计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20190113
56(6):1182-1191,2019

面向阻变存储器的长短期记忆网络加速器的训练和软件仿真

刘 鹤[1] 季 宇[1] 韩建辉[2] 张悠慧[1] 郑纬民[1]
[1](清华大学计算机科学与技术系 北京 100084)
[2](清华大学微电子学研究所 北京 100084)
(liuhe94@hotmail.com)

Training and Software Simulation for ReRAM-Based LSTM Neural Network Acceleration

Liu He[1], Ji Yu[1], Han Jianhui[2], Zhang Youhui[1], and Zheng Weimin[1]
[1](*Deparment of Computer Science and Technology, Tsinghua University, Beijing* 100084)
[2](*Institute of Microelectronics, Tsinghua University, Beijing* 100084)

Abstract Long short-term memory (LSTM) is mostly used in fields of speech recognition, machine translation, etc., owing to its expertise in processing and predicting events with long intervals and long delays in time series. However, most of existing neural network acceleration chips cannot perform LSTM computation efficiently, as limited by the low memory bandwidth. ReRAM-based crossbars, on the other hand, can process matrix-vector multiplication efficiently due to its characteristic of processing in memory (PIM). However, a software tool of broad architectural exploration and end-to-end evaluation for ReRAM-based LSTM acceleration is still missing. This paper proposes a simulator for ReRAM-based LSTM neural network acceleration and a corresponding training algorithm. Main features (including imperfections) of ReRAM devices and circuits are reflected by the highly configurable tools, and the core computation of simulation can be accelerated by general-purpose graphics processing unit (GPGPU). Moreover, the core component of simulator has been verified by the corresponding circuit simulation of a real chip design. Within this framework, architectural exploration and comprehensive end-to-end evaluation can be achieved.

Key words ReRAM; long short-term memory (LSTM); training algorithm; simulation framework; neural network

摘 要 长短期记忆(long short-term memory, LSTM)网络是一种循环神经网络,其擅长处理和预测时间序列中间隔和延迟较长的事件,多用于语音识别、机器翻译等领域.然而受限于内存带宽的限制,现今的多数神经网络加速器件的计算模式并不能高效处理长短期记忆网络计算;而阻变存储器交叉开关结构能够以存内计算形式完成高效、高密度的向量矩阵乘运算,从而成为一种高效处理长短期记忆网络的极具潜力的加速器设计模式.研究了面向阻变存储器的长短期记忆神经网络加速器模拟工具以及相应的神经网络训练算法.该模拟工具能够以时钟驱动的形式模拟设计者提出的以阻变存储器交叉开关结构为核心加速部件的长短期记忆加速器微体系结构,从而进行设计空间探索;同时改进了神经网络训练算法以适应阻变存储器特性.这一模拟工具基于 System-C 实现,且对于核心计算部分实现了图形处理器加速,可以提高阻变存储器器件的仿真速度,为探索设计空间提供便利.

收稿日期:2019-03-01;**修回日期**:2019-04-10
基金项目:国防科技创新特区项目
This work was supported by the Science and Technology Innovation Special Zone Project.
通信作者:张悠慧(zyh02@tsinghua.edu.cn)

关键词　阻变存储器;长短期记忆网络;训练算法;仿真框架;神经网络

中图法分类号　TP389.1; TP391.9

长短期记忆(long short-term memory, LSTM)神经网络擅长处理和预测时间序列中间隔和延迟很长的事件,因此多用于语音识别、机器翻译、控制机器人、合成音乐等领域.LSTM 的推断计算大概占用了谷歌数据中心的 30%的工作负载[1].然而,LSTM 网络在传统的神经网络加速器上的计算效率却很低.举例来说,对于 TPU[1]而言,相较于卷积神经网络(convolutional neural network, CNN)的 86 Tops/s 的速度,LSTM 神经网络的速度只能达到 2.8Tops/s.其主要原因在于 LSTM 的访存更为密集,因此其计算性能受到内存带宽的严格限制.对于 GPU[2]和 FPGA[3-4]等加速器件,也有类似的结论.

为了解决内存瓶颈对计算性能的影响,用 ReRAM 器件可作为神经网络加速器件的计算核心.非易失性阻变存储器(resistive random-access memory, ReRAM)是新一代的存储器件,其阻值可动态设置.利用此器件的物理特性,可以将 ReRAM 制作成交叉开关阵列(Crossbar)进行神经网络所需的矩阵向量乘法计算.其计算原理是基尔霍夫定律,计算速度快、功耗低,并且避免了一般神经网络加速器计算期间的权重数据移动,很大程度上削弱了内存带宽对计算效率的限制.

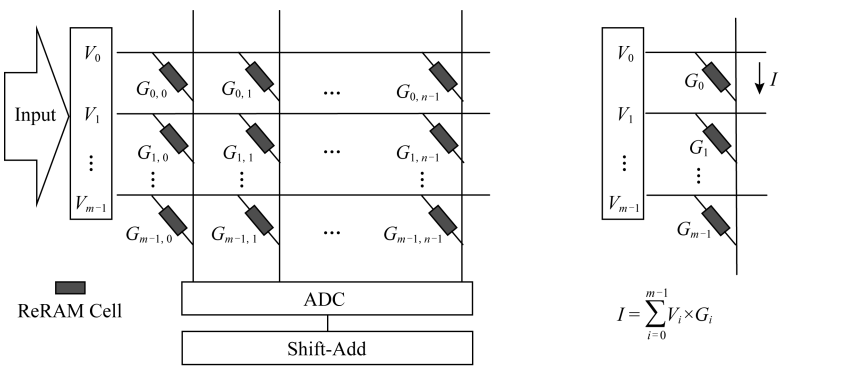
由于 ReRAM Crossbar 本身是模拟电路,相对于一般的数字电路设计有更多的约束条件,为此本文提出了定制化的训练算法,使得 LSTM 网络在训练时就充分考虑了相应的硬件约束,能够更好地反映推断计算的真实运行环境.同时本文还完成了一个针对 ReRAM 器件的 System-C 模拟器,采用行为级电路模拟技术避免了现有的基于 SPICE 的 ReRAM 电路仿真的复杂计算,可以大幅度提高模拟运行速度.相对于 SPICE 仿真,该模拟器引入的误差不超过 2.68%①;且核心计算部分实现了 GPU 加速,大幅提高了模拟速度,为探索设计空间提供了便利.

1　背景与相关工作

1.1　ReRAM

ReRAM 是新一代非易失性存储器件,其具有非易失性、高密度、低功耗、存算合一、易于 3D 堆叠等特点.在神经网络计算中可以利用其阻值可调的特点作为突触器件,并利用其存算合一的优点以 Crossbar 形式完成高速向量矩阵乘运算.

具体的计算方式为:ReRAM 是一种多级(multi-level)内存设备,理论上可以将 ReRAM 的电导值设为其取值范围中的任意值.我们可以将 ReRAM 在 Crossbar 交叉点的阻值设置为神经网络权重矩阵的对应值,这样就可以用一个 ReRAM Crossbar 来表示一个神经网络的权重矩阵,并且根据其物理特性就地(in-situ)完成矩阵向量乘操作.对于理想的 Crossbar 器件,将输入电压 V_i 加到每一行字线(word-line),电压与每一交叉点的电导值 G_{ij} 相乘,根据基尔霍夫定律 $I=GV$,每一列的输出电流 I 可累加得到.该计算过程能达到很高的并行度和速度,过程如图 1 所示:



(a) ReRAM Crossbar　(b) ReRAM multiply-add computation

Fig. 1　ReRAM Crossbar

图 1　ReRAM 交叉开关阵列

① 与真实芯片的电路设计作对比得出该结论.

但实际上基于 ReRAM Crossbar 的计算并不像上文提到的理想情况那样,主要包含 2 种非理想因素,分别是设备 Variation 和周期 Variation.设备 Variation 主要反映在多个 ReRAM 单元之间的参数变动,即当我们设置多个 ReRAM 单元的电导值至某一个给定值时,其实际设置的结果值并不准确地等于预期值,而是服从预期值附近的一个数值分布.周期级的 Variation 则表示单个 ReRAM 单元的不同次操作之间的电导值并不完全一致.

具体地,在设置 ReRAM 电导值的过程中,一个单元的电导值的变化受极性、磁性和输入电压等的影响.同时一些非理想情况,如电导值突变和波动会在反复循环设置电导的过程中出现,因此需要写验证电路来验证所设置的电导值是否符合要求.通常的设置过程如下:将电压脉冲加到 ReRAM 单元上来增大(或减小)单元的电导值,直到该电导值大于(或小于)期望的电导值.如果实际电导值与期望的电导值相差过大,则需要进行相反的操作来减小(或增大)电导值.同时由于器件不完美特性和设置过程开销(用于验证电路的参考电压数量有限),实际可用的电导取值通常是离散值.这是当前很多基于 ReRAM 进行神经网络计算加速的研究工作使用离散电导值的一个原因.另一个原因在于,使用离散的权重值有利于提升神经网络的抗噪能力.

由于整个加速芯片系统的主要部分仍是数字电路,因此数模转换(模数转换),即 DAC/ADC,是很大的硬件开销.在配置 DAC/ADC 的时候,需要综合考虑神经网络的准确率和硬件开销的均衡.

数模转换器(digital analog converter, DAC)的作用是将 Crossbar 每一行的数字输入向量转换成模拟信号.其重要参数是转换分辨率 d,即每次可转换的信号位数.在电路模块中,DAC 要占用不小的硬件面积,因此很多设计(如 ISAAC[5] 等)将 S-bit 宽度的数字输入信号先转换成一系列低精度的数字信号(通常精度是宽度为 1,即 $d=1$),再经过 DAC;这也意味着 Crossbar 要进行(S/d)次的顺序操作,这是一种用时间换空间的优化策略.DAC 的个数设为 $2^r\times C$.

模数转换器(analog digital converter, ADC)的作用是将 Crossbar 的每一列电流输出转换成数字信号,其传感分辨率为 a.为了减小硬件开销,通常使用时分复用的方式,即每个 Crossbar 列的输出依次进入到 ADC 中,ADC 将其逐个转换成输出信号.每个 Crossbar 的 ADC 的数量是 A.

1.2 LSTM 神经网络

LSTM[6] 网络是一种带门结构的循环神经网络(recurrent neural network, RNN),适合于处理和预测时间序列中间隔和延迟较长的重要事件.

LSTM 网络引入了判断旧有信息是否有用的单元(cell).Cell 的构成如图 2 所示:

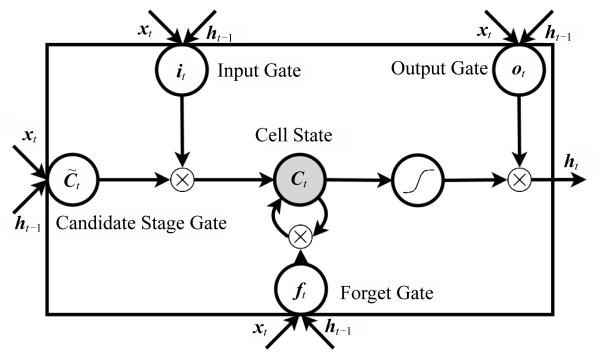


Fig. 2 LSTM Cell[7]

图 2 LSTM 计算单元[7]

遗忘门(forget gate)控制当前输入 $\boldsymbol{x}_t$ 对来自之前 Cell 状态删除信息的影响程度:

$$\boldsymbol{f}_t=\sigma(\boldsymbol{W}_{\text{for}}\cdot\boldsymbol{x}_t+\boldsymbol{R}_{\text{for}}\cdot\boldsymbol{h}_{t-1}+\boldsymbol{b}_{\text{for}}). \tag{1}$$

输入门(input gate)控制当前输入 $\boldsymbol{x}_t$ 对当前 Cell 状态中新信息的增量的影响程度:

$$\boldsymbol{i}_t=\sigma(\boldsymbol{W}_{\text{in}}\cdot\boldsymbol{x}_t+\boldsymbol{R}_{\text{in}}\cdot\boldsymbol{h}_{t-1}+\boldsymbol{b}_{\text{in}}). \tag{2}$$

输出门(output gate)控制当前输入 $\boldsymbol{x}_t$ 对当前网络输出的直接影响:

$$\boldsymbol{o}_t=\sigma(\boldsymbol{W}_{\text{out}}\cdot\boldsymbol{x}_t+\boldsymbol{R}_{\text{out}}\cdot\boldsymbol{h}_{t-1}+\boldsymbol{b}_{\text{out}}). \tag{3}$$

候选阶段门(candidate stage gate)代表当前输入 $\boldsymbol{x}_t$ 所创建的新的信息:

$$\tilde{\boldsymbol{C}}_t=\tanh(\boldsymbol{W}_{\text{can}}\cdot\boldsymbol{x}_t+\boldsymbol{R}_{\text{can}}\cdot\boldsymbol{h}_{t-1}+\boldsymbol{b}_{\text{can}}). \tag{4}$$

当前细胞状态(current cell state)是需要被遗忘和需要被吸收的内容的组合,其计算模式$\otimes$表示逐点乘:

$$\boldsymbol{C}_t=\boldsymbol{f}_t\otimes\boldsymbol{C}_{t-1}+\boldsymbol{i}_t\otimes\tilde{\boldsymbol{C}}_t. \tag{5}$$

网络输出(network output)是输入 $\boldsymbol{x}_t$ 与当前细胞状态的组合,其计算模式为逐点乘:

$$\boldsymbol{h}_t=\boldsymbol{o}_t\otimes\tanh(\boldsymbol{C}_t). \tag{6}$$

式(1)~(6)中,$\boldsymbol{x}_t$ 为当前输入向量,$\boldsymbol{h}_t$ 为当前 LSTM 的输出,$\boldsymbol{h}_{t-1}$ 为前一时刻 LSTM 的输出.$\boldsymbol{f}_t$, $\boldsymbol{i}_t$, $\boldsymbol{o}_t$ 分别为遗忘门、输入门和输出门的输出,$\tilde{\boldsymbol{C}}_t$ 为候选状态门的输出,$\boldsymbol{C}_t$ 为当前细胞状态,$\boldsymbol{C}_{t-1}$ 为前一时刻细胞状态.$\boldsymbol{W}_{\text{for}}$, $\boldsymbol{W}_{\text{in}}$, $\boldsymbol{W}_{\text{out}}$, $\boldsymbol{W}_{\text{can}}$ 分别是对应门的输入权重.$\boldsymbol{R}_{\text{for}}$, $\boldsymbol{R}_{\text{in}}$, $\boldsymbol{R}_{\text{out}}$, $\boldsymbol{R}_{\text{can}}$ 分别是对应门的递归权重.$\boldsymbol{b}_{\text{for}}$, $\boldsymbol{b}_{\text{in}}$, $\boldsymbol{b}_{\text{out}}$, $\boldsymbol{b}_{\text{can}}$ 分别是对应门的偏置权重.

在实际的神经网络计算过程中,由于 LSTM 各门计算均为矩阵向量乘操作,因此常常将 $\boldsymbol{f}_t$,$\boldsymbol{i}_t$,$\boldsymbol{o}_t$,$\widetilde{\boldsymbol{C}}_t$ 这 4 个门的计算置于一次矩阵运算之中,以减少计算的调用次数,提高计算的效能.在矩阵运算之后,再分别进行对应的激活函数和逐点乘法的操作.

1.3　相关工作

在训练层面,文献[8]提出了一种通过补偿设备变动的方式来增强鲁棒性的训练方案.文献[9]使用二值网络训练来降低 ReRAM 表示能力的需求.在基于 ReRAM 的神经网络加速器设计方面也有一些相关工作.ISAAC[5] 是一个使用 Crossbar 的 CNN 加速器,提出了流水线化的微体系结构和新的数据编码技术.PipeLayer[10] 是一个面向 CNN 的基于 ReRAM 的内存计算(processing in memory, PIM)加速器,可以支持训练和推断计算.

在模拟器层面,尚未有研究提出一个开放的模拟框架.NVSim[11] 对器件面积、时序、动态能量等非易失性存储技术进行建模,但不是针对神经网络加速计算.PIMSim[12] 是一个内存计算的模拟器,用于传统的内存技术而不支持内存和计算的本地化.NeuroSim[13] 是一个面向神经网络的基于非易失性存储阵列结构的集成模拟框架,但是其目标用户是希望利用自身的模拟突触设备快速评估系统级性能的设备工程师.该模拟架构结构简单(仅支持针对 MNIST 手写体识别数据集的 2 层 MLP)并且缺乏相应的训练和映射软件.MNSIM[14] 是一个针对基于 ReRAM 的神经形态系统的仿真平台,包括一个层次化的神经形态计算加速器结构和可以灵活定制的接口,以及一个行为级的计算准确度模型.但是 MNSIM 只实现了仿真功能而不支持实际神经网络应用的评估.因此,本文是第 1 个开源的针对 LSTM 的基于 ReRAM 的神经网络加速的训练和仿真平台.

2　针对 ReRAM 特性与加速器结构的 LSTM 神经网络训练

2.1　神经网络权重量化

神经网络权重量化(quantization)是神经网络的常用压缩算法.一般认为,神经网络参数过多、计算过于密集,因此需要通过压缩算法以降低神经网络的有效权重数量,这样既可以减少参数,又能简化计算;同时也能够降低数值表示精度,即用低精度计算来代替高精度浮点数计算,达到降低开销的目的.本文采用神经网络权重量化的出发点,主要是从 ReRAM 器件特性的角度来考虑.

正如 1.1 节介绍的,ReRAM 器件的优势在于速度快、功耗低、并行性非常好,但是由于当前工艺不够成熟以及器件本身的不完美特性,其数值表示精度低(每个单元所能表示的权重的离散值范围与个数都有限)且受器件噪声的影响较大,因此需要在训练阶段对神经网络权重进行量化,使得神经网络能够符合实际器件的约束.其主要思想是将原有的权重进行放缩至器件精度范围如 2^8,然后四舍五入为整数,并用该整数来表示权重值.

2.2　层间 I/O 精度限制

在层间数据的传输时,数据的精度也常常受到硬件开销的限制.若该层间传输的精度限制为 S-bit,权重精度为 b-bit,输入的行数有 2^c 行,则在 Crossbar 中计算时,其结果数据的数值范围为 $2^S\times 2^b\times 2^c=2^{S+b+c}$,即输出的结果精度为$(S+b+c)$-bit,该结果需要被截取到层间数据传输的限制 S-bit.截取策略一般有 2 种,最直接的方法是对输出结果等比例放缩到 S-bit.这种方法的弊端是,当结果分布不均匀的时候会有较大的精度损失.另一种策略是截取结果中连续的 S-bit 作为最终结果,截取的起始位置则由数据分布来确定.

2.3　DAC 与 ADC 及移位相加

在训练阶段还需要考虑 ReRAM 加速器所需的 DAC 和 ADC 以及移位相加(shift-add)对神经网络的影响.因此在训练的每次矩阵向量乘计算之前,需要将原有的输入值按位进行拆分,即将原有的 1 个 8-bit 输入向量拆分为 8 个 1-bit 输入向量,并将其合并成新的输入矩阵(训练阶段以此来加速训练过程),以此操作模拟 DAC 的转换过程.

在训练阶段的矩阵向量乘计算之后,首先需要统计该层所有输出结果的最大值,并根据此最大值确定 ADC 的参考电流(该参考电流值也将用于模拟器的 ADC 模块).在确定了该层 ADC 的参考电流之后,再将输出结果按照 DAC 的顺序,每 8 个行向量移位相加成 1 个行向量(即将扩展了 8 倍大小的输出矩阵还原成 DAC 拆分之前的同一层输出大小).

2.4　ReRAM 噪声分布

ReRAM 器件的噪声是指每个单元的噪声,其大小与单元的电导值(即神经网络权重)有关.每个单元的实际电导值为均值为单元目标电导值的正态

分布,则每个单元的噪音为均值为0、标准差受电导值影响的正态分布:$G_{noise}=N(0,\delta^2)$.本文采用的ReRAM器件标准差δ符合(由实际测量值拟合得出[15]):

$$\delta=-0.000\,603\,4x^2+0.061\,84x+0.724\,0, \quad (7)$$

其中,x是ReRAM单元的电导值.在Crossbar计算的过程中,由于每次读取权重都会有偏差(ReRAM特性),因此在以上正态分布的约束下,每次计算都需要重新生成G_{noise}值,并将其加在ReRAM单元的原有电导值上,作为新的读出值.

2.5 针对ReRAM特性的神经网络训练算法

在神经网络训练时,为了快速收敛,首先按照经典神经网络训练方法进行训练,得到高精度模型,在此基础上,使用本文提出的定制化的训练算法,训练得到符合ReRAM约束的神经网络模型.综合上文提到的ReRAM特性,本文提出的神经网络训练算法如下:

1) 前向过程(forward pass)

步骤1. 神经网络权重8-bit量化.针对各神经网络层,分别选取各层的最大值max($\boldsymbol{W}$);保存现有权重为$\boldsymbol{W}_{old}$;计算$\boldsymbol{W}_{new}$:

$$\boldsymbol{W}_{new}=round(\boldsymbol{W}_{old}/\max(\boldsymbol{W})\times(2^7-1))/(2^7-1)\times\max(\boldsymbol{W}). \quad (8)$$

步骤2. 层间I/O精度8-bit限制.对每一层输入$\boldsymbol{input}$,分别选取最大值max($\boldsymbol{input}$),继而计算出最大值位数$n_{max}=ceil(\mathrm{lb}(\max(\boldsymbol{input})))$;对$\boldsymbol{input}$进行缩放:$\boldsymbol{input}_{new}=round(\boldsymbol{input}\times 2^{8-n_{max}})$.

步骤3. DAC转换.将步骤2中获得的$\boldsymbol{input}_{new}$进行按位拆分,生成新的1-bit向量.工作流程如图3所示.

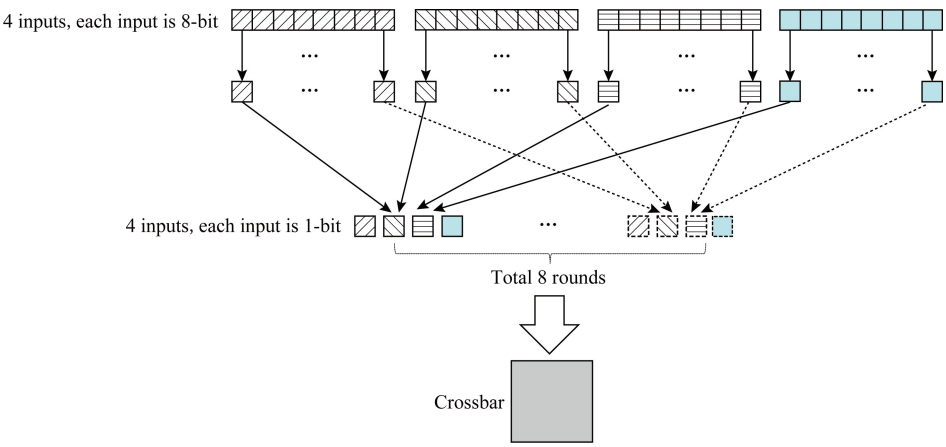


Fig. 3 DAC Workflow

图3 DAC工作流程图

步骤4. 计算并添加噪音.首先生成标准正态分布噪音$\boldsymbol{normal}$,进一步根据式(7)得到标准差为δ的$\boldsymbol{V}_{noise}$;将$\boldsymbol{V}_{noise}$加到现有$\boldsymbol{W}_{new}$上,得到带噪音的权重值$\boldsymbol{W}_{noise}$.将DAC生成的1-bit向量与$\boldsymbol{W}_{noise}$相乘,得到输出$\boldsymbol{output}$.

步骤5. ADC与shift-add.首先确定各层参考电压AD_V,其值为每层$\boldsymbol{output}$的最大值,即$AD_V=\max(\boldsymbol{output})$.将每个$\boldsymbol{output}$值转换成8-bit向量:$\boldsymbol{output}_t=round(\boldsymbol{output}/AD_V\times(2^7-1))$.按照DAC拆分的顺序,每8个$\boldsymbol{output}_{t_i}$进行移位相加:$\boldsymbol{res}=\sum_{i=0}^{7}\boldsymbol{output}_{t_i}\times 2^i$. $\boldsymbol{res}$为最终输出结果向量.

2) 反向过程(backward pass)

步骤1. 将现有网络权重值$\boldsymbol{W}_{noise}$恢复为$\boldsymbol{W}_{old}$.

步骤2. 使用反向传播算法(backpropagation)更新权重值.

以上为本文提出的针对ReRAM特性的神经网络训练算法.利用此算法可以训练得到符合ReRAM约束的神经网络模型.

3 模拟器架构

3.1 整体架构

模拟器的主要计算模块为LSTM模块和Linear模块,分别负责LSTM网络和全连接网络的计算.除了计算模块,还有数据输入和输出模块,以及相应的数据缓冲区模块(LSTM模块的缓冲区与全连接模块的缓冲区结构相似,仅有规模参数不同,因此视为同类模块).如图4所示,Linear模块中包括DAC、

若干 ReRAM 计算阵列、ADC 和 shift-add 以及激活函数,其输出值将发送给下一层数据缓冲区(或者作为模拟器的输出结果).LSTM 模块的特殊性在于,其激活函数为非线性函数 Sigmoid 和 tanh(而非 Linear 和 CNN 中常用的 ReLU),并且需要引入新的控制和计算功能——向量拼接和逐点乘.

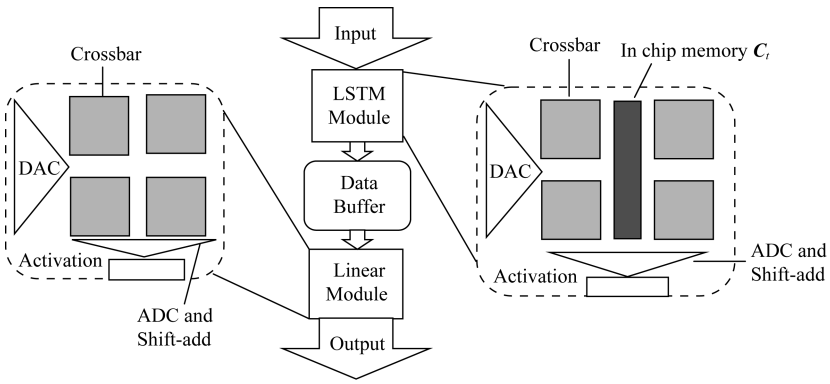


Fig. 4　Overall architecture

图 4　整体架构

3.2　处理单元

1) DAC 模块

DAC 模块的主要作用是将输入数据(8-bit)逐位转换成 1-bit 数据,即将一个 8-bit 向量转换成 8 个 1-bit向量,依次输送给 Crossbar 模块进行计算.如图 4 所示,对于 4 个 8-bit(层间 I/O 限制)输入数据,对其 bit 位进行拆分,每个 8-bit 输入数据按照从高位到低位拆分成 8 个 1-bit 数据;再按照高低位次序,分 8 次输入到 ReRAM Crossbar 中进行计算,因此 Crossbar 中的计算需要 8 个时间步完成.

2) Crossbar 模块

如图 1 所示,将一列输入电压与 ReRAM 单元的电导值(权重值)相乘加得到输出电流.在此过程中,同时需考虑 2.4 节提到的 ReRAM 单元的噪音,即对每一个 ReRAM 单元,其计算电流值为 $I_{ij}=V_i\times(G_{ij}+noise_{ij})$.Crossbar 中的权重值经预先训练和转换得到,在模拟器运行启动时读入.Crossbar 模块的大小预先设置为 1 152×128,其参数也是可调的.

3) ADC 与 shift-add 模块

本模块的主要作用是将 Crossbar 模块的计算输出转换成数字电路数值,并且根据 DAC 转换的高低位依次进行移位相加.在 ADC 的运行之前,需要预先确定参考电流 I.其作用是当 ADC 的输入电流 $i>I$ 或 $i<-I$ 时,要将其截取为 I 或 $-I$;否则,保持 i 不变.然后将 i 放缩到 8-bit 的区间(−127,127),该值为 ADC 的输出值 a.在经过 ADC 之后,根据 DAC 转换的高低位,将 a 依次左移相加,其计算公式为 $result=\sum_{i=0}^{7}a_i\times 2^i$,其中 a_i 为第 i 个 ADC 输出值.累加结果 *result* 即为本模块的计算结果.

4) LSTM 模块

在 LSTM 模块中,集成了从上层获取数据、DAC 转换、Crossbar 的计算、ADC 转换这 4 部分.除此之外,由于 LSTM 网络计算的特殊性,还需要引入逐点乘操作和非线性(激活)函数.另外还需要在模拟器中开辟专门的存储空间来存放本模块循环输入给下一个时间步的结果 $\boldsymbol{C}_t$.

与传统的 CNN 计算方式不同,LSTM 模块的计算需要对数据进行连接,即将输入数据 $\boldsymbol{x}_t$ 与隐层状态 $\boldsymbol{h}_{t-1}$(同时也是上一个时间步 $t-1$ 时刻的 LSTM 模块的输出)拼接成一个新的向量.受此影响,ReRAM Crossbar 中的权重在映射时也需要考虑将各门权重值进行拼接.如图 5 所示,在 Crossbar 计算和 ADC 转换之后,得到的中间结果为 $\boldsymbol{f}_t$,$\boldsymbol{i}_t$,$\boldsymbol{o}_t$,$\widetilde{\boldsymbol{C}}_t$ 各门的输出.该输出结果需要进一步进行逐点乘操作和激活函数计算.逐点乘的操作模式为 2 个向量对应位相乘,得到新的同维度的输出向量.LSTM 模块中的激活函数为 Sigmoid 和 tanh,都是非线性函数,直接使用数值计算的方式使其计算开销很大.考虑到 Crossbar 计算过程中数据的精度一般限制为 8-bit,因此在激活函数的实现上可以使用查表的方式(表项不超过 2^8 个).

经过激活函数的计算后得出的结果 $\boldsymbol{h}_t$ 作为下一时刻本层的输入,$\boldsymbol{C}_t$ 作为当前细胞状态值用于计算下一时刻的细胞状态,因此需要在本层中存入相应的数据缓冲区,待下一时刻本层计算 $\boldsymbol{C}_{t+1}$ 时使用.

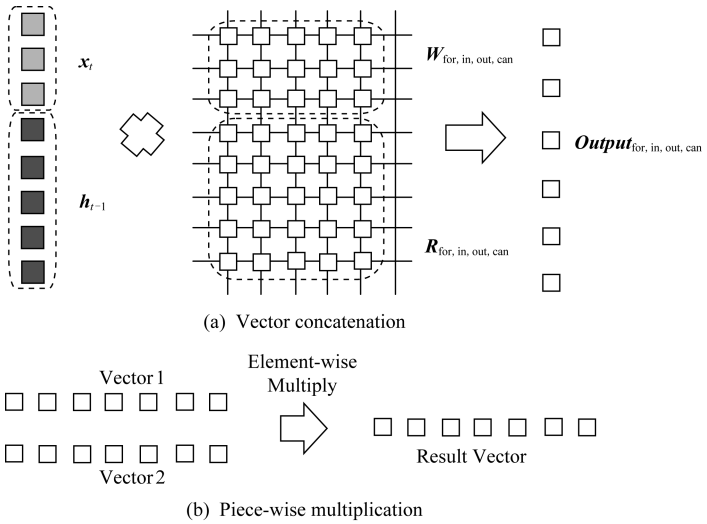


Fig. 5 Computation Mode in LSTM module

图 5 LSTM 模块的计算模式

3.3 GPU 加速计算

模拟器的主要计算部分为矩阵向量乘,同时还要在每次进行 Crossbar 计算时生成满足高斯分布的噪音(需要生成随机数),所以使用 CPU 进行计算时其时间开销很大.而 GPU 作为 SIMD 硬件,其计算并行性很高,适合做矩阵相乘的操作,目前也多用于神经网络计算,具有良好的加速效果.因此我们考虑使用 GPU 来加速模拟器,即将 Crossbar 的主要计算部分由 GPU 来完成,包括 ReRAM 单元噪音的生成.在 GPU 计算时,通过编写 cuda 代码,调用用于生成随机数的 curand 库生成符合条件的高斯分布噪声,并将其加到对应的 ReRAM 单元的电导值上.加速算法的步骤如下:

步骤 1. 初始化权重 $\boldsymbol{W}$.将所有层 Crossbar 的权重加载到 GPU 内存中,并根据权重值计算该层权重对应的噪音标准差 δ.

步骤 2. 在 System-C 模拟器中,将每层输入拼接成 1 个大输入向量 $\boldsymbol{input}$,将 $\boldsymbol{input}$ 发送给 GPU.

步骤 3. 调用 curand 库,生成标准差为 δ 的噪音 $\boldsymbol{V}_{\text{noise}}$,并将该 $\boldsymbol{V}_{\text{noise}}$ 与 $\boldsymbol{W}$ 相加,得到 $\boldsymbol{W}_{\text{noise}}$.GPU 调用 cublas 库进行 $\boldsymbol{input}$ 和 $\boldsymbol{W}_{\text{noise}}$ 的矩阵乘计算,得到输出向量 $\boldsymbol{output}$.

步骤 4. 将输出向量 $\boldsymbol{output}$ 进行转置(GPU 中显存的存储方式为列优先,而 CPU 内存中的存储方式为行有限).将结果拷贝回 CPU 内存.

在具体实现上,使用 CMake 进行编译,其中 System-C 相关代码需要调用 System-C 2.3.2 库,而 Crossbar 计算部分则单独拆分出来,使用 NVCC 进行编译.在程序链接时,将 2 部分代码生成一个可执行文件.

本项目所用的 GPU 为 NVIDIA Tesla P100,其显存大小为 12 GB.加速效果如表 1 所示:

Table 1 GPU Acceleration Result

表 1 使用 GPU 计算的加速效果

Data Format	No-noise		Noise	
	CPU/s	GPU/s	CPU/s	GPU/s
512×1 024×1 024	2.3	0.13	65.1	0.156
100×3 000×3 000	3.9	0.021	109	0.039
3 000×1 152×128	2.0	0.118	54.5	0.135

由表 1 所示,"数据规格"为所设置的 Crossbar 大小,其中第 1 维表示 Crossbar 个数,后两维表示单个 Crossbar 的长和宽.在表 1 中,第 1,2 行分别列出了中等矩阵规模(1 024×1 024)和较大矩阵规模(3 000×3 000)的加速数据.由于当前 ReRAM 工艺所限,其 Crossbar 阵列规模亦有限,因此在最后一行给出了 Crossbar 尺寸为 1 152×128 的计算加速数据.表 1 中的第 2 列和第 3 列首先使用无噪音的随机生成数据作为权重验证计算准确性,可以看出,在无噪音情况下,当阵列规模较小的时候(第 1 行和第 3 行)使用 GPU 进行计算可以达到 16～17×左右的加速比,在阵列规模较大的时候(第 2 行)更是能够发挥 GPU 的并行优势,达到 185×的加速比.之后再用有噪音的随机生成数据评估性能,在使用 GPU 进行加速计算的情况下,当阵列规模较小时

(第 1 行和第 3 行)加速比可达到 400×左右,而当阵列规模较大时(第 2 行)其加速比更可达 2 794×.由以上数据可以看出加速效果十分显著.

4　案例研究与结果评估

作为案例研究,我们实现了如第 3 节架构的一个模拟器,并据此评估我们的训练算法.

4.1　模拟器的实现

我们实现了一个 ReRAM Crossbar 仿真框架,并给出了基于 Pytorch 的定制化训练算法,包含第 3 节中提到的诸多模拟电路特性;模拟器仿真部分基于 System-C 实现,ReRAM 设备模型来自文献[15],默认的 I/O 和权重精度均为 8-bit(可以配置).

在具体的仿真实现上有诸多参数可以设置,包括电路参数和神经网络参数,如表 2 和表 3 所示:

Table 2　Circuit Parameters for Simulator

表 2　模拟器所需电路参数

Parameters	Value
DA Reference Voltage/V	1.0
AD Reference Voltage	Get after training
Crossbar Length	1 152
Crossbar Width	128
Crossbar Number in Each PE	4

Table 3　Neural Network Parameters for Simulator

表 3　模拟器所需神经网络参数

Parameters	Value
Input Length	118 848
Input Size	39
Hidden Size	128
Output Size	61

基于表 2 和表 3 所列参数,针对所需的仿真模型,通过提前设置 LSTM 模块数、Linear 模块数、数据缓冲区模块数,并通过 System-C 中的信号将相关模块串联,即可生成特定的模拟器,且该流程可通过代码脚本自动生成.

对 ReRAM 神经网络加速器而言,面积和功耗是重要指标之一,因此本项目模拟器对此进行了 32 nm 尺寸下的参考设计,具体参数如表 4 所示.

其中,缓冲区(buffer)相关的模型参数来自 CACTI[16],Crossbar 面积参数来源于文献[17],其他参数采用了 ISAAC[5]的参数.

Table 4　Hardware Parameters for Simulator

表 4　模拟器所需硬件参数

Module	Power/mW	Area/mm^2
ADC(8-bit)	2	0.0012
DAC(1-bit)	0.00391	1.66016E−07
S&H	9.76563E−06	3.90625E−08
X-bar	0.3	0.000148
Shift-add	0.05	0.00006
Input-buf(2 KB)	1.24	0.0021
Output-buf(256 KB)	0.23	0.00077
ReLU	0.003 2	8.9E−06

4.2　仿真结果的评估

4.2.1　仿真算法评估

我们实现的 LSTM 网络为应用在一个在 TIMIT 数据集上进行语音识别的网络.该数据集的训练集、验证集和测试集大小分别为 3 696,400 和 192.其中每条测试集语音帧数为 619,因此总计有 118 848 帧测试数据.其神经网络输入维度为 39(经过 MFCC 预处理后的语音数据,即每一帧的向量长度),输出维度为 61(语素分类数),包含一个 LSTM 层(隐层大小 128)和一个全连接层.表 5 为仿真算法评估结果.

Table 5　Evaluation Result for Training Algorithm

表 5　训练算法评估结果

Training	Inference	TIMIT Accuracy/%
No ReRAM crossbar constraint	No ReRAM crossbar constraint	84.22
8-bit weight quantize and I/O constraint	8-bit weight quantize and I/O constraint	84.17
No ReRAM crossbar constraint	With ReRAM crossbar constraint	82.53
Customized training algorithm(with ReRAM crossbar constraint)	With ReRAM crossbar constraint	84.05

我们使用通用的神经网络训练算法作为基准,经过 30 个周期的训练,其分类准确度可以达到 84.22%,此网络模型称为基准模型.在此基础上,我们在训练中引入了神经网络 8-bit 的权重量化和 I/O 限制,结果显示其准确度为 84.17%,即有轻微的下降.接下来将基准模型直接部署模拟器上(后者在模拟过程中引入了所有的限制因素,包括权重量化、I/O 限制、ReRAM 的非理想因素等),结果显示由于权重量化、器件噪音以及 DA/AD 等的影响,识别准确度下降为 82.53%.最后我们将基准模型在

ReRAM 约束下进行微调(fine-tune),引入了所有限制因素,在网络收敛后,可以得到其识别准确度 84.05%,这个数据与将该网络部署到模拟器上获得的精度一样.

表 5 表明:传统的训练算法在应用到 ReRAM 的推断过程中,权重的量化和 I/O 精度限制对神经网络的表达结果影响很小,但是由于器件噪音以及 DA/AD 等的约束,会导致性能下降.而我们的定制化训练算法将以上因素考虑到训练过程中,再将训练好的模型应用到 ReRAM Crossbar 的计算之上,可以有效减小器件因素带来的性能下降.

4.2.2 模拟器的仿真准确度评估

在模拟器本身的准确性方面,我们做了基于 SPICE 模拟的评估.这是因为模拟器本身需要能够满足实际器件的物理特性和约束.现有的基于 SPICE 的 ReRAM 模拟工作,其电路方程非常复杂(对于一个 $2^r \times 2^c$ 的 Crossbar,需要求解 $2^r \times 2^c + 2^r \times (2^c - 1)$个电压参数和 $3 \times 2^r \times 2^c$ 个电流参数,且方程是非线性的),因此求解难度很高,模拟仿真速度慢.

本项目模拟器在计算速度和仿真准确性上做了权衡:1)Crossbar 的输入电压为 1-bit 的值(只有高低电压之分),以此来消除非线性的影响;2)考虑到 Crossbar 阵列中连线上的电容和电感对于计算影响很小[18],因此将其忽略;3)每当读取电导值时,都要引入 ReRAM variation[19].

在此基础上,我们将结果与实际芯片的电路级仿真进行了比较.在阵列大小为 1 152×128 规模下,ReRAM 单元高阻和低阻分别为 800 KΩ 和 50 KΩ,导线电阻(bit-line, source-line, work-line)分别为 87 mΩ,100 mΩ,1.16 Ω(130 nm 工艺下的 CMOS 电路).在此条件下,我们得到的行为级模型计算的结果电流值与电路仿真的结果电流值其误差不超过 2.68%.

5 结束语

本文提出了基于 ReRAM 的长短期记忆网络加速器训练和仿真框架,包括针对 ReRAM 器件特性的定制训练算法、时钟驱动的行为级模拟器及其 GPU 加速.验证结果显示训练算法能够有效降低模拟器件带来的噪音和数值精度损失等不利因素的影响,而该模拟器与 SPICE 仿真的计算结果误差在 2.68%以内,并且避免了 SPICE 对电路仿真耗时过长的缺点.与现有的工作相比,本文提出的模拟器项目完成了端到端的训练和仿真,并将一个实际应用于 TIMIT 数据集语音分类的 LSTM 网络部署并运行.未来我们会继续完善该模拟器,增强各模块的可配置性,为相关的加速器硬件设计提供探索方案.

致谢 感谢北京市未来芯片技术高精尖创新中心的支持!

参 考 文 献

[1] Jouppi N P, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit [C] //Proc of the 44th Annual Int Symp on Computer Architecture. New York: ACM, 2017: 1-12

[2] Xcelerit. Benchmarks: Deep Learning Nvidia P100 vs V100 GPU [OL]. [2017-11-27]. https://www.xcelerit.com/computing-benchmarks/insights/benchmarks-deep-learning-nvidia-p100-vs-v100-gpu/

[3] Han Song, Kang Junlong, Mao Huizi, et al. ESE: Efficient speech recognition engine with sparse LSTM on FPGA [C]// Proc of the 2017 ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2017: 75-84

[4] Wang Suo, Li Zhe, Ding Caiwen, et al. C-LSTM: Enabling efficient LSTM using structured compression techniques on FPGAs [C] //Proc of the 2018 ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2018: 11-20

[5] Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars [C] //Proc of the 43rd Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 14-26

[6] Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural Computation, 1997, 9(8): 1735-1780

[7] Evangelopoulos G N. Efficient hardware mapping of long short-term memory neural networks for automatic speech recognition [D]. Belgium: KU Leuven, 2016

[8] Liu Beiye, Li Hai, Chen Yiran, et al. Vortex: Variation-aware training for memristor x-bar [C]// Proc of the 52nd ACM/EDAC/IEEE Design Automation Conf (DAC). Piscataway, NJ: IEEE, 2015: 1-6

[9] Tang Tianqi, Xia Lixue, Li Boxun, et al. Binary convolutional neural network on RRAM [C] //Proc of the 22nd Asia and South Pacific Design Automation Conf. Piscataway, NJ: IEEE, 2017: 782-787

[10] Song Linghao, Qian Xuehai, Li Hai, et al. PipeLayer: A pipelined ReRAM-based accelerator for deep learning [C] // Proc of 2017 IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2017: 541-552

[11] Dong Xiangyu, Xu Cong, Xie Yuan, et al. NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, 31(7): 994-1007

[12] Xu Sheng, Chen Xiaoming, Wang Ying, et al. PIMSim: A flexible and detailed processing-in-memory simulator [J]. IEEE Computer Architecture Letters, 2018, 18(1): 6-9

[13] Chen Paiyu, Peng Xiaochen, Yu Shimeng. NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(12): 3067-3080

[14] Xia Lixue, Li Boxun, Tang Tianqi, et al. MNSIM: Simulation platform for memristor-based neuromorphic computing system [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(5): 1009-1022

[15] Yao Peng, Wu Huaqiang, Gao Bin, et al. Face classification using electronic synapses [J]. Nature Communications, 2017, 8: 15199

[16] Muralimanohar N, Balasubramonian R, Jouppi N P. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0 [C] //Proc of the 40th Annual IEEE/ACM Int Symp on Microarchitecture (MICRO-40 2007). Piscataway, NJ: IEEE, 2007: 3-14

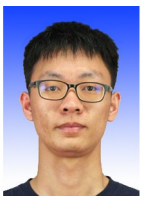
[17] Long Yun, Na T, Mukhopadhyay S. ReRAM-based processing-in-memory architecture for recurrent neural network acceleration [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2018, 26(12): 2781-2794

[18] Gu Peng, Li Boxun, Tang Tianqi, et al. Technological exploration of RRAM crossbar array for matrix-vector multiplication [J]. Journal of Computer Science and Technology, 2016, 31(1): 3-19

[19] Lee S R, Kim Y B, Chang M, et al. Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage class memory [C] //Proc of 2012 Symp on VLSI Technology. Piscataway, NJ: IEEE, 2012: 71-72

Liu He, born in 1994. Master candidate. His main research interests include simulation framework for neural network accelerator.

Ji Yu, born in 1993. PhD candidate. His main research interests include neural network accelerator and compiler and machine learning for system optimization.



Han Jianhui, born in 1994. PhD candidate. His main research interests include emerging technology-based machine learning accelerator design.

Zhang Youhui, born in 1976. Professor and PhD supervisor. Member of CCF, ACM and IEEE. His main research interests include computer architecture and neuromorphic computing.

Zheng Weimin, born in 1946. Professor and PhD supervisor. His main research interests include high performance computing, network storage and parallel compiler.