倍,在目标检测算法的应用场景下嵌入式设备实现具有更好的灵活性和适应性。

3.5.3 不同实现性能对比

将本文设计的目标检测算法嵌入式设备实现方式与其他方式在嵌入式设备上实现目标检测算法进行对比,结果如表 7 所示。

表 7 不同硬件实现方式对比

	文献[14]	文献[15]	本文方法
嵌入式平台	ZCU104	Arria-10 GX1150	ZCU104
实现算法	YOLOv2	YOLOv2	改进的 YOLOv4-Tiny
工作频率/MHz	100	190	150
数据格式	16 位定点数	2、16 位定点数	8 位定点数
DSP	152	1 092	1 179
吞吐量/GOPS	28.3	566.0	151.0
功耗/W	3.98	26.00	3.80
能效比/(GOPS/W)	7.10	21.77	39.70

从表 7 中可知,文献[14]采用与本文相同的嵌入式平台对目标检测算法进行实现,本文采用的改进算法在嵌入式设备上实现时,乘法器 DSP 资源利用率更高。本文改进算法相对 YOLOv2 算法更适合嵌入式设备实现,在设备功耗相近的情况下,本文能效比是文献[14]的 5.6 倍。文献[15]选用现场可编程门阵列(field programmable gate array,FPGA)作为目标检测算法的嵌入式实现平台,其设备拥有更丰富的逻辑资源,所以在吞吐量上优于本文设计,但其系统工作时功耗更高。本文嵌入式实现所使用的 ZCU104 平台是 ZYNQ 架构,其中包含控制部分相对文献[15]所使用的平台更加灵活,所以本文能效比略优于文献[15],是文献[15]的 1.82 倍。

4 结 论

本文对 YOLOv4-Tiny 算法进行适合嵌入式设备的轻量化改进,通过对主干网络的改进大幅降低了网络模型参数量和计算量,添加自深层向浅层特征融合的颈部结构降低了由于主干改进导致的精度损失,最后通过量化训练将网络模型参数类型在精度损失较小的情况下修改为适合嵌入式设备计算的参数类型。实验结果表明,改进后的目标检测算法模型尺度更小且精度损失有限,更适合嵌入式设备实现。同时本文设计了嵌入式设备上实现的加速计算结构,能够提高算法在嵌入式设备上实现的能效比,证明本文设计为嵌入式设备实现 YOLO 系列目标检测算法提供了有效的改进方式。

本文设计仍存在提升空间,未来研究考虑引入 Winogard 计算方式降低卷积计算时对功耗较高的乘法器利用,并通过乒乓的方式优化卷积神经网络加速器的计算过程。

参考文献

[1] 卢健, 何金鑫, 李哲, 等. 基于深度学习的目标检测综述[J]. 电光与控制, 2020,27(5): 56-63.

[2] 路昊, 石敏, 李昊, 等. 基于深度学习的动态场景相机姿态估计方法[J]. 高技术通讯, 2020,30(1): 41-47.

[3] 陈朋, 何建彬, 陈诺, 等. 基于 FPGA 的视频实时目标检测方法研究[J]. 高技术通讯, 2022,32(3): 239-

247.

[4] GIRSHICK R, DONAHUE J, DARRELL T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Columbus, USA:IEEE, 2014:580-587.

[5] GIRSHICK R. Fast R-CNN[C]// Proceedings of the IEEE International Conference on Computer Vision. Boston, USA: IEEE, 2015:1440-1448.

[6] REN S Q, HE K M, 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

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

[8] ANGUELOV D, ERHAN D, et al. SSD: single shot multibox detector[C]// Computer Vision-ECCV 2016: 14th European Conference. Amsterdam, Netherlands: IEEE, 2016:21-37.

[9] 程亮, 杨渊, 张云飞,等. 面向无人艇智能感知的水上目标识别算法研究[J]. 电子测量与仪器学报, 2021, 35(9):99-104.

[10] 张宝朋, 康谦泽, 李佳萌, 等. 轻量化的 YOLOv4 目标检测算法[J]. 计算机工程, 2022,48(8):206-14.

[11] 刘冬, 李庭鑫, 杜宇, 等. 基于 MCA-YOLO 的轻量级红外实时目标检测算法[J]. 华中科技大学学报(自然科学版), 2024,52(2):35-40,46.

[12] HAN K, WANG Y, TIAN Q, et al. GhostNet: more features from cheap operations[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Seattle, USA: IEEE, 2020:1580-1589.

[13] 冯鹏程, 禹龙, 田生伟, 等. 基于均方误差的 8 位深度神经网络量化[J]. 计算机工程与设计, 2022,43(5):1258-64.

[14] 高振. 基于 FPGA 的目标检测加速器设计[D]. 杭州: 杭州电子科技大学, 2022:49-55.

[15] XU K, WANG X, LIU X, et al. A dedicated hardware accelerator for real-time acceleration of YOLOv2[J]. Journal of Real-Time Image Processing, 2021,18:481-92.

Improvement methods for YOLO object detection algorithm targeting embedded devices

ZHANG Liguo, MENG Zijie, JIN Mei

(Institute of Electrical Engineering, Yanshan University, Qinhuangdao 066000)

Abstract

To address the problem of implementing algorithms on resource-limited embedded devices, a lightweight improvement is proposed based on the YOLO series of algorithms to adapt to embedded device implementation, specifically including: improving the network backbone by introducing GhostNet ideas based on the YOLOv4-Tiny algorithm structure to significantly reduce network parameters and computational complexity; strengthening the fusion effect of neck network features to reduce accuracy loss caused by model compression; and using quantization during training to convert network model parameters from 32-bit floating-point data to 8-bit fixed-point parameters suitable for embedded device computation. Experimental results show that after the improvement in this paper, the network's model size relative to the original algorithm is reduced by 57% when the detection accuracy meets application requirements, and the power consumption for embedded device implementation is only 3.795 W.

Key words: object detection, YOLOv4-Tiny, lightweight design, embedded implementation, accelerator