DOI:10.7544/issn1000-1239.2017.20170099
54(6):1367-1380,2017

基于忆阻器的 PIM 结构实现深度卷积神经网络近似计算

李楚曦[1] 樊晓桠[1,2] 赵昌和[1] 张盛兵[1,2] 王党辉[1,2] 安建峰[1,2] 张 萌[1,2]
[1](西北工业大学计算机学院 西安 710129)
[2](嵌入式系统集成教育部工程研究中心(西北工业大学) 西安 710129)
(lichuxi@mail.nwpu.edu.cn)

A Memristor-Based Processing-in-Memory Architecture for Deep Convolutional Neural Networks Approximate Computation

Li Chuxi[1], Fan Xiaoya[1,2], Zhao Changhe[1], Zhang Shengbing[1,2], Wang Danghui[1,2], An Jianfeng[1,2], and Zhang Meng[1,2]
[1](*School of Computer Science, Northwestern Polytechnical University, Xi'an* 710129)
[2](*Engineering and Research Center of Embedded Systems Integration (Northwestern Polytechnical University), Ministry of Education, Xi'an* 710129)

Abstract Memristor is one of the most promising candidates to build processing-in-memory (PIM) structures. The memristor-based PIM with digital or multi-level memristors has been proposed for neuromorphic computing. The essential frequent AD/DA converting and intermediate memory in these structures leads to significant energy and area overhead. To address this issue, a memristor-based PIM architecture for deep convolutional neural network (CNN) is proposed in this work. It exploits the analog architecture to eliminate data converting in neuron layer banks, each of which consists of two special modules named weight sub-arrays (WSAs) and accumulate sub-arrays (ASAs). The partial sums of neuron inputs are generated in WSAs concurrently and are written into ASAs continuously, in which the results are computed finally. The noise in proposed analog style architecture is analyzed quantitatively in both model and circuit levels, and a synthetic solution is presented to suppress the noise, which calibrates the non-linear distortion of weight with a corrective function, pre-charges the write module to reduce the parasitic effects, and eliminates noise with a modified noise-aware training. The proposed design has been evaluated by varying neural network benchmarks, in which the results show that the energy efficiency and performance can both be improved about 90% in specific neural network without accuracy losses compared with digital solutions.

Key words memristor; processing-in-memory (PIM); convolutional neural network (CNN); approximate computation; analog memory

摘 要 忆阻器(memristor)能够将存储和计算的特性融合,可用于构建存储计算一体化的 PIM (processing-in-memory)结构.但是,由于计算阵列以及结构映射方法的限制,基于忆阻器阵列的深度神经网络计算需要频繁的 AD/DA 转换以及大量的中间存储,导致了显著的能量和面积开销.提出了一种

收稿日期:2017-02-27;**修回日期**:2017-04-14
基金项目:国家自然科学基金项目(61472322);中央高校基本科研业务费专项资金项目(3102015BJ(Ⅱ)ZS018)
This work was supported by the National Natural Science Foundation of China (61472322), the Fundamental Research Funds for the Central Universities (3102015BJ(Ⅱ)ZS018).
通信作者:王党辉(wangdh@nwpu.edu.cn)

新型的基于忆阻器的深度卷积神经网络近似计算 PIM 结构,利用模拟忆阻器大大增加数据密度,并将卷积过程分解到不同形式的忆阻器阵列中分别计算,增加了数据并行性,减少了数据转换次数并消除了中间存储,从而实现了加速和节能.针对该结构中可能存在的精度损失,给出了相应的优化策略.对不同规模和深度的神经网络计算进行仿真实验评估,结果表明,在相同计算精度下,该结构可以最多降低 90%以上的能耗,同时计算性能提升约 90%.

关键词 忆阻器;PIM;卷积神经网络;近似计算;模拟存储

中图法分类号 TP303

新型器件忆阻器(memristor)具有能够将计算和存储功能相融合的特殊性质,因此,以忆阻器及其基本单元构造的 PIM(processing-in-memory)结构被认为是访存瓶颈的有效解决途径[1].同时,由于欧姆定律和神经网络矩阵乘法运算的高度契合,并且忆阻器的高存储密度也能满足神经网络的巨大数据量,因此基于忆阻器交叉阵列的 PIM 结构极其适合于实现神经网络近似计算[2].

此前的一些相关研究结果显示,以忆阻器交叉阵列为基础的神经网络计算结构能够有效克服访存瓶颈,并取得良好的加速效果[2-5].然而,数据在交换过程中需要不断经过转换器进行数模/模数转化以及大量的多位中间存储,造成了显著的面积和能量消耗.

因此,本文提出了新型的 PIM 结构实现基于忆阻器的深度卷积神经网络近似计算,在不降低神经网络计算准确度的前提下,取得了能耗更低、面积更优、执行时间更短的效果.

本文的主要贡献如下:

1) 提出了将卷积过程分解到不同形式的忆阻器阵列中分别计算的计算方案,取得了显著性能提升,最高达到 87%;

2) 设计了新型的基于忆阻器的深度卷积神经网络近似计算 PIM 结构,能够减少数据转换次数,消除中间存储,最高可降低 95%的能耗;

3) 针对所提出的结构中可能存在的精度损失,提出了相应的优化策略,提高了系统精度,保证系统精度在 95%左右.

1 背 景

1.1 忆阻器阵列的 CNN 计算特征

卷积神经网络(convolutional neural network, CNN)是神经网络架构中常见的一种,由生物学自然视觉认知机制启发而来.典型的卷积神经网络具有输入层、卷积层、降采样层、全连接层和输出层.一般地,一个卷积层之后配置一个降采样层,2 个层构成一组.一个完整的卷积神经网络由一个输出层、多组卷积/降采样层、多个全连接层以及一个输出层按照从输入到输出的顺序构成,如图 1 所示[6].深度神经网络突出的特征学习能力使其在大数据应用中有突出的表现,特别是语音识别、图像识别等基于分类的识别问题[7].

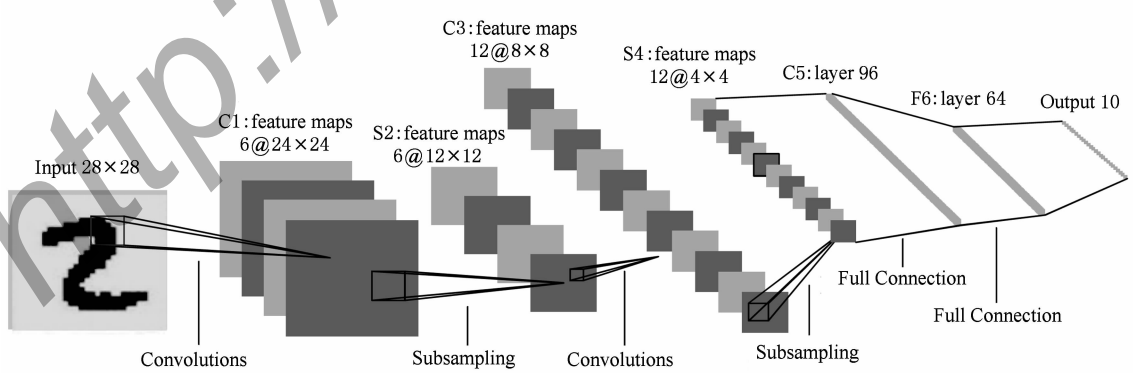

Fig. 1 The structure of LeNet-5 CNN

图 1 LeNet-5 卷积神经网络结构示意图

卷积神经网络的核心是乘累加运算,所有类型的层的算法都可以抽象成乘累加运算.参与乘累加的元素可以分成 2 类:权值和待处理的信号值.卷积神经网络中所有的运算过程,都可视为权值和待处理的信号值对应相乘之后再累加,其后加上偏置,最后通过激活函数(sigmoid 函数),得到计算结果的过程.即:

$$o_{ij} = \mathrm{sigmoid}\left(\sum_i \sum_j (m_{ij} \times k_{ij}) + bias\right), \quad (1)$$

其中,m_{ij} 为输入,k_{ij} 为权值,o_{ij} 为输出.

忆阻器是阻性器件,其电路特性满足阻性器件串并列规则.根据欧姆定律,假设若干忆阻器的一端连接在水平位线 WL 上,另一端连接在竖直字线 BL 上,构成忆阻器交叉阵列[8],第 i 行第 j 列忆阻器的阻值为 M_{ij},电导值为 $g_{ij}=1/M_{ij}$,施加在其 WL 的电压值为 U_i,不考虑外加激励导致的忆阻器阻值变化,则第 j 条 BL 的总电流 $I_{\text{total},j}$ 为

$$I_{\text{total},j}=\sum_i\sum_j U_i\times g_{ij}. \tag{2}$$

并行地完成多次乘累加运算,如图 2 所示.所有乘累加有一个元素完全相同,为输入电压 U_i;另一个元素不同,为不同列忆阻器的电导值.这与卷积神经网络运算形式的需求一致,如果配置好忆阻器的阻值,将其作为权值,将待处理的信号值编码成输入电压,就可以完成权值和待处理信号值的乘累加[9].

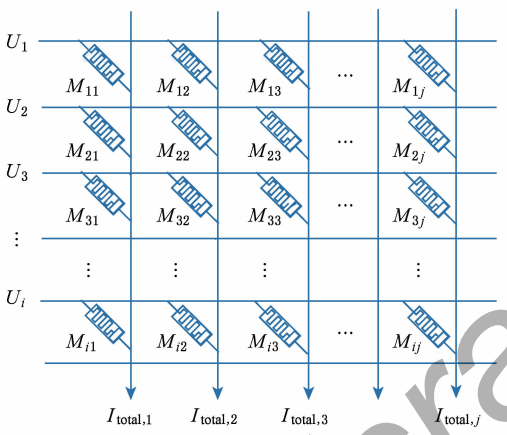

Fig. 2 Circuit structure of the memristor array

图 2 忆阻器阵列的电路结构

乘累加的计算结果形式为电流值,需要经 BL 输出到外部,以稳定的形式存放在存储单元中.如果神经网络的规模或深度超过单个忆阻器阵列的计算能力,则可分离出一部分忆阻器阵列作为缓存,用于存储中间结果,减少与外部的数据交换.

1.2 相关工作分析

文献[10-12]研究了基于非易失性存储器的 PIM 结构,证明了该结构在加速和降低能耗 2 方面都有显著优势.文献[3]提出了一种新型的 PIM 结构,利用忆阻器交叉阵列和 CMOS 辅助电路实现了 CNN,并将忆阻器交叉阵列划分为 2 种不同功能的模块:1)只用于存储;2)兼用于存储和计算.该电路结构取得了明显的加速效果,计算精度也较高.然而,其计算过程中产生的中间结果近似于模拟值,而忆阻器单元使用的是数字忆阻器,数据无法缓存,需要不断地进行数模/模数转化以及大量多位的中间存储,造成了额外的面积开销和能量消耗.

如果使用模拟忆阻器,即阻值连续可变的忆阻器,则可以保证中间结果能够缓存在忆阻器阵列中,消除数据转换带来的额外消耗.并且将读写次数由多位读写压缩至一次读写,大大提高数据密度,降低读写消耗.虽然理论上模拟忆阻器存储时拥有无穷高的精度,但是在实际应用中,由于相邻存储单元之间的漏流干扰和读写误差等原因,模拟忆阻器的精度会受到影响.同时,数字忆阻器存储多位数据时也存在截断误差,因此不能简单地判定二者的精度大小.如果能够通过合理的措施保证模拟忆阻器的精度,就能够利用其显著提高数据密度,大大降低计算电路的复杂性.

文献[8]提出了一种基于模拟忆阻器的非线性神经网络计算方案并验证了其可靠性,结果表明其平均误差在 5%左右.

基于以上分析,本文采用模拟忆阻器构建深度 CNN 近似计算结构,通过分析卷积神经网络的计算特征,将计算过程分解成利于并行的 2 部分,将中间结果缓存在部分忆阻器阵列中,消除数据转换.同时,给出提高精度的措施保证系统的可靠性.

2 基于 CNN 计算分解的忆阻器阵列结构

2.1 面向高效并行的计算方案

考虑到卷积层是卷积神经网络中复杂度最高、最具有普遍意义的类型,降采样层和全连接层都可以看作是卷积的特殊形式,只要能够实现卷积层,采用类似的方法也能实现降采样层和全连接层.因此以下重点讨论卷积层的实现方法.

假设卷积层的输入图尺寸是 $n\times n$,卷积核尺寸是 $k\times k$,滑步长度是 l,则该层的输出特征图尺寸为 $(n-k+l)/l\times(n-k+l)/l$.以 MNIST 手写体识别[13]的第 1 层卷积层计算为例,输入图大小是 28×28,卷积核的大小是 5×5,滑步是 1,得到的特征图大小是 24×24.

单次卷积运算由一个 $k\times k$ 的卷积核和一个 $k\times k$ 的卷积计算窗口参与.具体到 MNIST 手写体识别的第 1 层卷积层中,单次卷积运算是 5×5 的卷积核和 5×5 的卷积计算窗口.把计算过程进行分解,可视为 6 次乘累加运算.其中前 5 次是并列的,每次由卷积核中的一行和卷积窗口中的一行进行乘累加;最后一次是在前 5 次计算的中间结果得出后,将 5 个中间结果全部累加.这样拆解的原因是:在忆阻器阵列电路中,批量读取数据往往是以行为单位,在

同一行的数据可以在一个周期内同时读出,同时参与运算;而不在同一行的数据,需要在另外一个周期才能被读出.通过此拆解方法,将单次卷积运算分成了二维方向的2种类型运算:1)横向 k 次乘累加;2)纵向1次累加.

图3给出了一个卷积运算拆解的例子,原本的四次乘法、一次加法被拆分成了2组三次乘累加.

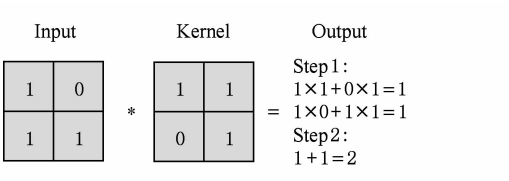

Fig. 3 An assembly example of 2×2 convolutional computation

图3 一个2×2卷积运算的拆解示例

本文将拆解后的2种运算分到2个忆阻器阵列子电路模块中分别计算,能够减少读取次数、提高效率并保证数据大规模并行.用于计算横向乘累加的模块命名为权重子阵列(weight sub-array, WSA),因为权重的存储和参与运算都在这部分中;用于计算纵向累加的模块命名为累加子阵列(accumulate sub-array, ASA),因为它一方面起到缓存卷积中间结果的作用,一方面将这些中间结果进行累加得到卷积值.

2.2 WSA模块

WSA模块的作用是完成横向乘累加,即卷积核中的每一行与卷积窗口中的每一行的乘累加运算.为了消除潜通路径(sneak path)的影响,其结构是1S1R的忆阻器单元构成的阵列[14](为简化电路结构图,便于阅读,以下模块结构图中省略了与忆阻器串联的选择管).

传统的策略是将卷积核(即权值)数据和卷积窗口数据编码为忆阻值,以矩阵排列方式分别存储在2个阵列中.计算时将卷积窗口数据读出,转换为电压形式,再作为输入传递给权值数据所在的阵列,最后在这其中将二者相乘并叠加.每采集一次卷积窗口数据需要进行一次读操作读取一行,进行一次 $k\times k$ 大小的卷积需要进行 k 次读操作才能采集完全部输入数据.进行完本次乘累加运算后,则滑步一次进行下一次卷积运算.由于前一次的读取结果不能以电压形式被缓存,此时需要再次读取同样的 k 行.当滑步进行到一行的末尾,则回到起始位置并向下滑步一次进行下一次卷积运算.此时需要读取的数据中只有一行不重复,其他 $k-1$ 行都与前一次卷积读取的行相同.

由于卷积层算法中每个卷积窗口都存在重叠,以及阵列按行读取数据的特点,大量的时间和能耗消耗在了对重复数据的多次读取上.要避免重复读取,首先要观测每次读取的数据参与的运算次数.下文继续以MNIST手写体识别的第1层卷积层计算为例,输入图大小是28×28、卷积核的大小是5×5,滑步是1,得到的特征图大小是24×24.将输入图的第 i 行命名为 L_i,将卷积核按照从左到右、从上到下的顺序命名为 $w_1\sim w_{25}$.

以输入图为观测对象,逐行考查:

对第1行 L_1,只有卷积核覆盖在 $L_1\sim L_5$ 上做卷积时参与运算,所涉及到的运算为24次乘累加.从 L_1 最左边开始,每次取5个连续数与 $w_1\sim w_5$ 分别相乘并累加,然后右移一个单位继续取数,直到取到最右边的5个数,共24次乘累加.

对第2行 L_2,卷积核覆盖在 $L_1\sim L_5$ 或者 $L_2\sim L_5$ 上做卷积时都要参与运算,所涉及到的运算为24×2次乘累加.从 L_2 最左边开始,每次取5个连续数,不仅要与 $w_1\sim w_5$ 相乘并累加,还要与 $w_6\sim w_{10}$ 进行相同的操作.

对第3行 L_3,卷积核覆盖在 $L_1\sim L_5$,$L_2\sim L_5$ 或者 $L_3\sim L_4$ 上做卷积时都要参与运算,所涉及到的运算为24×3次乘累加.从 L_2 最左边开始,每次取5个连续数,不仅要与 $w_1\sim w_5$ 相乘并累加,还要与 $w_6\sim w_{10}$,$w_{11}\sim w_{16}$ 进行相同的操作.

以此类推,L_4 要与 $w_1\sim w_5$,$w_6\sim w_{10}$,$w_{11}\sim w_{15}$,$w_{16}\sim w_{20}$ 进行24×4次乘累加,而 $L_5\sim L_{24}$ 要与 $w_1\sim w_5$,$w_6\sim w_{10}$,$w_{11}\sim w_{15}$,$w_{16}\sim w_{20}$,$w_{21}\sim w_{25}$ 这5组数进行24×5次乘累加.

最后5行的规律与前5行对称,L_{25} 与 $w_6\sim w_{10}$,$w_{11}\sim w_{15}$,$w_{16}\sim w_{20}$ 和 $w_{21}\sim w_{25}$ 进行24×4次乘累加,L_{26} 与 $w_{11}\sim w_{15}$,$w_{16}\sim w_{20}$ 和 $w_{21}\sim w_{25}$ 进行24×3次乘累加,L_{27} 与 $w_{16}\sim w_{20}$ 和 $w_{21}\sim w_{25}$ 进行24×2次乘累加,L_{28} 与 $w_{21}\sim w_{25}$ 进行24次乘累加.

至此,输入图每一行参与运算的情况分析完毕,其对称性和规律性有利于在电路中进行优化.由于计算的总次数是固定的,或者扩大阵列面积,在一个时间周期内并行地进行多次计算;或者复用阵列,在每个时间周期进行一次计算,因此计算时间和面积相互制约.由于忆阻器的纳米级尺寸和高密度集成,其面积明显小于CMOS电路,阵列面积的增加并不是电路的主要制约因素,因此将时间作为主要优化参数.WSA的设计目标是:增加并行性,在每次读操作之后,将读出的一行所涉及到的所有乘累加运算在一个周期内全部完成.

根据以上分析,将权重分为 $w_1\sim w_5$,$w_6\sim w_{10}$,$w_{11}\sim w_{15}$,$w_{16}\sim w_{20}$,$w_{21}\sim w_{25}$ 五组,构建 5 个阵列分别存放.在每个阵列内,不再按照传统的数据存放方式,而是将 5 个数据扩展成一个 28×24 的阵列,每一列由前一列的数据向下移位一次构成.以 $w_1\sim w_5$ 所在的阵列为例,排列方式如图 4 所示,第 1 列为 $w_1\sim w_5$ 从上到下排列,第 2 列为第 1 列向下移位一个单位,以此类推,直到填满第 24 列.

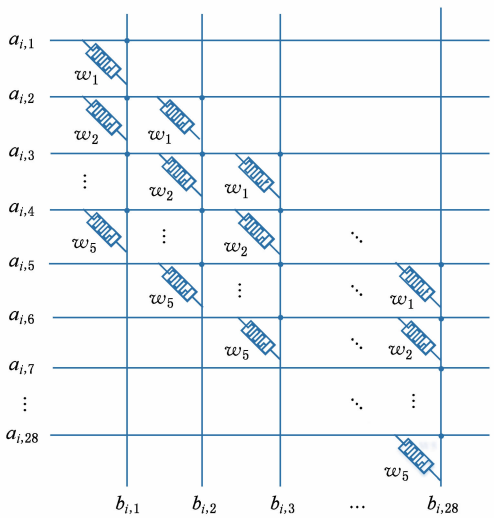

Fig. 4 Structure of the first array of 28×24 WSA

图 4 28 输入 24 输出的 WSA 中第 1 个阵列的结构图

每个周期内读出一行输入图的值,按照图 5 所示传递给与之相关的阵列.对于每一个阵列,输入为

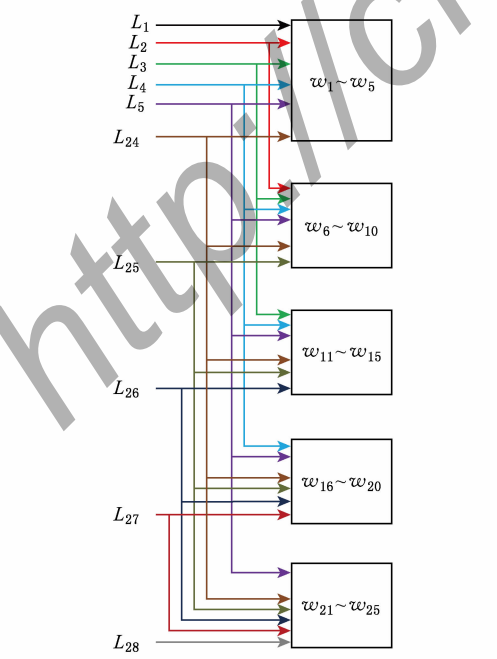

Fig. 5 The data transmission relationship between the rows of input map and the WSA

图 5 输入图的行数与 WSA 阵列的数据传输关系

一行的 28 个数据,以电压形式从 28 条 WL 同时并行输入.输出为 24 个乘累加的计算结果,以电流形式从 24 条 BL 同时输出.

假设一个周期内 5 个阵列的输入为 $a_{i,j}$,输出分别为 $b_{i,n}$,$c_{i,n}$,$d_{i,n}$,$e_{i,n}$,$f_{i,n}$,其中 $a_{i,j}$ 表示第 i 行第 j 列的输入数据,每个周期输入一行,在此周期内 i 不变,j 从 0 到最大值遍历一次.$b_{i,n}\sim f_{i,n}$ 分别表示 5 个阵列中当输入数据为第 i 行时的第 n 条 BL 上的输出数据($1\leqslant i,j\leqslant 28$,$1\leqslant n\leqslant 24$).

以 $w_1\sim w_5$ 所在的第 1 个阵列为例,BL_1 从上到下存放 $w_1\sim w_5$,则 BL_1 的输出为 $a_{i,1}\sim a_{i,5}$ 与 $w_1\sim w_5$ 的按序乘累加.BL_2 是 BL_1 的数据下移一位,则 BL_1 的输出为 $a_{i,2}\sim a_{i,6}$ 与 $w_1\sim w_5$ 的按序乘累加.可以得到:

$$b_{i,n}=a_{i,n}\times w_1+a_{i,n+1}\times w_2+a_{i,n+2}\times w_3+a_{i,n+3}\times w_4+a_{i,n+4}\times w_5. \tag{3}$$

其他 4 个 WSA 阵列以相同的形式分别将 $w_6\sim w_{10}$,$w_{11}\sim w_{15}$,$w_{16}\sim w_{20}$,$w_{21}\sim w_{25}$ 错位展开存放.同理可得:

$$c_{i,n}=a_{i,n}\times w_6+a_{i,n+1}\times w_7+a_{i,n+2}\times w_8+a_{i,n+3}\times w_9+a_{i,n+4}\times w_{10}, \tag{4}$$

$$d_{i,n}=a_{i,n}\times w_{11}+a_{i,n+1}\times w_{12}+a_{i,n+2}\times w_{13}+a_{i,n+3}\times w_{14}+a_{i,n+4}\times w_{15}, \tag{5}$$

$$e_{i,n}=a_{i,n}\times w_{16}+a_{i,n+1}\times w_{17}+a_{i,n+2}\times w_{18}+a_{i,n+3}\times w_{19}+a_{i,n+4}\times w_{20}, \tag{6}$$

$$f_{i,n}=a_{i,n}\times w_{21}+a_{i,n+1}\times w_{22}+a_{i,n+2}\times w_{23}+a_{i,n+3}\times w_{24}+a_{i,n+4}\times w_{25}, \tag{7}$$

对一幅输入图,28 行被分成 28 个周期按序输入,在第 k 个周期可以得到 $b_{k,n}$,$c_{k,n}$,$d_{k,n}$,$e_{k,n}$,$f_{k,n}$.每一个卷积运算分成的 5 次乘累加在 5 个周期内完成,例如输出图的第一个卷积结果 $k_{1,1}$:

$$k_{1,1}=\sum_{n=1}^{5}a_{1,n}\times w_n+\sum_{n=1}^{5}a_{2,n}\times w_{n+5}+\sum_{n=1}^{5}a_{3,n}\times w_{n+10}+\sum_{n=1}^{5}a_{4,n}\times w_{n+15}+\sum_{n=1}^{5}a_{5,n}\times w_{n+20}, \tag{8}$$

即

$$k_{1,1}=b_{1,1}+c_{2,1}+d_{3,1}+e_{4,1}+f_{5,1}. \tag{9}$$

5 次乘累加被分在第 1 个到第 5 个周期完成.同时,与 $k_{1,1}$ 同在一行的第 1 行 24 个数据都是在第 5 个周期将 5 次乘累加全部计算完.而第 2 行的 24 个数据在第 6 个周期能将 5 次乘累加全部计算完.第 28 个周期结束后,全部输出数据都被计算完毕,共 24×24×5 个.每个周期结束后,这些中间结果值需要被缓存进行下一步累加,这部分工作传递给

ASA,由 ASA 完成. 将 WSA 中完成横向乘累加输出的中间结果命名为部分和.

在以上示例中,整个 WSA 需要 $28\times24\times5$ 个忆阻器单元. 更一般地,若输入图大小是 $n\times n$、卷积核的大小是 $k\times k$,滑步是 l,则需要 $n\times(n-k+l)\times k$ 个忆阻器单元.

对于降采样层,可将其视为特殊的卷积层,其特殊之处在于:1)卷积窗口没有重叠;2)所有权值都是相同的(取均值算法). 除此之外,乘累加的顺序和方法均无改变. 在降采样层的计算初始时,将滑步设置为与卷积核宽度同样的大小 k,将权值设置为同一个固定的数,则采用的计算电路和卷积层可统一成同种形式. 对于全连接层,与前 2 种层不同,只有 1 次乘累加过程,而没有部分和. 这是由于全连接层的输入数据只有 1 行,不需要进行纵向叠加. 因此,全连接层可以直接映射到 WSA,在 1 个周期内计算完成,而不需要传输给 ASA 再做叠加.

该设计可以避免电路的特殊化,同样的硬件电路通过配置可以任意改变为卷积层、降采样层或全连接层,保证了电路的可配置性.

2.3 ASA 模块

ASA 模块的作用是完成纵向乘累加,即接收由 WSA 计算得出的部分和,完成进一步叠加,从而完成卷积运算. 叠加过程没有加权,直接将所有值相加. 其结构是若干模拟忆阻器缓存单元构成的阵列,每个缓存单元由 1 个忆阻器和 1 个双向选择管构成,如图 6 所示,圆形虚线框内为 1 个单元.

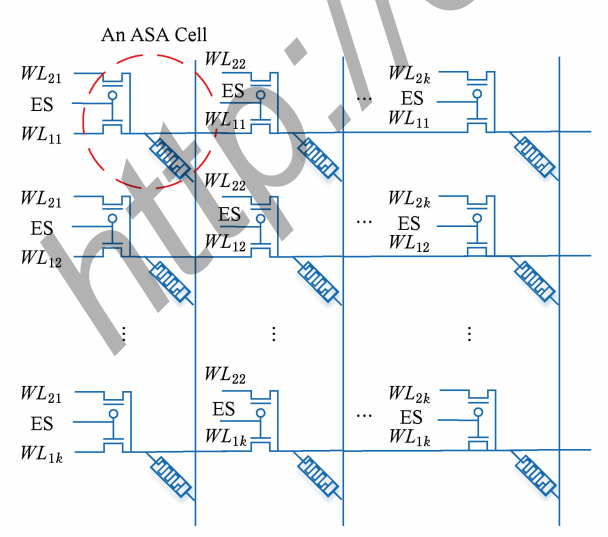

Fig. 6 The structure of ASA

图 6 ASA 的结构

ASA 的输入信号分为 2 部分:1)控制器以高/低电平形式输入到双向选择管的控制信号 ES (extra selector);2)从 WL 以电压形式输入的部分和数据.

双向选择管有 2 个作用:1)充当忆阻单元选择管,消除 Sneak Path;2)将 WL 扩展成 2 个不同的输入端 WL_1 和 WL_2,在不同情况下为其对应的忆阻器选择合适的输入信号. 当 ES 信号为高电平 1 时,忆阻器缓存单元中的 NMOS 晶体管导通,PMOS 晶体管关断,信号从 WL_1 输入;当 ES 信号为低电平 0 时,PMOS 晶体管导通,NMOS 晶体管关断,信号从另外一条扩展字线 WL_2 输入.

根据 WSA 模块的设计,需要在同一个 ASA 中叠加的 k 个部分和数据,即一个卷积运算的所有部分和数据将会在连续的 k 个周期内产生,以模拟电压信号的方式输出. 部分和电压信号输入给 ASA,由电压的幅值大小决定对 ASA 中的忆阻器的写入值. 写电路驱动电压的幅值与写入的电阻值一一对应,理想条件下,二者呈线性关系. ASA 中所有忆阻器的阻值在初始化过程中都被写入为最小阻值,可以视为 0. 随着 WSA 不断输出部分和,ASA 也不断地被连续写入阻值,在每个单元内积累写入值,从而完成对部分和的累加,1 个计算周期结束后忆阻器的阻值对应着卷积结果.

继续以 MNIST 手写体识别的第 1 层卷积层计算为例,输入图大小是 28×28、卷积核的大小是 5×5. 在每个周期内,WSA 电路的 5 个阵列各输出 24 个部分和. 从第 1 个运算周期开始,每 5 个周期视为一个循环,同一周期内输出的所有部分和不存在需要叠加的关系,需要叠加的是第 i 个周期 WSA 的第 i 个阵列内同一编号的 BL 上输出的结果($1\leqslant i\leqslant5$). 每一个卷积运算的最终结果都是由 5 个部分和叠加而来的,以下集中讨论输入图前 5 行与卷积核进行卷积的计算结果,将这一部分输入图的第 j 行与卷积核的第 j 行乘累加产生的部分和称为部分和 j(partial sum j).

卷积运算开始的第 1 个周期,WSA 的第 1 个阵列输出的是输入图第 1 行的每 5 个元素与 $w_1\sim w_5$ 分组相乘累加的 24 个结果,即部分和 1,其他 4 个阵列没有输出结果. 将这 24 个结果通过 WL_{1k} 同时传输到 ASA 的第 1 列上,以阻值形式缓存.

第 2 个周期,WSA 的第 1 个阵列输出的是输入图第 2 行的每 5 个元素与 $w_1\sim w_5$ 分组相乘累加的 24 个结果,第 2 个阵列输出的是输入图第 2 行的每 5 个元素与 $w_6\sim w_{10}$ 分组相乘累加的 24 个结果,即部分和 2,其他 3 个阵列没有输出结果. 根据卷积神

经网络的算法,部分和 1 应该与部分和 2 相叠加.那么,将第 1 个阵列输出的结果通过 WL_{1k} 传输到 ASA 的第 2 列上缓存,而将部分和 2 通过 WL_{21} 再次传输给 ASA 的第 1 列,给忆阻器写入叠加值,叠加在部分和 1 上.

同样地,在第 3 和第 4 个周期,每个周期利用 WL_{1k} 缓存一列数据,利用 WL_{2k} 叠加一次部分和.

第 5 个周期结束时,ASA 的前 5 列中存有缓存数据,第 1 列上已经完成了 5 个部分和的累加,即卷积结果图中第 1 行的 24 个数据.为了节省面积开销,计算完毕的数据可以马上输出到外部或者下一个层,将第 1 行的忆阻器单元回复初始状态,即最小阻值状态,可以在下个周期缓存新的数据.

需要注意的是,此时 ASA 的第 2 列上已经完成了卷积窗口下移一个单位时,即输入图的第 2 行到第 6 行与卷积核做运算的前 4 个部分和的叠加,只要第 6 个周期结束,这部分卷积运算,即卷积结果图中第 2 行的 24 个数据就计算完成,也可立即输出到外部,重置第 2 列的忆阻器单元.

通过这样流水的方式,完成全部卷积运算的周期数被大大缩减,只需 28 个读周期结束,WSA 计算出所有部分和后,ASA 就能立即输出所有的卷积结果.整个计算过程中每个周期的输出如表 1 所示,表 1 中的 $LINE_i$ 表示输入图中的第 i 行,$O_{k,n}$ 表示输出卷积结果的第 k 行中所有元素.

Table 1 The Input and Output of WSA and ASA in Each Cycle

表 1 各个周期 WSA 和 ASA 的输入输出

Cycle	Input of WSA	Output of WSA	Input of ASA
1	$LINE_1$	Partial sum1 of $O_{1,n}$	
2	$LINE_2$	Partial sum1 of $O_{2,n}$ Partial sum2 of $O_{1,n}$	
3	$LINE_3$	Partial sum1 of $O_{3,n}$ Partial sum2 of $O_{2,n}$ Partial sum3 of $O_{1,n}$	
⋮	⋮	⋮	⋮
i	$LINE_i$	Partial sum1 of $O_{i,n}$ Partial sum2 of $O_{i-1,n}$ Partial sum3 of $O_{i-2,n}$ Partial sum4 of $O_{i-3,n}$ Partial sum5 of $O_{i-4,n}$	$O_{i-4,j}$
⋮	⋮	⋮	⋮
27	$LINE_{27}$	Partial sum4 of $O_{28,n}$ Partial sum5 of $O_{27,n}$	$O_{27,j}$
28	$LINE_{28}$	Partial sum5 of $O_{28,n}$	$O_{28,j}$

此外,由于 WSA 中忆阻器的乘累加计算依赖于欧姆定律,当电压激励给出后,电流响应的延迟时间极短;ASA 中的叠加依赖于忆阻器的连续写入,由于忆阻器的读写速度快,ASA 延迟时间也极小.因此整个计算结构无论在周期数还是电路延迟时间上都体现出显著的速度优势,对加速卷积神经网络有着极大提升.

整个 ASA 需要 $28\times24\times5$ 个忆阻器单元,包含 1 个忆阻器和 2 个互补 MOS 晶体管.更一般地,若输入图大小是 $n\times n$,卷积核的大小是 $k\times k$,滑步是 l,则需要 $n\times(n-k+l)\times k$ 个忆阻器单元.

3 面向 CNN 近似计算的 PIM 结构

3.1 PIM 系统结构

本文设计了一种基于忆阻器阵列的可配置 CNN 近似计算 PIM 结构(如图 7 所示).系统包括 16 个分块(bank),被划分为 7 种模块,分别为:全局控制器(global controller, GC)、数模(模数)转换器(DAC/ADC)、三输入多路选择器(multiplexer, MUX)、读写模块(write and read, WR)及其寻址模块(addressing, AD)、开关模块(switch, SW)、WSA 和 ASA.

1) 全局控制器.①在系统开始工作之前,根据不同深度和规模的卷积神经网络应用,从 bank 和 bank 内的模块 2 个层面来进行配置和调度.在 bank 层面,当所需求的 CNN 应用的深度和规模超过一个 bank 的运算范围时,全局控制器将配置多个 bank 协同运算,把需要运算的数据分解到各个 bank 中并行计算,每个 bank 运算后的数据会按照层的顺序输出给下一个 bank 作为输入.在阵列层面,全局控制器将根据所需规模调用适合大小的阵列进行计算,剩余不被调用的部分不参与计算,保持在存储模式.②在系统工作过程中,通过控制信号控制整个系统的工作流程.全局控制器中配有计数器和时钟产生模块,在每个周期给 WR,WSA,ASA 等模块发送控制信号,使其正常工作.

2) 数模(模数)转换器.将外部输入的数字信号转换为模拟电压信号,作为 bank 的输入信号参与 CNN 运算.运算完成后,再将输出到外部的模拟信号转换为数字信号.为了降低功耗和面积开销,一组数模(模数)转换器可以由多个 bank 共享,分时使用.本设计中采用 4 个 bank 共享一组数模(模数)转换器的方案,即一个系统中含有 4 组数模(模数)转换器.

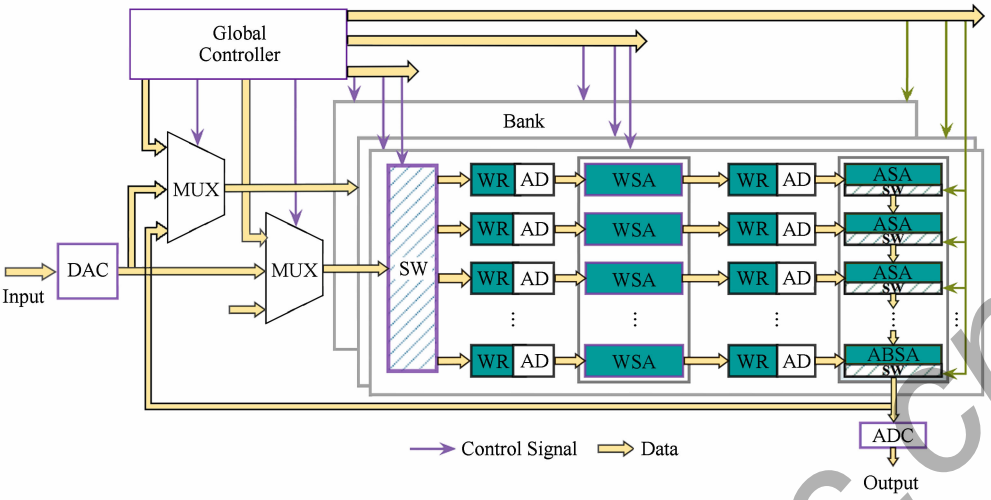

Fig. 7 Configurable PIM structure for CNN approximate computation

图 7 可配置卷积神经网络近似计算 PIM 结构图

3) 三输入多路选择器. 在不同阶段为 bank 选择合适的输入信号. 输入信号来自于不同来源:①初始化过程中来自全局控制器的配置信号;②外部输入信号经过数模转换后产生的模拟信号;③前一层的输出信号.

4) 读写模块. 为阵列中的忆阻器写入值,写入包括 2 个过程:①初始化阶段. 根据线下训练的结果为 WSA 写入权重矩阵对应的阻值,为 ASA 写入最小阻值 R_{ON}. ②计算阶段. 根据 WSA 输出的驱动电压为 ASA 实时地连续写入阻值. 每个读写模块附带一组寻址模块(AD).

5) 开关模块. 为 WSA 选择输出计算结果到外部或继续缓存数据. 开关模块的开断情况收到全局控制器的控制.

3.2 PIM 系统控制流程

WSA 可以被配置计算或存储 2 种模式,而 ASA 只能在计算模式工作. 在没有计算请求时,WSA 全部保持存储模式(memory mode). 当一个 CNN 应用的计算请求命令到来,系统将进入计算模式(computing mode),整个过程分为初始化(initial phase)、计算求值(valuing phase)2 个阶段:

1) 初始化阶段

① 输入神经网络配置参数并进行配置

分析应用需求,给全局控制器输入神经网络的参数,包括:网络的层数、各个层的种类、每层的输入和输出数量等. 根据参数对阵列进行配置,分配好参与计算的 bank,划分出 bank 中参与计算的单元,按照顺序配置成不同的层. 对于卷积层或降采样层,配置一个 WSA 和一个 ASA 结合;对于全连接层,只配置一个 WSA,没有 ASA. 设置好数据接口,令不参与计算的单元保持在存储模式.

② 初始化所有阵列的忆阻值

对于卷积层和全连接层的 WSA,通过 MUX 的选择,从外部经过数模转换输入线下训练的权值给读写模块,然后按照错位展开的方式排列,通过 WR 将阻值写入到对应 WSA 的各个单元中.

对于降采样层的 WSA,写入值为固定的某常数,这里设定为忆阻值最大值和最小值的中位数,即 $(R_{ON}+R_{OFF})/2$. 对于卷积层和降采样层的 ASA,写入值为忆阻值的最小值,即 R_{ON}.

2) 计算求值阶段

① 从外部输入计算数据,经过 DAC 转换成模拟值,再通过 MUX 的选择输入给第 1 层对应的 WSA. 根据神经网络应用的不同,第 1 层有可能是卷积层,也有可能是降采样层. 在第 1 层的 WSA 中计算出所有部分和,实时地通过 BL 输出给第 1 层的 ASA 的写电路,写电路对 ASA 进行改写,完成部分和的叠加. 最后将计算结果通过 ASA 的 BL 输出.

② 第 1 层计算完毕之后,将输出数据通过 MUX 传输给下一层,进行下一步计算. 自此计算流程与第 1 步相同,直到所有卷积层和降采样层计算完成. 在计算电路中流动的数据全部为模拟电流或电压值,不需要经过数模/模数转换.

③ 所有卷积层和降采样层计算完成之后,数据传输给全连接层. 全连接层没有 ASA,在 WSA 中进行乘累加即可得出计算结果. 将计算结果经过 ADC 传输到外部,标志着整个计算过程完成.

④ 计算完成之后,全局控制器将 WSA 恢复为存储模式,将 ASA 中缓存的所有值释放.

4 精度提高的优化策略

模拟忆阻器的噪声敏感度远远高于数字忆阻器,噪声干扰会影响系统的准确性和灵敏性.经过分析,本系统噪声来自于若干并列的噪声来源,可以表示为由这 4 个主要元素构成的表达式,如式(10)所示:

$$Noise=E\{\theta_{\mathrm{l}},\theta_{\mathrm{m}},\theta_{\mathrm{w}},\theta_{\mathrm{o}}\}, \tag{10}$$

其中 θ_{l} 表示来自于漏电流(leakage)的噪声,包括 Sneak Path 和 MOS 器件漏电等;θ_{m} 表示来自于从权值到非线性忆阻值(memristance)的映射误差;θ_{w} 表示来自于写入过程(writing)的误差;θ_{o} 代表其他来源(other)的噪声,包括但不限于器件噪声和过程噪声等.

θ_{l} 和 θ_{w} 可以在电路设计过程中采取措施来消除对电路的干扰影响.对于 θ_{l},本文中使用给忆阻器串联选择管构成忆阻单元的方法加以消除,对于 θ_{m},θ_{w} 和 θ_{o},分别采用权值映射的修正函数、读写模块的精度提升和噪声感知的训练方法来消除.

4.1 权值映射的修正函数

θ_{m} 来自于从权值到非线性忆阻值的映射误差.在神经网络计算的初始化阶段,要完成从训练好的权值到 WSA 中忆阻值的一一映射,并将其写入相应的忆阻器中.一般采用的映射函数是线性函数,也就是说,2 个权值 w_1,w_2 的差值和与之对应的 2 个忆阻值 M_1,M_2 成正比关系,即 $\Delta M/\Delta w$ 是恒定常数.然而,忆阻器物理器件是一个非线性器件,忆阻值 M 和流经的电荷量 Q 呈非线性关系,2 个忆阻值 M_1,M_2 的差值和与之对应的 2 个电荷量 Q_1,Q_2 不成正比关系,即 $\Delta M/\Delta w$ 不是恒定常数.也就是说,在不同的初始条件下流过相同的电荷量时,忆阻值的变化量不恒定.本文所用的忆阻器模型仿真曲线如图 8 所示,其中图 8(a)是 *I-V* 曲线;图 8(b)是 *M-Q* 曲线,即忆阻器阻值和电荷量的关系曲线.

如果采用传统的线性映射方法,忆阻器器件的非线性成分就会导致在计算值累积的过程中产生误差.实验表明,在一层输入为 28×28、卷积核大小为 5×5、滑步为 1 的卷积层计算情况下,线性映射方法会造成的输出值相对误差大约为 10%.在多层卷积神经网络中,这个误差将会进一步累积,如果不采取优化措施,有可能会直接导致计算结果错误.

为了解决这一问题,本文提出了一种新的修正映射方法.假设训练好的权值位于$[B_{\mathrm{l}},B_{\mathrm{h}}]$之间,其中 B_{l} 表示权值的下边界,是所有权值中的最小值;

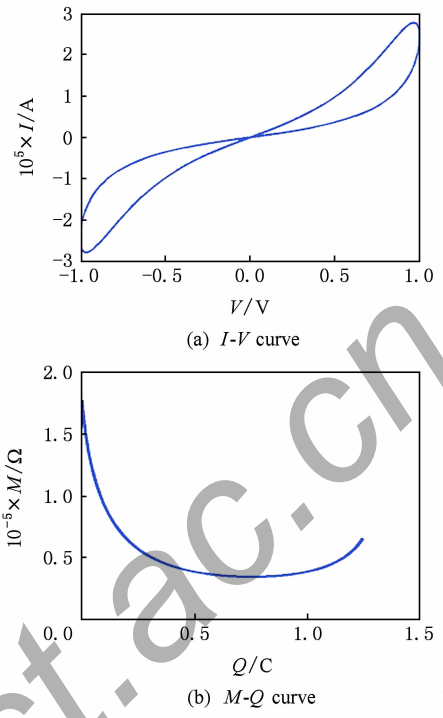

Fig. 8 The memristor model simulation curve

图 8 忆阻器模型仿真曲线

而 B_{h} 表示权值的上边界,是所有权值中的最大值.确定好范围之后,将其与位于$[R_{\mathrm{ON}},R_{\mathrm{OFF}}]$之间的忆阻值通过修正的映射函数进行映射.

修正映射函数是通过将原本的线性映射函数和非线性的 *M-Q* 曲线相除,再利用 Matlab 进行多项式拟合得到.经过试验,当多项式的次数为 9 时,能够在$[R_{\mathrm{ON}},R_{\mathrm{OFF}}]$取值范围内对本文所使用的忆阻器模型取得较贴近的拟合效果.该修正映射函数为

$$f(M)=\sum_{n=0}^{9}p_n\times M^n, \tag{11}$$

其中,M 为自变量,代表忆阻值;$f(M)$为因变量,代表修正映射函数;$p_0\sim p_9$ 为 10 个常数,是通过 Matlab 拟合得到的系数.修正映射函数的曲线图如图 9 所示:

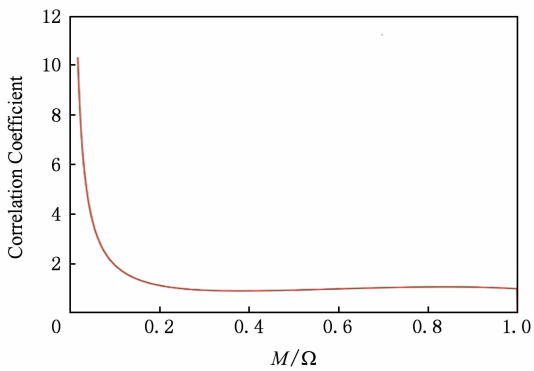

Fig. 9 Mapping function correction curve

图 9 修正映射函数曲线

采用这个修正拟合函数,可以消除非线性因素带来的误差,将识别准确度提高 7%左右.

4.2 读写模块的精度提升

在忆阻器阵列中,大量的导线 WL/BL 平行放置,会造成寄生电容的引入.寄生电容在忆阻器的写入过程中会被写驱动充电,发生电荷共享,消耗一部分写驱动电荷,造成写干扰.由于大规模阵列中的连接线长度大、数量多,寄生电容的值较大,造成的干扰也更加严重.根据文献[15]中的分析,写干扰的程度与位线寄生电容 C_{BL} 成正比,与写电流 I_{SET} 成反比,如式(12)所示.也就是说,阵列尺寸的增大或者写电流的减小都会增大写干扰,这与高密度、低功耗的目标相悖.

$$Write\_Disturbance \propto \frac{I_{surge}}{I_{SET}} = \frac{C_{BL} \times \Delta V_{BL}}{\Delta T_{SET}} \times \frac{1}{I_{SET}} = \alpha \times \frac{C_{BL}}{I_{SET}}. \tag{12}$$

本文通过对寄生电容进行预充电的策略消除寄生电容的影响.带预充电电路的写电路如图 10 所示,在文献[16]的写电路基础上增加了预充电单元.图 10 中 S_{pre} 是预充电控制开关,I_{pre} 是预充电电流源.在写操作前添加预充电阶段,将开关 S_{pre} 接 Port1,给寄生电容施加一个短时间的预充电电流.预充电电流的持续时间和幅值选择基于阵列的大小和写入电压的值.预充电完成后,将开关 S_{pre} 接 Port2 进行忆阻器的脉冲写操作.

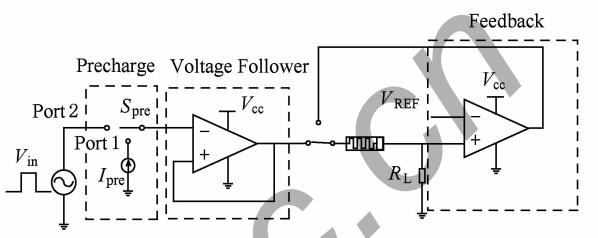

Fig. 10 The write circuit with feedback and precharge circuit

图 10 带反馈回路和预充电电路的写电路

图 11 所示的是使用预充电策略的写电路仿真结果,其相对误差为 1.36%.

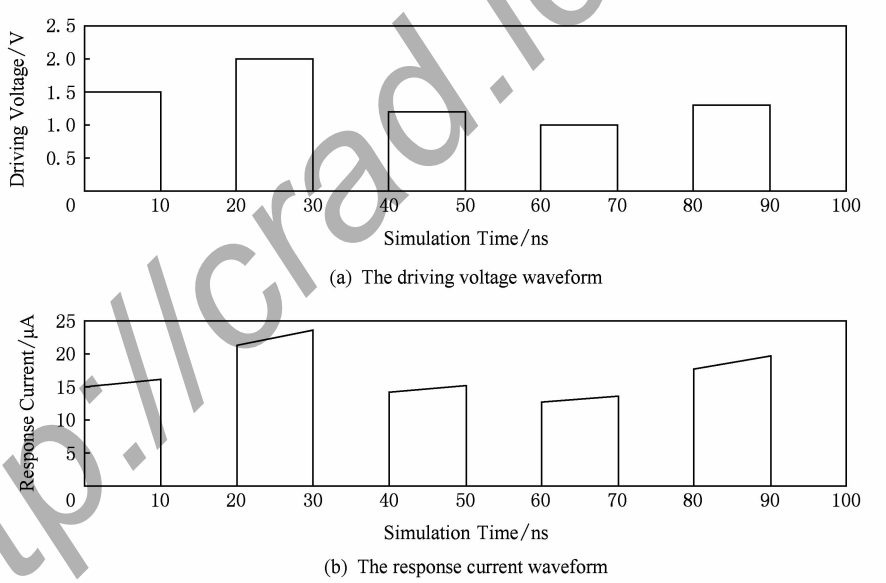

Fig. 11 The write process simulation results

图 11 写入过程仿真结果

4.3 噪声感知的训练方法

θ_o 是来自其他来源的噪声,包括但不限于器件电子噪声和过程噪声等.器件电子噪声来自于电路器件本身,包括热噪声和闪烁噪声等;过程噪声来自于电路工作中的各种随机变化.θ_o 成分都是随机过程量,通常用统计模型来表示,并且,成分难以通过优化电路设计的方法来抑制消除.

本文提出了一种噪声感知的训练方法,能够将这部分无法靠优化电路消除的噪声成分在线下训练时抵消掉.文献[17]中提供了 MOS 晶体管的噪声模型,文献[18]中提供了忆阻器的噪声模型,过程噪声可以使用正态分布来建立模型.确立了噪声的统计模型之后,需要将每一个权重经过的电路路径上的所有噪声分量进行叠加,详细的公式推导请参阅参考文献[17-18].在进行线下训练时,根据神经网络应用的规模和深度确定模型参数,将这些噪声分量模型注入到原本的训练过程中,使得权值得到调整.通过这种方式,能够将识别准确率提高 3%左右.

5 实验与结果分析

5.1 仿真平台与配置

电路的仿真平台为 HSPICE,使用到的基本电路元件有:电阻、电容、NMOS 晶体管、PMOS 晶体管和模拟忆阻器等.

电阻和电容使用 HSPICE 基本元件库中的模型;NMOS 晶体管和 PMOS 晶体管使用文献[19]中的 0.13um 工艺模型,该工艺库中有 2 种规格的 MOS 管,分别为驱动电压 1.5 V 和驱动电压 3.3 V,本文选择 3.3 V 规格.

忆阻器使用文献[20]的 Spice 模型,将其参数按照表 2 设置.其中,R_{OFF}代表最大阻值,R_{ON}代表最小阻值,R_{INT}代表初始阻值,D 代表忆阻器薄膜厚度,μ_v 代表迁移速度,p 代表窗口函数的参数.

Table 2 Memristor Parameter Setting

表 2 忆阻器参数设置

R_{OFF}/kΩ	R_{ON}/Ω	R_{INT}/kΩ	D/nm	μ_v/($m^2 \cdot V^{-1}$)	p
200	200	11	10	10^{-14}	1

5.2 仿真基准的选择

本文选择了 3 类 6 种基准,包括 2 种不同规模的 28×28 输入卷积神经网络、3 种不同规模的多层感知器(MLP)和 1 种 VGG-19 网络,前 2 类被广泛运用于 MNIST 手写体识别,VGG-19 运用于 RGB 图像识别.表 3~表 5 分别给出了仿真基准的所有参数[21].

多层感知器是神经网络中较简单的一种,含有一个输入层、一个输出层和若干个隐藏层.隐藏层进行的是乘累加运算,比起卷积神经网络往往深度和复杂度较低.事实上,卷积神经网络是多层感知器的一种特殊化,是将多层感知器中的隐藏层特定为卷积和降采样操作的结果.VGG-19 网络是一种规模较大的多通道神经网络.神经网络的规模大小和深度会直接影响本文所设计的电路结构的加速和节省能耗的效果.因此这里选择多种不同规模和深度的神经网络,以便分析各项性能与神经网络尺寸和深度的关系.

Table 3 Two Kinds of CNN Benchmarks Parameters

表 3 2 种卷积神经网络仿真基准参数

Layer	CNN-1	CNN-2
Input Layer	28×28	28×28
Coventional Layer1	6×24×24	32×24×24
Pooling Layer1	6×12×12	32×12×12
Coventional Layer2	12×8×8	17×8×8
Pooling Layer2	12×4×4	17×4×4
Full Connection Layer	192-64	272-46
Output Layer	10	10

Table 4 Three Kinds of MLP Benchmarks Parameters

表 4 3 种多层感知器仿真基准参数

Layer	MLP-1	MLP-2	MLP-3
Input Layer	784	784	784
Hidden Layer1	500	1 000	1 500
Hidden Layer2	250	500	1 000
Hidden Layer3		250	10
Output Layer	10	10	10

Table 5 VGG-19 Benchmarks Parameters

表 5 VGG-19 网络仿真基准参数

Benchmark	Parameters
VGG-19	conv3x64-conv3x64-pool-conv3x128-conv3x128-pool-conv3x256-conv3x256-conv3x256-conv3x256-pool-conv3x512-conv3x512-conv3x512-conv3x512-pool-conv3x512-conv3x512-conv3x512-conv3x512-pool-4096-4096-1000

5.3 仿真结果与评估分析

文献[3]中提出了一种基于数字忆阻器的神经网络 PIM 结构,与本文相比主要有以下 3 点不同:

1) 使用的忆阻器模型不同.文献[3]中的所有忆阻器都采用数字忆阻器模型,这对数模/模数转换器提出了更高要求,引入了大量转换单元,进而造成大量的功耗和面积消耗.而本文中的设计采用模拟忆阻器模型,只在 bank 的输入输出接口需要使用数模/模数转换单元,可以节省数模/模数转换带来的额外消耗.同时,数字忆阻器的数据密度低于模拟忆阻器,文献[3]中使用 8 个忆阻器存储一个权重数据,每读写一次数据需要对 8 个忆阻单元进行读写,而本文的设计使用 1 个忆阻器存储一个权重数据,节省了 7/8 的存储面积消耗和读写功耗.

2) 从卷积神经网络到阵列的映射方法不同.本文提出的设计将一次卷积运算拆解为横向和纵向 2 个方向,在 2 个阵列中分别计算,并且将权重阵列展开错位放置,使得计算得以加速.

3) 噪声敏感度不同.文献[3]中的数字忆阻器对噪声敏感度较低,噪声需要积累一定量才会使得忆阻器的状态发生错误改变.而本文中的设计采用模拟忆阻器,对噪声敏感度高,少量的噪声积累就可

能发生错误的结果,因此第 4 节中提出了一些适应于模拟电路的噪声消除方法.

下面将利用 HSPICE 平台,根据 6 种不同的 Benchmark 分别搭建电路系统,从图像识别准确率、能耗和性能这 3 个方面分别对以上本文和文献[3]中的设计进行仿真并比较和分析仿真结果.

图 12 为图像识别准确率的仿真结果. 5 组柱状图中的每一组代表一种神经网络仿真基准,假设使用 Matlab 进行浮点运算所得的该神经网络识别准确率为单位 1. 每组的柱体 1(Before Adjust)代表使用本文设计的系统,但不采用任何噪声消除措施的准确率;柱体 2(After Adjust)代表使用本文设计的系统,采用第 4 节所有噪声消除措施的准确率;柱体 3(Design of Ref[3])代表文献[3]中的数字忆阻器系统的准确率. 折线上的点(Accuracy Improvement)代表经过噪声消除措施后准确率的提高量.

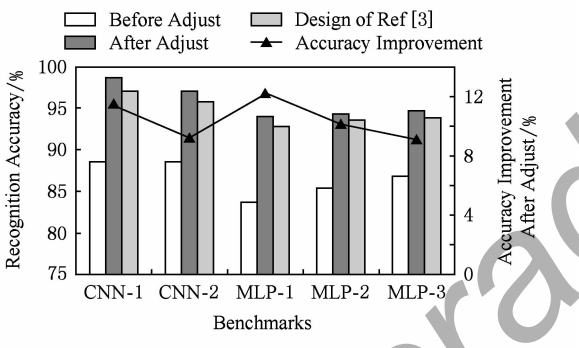

Fig. 12 Simulation results of recognition accuracy

图 12 图像识别准确率仿真结果

由图 12 的仿真结果可以看出:

1) 文献[3]中的数字忆阻器系统的准确率在 95%左右,文本所设计的模拟系统如果不使用任何噪声消除措施,将会比数字系统降低 8%左右的准确率,这是由于模拟电路对噪声的高度敏感性. 而采用了第 4 节介绍的所有噪声消除措施之后,准确率可以比噪声消除前提高 10%左右,比数字系统的准确度略高一些,是可以接受的识别准确率.

2) 每组测试基准中,本文设计的系统与数字系统的准确率差异相差不大,与测试基准的规模、深度并没有明显的关系.

图 13 为能耗节省量的仿真结果. 6 组柱状图中的每一组代表 1 种神经网络仿真基准,每组中柱体 1(Design of Ref[3])代表文献[3]中的数字忆阻器系统的能耗;柱体 2(Proposed Design)代表本文设计的系统的能耗. 折线上的点(Energy Saving)代表本文设计的系统相对于数字系统的能耗节省比.

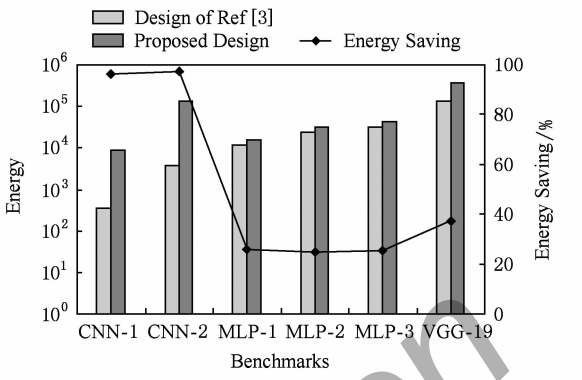

Fig. 13 Simulation results of energy saving

图 13 能耗优化仿真结果

正如 1.2 节所分析的,本文设计的系统拥有比数字系统更低的能耗,节省的能耗主要包括:1)在数模/模数转换模块上的额外功耗;2)在进行多位读写时消耗的额外能耗.

从图 13 的仿真结果可以看出:

1) 对于 6 层的 CNN,如 CNN-1 和 CNN-2,本文设计的系统能够节省超过 95%的能耗.

2) 对于 MLP,本文设计的系统能够节省超过 24%的能耗.

3) 对于 VGG-19,本文设计的系统能够节省 40%的能耗.

4) 综合上面 3 条结论,可以发现当神经网络的规模和深度增加时,能耗的节省量也随之增加. 这是因为在大型网络中具有更多数模/模数转换模块,也需要更多次数的多位读写,能够被优化的能耗占比随着规模和深度增加而快速增长. 而 VGG-19 的网络结构较为特殊,多个卷积层相连,而降采样层数量少,因此取得的效果低于 CNN.

图 14 为性能提升量的仿真结果. 5 组柱状图中的每一组代表一种神经网络仿真基准. 每组中的柱

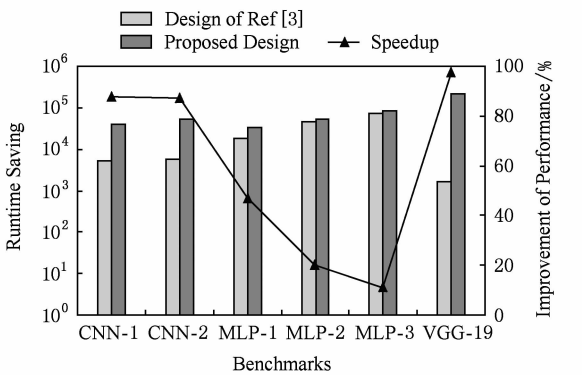

Fig. 14 Simulation results of performance improvement

图 14 性能提升仿真结果

体 1(Design of Ref[3])代表文献[3]中的数字忆阻器系统的执行时间;柱体 2(Proposed Design)代表本文设计的电路的执行时间.折线上的点(Speedup)代表本文设计的系统相对于数字系统的性能提升比.

性能提升主要得益于 WSA 中将权值展开错位放置的策略.假设输入图大小是 $n\times n$,卷积核的大小是 $k\times k$,滑步是 l,该策略能够将原本需要 $(n-k+l)/l\times k$ 个周期的一层卷积层运算缩减到 n 个周期内.

从图 14 的仿真结果可以看出:

1) 对于 6 层的 CNN,如 CNN-1 和 CNN-2,本文设计的系统能够节省超过 90%的运行时间.

2) 对于 MLP,本文设计的系统能够节省 22%~47%的运行时间.

3) 对于 VGG-19,本文设计的系统能够节省超过 98%的运行时间.

4) 综合上面 3 条结论,可以发现当神经网络的规模和深度增加时,运行时间的节省量也随之增加.这是因为当层的规模越大时,每层节省的周期数占比越大;同时,当层的数量越多时,节省的周期数累积占比越多.

6 结 论

此前,基于忆阻器的神经形态近似计算的 PIM 结构中存在面积和能耗浪费,本文设计了新型的基于忆阻器的 PIM 结构实现深度卷积神经网络近似计算,选择模拟忆阻器来减少数模/模数转换次数,改变忆阻器阵列的组织方式来增加数据并行性,并针对该结构中可能存在的精度损失给出了相应的优化策略.本文选取 6 种不同规模和深度的神经网络,在 HSPICE 平台上进行了仿真.实验结果表明,本文所设计的结构在准确度不下降的前提下,在能耗和执行时间上都有明显优势.

参 考 文 献

[1] Hamdioui S, Xie L, Nguyen H A D, et al. Memristor based computation-in-memory architecture for data-intensive applications [C] //Proc of the 2015 Design, Automation & Test in Europe Conf & Exhibition (DATE'15). Piscataway, NJ: IEEE, 2015: 1718-1725

[2] Jo S H, Chang T, Ebong I, et al. Nanoscale memristor device as synapse in neuromorphic systems [J]. Nano Letters, 2010, 10(4): 1297-1301

[3] Chi Ping, Li Shuangchen, Xu Cong, et al. PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory [C] //Proc of the 43rd Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 27-39

[4] Liu Chenchen, Yan Bonan, Yang Chaofei, et al. A spiking neuromorphic design with resistive crossbar [C] //Proc of the 52nd ACM/EDAC/IEEE Design Automation Conf. New York: ACM, 2015: 1-6. Article No. 14, DOI: 10.1145/2744769.2744783

[5] Duan Shukai, Hu Xiaofang, Dong Zhekang, et al. Memristor-based cellular nonlinear/neural network: Design, analysis, and applications [J]. IEEE Trans on Neural Networks & Learning Systems, 2015, 26(6): 1202-1213

[6] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks [C] //Proc of the 25th Int Conf on Neural Information Processing Systems. New York: Curran Associates Inc, 2012: 1097-1105

[7] Zhang Lei, Zhang Yi. Big data analysis by infinite deep neural networks [J]. Journal of Computer Research and Development, 2016, 53(1): 68-79 (in Chinese)
(张蕾, 章毅. 大数据分析的无限深度神经网络方法[J]. 计算机研究与发展, 2016, 53(1): 68-79)

[8] Williams R S. How we found the missing memristor [J]. IEEE Spectrum, 2008, 45(12): 28-35

[9] Tang Tianqi, Xia Lixue, Li Boxun, et al. Spiking neural network with RRAM: Can we use it for real-world application? [C] //Proc of 2015 Design, Automation & Test in Europe Conf & Exhibition. Piscataway, NJ: IEEE, 2015: 860-865

[10] Wang Ying, Zhang Lei, Han Yinhe, et al. ProPRAM: Exploiting the transparent logic resources in non-volatile memory for near data computing [C] //Proc of the 52nd IEEE/ACM Design, Automation Conf. Piscataway, NJ: IEEE, 2015: 1-6. Article No. 47, DOI: 10.1145/2744769.2744897

[11] Wang Ying, Xu Jie, Han Yinhe, et al. Deep Burning: Automatic generation of FPGA-based learning accelerators for the neural network family [C] //Proc of the 53rd IEEE/ACM Design, Automation Conf. Piscataway, NJ: IEEE, 2016: 1-6. Article No. 110, DOI: 10.1145/2897937.2898002

[12] Wang Ying, Li Huawei, Li Xiaowei, Rearchitecting the on-chip memory subsystem of machine learning accelerator for embedded devices [C] //Proc of 2016 IEEE/ACM Int Conf on Computer Aided Design. Piscataway, NJ: IEEE, 2016: 1-6. Article No. 13, DOI: 10.1145/2966986.2967068

[13] Lécun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324

[14] Manem H, Rose G, He Xiaoli, et al. Design considerations for variation tolerant multilevel CMOS/NANO memristor memory [C] //Proc of ACM Great Lakes Symp on VlSI 2009. New York: ACM, 2010: 287-292

[15] Byeon D S, Yoon C W, Park H K, et al. Disturbance-suppressed ReRAM write algorithm for high-capacity and high-performance memory [C] //Proc of the 14th Non-Volatile Memory Technology Symp. Piscataway, NJ: IEEE, 2014: 1-4. DOI: 10.1109/NVMTS.2014.7060837

[16] Huang G M, Ho Yenpo, Li Peng. Memristor system properties and its design applications to circuits such as nonvolatile memristor memories [C] //Proc of 2010 Int Conf on Communications, Circuits and Systems. Piscataway, NJ: IEEE, 2010: 805-810

[17] Archanaa M, Balamurugan K. Analysis of thermal noise and noise reduction in CMOS device [C] //Proc of 2014 Int Conf on Green Computing Communication and Electrical Engineering. Piscataway, NJ: IEEE, 2014: 1-5. DOI: 10.1109/ICGCCEE.2014.6922246

[18] Georgiou P S, Koymen I, Drakakis E M. Noise properties of ideal memristors [C] //Proc of 2015 IEEE Int Symp on Circuits and Systems. Piscataway, NJ: IEEE, 2015: 1146-1149

[19] Yang M T, Ho P P C, Lin C K, et al. BSIM4 high-frequency model verification for 0.13 μm RF-CMOS technology [C] //Proc of 2004 IEEE MTT-S Int Microwave Symp Digest. Piscataway, NJ: IEEE, 2004: 1049-1052

[20] Biolek Z, Biolek D, Biolkova V. SPICE model of memristor with nonlinear dopant drift [J]. Radioengineering, 2009, 18(2): 210-214

[21] Kussul E M, Baidyk T N, Nd W D, et al. Permutation coding technique for image recognition systems [J]. IEEE Trans on Neural Networks, 2006, 17(6): 1566-1579

Li Chuxi, born in 1991. PhD candidate of Northwestern Polytechnical University. Student member of CCF. Her main research interests include computer architecture and neuromorphic architecture.

Fan Xiaoya, born in 1963. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include computer architecture, machine learning, neuromorphic architecture and very large scale integrated circuit design (VLSI).

Zhao Changhe, born in 1994. Master candidate of Northwestern Polytechnical University. His main research interests include computer architecture and neuromorphic architecture.

Zhang Shengbing, born in 1968. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include computer architecture, machine learning, neuromorphic architecture and microelectronic circuit design.

Wang Danghui, born in 1975. PhD, associate professor. Member of CCF. His main research interests include computer architecture, emerging memory, neuromorphic architecture and very large scale integrated circuit design (VLSI).

An Jianfeng, born in 1977. PhD, associate professor. Member of CCF. His main research interests include computer architecture, advanced micro architecture and reconfigurable computing.

Zhang Meng, born in 1978. PhD, assistant professor. Member of CCF. His main research interests include computer architecture, machine learning, neuromorphic architecture and very large scale integrated circuit design (VLSI).