文章编号:1671-4598(2023)06-0176-07 DOI:10.16526/j.cnki.11—4762/tp.2023.06.027 中图分类号:TP183;TN791 文献标识码:A

神经网络训练处理器的浮点运算优化架构

张立博[1],李昌伟[1],齐 伟[1],王 刚[2],戚鲁凤[2]
(1.中国绿发投资集团有限公司,北京 100010;2.山东鲁软数字科技有限公司,济南 250001)

摘要:针对神经网络训练加速器中存在权重梯度计算效率低的问题,设计了一种高性能卷积神经网络(CNN)训练处理器的浮点运算优化架构;在分析CNN训练架构基本原理的基础上,提出了包括32 bit、24 bit、16 bit和混合精度的训练优化架构,从而找到适用于低能耗且更小尺寸边缘设备的最佳浮点格式;通过现场可编程门阵列(FPGA)验证了加速器引擎可用于MNIST手写数字数据集的推理和训练,利用24 bit自定义浮点格式与16 bit脑浮点格式相结合构成混合卷积24 bit浮点格式的准确率可达到93%以上;运用台积电55 nm芯片实现优化混合精度加速器,训练每幅图像的能耗为8.51 μJ。

关键词:卷积神经网络;浮点运算;加速器;权重梯度;处理器

Floating Point Optimization Architecture of Neural Network Training Processor

ZHANG Libo[1], LI Changwei[1], QI Wei[1], WANG Gang[2], QI Lufeng[2]
(1. China Green Development Investment Group Co., Ltd., Beijing 100010, China;
2. Shandong Luruan Digital Technology Co., Ltd., Jinan 250001, China)

Abstract: Aiming at the low efficiency of weight gradient calculation in a neural network training accelerator, a floating-point operation optimization architecture based on the high performance convolutional neural network (CNN) training processor is designed. On the basic principle of CNN training architecture, a training optimization architecture with 32 bit, 24 bit, 16 bit and mixed accuracy is proposed, the best floating-point format for edge devices with low energy consumption and smaller size is found. By field programmable gate array (FPGA), the accelerator engine is used to verify the reasoning and training of MNIST handwritten digital data sets. The data with 24 bit custom floating-point format and 16 bit brain floating-point format are used to constuct that of hybrid convolution 24 bit floating-point format, which realizes the accuracy of more than 93%. TSMC 55 nm chip is used to realize the optimized hybrid accuracy of the accelerator, and the energy consumption of each image is 8.51 μJ.

Keywords: convolutional neural network; floating point operation; accelerator; weight gradient; processor

0 引言

随着人工智能(AI)蓬勃发展,网络模型的精准度不断提升,应用场景愈发广泛,但网络模型存储压力不断增加,模型运算量日益增大,资源受限的互联网设备面临新的挑战[1]。AI技术需要利用低能耗的神经网络处理器,但最近的大多数研究都集中在设计仅用于推理的加速器上[2]。然而,为了能够真正利用AI服务于现实需求,还需实现自我监督和半监督学习,这就要求神经网络处理器除了具有推理功能外,还要减少训练过程。

大多数关于神经网络加速器的研究都集中在确定输入数据准确性的正向架构和电路结构上[3]。然而,为了尽可能地模拟人类或动物的神经网络,有必要在反向传播中设计一个通过精度反馈的神经网络电路,从而能够使用尺寸更小且功耗更低的运算架构。文献[4]指出神经网络的学习可分为芯片外学习和芯片内学习。其中,芯片外学习通过借助互联网设备分散大量运算,文献[5]总结出使用边缘设备收集、处理和分析神经网络类数据更有效。文献[6]验证了与服务器重复交互的边缘设备更能显著减少电能消耗。文献[7]开发了基于树的算法用于预测资源受限的互联网设备。芯片内学习则是通过设计优化的轻量级神经网络模型在低功耗且小尺寸的芯片上实现训练[8]。

为了实现低功耗,必须为所涉及的数学计算选择适当的格式。文献[9]研究表明整数运算足以设计低功耗推理加速器,然而,还需要大量的研究来训练具有合理精度的神经网络。文献[10]的实证结果表明,训练神经网络需要至少16 bit精度。文献[11]中实现了使用整数运算的混合精度训练,将两个整数相乘并输出存储到整数累加器中。然而,整数运算符无法代表广泛的数字,严重阻碍了训练

收稿日期:2022-10-17; **修回日期**:2022-11-06。
基金项目:中国绿发投资集团有限公司科技项目(CGDG529000220008)。
作者简介:张立博(1980-),男,山东济南人,硕士,工程师,主要从事网络技术、网络安全、信息化技术等方向的研究。
引用格式:张立博,李昌伟,齐 伟,等.神经网络训练处理器的浮点运算优化架构[J].计算机测量与控制,2023,31(6):176-182.

引擎中使用。

当用于训练神经网络时，浮点运算比定点运算具有更高的精度[12]。传统的神经网络电路设计研究集中在利用 GPU 或定点计算硬件进行浮点运算。然而，大多数现有的基于浮点的神经网络仅限于推理操作，只有少数包含针对高速服务器而非低功耗移动设备的训练引擎[13]。将训练与高精度目标结合起来需要使用浮点运算符。神经网络中的高精度浮点运算结构往往会大量能耗。因此，需要设计出优化浮点运算的加速器。通过计算近似技术可以有效降低计算复杂性，最大限度地减少浮点算子的显著能耗[14]。虽然计算近似技术在能耗和吞吐量优化方面表现出较好的性能，但现有研究热点集中在反向传播期间保持权重更新的精度。文献［15］通过维护浮点权重的主副本来实现混合精度训练，并且在向前和向后传播期间使用浮点的另一副本。然后，权重梯度需要更新浮点运算主副本并在每次迭代中重复该过程。尽管减少了整体内存使用量，但保持权重的副本会将权重的内存需求增加两倍[16]，并且由于额外的内存访问将不可避免地导致整体性能发生延迟。

为了能够兼顾神经网络训练中浮点运算的训练过程和高精度，本文设计了一种具有高性能卷积神经网络（CNN）训练处理器的浮点运算优化架构。在不牺牲精度的情况下找到最佳浮点格式。利用混合精度架构，将需要更高精度的层分配了更多的存储，而需要更少精度的层分配了更少的存储。通过评估不同的浮点格式及其组合来实现浮点运算符，从而以更少的能耗实现更高的精度结果。通过 CNN 和浮点训练设计一种混合精度加速器实现更高精度的卷积块，并运用 MNIST 手写数字数据集验证了优化架构的有效性。

1　卷积神经网络（CNN）训练加速器

1.1　CNN 架构

CNN 作为人工神经网络（ANN）的分支，最常用于特征提取和分类[17]。使用 CNN 训练加速器训练电路，从而高效准确地对输入图像进行分类。CNN 训练加速器的体系结构可在正向传播过程中使用经过训练的权重从输入值推导输出，并在反向传播过程中更新权重，进而提高正向传播中下一幅图像的总体精度[18]。CNN 的总体架构，如图 1 所示。其中，Conv 为卷积，ReLu 为线性整流函数。Softmax 为归一化指数函数。

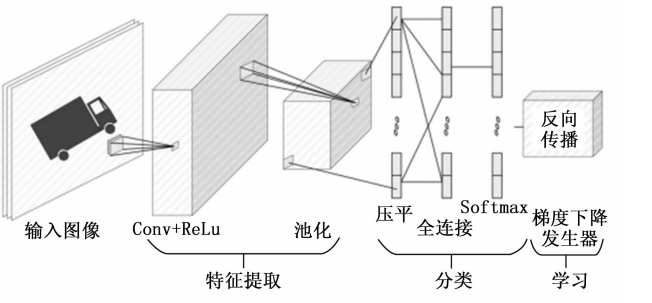


图 1　CNN 的总体架构

1.2　Softmax 模块

Softmax 函数作为激活函数，根据概率分布将网络的输出标准化为输出类。Softmax 函数将 N 个实数组成的向量 x_i 作为输入，由 N 个概率组成的概率分布建立标准化，且概率与输入值的指数成正比。因此，对每个输入向量应用标准指数函数，每个输入向量指数值除以所有累加指数对每个输入向量进行标准化。则 SoftMax 函数可以表示为：

$$S(x)_i = \frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}} \tag{1}$$

从公式（1）可以看出，计算 SoftMax 值需要 1 个除法器模块和 1 个加法器模块。

1.3　梯度下降发生器模块

CNN 训练模型利用反向传播算法将输出层的误差向后传播，并使用基于梯度下降的优化算法逐层更新变量[19]。本文利用加速变量更新收敛方式平滑梯度下降学习过程中遇到的波动，CNN 模块中的权重更新为：

$$\theta^{\tau} = \theta^{\tau-1} - \eta \cdot \Delta\nu^{\tau} \tag{2}$$

其中：θ^{τ} 为时刻 τ 的权重，η 为学习率，$\Delta\nu^{\tau}$ 为梯度。从公式（2）可以看出，梯度下降权重更新需要 1 个乘法器和 1 个减法器模块。

2　浮点运算体系结构

2.1　通用浮点数与运算

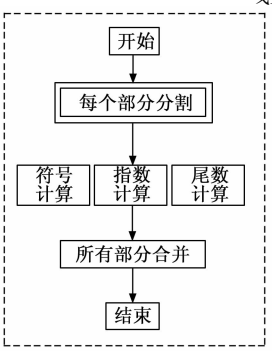
根据 IEEE 754 标准，由 3 个元素定义浮点格式：1）符号（正/负）；2）精度（实数的有效位数，尾数）；3）位数（索引范围）。浮点数可以表示：

$$FPN = (-1)^{Sign} \cdot M \cdot 2^{E-eb} \tag{3}$$

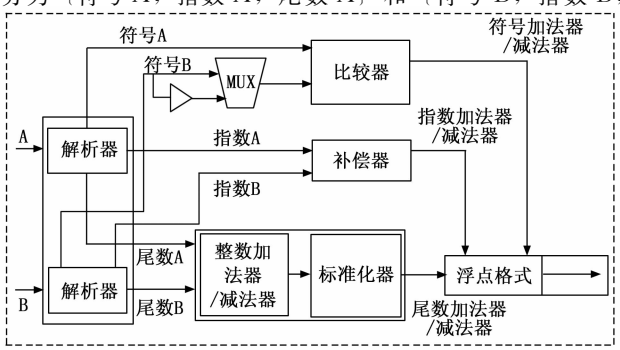
其中：$Sign$ 为符号指数，E 为指数的二进制值，eb 为指数范围的中值，M 为尾数，即小数点后的数字部分。浮点运算的流程图，如图 2 所示。其中，MUX 为多路复用器，浮点数的每个部分单独计算。

浮点运算的步骤为：

步骤 1：在执行实际计算之前，原始的浮点数 A 和 B 划分为｛符号 A，指数 A，尾数 A｝和｛符号 B，指数 B，



(a)一般运算流程



(b)详细的浮点运算流程

图 2　浮点运算的流程图

尾数 B}。

步骤 2：对于每个分离的元素分别进行运算：

1）符号：在加法/减法运算中，通过比较两个输入的尾数和指数来确定输出符号，在输入 B 的符号处放置 1 个非门和 1 个 MUX，使用反向符号在相同模块进行减法运算。在乘法/除法运算，通过异或（XOR）运算得到两个输入符号。

2）指数：如果指数值不同，则在两个输入中选择较大的指数值。对于指数值较小的输入，有效数位右移一位使得两个数字与同一个小数点对齐。两个输入的指数大小之差决定了要执行右移的次数。

3）尾数：通过无符号运算计算尾数的值。尾数位的加法/减法运算结果可能比两个输入的尾数位大 1 bit。因此，为了获得精确的结果，本文将两个输入的尾数位大小增加了两次，然后根据尾数的计算结果，无论最高有效位（MSB）为 0 或 1，对尾数进行加法或减法运算。如果 MSB 为 0，则不需要标准化。如果 MSB 为 1，则标准化器移动先前计算的指数位和尾数位来得到最终合并结果。

步骤 3：每个计算出的元素合并成一个浮点前置模块，从而生成结果以浮点形式输出。

2.2 浮点数格式的变体

本文利用 4 种不同的浮点格式优化 CNN。每种浮点格式的详细信息，如表 1 所示。

表 1 4 种不同的浮点格式

浮点格式	总位数	有效位	指数位	eb
自定义浮点	16	10	6	$2^5-1=31$
脑浮点	16	8	8	$2^7-1=127$
自定义浮点	24	16	8	$2^7-1=127$
单精度浮点	32	24	8	$2^7-1=127$

表 1 中的有效位表示位，包括符号位和尾数位。本文中使用的每种浮点格式，如图 3 所示。

16 bits 自定义浮点格式用于与现有的 16 bits 脑浮点格式进行比较，24 bits 自定义浮点格式用于与其他浮点格式进行比较。本文还在 16 bits 卷积块中使用 24 bit 自定义浮点格式进行累加，从而提高网络的精度。

2.3 基于倒数的除法运算

本文使用牛顿迭代法[20]作为除法运算的算法，只需要减法和乘法即可计算出一个数的倒数。在数值分析中，实值函数 $f(y)$ 近似为一条切线，其方程由 $f(y)$ 的值及其在初始近似处的一阶导数求得。如果 y_n 是真实值的当前估计值，则下一个估计值 y_{n+1} 可以表示为：

$$y_{n+1}=y_n-\frac{f(y_n)}{f'(y_n)} \tag{3}$$

其中：$f'(y_n)$ 为 $f(y_n)$ 对 y_n 的一阶导数。

在倒数生成器中有 3 个整数乘法器，其中两个乘法器负责在多次迭代后生成倒数。为了在倒数生成器中执行快速计算，本文使用 Dadda 乘法器[21]，其中部分乘积在半加法器和全加法器阶段求和，然后使用常规加法器将最终结果相加。基于倒数的浮点除法器的结构，如图 4 所示。

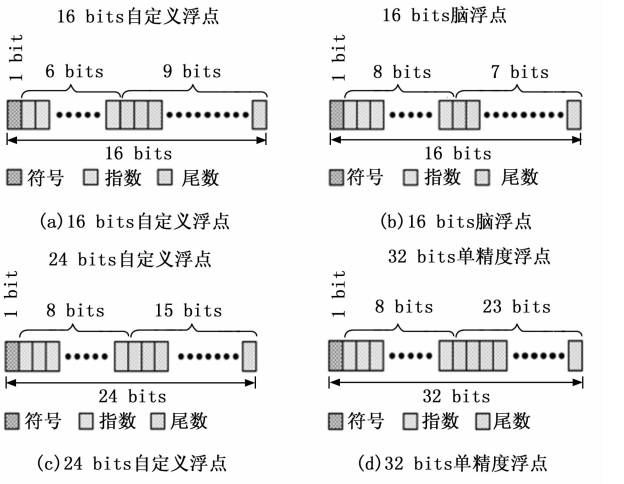


图 3 浮点表示法

3 浮点运算优化架构

3.1 基于符号数组的除法运算

由于基于倒数的除法器需要许多迭代和乘法器，因此，不可避免地受到处理延迟和过多硬件能耗的影响。本文使用符号数组来计算二进制数的除法[22]，与使用倒数的除法计算相比，符号数组除法没有重复的乘法运算。为了对分割进行优化，使用专门设计的处理单元，通过减法运算和执行反馈来选择除法的商值。符号数组除法处理单元（PU）的结构，如图 5 所示。

基于符号数组的浮点除法运算符整体结构，如图 6 所示。其中，LSB 为最低有效位。首先计算符号数组模块中的尾数，使用减法器/补偿器计算指数，使用 XOR 独立计算结果的符号位。符号数组模块中的每行都计算部分除法（从先前的部分除法中减去），然后将其传递到下一行。每行右移一位，使每个部分除法对应于被除数的下一位位置。数组除法器的每行决定了除法的商值的下一个最高位，将 3 个元素进行合并得到最终的除法器结果。

本文设计的加速器结构将能耗和硬件尺寸降至足以满足互联网应用需求的程度。为了比较硬件尺寸、处理延迟和总能耗，使用设计编译器合成工具和台积电（TSMC）55 nm 标准单元实现倒数和符号数组这两个除法器。倒数和符号数组进行除法计算的比较，如表 2 所示。

表 2 倒数和符号数组进行除法计算的比较

时钟频率	50 MHz		100 MHz	
除法器	倒数	符号数组	倒数	符号数组
硬件尺寸/μm^2	38 018.24	6 253.19	38 039.84	8 254.21
处理延迟/ns	70.38	21.36	64.23	10.79
总能耗/pJ *	78.505	4.486	112.927	5.019

* 总能耗为每个分区的能耗。

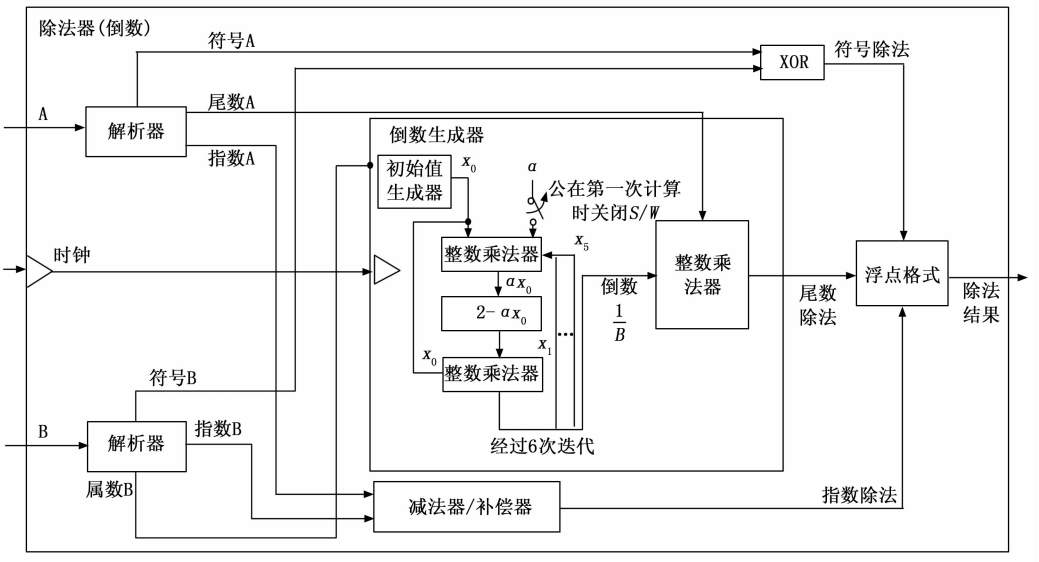


图 4　基于倒数的浮点除法器的结构

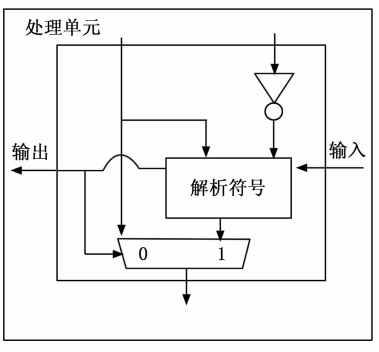


图 5　符号数组除法处理单元的结构

对于 50 MHz 和 100 MHz 的运算时钟，本文所提出的符号数组除法器比基于倒数的除法器分别降低了 6.1 倍和 4.5 倍。对于这两个时钟，符号数组除法器的处理延迟分别缩短了 3.3 倍和 6 倍。此外，与基于倒数的除法器相比，显著降低了 17.5～22.5 倍的能耗。因此，在实现 CNN 训练加速器的 Softmax 函数时选择所提出的符号数组除法器。

3.2　浮点乘法器

与浮点加法器/减法器不同，浮点乘法器计算尾数和指数时不依赖于符号位。符号位通过两个输入 XOR 门计算。加法器和补偿器块通过将两个输入数字的指数相加并从结果中减去偏移量“127”来计算得到的指数。如果计算的指数结果不在 0 和 255 之间，则视为上溢/下溢，并饱和到界限。即任何小于 0 的值（下溢）饱和为 0，而大于 255 的值（上溢）饱和为 255。通过两个输入尾数的整数乘法计算得

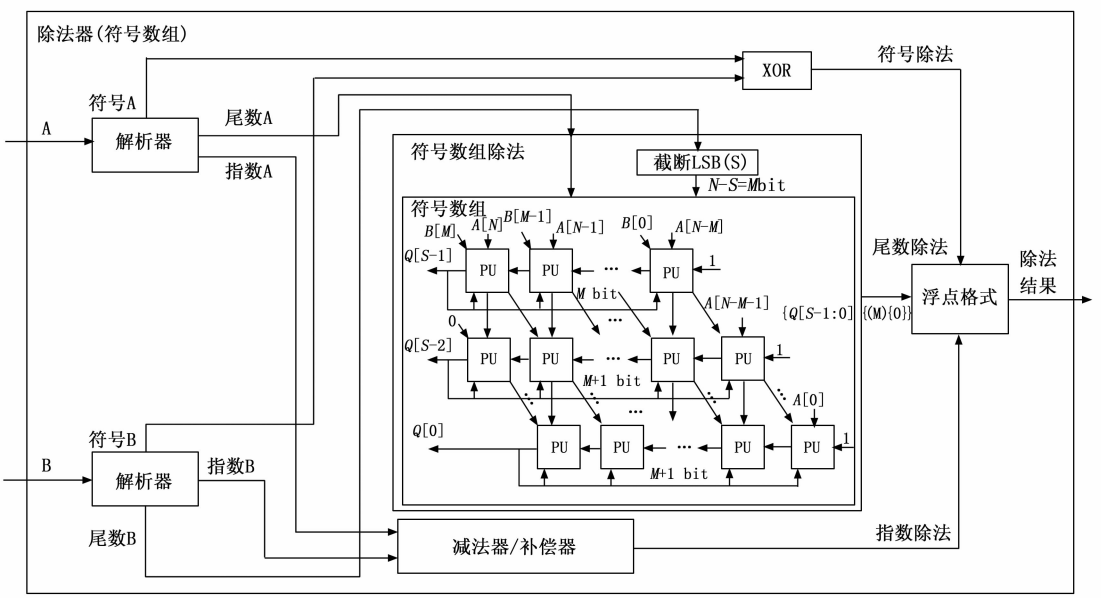


图 6　基于符号数组的浮点除法运算符整体结构

到尾数输出，使用指数值重新排列尾数位，然后合并生成最终的浮点格式。浮点乘法器的体系结构，如图 7 所示。

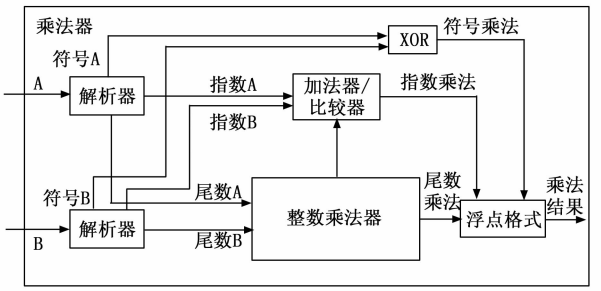


图 7 浮点乘法器的体系结构

3.3 CNN 加速器的总体架构

为了评估所提出的浮点运算符的性能，本文设计了可支持两种运算模式的 CNN 加速器，即推理和训练。加速器配备了用于发送图像、滤波器、权重和控制信号的芯片外接口。CNN 加速器的总体架构，如图 8 所示。其中，FC 为全连接网络，dout 为梯度，dW 为权重导数。

在发送大小为 28×28 的 MNIST 手写数字图像之前，通过芯片外接口将大小为 3×3 的 4 个滤波器、大小为 196×10 的 FC1 和大小为 10×10 的 FC2 的初始权重写入芯片内存储器。接收到启动信号后，将训练图像和滤波器权重传递给卷积模块，并根据矩阵的点积计算卷积。最大池模块通过在输出特征数据的每个 2×2 子阵选择最大值，对卷积模块的输出进行下采样。对 FC1 和 FC2 两个全连通层进行 SoftMax 运算，将预测输入图像的分类作为推理结果。

在训练模式下，反向传播按照 CNN 层的倒序计算矩阵点积的梯度下降。Softmax 层和反向传播层也用于进一步训练部分权重。从图 8 可以看出，全连接层分为 dout 模块和 dW 模块，其中，dout 模块用于计算梯度[23]，dW 模块用于计算权重导数[24]。由于反向卷积为最后一层，因此没有 dout 模块。通过反复计算 dout 和 dW 来训练每个层的权重值，直至权重值达到所需的精度。经过训练的权重存储在各自的存储器中，并用于训练过程的下一次迭代的推理。

3.4 CNN 结构优化

为了找到最佳浮点格式来优化 CNN 体系结构，本文计算了不同精度格式的准确率和动态功率，从而找到具有合理精度的浮点，如表 3 所示。

表 3 不同精度格式的准确率和动态功率

精度格式	单独层格式总位数	尾数位	指数位	训练准确率/%	测试准确率/%	动态功率/mW
单精度－32	32	24	8	96.42	96.18	36
自定义－24	24	16	8	94.26	93.15	30
单精度－16	16	11	5	12.78	11.30	19

本文选择 93%的目标准确率，尽管自定义－24 格式满足 93%的准确率阈值，但其动态功率为 30 mW，比单精度－16 浮点格式高 58%。因此，在满足目标准确率的同时，搜索各个层的最佳浮点格式来实现最小的能耗。本文中设置为单精度－16 的初始浮点格式计算准确率，将指数逐渐增加 1 bit，直至准确率停止增加或开始降低。因此，在第 k 次迭代中，指数位增加，而随着尾数位的减少，整

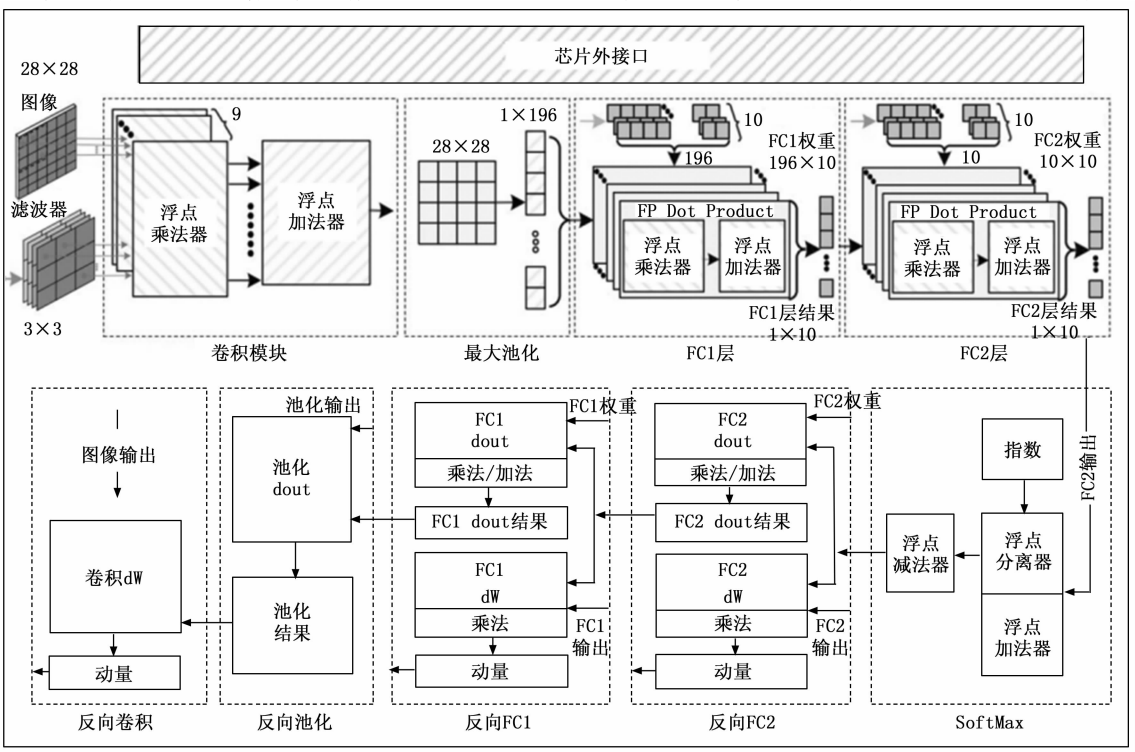


图 8 CNN 加速器的总体架构

体数据宽度（DW）保持不变。在确定指数位宽度后，算法使用新的浮点数据格式计算性能指标（准确率和能耗）。在本文实验中，发现尾数优化之前的新浮点格式为（Sign，Exp，DW-Exp-1），DW＝16 bit，Exp＝8，尾数＝16-8-1＝7 bit。通过逐渐将尾数增加 1 bit 对每层的精确度格式进行优化，直至当前 DW 满足目标准确率。当所有层在满足目标准确率的同时以最小能耗进行优化时，将为所有层存储最佳格式的组合。然后，将所有层的 DW 增加 1 bit，重复上述过程来搜索其他最佳格式，直至 DW 达到最大数据宽度（MAX DW），本文实验中 DW 设置为 32 bit。完成上述搜索过程后，将比较所有搜索结果的准确率和功率，在保持目标准确率的情况下，确定功率最小的格式构建最佳组合。

4　实验分析

4.1　浮点运算符的比较

使用 Synopsys 软件设计编译器对不同格式的比较进行了评估，编译器可以将 HDL 设计合成为 SoC 数字电路。使用台积电（TSMC）55 nm 工艺技术和 100 MHz 的固定频率进行评估。不同位宽的浮点加法器和减法器的比较，如表 4 所示。

表 4　不同位宽的浮点加法器和减法器的比较

浮点类型	不同位宽	硬件尺寸/μm^2	处理延迟/ns	总能耗/pJ
脑浮点	16(1,8,7)	1749.96	10.79	0.402
自定义浮点	24(1,8,15)	2610.44	10.80	0.635
单精度浮点	32(1,8,32)	3895.16	10.75	1.023

由表 4 可见，浮点加法器/减法器运算的位数越少，硬件尺寸或能耗就越小。每个位宽的浮点格式由 N（S，E，M）表示，其中 N 为总位数，S 为符号位，E 为指数，M 为尾数。使用不同位宽的浮点乘法器的比较，如表 5 所示。

表 5　不同位宽的浮点乘法器的比较

浮点类型	不同位宽	硬件尺寸/μm^2	处理延迟/ns	总能耗/pJ
脑浮点	16(1,8,7)	1 989.32	10.80	0.875 1
自定义浮点	24(1,8,15)	2 963.16	10.74	1.576 6
单精度浮点	32(1,8,32)	5 958.07	10.76	3.399 8

由表 5 可见，浮点乘法器运算的位数越少，硬件尺寸和能耗就越小。与加法器/减法器不同，乘法器能耗大幅增加。使用不同位宽的浮点除法器的比较，如表 6 所示。

表 6　不同位宽的浮点除法器的比较

浮点类型	不同位宽	硬件尺寸/μm^2	处理延迟/ns	总能耗/pJ
脑浮点	16(1,8,7)	1 442.16	10.80	0.623 6
自定义浮点	24(1,8,15)	3 624.12	10.79	1.912 5
单精度浮点	32(1,8,32)	8 254.21	10.85	5.019 0

由表 6 可见，与其他运算符相比，浮点除法器运算延迟时间是恒定的，但浮点除法运算的位数越少，硬件尺寸和能耗就越小。

4.2　CNN 训练加速器的性能评估

本文所提出的 CNN 训练加速器使用 Verilog 在寄存器传输级设计中实现，并使用 Vivado Verilog 模拟器进行验证。确认结果后在现场可编程门阵列（FPGA）上实现了加速器，并针对 50 K 图像训练了所有测试用例模型。训练后，通过将 MNIST 手写数字数据集中的 10 K 测试图像用于训练模型来计算推理准确率。

表 7 给出了优化算法找到的几个突出的格式组合搜索结果。在格式组合中，混合卷积－24 选为准确率和功率方面最优的格式组合。这种格式组合在卷积层（正向和反向传播）中使用 24 bit 格式，同时为池、FC1、FC2 和 Soft-Max 层（正向和反向传播）分配 16 bit 格式。

表 7　不同精度格式下优化算法准确率和动态功率的比较

精度格式	单独层格式	尾数位	指数位	训练准确率/%	测试准确率/%	动态功率/mW
单精度浮点－16	16 bits	11	5	11.52	10.24	19
自定义浮点－16	16 bits	10	6	15.78	13.40	19
自定义浮点－16	16 bits	9	7	45.82	32.54	19
脑浮点－16	16 bits	8	8	91.85	90.73	20
混合卷积－18	Conv/Back ConV-18 Rest 16 bits*	10/8	8	92.16	91.29	21
混合卷积－20	Conv/Back ConV-20 Rest 16 bits*	12/8	8	82.48	91.86	22
混合卷积－23	Conv/Back ConV-23 Rest 16 bits*	15/8	8	92.91	92.75	22
混合卷积－24	Conv/Back ConV-24 Rest 16 bits*	16/8	8	93.32	93.12	23
FC1 混合－32	Conv/Back ConV-32 Rest 20 bits#	24/12	8	93.01	92.53	26
FC2 混合－32	Conv/Back ConV-32 Rest 22 bits^	24/14	8	93.14	92.71	27

* Rest 16 bits 模块包括池化、FC1、FC2、Softmax、反向 FC1、反向 FC2 和反向池化；Rest 20 bits 模块是卷积、池化、FC2、Softmax、反向 FC2、反向池化和反向卷积；^ Rest 16 bits 模块是卷积、池化、FC1、Softmax、反向 FC1、反向池化和反向卷积。

4.3　优化架构比较

为了验证本文提出的浮点运算优化架构比现有 FGPA 加速器大幅减少硬件资源，利用 MNIST 手写数字数据集，将本文的最优架构（混合卷积－24）与文献［25］使用的传统 CNN、文献［26］采用滑动滤波器进行 CNN 和乘与

积（MAC）的并行运算、文献［27］提出的基于 Spike 网络加速器的 CNN 等方法进行比较，如表 8 所示。

表 8 优化架构比较

方法	准确率/%	计算时间/μs	功率/W	能耗/μJ
传统 CNN[25]	81.22	355.34	27.31	9 691.50
CNN＋MAC[26]	90.13	58.17	12.25	696.27
Spike＋CNN[27]	91.56	236.94	20.87	4 720.55
本文方法	93.32	13.40	0.64*	8.51

* 由 Xilinx Vivado 计算(功率＝静态功率＋动态功率)。

由表 8 可见，在本文的最优架构（混合卷积－24）加速器中，每幅图像的能耗仅为 8.5 μJ，每幅图像的能耗分别比传统 CNN 方法、CNN＋MAC 方法和 Spike＋CNN 方法低 1 140、81 和 555 倍。

5 结束语

本文评估了不同的浮点格式并优化了 CNN 训练中的浮点运算符，在高达 100 MHz 的频率下增加了吞吐量。使用 MNIST 手写数字数据集进行评估，使用混合精度体系结构实现了 93%以上的准确率，并且每幅图像的能耗仅为 8.5 μJ，比其他方法显著地降低了能耗。由于加速器仅需 55 nm 芯片即可实现低能耗和高精度，因此，本文设计的加速器适用于现实 AI 应用。在未来的研究中，尝试在 CNN 加速器中添加 8bit 配置，使其更进一步降低能耗。

参考文献：

[1] 安 婷，郭 辉．深度学习的人工智能应用处理系统设计研究［J］．现代制造技术与装备，2022，58（4）：191－193.

[2] 王 荣，杨 璐．大数据时代人工智能在计算机网络技术中的应用［J］．数字通信世界，2022，51（5）：100－102.

[3] 凤 雷，王宾涛，刘 冰，等．基于 FPGA 的深度强化学习硬件加速技术研究［J］．计算机测量与控制，2022，30（6）：242－247.

[4] 王永甲，王瑞博，赵一阳，等．基于 AgInSbTe 忆阻器的高效 MLP 神经网络［J］．信息技术，2022（1）：1－5.

[5] HUANG T，LUO T，ZHOU T，et al. APT：The master-copy-free training method for quantised neural network on edge devices［J］. Journal of Parallel and Distributed Computing，2022，166（8）：95－103.

[6] 邝祝芳，陈清林，李林峰，等．基于深度强化学习的多用户边缘计算任务卸载调度与资源分配算法［J］．计算机学报，2022，45（4）：812－824.

[7] 杨晶晶，薛明浩，王继禾．一种面向异构边缘架构的实时高能效图像分类任务划分策略［J］．小型微型计算机系统，2021，42（9）：1962－1966.

[8] 吴 恋，赵晨洁，韦萍萍，等．基于轻量级深度网络的计算机病毒检测方法［J］．计算机工程与设计，2022，43（3）：632－638.

[9] CHEN Y M，LI C，GONG L Q，et al. A deep neural network compression algorithm based on knowledge transfer for edge devices［J］. Computer Communications，2020，163（1）：186－194.

[10] 王 恺，严迎建，郭朋飞，等．基于改进残差网络和数据增强技术的能量分析攻击研究［J］．密码学报，2020，7（4）：551－564.

[11] 梅志伟，王维东．基于 FPGA 的卷积神经网络加速模块设计［J］．南京大学学报（自然科学），2020，56（4）：581－590.

[12] 宗德才，王康康．TEC-XP16 教学机浮点运算指令快速设计方法［J］．计算机与现代化，2021，12（8）：77－84.

[13] 张慧明．基于多核的卷积神经网络加速方法与系统实现［J］．集成电路应用，2020，37（5）：10－13.

[14] 栾桂芬．基于人工智能技术的网络多节点通信系统设计［J］．自动化技术与应用，2022，41（5）：71－74.

[15] KWON D，JIN C，KIM M. Prediction of dynamic and structural responses of submerged floating tunnel using artificial neural network and minimum sensors［J］. 2022，244（15）：110－124.

[16] 张 力，高 迪，陈 烁，等．一种嵌入铁电晶体管内容寻址存储器的高能效浮点运算结构［J］．电子与信息学报，2021，43（6）：1518－1524.

[17] 杨小琴，朱玉全．基于 CNN 的轻量级神经网络单幅图像超分辨率研究［J］．计算技术与自动化，2022，41（1）：98－105.

[18] 王 滨，郭艳凯，钱亚冠，等．针对卷积神经网络流量分类器的对抗样本攻击防御［J］．信息安全学报，2022，7（1）：145－156.

[19] 孟 浩，刘 强．基于 FPGA 的卷积神经网络训练加速器设计［J］．南京大学学报（自然科学），2021，57（6）：1075－1082.

[20] 李依肖，张 方．基于牛顿迭代法的时域动载荷识别 SISO 修正算法［J］．国外电子测量技术，2022，41（3）：52－55.

[21] 唐俊龙，汤孟媛，吴圳羲，等．32 位 RISC-V 处理器中乘法器的优化设计［J］．电子设计工程，2022，30（6）：61－65.

[22] ZHANG Z，ZHANG B，LIU Y，et al. An approach for predicting multiple-type overflow vulnerabilities based on combination features and a time series neural network algorithm［J］. Computers & Security，2022，114（3）：102－114.

[23] 吴婷婷，许晓东，吴云龙．卷积神经网络中 SPReLU 激活函数的优化研究［J］．计算机与数字工程，2021，49（8）：1637－1641.

[24] 郑长亮，庞 明．基于卷积神经网络的时空权重姿态运动特征提取算法［J］．应用科学学报，2021，39（4）：594－604.

[25] 许 杰，张子恒，王新宇，等．一种基于 Zynq 的 CNN 加速器设计与实现［J］．计算机技术与发展，2021，31（11）：108－113，121.

[26] 曹学成，廖湘萍，李盈盈，等．一种基于 FPGA 的高性能卷积神经网络加速器的设计与实现［J］．智能物联技术，2021，4（5）：11－17.

[27] 王红亮，程佳风．基于嵌入式设备应用的 CNN 加速器的设计研究［J］．电子器件，2021，44（4）：797－801.