一种注意力机制优化方法及硬件加速设计

王 莹[1,2],王 晶[1],高 岚[1],吕 旭[1],张伟功[1]
(1. 首都师范大学信息工程学院,北京 100048; 2. 首都师范大学数学科学学院,北京 100048)

摘 要: 针对注意力机制在卷积神经网络的应用过程中无法避免的计算量增大、延迟增加问题,本文提出一种优化后的CBAM(Convolutional Block Attention Module)算法模型,并进行了硬件设计实现. 论文基于传统CBAM模型结构,分析算法内部隐藏的潜在问题,设计更加符合注意力重要性参数提取初衷的算法模型;同时,通过计算过程优化,减少数据计算量、对算子进行最大并行组合;利用FPGA(Field Programmable Gate Array)可设计高效灵活并行阵列的优势,为改进后的CBAM算法设计一种硬件加速引擎结构. 实验结果表明,与传统CBAM机制相比,改进后的注意力机制可以保持与原有算法模型几乎相同的精度,部署在FPGA的硬件加速计算引擎以180 MHz工作频率进行推理实验,经分析可得,本文提出的设计方案在同等硬件资源条件下,针对注意力机制电路可实现10.2%的计算速度提升,针对VGG16网络模型可实现4.5%的推理速度提升.

关键词: 注意力机制;CBAM;卷积神经网络;FPGA;硬件加速器
基金项目: 国家自然科学基金面上项目(No.62076168)
中图分类号: TP302.8 **文献标识码:** A **文章编号:** 0372-2112(2023)04-1021-09
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20211229

An Improved Attention Mechanism Algorithm Model and Hardware Aceleration Design Method

WANG Ying[1,2], WANG Jing[1], GAO Lan[1], LÜ Xu[1], ZHANG Wei-gong[1]
(1. *College of Information Engineering*, *Capital Normal University*, *Beijing* 100048, *China*;
2. *School of Mathematical Science*, *Capital Normal University*, *Beijing* 100048, *China*)

Abstract: Aiming at the problem of increased calculation and delay that cannot be avoided in the application of convolutional neural network in the attention mechanism, this paper proposes an optimized CBAM (Convolutional Block Attention Module) algorithm model. Based on the traditional CBAM model structure, we analyze the hidden problems inside the algorithm, and design an algorithm model that is more fit for the original intention of attention importance parameter extraction; at the same time, through the optimization of the calculation process, the amount of data calculation is reduced, and the maximum parallel combination of operators is used; taking advantage of FPGA (Field Programmable Gate Array) to design efficient and flexible parallel arrays, we design a hardware acceleration engine structure for the improved CBAM algorithm. The experimental results show that compared with the traditional CBAM mechanism, the improved attention mechanism can maintain almost the same accuracy as the original algorithm model. The hardware accelerated computing engine deployed on the FPGA performs inference experiments at a working frequency of 180 MHz. After analysis, it can be found that the design proposed in this paper can achieve a 10.2% increase in calculation speed for the attention mechanism circuit and a 4.5% increase in inference speed for the VGG16 network model with the same hardware resources.

Key words: Attention mechanism; CBAM; convolutional neural network; FPGA; hardware accelerator
Foundation Item(s): National Natural Science Foundation of China (No.62076168)

1 引言

近年来,计算机视觉在机器人、自动驾驶、人脸识别等领域的应用卓有成效,而卷积神经网络在诸多识别任务中,获得许多出色的表现,已经成为深度学习的研究重点. 注意力机制起源于对人类视觉的研究,早期成功用于自然语言理解,后来也逐步在计算机视觉方

收稿日期:2021-09-06;修回日期:2021-12-27;责任编辑:梅志强

向得到广泛应用. 注意力机制具有即插即用的特性,通过捕捉原始特征数据的正向信息,重点区分有用数据和无关数据,由此助力卷积神经网络的特征提取,达到提高网络精度的效果. 但是,在网络结构规模逐渐变大、变复杂的情况下,将注意力机制接入到网络结构中,除了具有上述积极作用外,也伴随着副作用问题,即:网络模型的参数量和计算量增多、复杂度增加;特别是在计算资源和存储资源有限的终端设备上,问题会更加突出.

注意力机制研究方向主要包括:通道注意力、空间注意力及时空注意力. SENet[1]是典型的通道注意力模型,应用于多种基础网络,例如:SKNet[2]将SENet的通道加权设计思路和Inception[3]的多分支网络进行结合,实现了性能的提升. CBAM(Convolutional Block Attention Module)[4]模型在SENet基础上进行改进,将空间和通道注意力机制相融合,提升了模型准确性. 而BAM(Bottlenect Attention Module)机制[5]基于CBAM方法,为通道方向和空间方向选择不同的池化模式,在单张图像超分辨率研究中获得了更好的效果;文献[6]借助CBAM模块,在遥感图像中用于抑制杂乱背景信息,同时突出对象区域;RHAM-MResNet-10[7]网络模型,将残差混合注意力模块RHAM嵌入MResNet-10模型中. Non-local Neural Networks[8]是更高维度的混合注意力模型,Non-Local Block可以直接获取两个像素位置的依赖关系,融入时间因素,更适合视频图像研究. 在软硬件架构结合层面[9~11],NLP(Natural Language Processing)领域首先出现了注意力机制模型算法与硬件架构结合的研究,2020年,文献[12]介绍了A3基于注意力机制的近似加速算法和专用硬件设计;2021年,文献[13]介绍了一种名为SpAtten的算法架构协同设计方案,进一步减少了内存的访问次数. 然而,尽管注意力机制的研究在不断的丰富,在图像分类、目标识别等领域关于注意力模型算法与硬件加速器架构结合的研究尚且不多.

另外,在传统CPU(Central Processing Unit)平台上进行网络模型推理任务,已经不能满足人们对推理延迟的容忍度要求,研究重点从CPU(Graphics Processing Unit)转向具有高度并行计算能力的GPU;但由于GPU无法避免的能耗问题[14],研究重点又从GPU逐步转向了FPGA(Field Programmable Gate Array)、ASIC(Application Specific Integrated Circuit)等高性能、低功耗的硬件平台.

本文基于传统CBAM算法进行分析,提出一种改进后的注意力重要性参数提取模型,并设计实现了一种高效的注意力机制硬件加速引擎. 首先,找到通道域重要性向量和空间域重要性矩阵之间的关系,构建近似等价的计算操作,减少冗余计算. 其次,专门为注意力机制设计了硬件加速模块,通过并行电路设计减少关键路径延迟和存储器访问次数、通过数据流水设计掩盖存储器访问的时间开销;采用参数可配置的设计方式,提高注意力机制模块在硬件电路中的通用性和灵活性. 基于FPGA硬件平台进行实验,与传统CBAM算法相比,改进后的CBAM算法可以在不影响推理精度的情况下,提升神经网络的推理速度.

2 改进的CBAM算法模型

2.1 CBAM算法模型

CBAM模型的研究初衷是期望建立特征数据分别在通道域和空间域内的相互关系,改善网络的表达能力,用加权计算的方式,增强有效数据并抑制无效数据. 基于Shortcut结构[15~17]的CBAM算法计算过程如下:首先,通道域重要性提取,一个输入大小为$H\times W\times C$的原始特征数据$\boldsymbol{F}_{org}$,计算每个通道中$H\times W$个参数的全局最大池化(Max-pool)及全局平均池化(Avg-pool),得到两个$1\times1\times C$的向量;将这两个向量分别输入到一个参数共享的MLP[18~20]进行计算,之后进行对应元素加和操作,生成一个$1\times1\times C$的向量,再经过Sigmoid激活操作(图1中S表示),得到通道域注意力重要性因子向量$\boldsymbol{\beta}_{CH}$;将$\boldsymbol{\beta}_{CH}$与$\boldsymbol{F}_{org}$在相同通道上进行对应元素乘操作,得到一个临时特征数据$\boldsymbol{F}_{tmp}$,计算过程如图1所示.

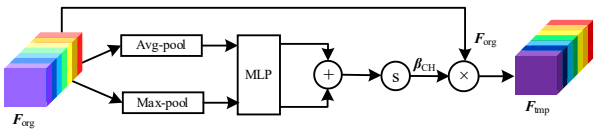

图1 通道域注意力计算过程

其次,将临时特征数据$\boldsymbol{F}_{tmp}$作为输入,分别基于C个参数计算全局最大池化(Max-pool)及全局平均池化(Avg-pool),得到两个$H\times W$的结果矩阵;将两个矩阵进行拼接操作(图2中C表示),输入到一个卷积核大小为7×7的卷积计算过程Conv,得到一个$H\times W\times1$的特征数据,再经过Sigmoid激活操作(图2中S表示),生成空间域注意力重要性因子向量$\boldsymbol{\beta}_{SP}$;将$\boldsymbol{\beta}_{SP}$与$\boldsymbol{F}_{tmp}$针对每个通道进行相同空间位置点的对应元素乘操作,最终得到一个新特征数据$\boldsymbol{F}_{new}$,计算过程如图2所示.

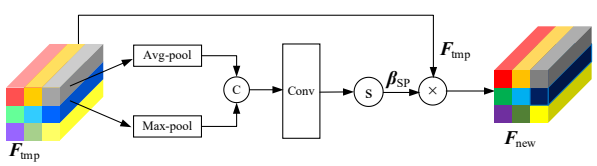

图2 空间域注意力计算过程

2.2　算法模型改进

2.2.1　共享参数合并运算

根据2.1节分析,对于原始特征数据$\boldsymbol{F}_{\text{org}}\in\mathbf{R}^{H\times W\times C}$,建立$\boldsymbol{F}_{\text{org}}$到$\boldsymbol{F}_{\text{tmp}}$的复合函数关系如下:

$$\boldsymbol{F}_{\text{tmp}}=\sigma\left(\text{MLP}\left(\boldsymbol{F}^{\text{c}}_{\text{avg}}\right)+\text{MLP}\left(\boldsymbol{F}^{\text{c}}_{\text{max}}\right)\right)\odot\boldsymbol{F}_{\text{org}}$$

$$\boldsymbol{F}^{\text{c}}_{\text{avg}}=\text{AvgPool}\left(\boldsymbol{F}_{\text{org}}\right)$$

$$\boldsymbol{F}^{\text{c}}_{\text{max}}=\text{MaxPool}\left(\boldsymbol{F}_{\text{max}}\right)$$

其中,σ表示Sigmoid函数,MLP仅包含一个隐藏层且MLP计算过程对于全局平均池化和全局最大池化两种输入是参数共享的,因此MLP计算过程满足乘法分配律,$\boldsymbol{F}_{\text{tmp}}$等价于:

$$\boldsymbol{F}_{\text{tmp}}=\sigma\left(\text{MLP}\left(\text{AvgPool}\left(\boldsymbol{F}_{\text{org}}\right)+\text{MaxPool}\left(\boldsymbol{F}_{\text{org}}\right)\right)\right)\odot\boldsymbol{F}_{\text{org}}$$

调整MLP和向量加法的运算顺序后,如图3所示,可以在保持等价变换的前提下,将MLP运算量减少50%.

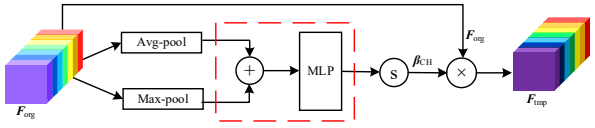

图3　变换后通道域注意力计算过程

2.2.2　叠加运算合理拆分

首先,根据2.1节分析,基础CBAM在空间域中,存在从特征数据$\boldsymbol{F}_{\text{tmp}}$到$\boldsymbol{F}_{\text{new}}$的复合函数关系,计算表达式如下:

$$\boldsymbol{F}_{\text{new}}=\sigma\left(f^{7\times7}\left(\left[\boldsymbol{F}^{\text{s}}_{\text{avg}}\cdot\boldsymbol{F}^{\text{s}}_{\text{max}}\right]\right)\right)\odot\boldsymbol{F}_{\text{tmp}}$$

$$\boldsymbol{F}^{\text{s}}_{\text{avg}}=\text{AvgPool}\left(\boldsymbol{F}_{\text{tmp}}\right)$$

$$\boldsymbol{F}^{\text{s}}_{\text{max}}=\text{MaxPool}\left(\boldsymbol{F}_{\text{tmp}}\right)$$

其中,σ表示Sigmoid函数,$\boldsymbol{F}^{\text{s}}_{\text{avg}}$和$\boldsymbol{F}^{\text{s}}_{\text{max}}$分别表示全局的空间平均池化和空间最大池化,符号[·]表示拼接操作,$f^{7\times7}$表示卷积核大小为7×7的卷积运算,⊙表示相同空间位置点的对应元素乘法操作.

其次,根据2.2.1节推导,通道域注意力重要性因子是一个具有C个元素的向量,用$\boldsymbol{\beta}_{\text{CH}}$表示:

$$\boldsymbol{\beta}_{\text{CH}}=\sigma\left(\text{MLP}\left(\text{AvgPool}\left(\boldsymbol{F}_{\text{org}}\right)+\text{MaxPool}\left(\boldsymbol{F}_{\text{org}}\right)\right)\right)$$

将$\boldsymbol{\beta}_{\text{CH}}$的每个元素都同等作用在与$\boldsymbol{F}_{\text{org}}$对应平面元素中,$\boldsymbol{F}_{\text{tmp}}$可进一步简化如下:

$$\boldsymbol{F}_{\text{tmp}}=\boldsymbol{\beta}_{\text{CH}}\odot\boldsymbol{F}_{\text{org}}$$

由此可以分析出,传统CBAM具有一个重要的隐藏含义:空间域重要性因子提取基础是各通道在不同缩放后的特征数据,并非原始特征数据.因此,在数据分布不均衡的通道中,通道域注意力重要性因子$\boldsymbol{\beta}_{\text{CH}}$对远离平均池化结果的数据,会产生与数值大小相反的作用效果,进一步代入到空间域计算时,会对空间域注意力重要性因子$\boldsymbol{\beta}_{\text{SP}}$的提取起到错误的指引作用.本文改进更为合理的注意力重要性因子提取过程原则是:无论是通道域还是空间域,均选择原始特征数据作为全局池化的对象数据;依据通道域生成的重要性向量因子$\boldsymbol{\beta}_{\text{CH}}$和空间域全局池化结果,生成空间域注意力重要性因子$\boldsymbol{\beta}_{\text{SP}}$.同时,$\boldsymbol{\beta}_{\text{SP}}$即为可以近似表示数据全局特征的全局重要性因子.

2.2.3　全局重要性因子生成

全局重要性因子$\boldsymbol{\beta}_{\text{SP}}$将通道域及空间域重要性信息相融合,由2个$H\times W$的平面拼接组成,对应最大和平均池化结果.全局重要性因子生成过程分为4步.

第1步,基于原始特征数据$\boldsymbol{F}_{\text{org}}$,在其空间域中进行全局最大池化和全局平均池化,运算结果分别定义为:$\boldsymbol{\beta}'_{\text{SP_MAX}}$和$\boldsymbol{\beta}'_{\text{SP_AVG}}$,具体如算法1所示.以全局最大池化为例:

首先,设原始特征数据为$\boldsymbol{F}_{\text{org}}\in\mathbf{R}^{H\times W\times C}$,按照$W\rightarrow H$方向逐个特征点进行滑动,按顺序依次读取$(H_i,\ W_j)$位置上所有特征数据的各个通道值$C_k$,$k\in(1,\text{Num\_}C)$,Num_$C$代表通道数量,计算每组$k$个数据的最大池化值,产生$H\times W$个最大池化结果;

其次,按照计算时的$(H_i,\ W_j)$对应位置进行存储,得到一个$H\times W$的矩阵,即为全局最大池化$\boldsymbol{\beta}'_{\text{SP_MAX}}$.

同理按照平局池化原则,计算得出全局平均池化结果为$\boldsymbol{\beta}'_{\text{SP_AVG}}$.

算法1　全局池化计算

```
FOR (i=0, i<Num_H, i++)
  FOR (j=0, j<Num_W, j++)
  {
    β'SP_MAX[i][j] = 0; β'SP_AVG[i][j] = 0;
    val_max = Forg[i][j][0]; val_avg = 0;
    FOR (k=0, k<Num_C, k++)
    {
      IF (val_max < Forg[i][j][k])
      {
        val_max = Forg[i][j][k];
        Lmax[i][j] = k;
      }
      val_avg = val_avg + Forg[i][j][k];
    }
    β'SP_MAX[i][j] = val_max;
    β'SP_AVG[i][j] = val_avg / Num_C;
  }
```

第2步,基于$\boldsymbol{\beta}'_{\text{SP_AVG}}$和$\boldsymbol{\beta}'_{\text{SP_MAX}}$,建立与1×1×$C$向量形式的通道域重要性因子$\boldsymbol{\beta}_{\text{CH}}$之间对应关系;为$\boldsymbol{\beta}'_{\text{SP_AVG}}$和$\boldsymbol{\beta}'_{\text{SP_MAX}}$中的每个数据,从$\boldsymbol{\beta}_{\text{CH}}$向量中选择合适的元素作为融合因子,融合后分别得到$\boldsymbol{\beta}_{\text{SP_AVG}}$和$\boldsymbol{\beta}_{\text{SP_MAX}}$,具体

如算法2所示.

（1） 最大池化融合因子选择和计算

在算法1中，$\boldsymbol{L}_{\max}[i][j]$记录了每个$(H_i, W_j)$位置点全局最大池化值所在通道编号，对应选择相同通道编号的$\boldsymbol{\beta}_{\mathrm{CH}}$向量元素作为该位置点的融合因子，与$\boldsymbol{\beta}'_{\mathrm{SP_MAX}}[i][j]$相乘. 完成$H\times W$个融合因子选择和计算后，得到空间域最大池化矩阵因子$\boldsymbol{\beta}_{\mathrm{SP_MAX}}$.

（2） 平均池化融合因子选择和计算

依次计算每个$\boldsymbol{\beta}'_{\mathrm{SP_AVG}}[i][j]$值与各个$\boldsymbol{\beta}_{\mathrm{CH}}$向量元素的差值diff，选择对应差值结果最小的$\boldsymbol{\beta}_{\mathrm{CH}}$作为该位置的融合因子，与$\boldsymbol{\beta}'_{\mathrm{SP_AVG}}[i][j]$相乘. 完成$H\times W$个融合因子选择和计算后，得到空间域平均池化矩阵因子$\boldsymbol{\beta}_{\mathrm{SP_AVG}}$.

算法2 全局池化因子生成

```
FOR (i=0, i<Num_H, i++)
  FOR (j=0, j<Num_W, j++)
  {
    // 最大池化融合因子选择和β'SP_MAX计算
    βSP_MAX[i][j] = βCH[Lmax[i][j]] * β'SP_MAX[i][j];
    // 平均池化融合因子选择和β'SP_AVG计算
    FOR (k=0, k<Num_C, k++)
    {
      diff = abs(β'SP_AVG[i][j] − βCH[k]);
      IF (tmp > diff)
      {
        tmp = diff;
        loc_ch = k;
      }
    }
    βSP_AVG[i][j] = βCH[loc_ch] * β'SP_AVG[i][j];
  }
```

第3步，与原始CBMA算法在空间域计算过程类似，顺序进行$\boldsymbol{\beta}_{\mathrm{SP_AVG}}$与$\boldsymbol{\beta}_{\mathrm{SP_MAX}}$的拼接操作，$f^{7\times 7}$的卷积运算及Sigmoid函数，得到空间域注意力重要性因子向量$\boldsymbol{\beta}_{\mathrm{SP}}$. 此时$\boldsymbol{\beta}_{\mathrm{SP}}$，即为可以近似表示数据全局特征的全局重要性因子.

第4步，将全局重要性因子向量$\boldsymbol{\beta}_{\mathrm{SP}}$与原始特征数据$\boldsymbol{F}_{\mathrm{org}}$进行相同空间位置点的对应元素乘法操作，最终得到新特征数据$\boldsymbol{F}_{\mathrm{new}}$.

全局重要性因子的完整生成过程，如图4所示.

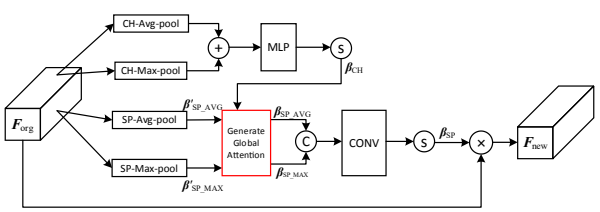

图4 基础CBAM算法叠加运算拆分过程

3 硬件加速结构设计

在通用性和灵活性上更具有优势的硬件加速结构是算子模块化和逻辑并行化[21~24]设计，将注意力机制算法过程的复合函数运算分解为子函数映射到算子模块，并行计算独立的子函数，可以有效减少数据加载次数、降低计算延迟. 同时，在算子模块内部采用流水方式[24~27]，配合适当的存储结构，进一步隐藏数据加载的延迟时间.

3.1 全局池化算法建模

设原始特征输入数据$\boldsymbol{F}_{\mathrm{org}}\in\mathbf{R}^{H\times W\times C}$，通道方向和空间方向的池化核大小分别为$H\times W$和$1\times C$. 因此，通道域全局池化结果输出为$1\times C$，空间域输出为$H\times W\times 2$，2代表平均池化运算和最大池化运算结果的拼接. 直观上看，如果只通过一次原始特征数据加载，同时完成通道域和空间域的全局池化运算，可以实现计算最大并行化及最低延迟时间.

因此，设计一个跟随顺序读入的全局池化计算过程及一个可支持多个数据并行操作的全局池化过程. 跟随读入池化针对$1\times 1\times C$的数据块进行连续的Max-pool和Avg-pool操作，得到空间注意力池化集合Atten_SP_out；同时，每C个数据按序送至多数据并行池化，等待下一次C个数据输入后进行并行Max-pool和Avg-pool操作，将两种运算结果相加，得到通道注意力池化集合Atten_CH_out，如图5所示.

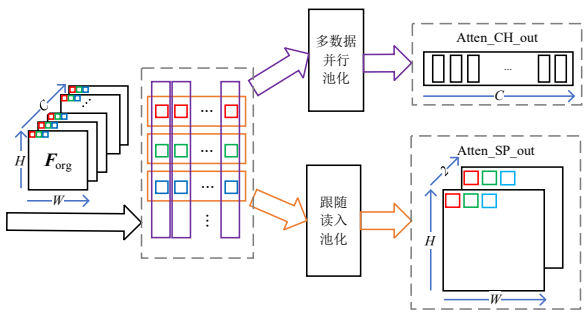

图5 全局池化算法硬件建模示意图

3.2 全局池化计算硬件结构

根据3.1节的全局池化算法，将原始特征数据按照$W\rightarrow H\rightarrow C$的顺序读入到空间域池化单元SP_Pooling和通道域池化单元CH_Pooling. 具体操作中，将一组$1\times 1\times C$的数据进行切块，每个子块大小为T_C，共需要$\lceil C/T_C\rceil$次读操作. 由于空间域池化与输入特征子块的数据流方向一致，因此采用寄存器进行数据缓存. 另外，按照$W\rightarrow H\rightarrow C$的顺序进行数据子块运算控制，需要为空间域池化设计一个片上RAM_SP存储器，保存空间域全局池化中间结果. 通道域池化需要在每读入相邻两个T_C大小数据块后，进行池化运算，为减少数据计算的延

迟等待时间,设计两个FIFO存储体组成乒乓结构,用于隐藏数据加载的延迟等待时间. 全局池化计算硬件结构见图6. 缓存空间大小取决于特征平面的数据个数,对于int16数据类型,RAM_SP容量为16×H×W(bit);FIFO空间大小取决于特征数据子块T_C长度,双缓冲结构FIFO容量为2×16×T_C(bit). 通常池化操作需要设计两级缓冲结构,对特征数据进行T_C、T_H、T_W进行子块划分. 注意力机制采用的全局池化不存在滑动步长重叠区域,因此可以直接进行数据切块划分,按照W→H→C的顺序进行数据子块加载.

单一的BRAM片上储存体容易受到访问带宽的限制,读写口数据位宽有限. 因此,在硬件电路中设计多个双口存储体,用于提高带宽和吞吐量. 具体地,对于通道域池化结果,设计T_C个位宽为int16、深度为⌈C/T_C⌉的双口RAM,A口用于全局池化结果的写入,B口用于注意力因子计算的读取,访问带宽为16×T_C(bit).

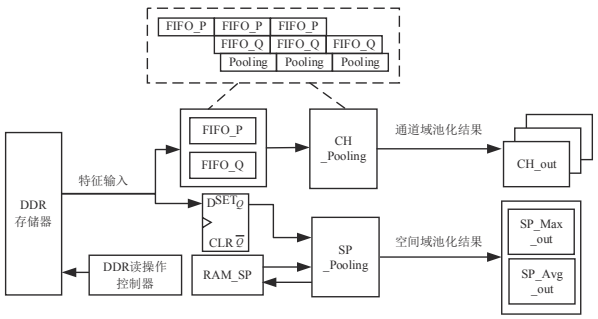

图6　全局池化计算硬件电路结构

设每次从片外加载T_C个数据共需要N个时钟周期,从第二个周期开始可以输出空间域池化临时结果,在N+1周期完成T_C个数据的池化计算,在(N+1)×H×W×⌈C/T_C⌉周期后,完成空间内特征数据全局池化;通道域池化按照T_C个数据量并行计算,通过基于并行流水的设计,可以达到与空间域计算延迟一致的时间开销.

3.3　注意力因子生成建模

3.3.1　通道域注意力因子计算

改进后的算法模型实现了计算顺序的优化,在不改变计算结果前提下,减少一半的乘法计算量. MLP属于全连接运算,设计一组可重配置的乘加器阵列,可以通过配置支持不同维度大小的向量与权重的矩阵计算,如图7所示. 首先,通道域池化结果与一组权重$\boldsymbol{W}0$进行全连接计算,得到中间临时结果tmp;其次,tmp与一组权重$\boldsymbol{W}1$进行全连接计算,之后再进行Sigmoid函数,得到通道域注意力因子集合为:

$$U_{\mathrm{CH}}=\left\{\mathrm{AttenCH}_i \middle| i\subseteq(0,C-1)\right\}$$

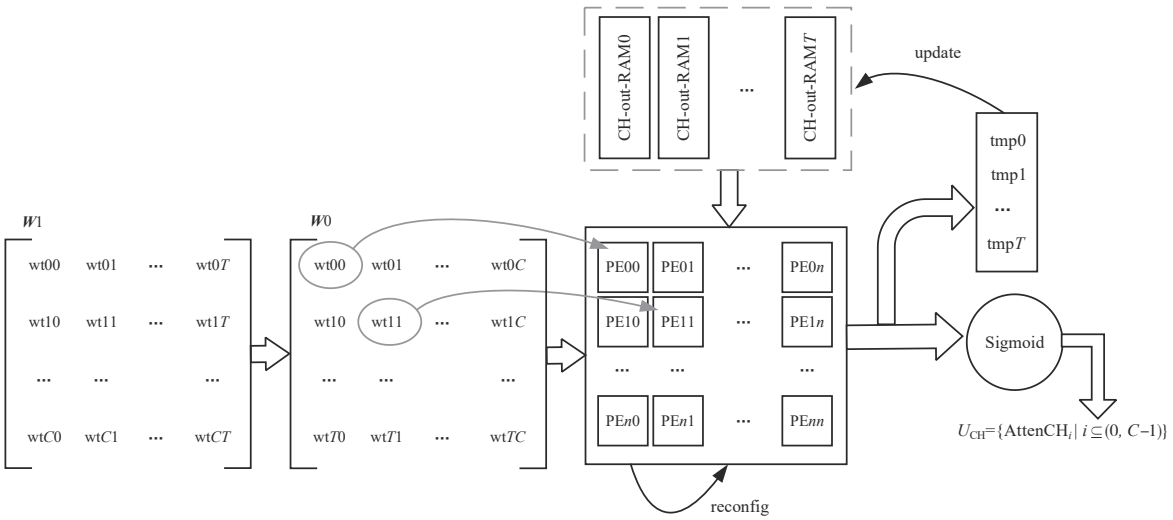

图7　乘加器阵列数据映射建模示意图

3.3.2　空间域注意力因子计算

空间域注意力池化集合Atten_SP_out是一个H×W×2的数据,分别代表H×W大小的最大池化平面和平均池化平面. 依次读取最大(平均)池化结果,计算每个空间域数据与通道域全局最大(平均)池化结果数据的差值,与C个数据比较后,找到差值最小的近似数据进行融合计算(加/乘),如图8所示.

3.4　注意力因子生成硬件结构

根据3.3节的注意力机制因子计算流程,需要将空间域全局最大(平均)池化结果数据按照W→H的顺序依次取出,每个数据和通道域最大(平均)池化结果数据进行取差值绝对值最小操作,取最接近的通道注意力作用在空间域中. 注意力因子生成的核心电路如图9所示.

完成两阶段的全局池化后,Dataflow_Seq_ctrl根据

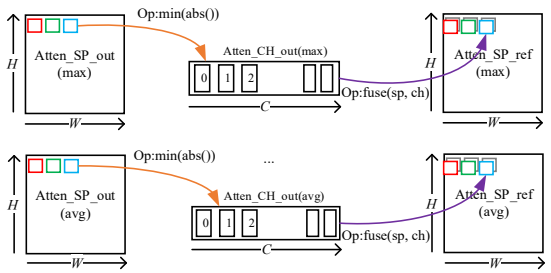

图8 融合因子计算过程建模示意图

PE-Group和Op-Group的运算操作顺序取数,计算完成后输出调节后的注意力因子SP_Max_Refine和SP_Avg_Refine. 通道域池化结果的片上存储采用多块并行BRAM,支持一次读取 T_C×16(bit)数据量,送至PE-Group作为乘加运算的一个向量输入进行MLP计算. PE-Group乘加运算的另一个输入是权重 $\boldsymbol{W}0$ 和 $\boldsymbol{W}1$,数据来源于片外存储器. 由于MLP计算是一个压缩-展开的过程,缩放系数为0.6,权重参数量Num_MLP为 $2\times C\times(0.6\times C)\times 16$(bit). 因此,复用权重参数可以有效降低数据加载的时间和能耗开销,通过 $\lceil C/T_C \rceil$ 次全局池化结果向量的轮换,完成一次权重读取的全部复用. 经过压缩和展开两次PE阵列运算后,再将得到的 C 个注意力临时结果进行Sigmoid函数变换,输出最终的通道注意力因子集合,存储在带索引的寄存器组Regs_CH_VAL中,寄存器组内数据通道连接方式如图10所示.

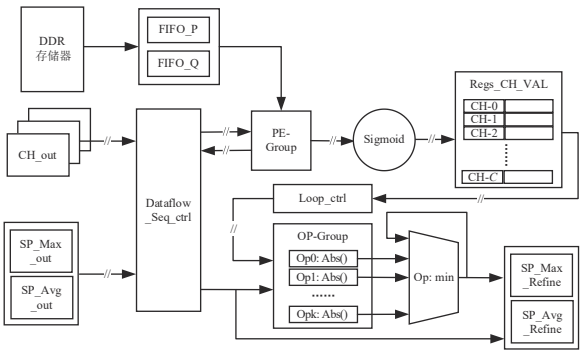

图9 注意力因子生成计算硬件电路结构

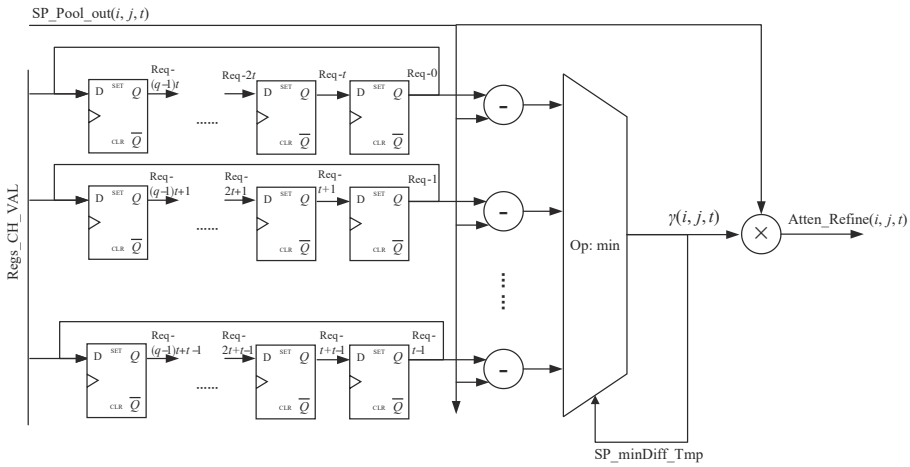

图10 寄存器组内数据通道连接方式电路图

在Loop_ctrl和Dataflow_Seq_ctrl的共同控制下,Regs_CH_VAL每次输出 t 个寄存器值,空间域池化结果每次输出一个1×1×1的数值SP_Pool_out(i,j,t),送至OP-Group,用于计算比较出数值距离最小的SP_minDiff_Tmp;Loop_ctrl控制下一组Regs_CH_VAL的 t 个寄存器值并行移位输出,与上一次结果SP_minDiff_Tmp进行迭代运算;经过 $\lceil C/k \rceil$ 次迭代后,得到与SP_Pool_out(i,j,t)更匹配的空间注意力关系值 $\gamma(i,j,t)$,计算SP_Pool_out(i,j,t)和 $\gamma(i,j,t)$ 的乘法结果后,得到一个最终的注意力机制因子Atten_Refine(i,j,t),同时触发Dataflow_Seq_ctrl模块按照 $W\rightarrow H\rightarrow C$ 顺序读取下一个SP_Pool_out(i,j,t),直到完成 $W\times H\times 2$ 个注意力机制因子Atten_Refine输出.

4 实验

为了证明文章提出改进算法的有效性和硬件设计方法的可行性,实验设计如下:首先,选用典型网络进行对比实验,证明文本提出的改进算法对网络模型精度的影响;其次,通过硬件加速器时序分析,证明进行注意力机制专用硬件加速器的可行性.

4.1 改进算法有效性验证

4.1.1 数据集与评估方法

选择标准数据集Cifar100和ImageNet-1K,进行图片分类测试;采用ResNet34和ResNet50网络模型,使用更具灵活性的PyTorch框架在NVIDIA型号为GeForce GTX TITAN X的GPU平台上进行实验;本文选择单标签图像分类任务进行实验评估,使用Top1 Accuracy和

Top5 Accuracy作为分类准确性的衡量标准.

4.1.2　模型参数设置

基于Cifar100的数据集,超参数设置为:批次大小batch_size=128,迭代次数epoch=100,权重衰减率为0.000 5的SGD优化器. 学习率的初始值为learning_rate=0.1,在每次训练周期结束时,学习率乘以0.02. 一共包含5个训练周期,每个训练周期分别迭代50、20、10、10、5次,一共100次. 基于ImageNet-1K的数据集,采用相同的超参数设置,区别是迭代次数为epoch=200.

4.1.3　实验结果

本文选择了模型深度略高的ResNet34和ResNet50进行性能对比实验,期望改进后的CBAM算法可以实现精度损失尽可能小,甚至没有精度损失,只有保证与原有CBAM相当的精度,才能不会影响网络模型性能. 为此本文进行了如下实验,结果如表1和表2所示,其中,CBAM_org代表传统CBAM算法、CBAM_ref代表改进的CBAM算法.

表1是基于Cifar100数据集的图片分类实验,实验结果及结果分析如下:

ResNet34网络与传统CBAM算法结合的Top1和Top5分类精度分别为79.71%和95.39%,ResNet34网络与改进的CBAM算法结合的Top1和Top5分类精度分别为79.56%和95.16%,可以得出改进后的CBAM算法可以保持与传统CBAM几乎相同的分类精度,下降仅为0.15%和0.23%,这样的损失在实际推理任务中可忽略.

ResNet50网络与传统CBAM算法结合可实现Top1和Top5分类精度分别为79.34%和96.01%,ResNet50网络与改进的CBAM算法结合可实现Top1和Top5分类精度分别为79.42%和96.92%,可以看出改进后的CBAM算法可以保持与传统CBAM几乎相同的分类精度,提升幅度为0.08%和0.91%.

表1　残差网络模型Cifar100分类结果对比

Model	Top1-Accuracy/%	Top5-Accuracy/%
ResNet34+CBAM_org	79.71	95.39
ResNet50+CBAM_org	79.34	96.01
ResNet34+CBAM_ref	79.56	95.16
ResNet50+CBAM_ref	79.42	96.92

表2是基于ImageNet-1K数据集的图片分类实验,实验结果及结果分析如下:

ResNet34网络与传统CBAM算法结合可实现Top1和Top5分类精度分别为74.01%和91.76%,ResNet34网络与改进的CBAM算法结合可实现Top1和Top5分类精度分别为72.52%和91.02%,可以看出改进后的CBAM算法可以保持与传统CBAM几乎相同的分类精度,精度损失仅为1.49%和0.84%,这与数据集类型规模、网络模型自身能力和训练时的参数选择有一定关系,这样的损失在实际推理任务中可忽略不计.

ResNet50网络与传统CBAM算法结合可实现Top1和Top5分类精度分别为77.34%和93.69%,ResNet50网络与改进的CBAM算法结合可实现Top1和Top5分类精度分别为76.89%和93.36%,可以看出改进后的CBAM算法可以保持与传统CBAM几乎相同的分类精度,精度损失仅为0.45%和0.33%,这样的损失在实际推理任务中可忽略不计.

此外,本文对比了基于FPGA平台上分别部署传统CBAM算法和改进后CBAM算法的推理精度,结果显示改进后的CBAM算法可以实现0.36%的小幅提升,具体见4.2.2节实验及分析.

表2　残差网络模型ImageNet-1K分类结果对比

Model	Top1-Accuracy/%	Top5-Accuracy/%
ResNet34+CBAM_org	74.01	91.76
ResNet50+CBAM_org	77.34	93.69
ResNet34+CBAM_ref	72.52	91.02
ResNet50+CBAM_ref	76.89	93.36

4.2　硬件电路功能验证

4.2.1　硬件环境

本文需要测试的注意力机制加速硬件模块,采用Xilinx公司的VCU118作为目标设计平台,使用VHDL硬件描述语言设计并搭建RTL逻辑电路. 采用硬件参数可寄存器配置的设计,数据精度选择16位定点数进行测试,核心的PE阵列大小配置为64×64的方阵形式.

4.2.2　性能评估

基于FPGA开发平台的性能测试选择具有13个卷积层、5个池化层、3个全连接层的VGG16[28]网络结构,针对第2~13个卷积层进行注意力机制模块功能接入,分别评估模型推理精度和时间开销,结果如表3所示. 表3中,VGG16+CBAM_org行给出了传统CBAM机制与VGG16结合并部署在FPGA平台的实验结果,VGG16+CBAM_ref行是改进后的CBAM机制与VGG16结合并部署在FPGA平台的实验结果. 具体分析如下.

推理精度对比:在ImageNet-1K数据集上,使用相同的权重参数,进行同等条件下传统CBAM和改进后CBAM推理精度实验,结果如表3所示. 可以看出,改进后的CBAM算法运行在FPGA硬件加速平台上可以实现推理精度的小幅度提升. 传统CBAM机制Top1精度为76.87%,改进后的CBAM机制Top1精度为77.23%,实现了约0.36%提升;传统CBAM机制Top5精度为94.52%,改进后CBAM机制Top5精度为95.05%,实现了约0.53%提升.

时间开销对比:测量传统CBAM和改进后的CBAM推理时间开销,以工作频率为180 MHz计算,结果如表3所示. 传统CBAM机制模式下,推理一张图片时间为61.725 ms,其中CBAM占用时间约26.018 ms;改进后

CBAM机制模式下,推理一张图片时间为59.069 ms,其中CBAM占用时间约23.362 ms. 经分析可得,本文提出的设计方案针对注意力机制电路可以加快10.2%的计算速度,针对VGG16网络模型可以加快4.5%的推理速度.

由此可见,本文提出的基于FPGA的硬件平台加速设计方案,采用改进后的CBAM算法,可以保持与传统CBAM算法基本一致的推理精度. 同时,可以实现推理速度的显著提高,因此该方案可行.

表3 VGG16网络模型分类精度和推理速度性能对比

Model	Top1-Accuracy/%	Top5-Accuracy/%	推理总耗时/ms	CBAM计算耗时/ms
VGG16+CBAM_org	76.87	94.52	61.725	26.018
VGG16+CBAM_ref	77.23(↑0.36%)	95.05(↑0.53%)	59.069(↓4.5%)	23.362(↓10.2%)

5 结论

本文分析了传统CBAM注意力机制算法模型隐藏的副作用问题,提出一种改进的CBAM算法模型,使其更加符合重要性提取的设计初衷. 特别是,改进后的算法实现了计算过程的并行优化,减少了MLP环节的乘加运算量. 为了更好的提高推理任务的加速性能,本文设计了一种基于FPGA的硬件加速器引擎电路,优化了大量乘加运算的计算延迟、减少了带宽访问量、有效降低了实际硬件功耗. 实验证明,改进后的算法模型,可以在保持原有传统模型算法精度不变的前提下,有效减少注意力环节计算量;基于FPGA平台部署改进后的CBAM算法模型,可以提高10.2%的注意力机制计算速度及4.5%的推理速度.

参考文献

[1] HU J, SHEN L, SUN G. Squeeze-and-excitation networks [C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 7132-7141.

[2] LI X, WANG W H, HU X L, et al. Selective kernel networks[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2019: 510-519.

[3] SZEGEDY C, VANHOUCKE V, IOFFE S, et al. Rethinking the inception architecture for computer vision[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2016: 2818-2826.

[4] WOO S, PARK J, LEE J Y, et al. CBAM: Convolutional block attention module[C]//Computer Vision-ECCV 2018. Cham: Springer International Publishing, 2018: 3-19.

[5] PARK J, WOO S, LEE J Y, et al. BAM: Bottleneck attention module[EB/OL]. (2018-07-17)[2021-09]. https://arxiv.org/abs/1807.06514.

[6] GAO G S, LIU Q J, WANG Y H. Counting dense objects in remote sensing images[EB/OL]. (2020-02-14)[2021-09]. https://arxiv.org/abs/2002.05928.

[7] 乔思波, 庞善臣, 王敏, 等. 基于残差混合注意力机制的脑部CT图像分类卷积神经网络模型[J]. 电子学报, 2021, 49(5): 984-991.
QIAO S B, PANG S C, WANG M, et al. A convolutional neural network for brain CT image classification based on residual hybrid attention mechanism[J]. Acta Electronica Sinica, 2021, 49(5): 984-991. (in Chinese)

[8] WANG X L, GIRSHICK R, GUPTA A, et al. Non-local neural networks[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 7794-7803.

[9] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[EB/OL]. (2017-06-12)[2021-09]. https://arxiv.org/abs/1706.03762.

[10] BAHDANAU D, CHO K, BENGIO Y. Neural machine translation by jointly learning to align and translate[EB/OL]. (2014-09-01)[2021-09]. https://arxiv.org/abs/1409.0473.

[11] GEHRING J, AULI M, GRANGIER D, et al. Convolutional sequence to sequence learning[EB/OL]. (2017-05-08)[2021-09]. https://arxiv.org/abs/1705.03122.

[12] HAM T J, JUNG S J, KIM S, et al. A^3: Accelerating attention mechanisms in neural networks with approximation [C]//2020 IEEE International Symposium on High Performance Computer Architecture. Piscataway: IEEE, 2020: 328-341.

[13] WANG H R, ZHANG Z K, HAN S. SpAtten: Efficient sparse attention architecture with cascade token and head pruning[EB/OL]. (2020-12-17)[2021-09]. https://arxiv.org/abs/2012.09852.

[14] HAN Y Z, HUANG G, SONG S J, et al. Dynamic neural networks: A survey[EB/OL]. (2021-02-09)[2021-09]. https://arxiv.org/abs/2102.04906.

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

[16] HE K M, ZHANG X Y, REN S Q, et al. Identity mappings in deep residual networks[C]//European Conference on Computer Vision. Cham: Springer, 2016: 630-645.

[17] XIE S N, GIRSHICK R, DOLLÁR P, et al. Aggregated residual transformations for deep neural networks[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2017: 5987-5995.

[18] TOLSTIKHIN I, HOULSBY N, KOLESNIKOV A, et al. MLP-mixer: An all-MLP architecture for vision[EB/OL]. (2021-05-04)[2021-09]. https://arxiv.org/abs/2105.01601.

[19] GUO M H, LIU Z N, MU T J, et al. Beyond self-attention: External attention using two linear layers for visual tasks [EB/OL]. (2021-05-05) [2021-09]. https://arxiv. org/abs/2105.02358.

[20] DING X H, XIA C L, ZHANG X Y, et al. RepMLP: Re-parameterizing convolutions into fully-connected layers for image recognition[EB/OL]. (2021-05-05) [2021-09]. https://arxiv.org/abs/2105.01883.

[21] 刘杰, 葛一凡, 田明, 等. 基于ZYNQ的可重构卷积神经网络加速器[J]. 电子学报, 2021, 49(4): 729-735.
LIU J, GE Y F, TIAN M, et al. Reconfigurable convolutional network accelerator based on ZYNQ[J]. Acta Electronica Sinica, 2021, 49(4): 729-735. (in Chinese)

[22] 乔瑞秀, 陈刚, 龚国良, 等. 一种高性能可重构深度卷积神经网络加速器[J]. 西安电子科技大学学报, 2019, 46(3): 130-139.
QIAO R X, CHEN G, GONG G L, et al. High performance reconfigurable accelerator for deep convolutional neural networks[J]. Journal of Xidian University, 2019, 46(3): 130-139. (in Chinese)

[23] 蹇强, 张培勇, 王雪洁. 一种可配置的CNN协加速器的FPGA实现方法[J]. 电子学报, 2019, 47(7): 1525-1531.
JIAN Q, ZHANG P Y, WANG X J. An FPGA implementation method for configurable CNN co-accelerator[J]. Acta Electronica Sinica, 2019, 47(7): 1525-1531. (in Chinese)

[24] PENG X Y, YU J X, YAO B W, et al. A review of FPGA-based custom computing architecture for convolutional neural network inference[J]. Chinese Journal of Electronics, 2021, 30(1): 1-17.

[25] CHEN Y H, KRISHNA T, EMER J, et al. 14.5 Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks[C]//2016 IEEE International Solid-State Circuits Conference. Piscataway: IEEE, 2016: 262-263.

[26] CHEN Y H, YANG T J, EMER J, et al. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019, 9(2): 292-308.

[27] 赵博雅. 基于卷积神经网络的硬件加速器设计及实现研究[D]. 哈尔滨: 哈尔滨工业大学, 2018.
ZHAO B Y. Study on Design and Implementation of Hardware Accelerators Based on Convolutional Neural Networks[D]. Harbin: Harbin Institute of Technology, 2018. (in Chinese)

[28] SIMONYAN K,ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[EB/OL]. (2014-09-14)[2021-09]. https://arxiv.org/abs/1409.1556.

作者简介

王　莹　女,1984年5月出生于北京市. 现为首都师范大学数学科学院博士研究生. 主要研究方向为高可靠嵌入式计算机体系结构和容错计算.
E-mail: wangyingstudio@163.com

王　晶　女,1982年8月出生于黑龙江省哈尔滨市. 现为首都师范大学信息工程学院教授、硕士生导师. 主要研究兴趣为计算机系统结构、智能芯片设计、高能效计算、容错计算.
E-mail: jwang@cnu.edu.cn

高　岚　女,1987年8月出生于河北省涿州市. 现为首都师范大学信息工程学院讲师,硕士生导师. 主要研究兴趣包括计算机体系结构及并行编程优化、并行编程模型、以及计算机片上存储系统的设计及优化.
E-mail: gaolan@cnu.edu.cn

吕　旭　男,1995年1月出生于江西九江. 现为首都师范大学硕士研究生,专业为高可靠嵌入式系统. 研究方向为计算机体系结构、领域专用加速器.
E-mail: introspecting2031@outlook.com

张伟功(通讯作者)　男,1967年4月出生于山西临猗. 现为首都师范大学信息工程学院教授、博士生导师. 主要研究方向为高可靠嵌入式计算机体系结构与应用技术. 中国电子学会会员编号:E190004415S.
E-mail: zwg771@cnu.edu.cn