doi:10.3772/j.issn.1002-0470.2019.01.002

低面积低功耗的机器学习运算单元设计①

周圣元②*,**,*** 杜子东* 刘道福*,*** 支 天* 陈云霁③*,**

(*中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)

(**中国科学院大学 北京 100049)

(***上海寒武纪信息科技有限公司 上海 201203)

摘 要 随着机器学习(ML)算法的日益流行,研究人员提出了很多专用于机器学习算法的加速器。然而,这些加速器会被其特定用途的狭窄范围所限制。另外,尽管芯片制造工艺有所提高,但是待处理问题规模的急剧增大依然加剧了这些机器学习加速器的低效程度。针对这种现象,本文研究了 4 种流行的机器学习算法——k-近邻算法(k-NN),k-均值算法(k-Means),支持向量机(SVM)和逻辑回归(LR),并对这些算法中最为耗时的运算部分进行了深入分析,此外,还针对数据位宽对运算精度、硬件开销的影响进行了分析。根据以上分析,本文设计了一款可以支持多种机器学习算法的运算单元,该运算单元混合使用 16 位浮点数和 32 位浮点数的运算器,实现了低面积、低功耗的需求。实验结果表明,本文提出的运算单元可以在几乎不损失正确率的情况下,减少 69.80% 的总面积开销以及 68.98% 的总功耗开销。

关键词 机器学习(ML),运算单元,加速器,低面积,低功耗

0 引 言

当前的时代是一个数据爆炸的时代,互联网上的数据量一直呈爆炸式的增长。据 IDC(国际数据公司)估计,2011 年生成和复制的信息量超过了 1.8 ZB(1.6 万亿 GB),其中 75% 来自个人,包括图片、视频和音乐等,这个信息量远远高于过去世界印刷品的信息量的总和(200PB)[1]。随着数据规模越来越大,机器学习算法由于其优异的处理数据的能力得到了越来越广泛的应用,但是也正因数据量的急剧增大,导致运算量急剧扩增,使得机器学习算法的运算能力和运算速度面临着巨大的挑战。为了加快机器学习技术的运算速度,许多专用于加速某种机器学习算法的加速器应运而生[2-4]。

在机器学习加速器的研究中,运算单元的设计占有举足轻重的地位。这是因为,一方面,随着片上数据的不断增加,尽管芯片制造工艺水平有所提高,但是,对机器学习加速器的设计而言,面积开销和功耗开销依旧是一个不得不考虑的问题,而这又和运算单元的设计紧密相关;另一方面,种类繁多的机器学习算法具有多种多样的运算特点和访存特点,因此,多数的机器学习加速器往往只针对某一种特定的机器学习算法来进行设计和加速。

目前的机器学习加速器有 3 个主要限制。第一,大多数机器学习加速器是单一算法导向的,即这

① 国家重点研发计划(2017YFA0700900, 2017YFA0700902,2017YFA0700901,2017YFB1003101),国家自然科学基金(61472396, 61432016, 61473275, 61522211, 61532016, 61521092, 61502446, 61672491, 61602441, 61602446, 61732002, 61702478,61732020),北京市自然科学基金(JQ18013),973 计划(2015CB358800),“核心电子器件、高端通用芯片及基础软件产品”科技重大专项(2018ZX01031102),中国科学院科技成果转移转化重点专项(KFJ-HGZX-013)和中国科学院战略性先导科技专项(B 类)(XDBS01050200)资助项目。

② 女,1991 年生,博士生;研究方向:机器学习加速器,神经网络加速器;E-mail:zhousy@ict.ac.cn

③ 通信作者,E-mail:cyj@ict.ac.cn

(收稿日期:2018-03-21)

些加速器无法适用于其他多种不同的算法和应用。第二,目前的并行机制是基于指令级的并行机制,这是由于受限于 CPU 和 GPU 的通用架构的影响。以 k-近邻算法的加速器[5]为例,它根据算法的 3 个步骤:距离计算,距离排序和分类选择,直接将加速器设计为 3 个部分,3 个部分间并行执行,每个部分用于完成一个步骤的运算。第三,对数据格式的关注较少。大多数机器学习加速器直接使用单一的 64 位浮点数或 32 位浮点数,忽略了数据位宽的影响[6,7]。

本文提出了一个面积小、功耗低的运算单元,其具有以下几个特点。

- 该运算单元面向多种机器学习算法,包括 k-近邻算法,k-均值算法,支持向量机和逻辑回归等,能够适用于多种不同的应用场景。
- 对多种机器学习算法的运算算子进行了分析,提取其运算的共性特征,细化了运算单元的并行运算的粒度,即使用 3 级流水线的方式,包括加法器阶段、乘法器阶段和加法树阶段,从而进一步提高了运算效率。
- 对数据位宽进行了分析,放弃了在一个运算单元中采用单一数据格式的设计,而是使用 16 位浮点数和 32 位浮点数混合的方式。通过实验来测试运算单元,并验证该方法可以大大减少运算单元的面积和功耗开销。

论文的结构安排如下。第 1 节对多种机器学习算法进行了简要分析,包括 k-近邻算法,k-均值算法,支持向量机和逻辑回归。第 2 节分析了算法运算的共性特征、数据位宽对运算和硬件的影响,从而提出了机器学习加速器运算单元的结构。第 3 节通过实验对该运算单元的正确度和硬件特性进行了检验和分析。最后,第 4 节总结了本文的主要贡献。

1 算法介绍

由于应用范围的不同,机器学习算法具有多样性以满足处理数据和任务的不同需求。本文选择了 4 种常用的机器学习算法,包括 k-近邻算法、k-均值算法、支持向量机和逻辑回归等算法。本节将对这 4 种算法进行简要分析,并指出每种算法中最为耗时的运算步骤。

K-近邻算法(k-NN),最常用的分类算法之一。测试样例会根据距离最近的 k 个参考样例的类别进行分类。显然,计算距离是最为耗时的步骤,可占到整体运算时间的 84.44%[4]。其中,最常用的距离为欧式距离,如式(1)所示:

$$dist(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{1}$$

其中,n 为维度。

K-均值算法(k-Means),最常见的无监督聚类算法之一。该算法的运算过程是不断迭代执行 2 个步骤,即将给定样例归为距离最小的聚类中心和更新聚类中心的位置。显然,k-均值算法与 k-近邻算法在计算上有一些相似之处,即其最耗时的操作都是计算 2 个样例之间的欧式距离[4]。

支持向量机(support vector machine, SVM)是一种常用的有监督机器学习算法。其基本思路是通过一个合适的核函数来将给定样例映射到一个新空间中,在该空间对给定样例进行分类。其中,核函数具有多种选择,如多项式函数(如式(2)所示),高斯径向基函数(如式(3)所示),双曲正切函数(如式(4)所示)等。

$$k(x_i, x_j) = (x_i \cdot x_j)^d \tag{2}$$

$$k(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2) \tag{3}$$

$$k(x_i, x_j) = \tanh(\kappa x_i \cdot x_j + c) \tag{4}$$

其中,在整个运算过程中,最为耗时的是两两样例之间根据核函数计算得到一个核矩阵。尽管核函数有多种,但是可以发现,上述几种最为常见的核函数中,其主要运算是类点积运算(如式(2)、(4)所示)和类距离运算的运算(如式(3)所示)。

逻辑回归(logistic regression, LR),一种常见的二类分类问题。其基本思想是构建一个线性回归模型来预测测试样例属于某一类的概率值,即如式(5)所示:

$$f(i) = \beta \cdot x_i \tag{5}$$

其中,β 是回归系数的向量,x 是测试样例。显然,该过程也类似于点积运算。

2 结构设计

大多数专用加速器都是针对单一算法设计的,

而且多为指令级并行，不会对运算算子进行更细粒度的分析，也更不会对数据位宽对面积和功耗的影响进行进一步研究和讨论。本节将对多种算法的运算特性进行细粒度的研究，并考虑数据位宽对运算单元、面积和功耗等的影响，从而设计出一个支持多种机器学习算法的面积小、功耗低的运算单元，并对其架构进行详细描述。

2.1　运算共性分析

由上一节分析可得，最为耗时的操作距离计算（k-近邻算法、k-均值算法、支持向量机的高斯径向基函数等）和点积运算（支持向量机的多项式函数、双曲正切函数、逻辑回归等），其伪代码分别如图 1、2 所示。显然，当对其包含的运算算子进行进一步分析，就会发现距离计算和点积运算的运算算子是相似的，即主要由加法（包括减法）和乘法运算组合完成。其中，距离计算的运算算子依次按以下运算顺序执行：对应元素相加，对应元素相乘和加和。而点积运算的运算算子按以下顺序执行：对应元素相乘和加和。因此，本文利用这个特点，对这 2 个最耗时的步骤进行加速。

```
dist = 0;
for (int i = 0; i < n; i++){
  tmp = x[i] - y[i];
  dist += tmp * tmp;
}
```

图 1　距离计算的伪代码

```
sum = 0;
for (int i = 0; i < n; i++)
  sum += x[i] * y[i];
```

图 2　点积运算的伪代码

2.2　数据位宽分析

常见的加速器均是采用单一的 32 位浮点数或者 64 位浮点数进行设计和运算。为了进一步节省运算单元的面积开销和功耗开销，本节对采用不同数据位宽来实现运算单元进行了研究，并分别从不同数据位宽对运算精度的影响和对硬件开销的影响两方面进行分析。

（1）运算精度

本文选取 UCI 机器学习知识库[8] 中的 4 个数据集（包括 wine，glass，ionosphere 和 iris）作为实验样例，将每 2 个数据样例划分为一组，利用不同的数据位宽进行数据表示，并进行距离计算和点积运算，以尽可能多地减少运算器的位宽，并评估它们对精度的影响。

选用 64 位浮点数格式（即 IEEE 754 标准双精度格式）和 32 位浮点数格式（即 IEEE 754 标准单精度格式）进行数据运算，以 64 位运算结果为标准，无论是否预先对数据进行归一化处理，二者运算的平均的均方误差均小于 0.01，即二者的运算精度差距很小。所以，本文的运算单元无需考虑采用 64 位浮点数格式实现。

本文以 32 位浮点数格式（即 IEEE 754 标准单精度格式）的运算结果为标准，讨论采用 16 位浮点数格式（即 IEEE 754 标准半精度格式）时对运算结果的影响。首先对数据进行归一化，而后测得 16 位浮点数的运算结果相比于 32 位浮点数的均方误差如图 3 所示，距离计算和点积运算的误差约为0.010 和 0.011。可以说明，对于距离计算和点积运算而言，采用 16 位浮点数对归一化后的数据的运算结果的影响是很小的。

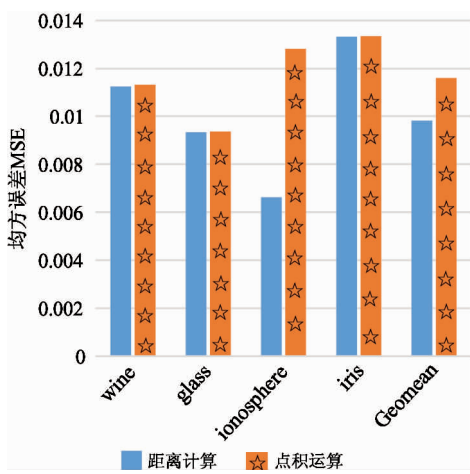

图 3　UCI 数据集的利用 16 位浮点数进行距离计算和点积运算相比于 32 位浮点数的均方误差（归一化）

然而，如果数据未预先进行归一化处理，当样例中的参数数值过大的时候，就很有可能会发生数据溢出的情况。如图 4 所示，由于 ionosphere 和 iris 数

据集中的参数数值较小,未归一化后的运算结果的均方误差很小,对于距离计算和点积运算分别平均为0.016 和0.014;对于 wine 和 glass 数据集,由于有某一项或某几项参数的数值过大,导致运算的误差较大(如 glass 的点积运算),甚至部分结果发生溢出(如 wine 的距离计算和点积运算)。进一步地,对产生溢出的运算进行测试,发现数据溢出主要是发生在数据加和部分,这是因为前面的一个加法或者乘法只是一个维度的单独的数据运算,而最后的加和是很多项的结果的累积,所以对误差也更为敏感,而且对数据的表示范围需求更高。因此,对于维度较多或参数数值较大的数据样例,全部采用 16 位浮点数完成距离计算和点积运算是存在溢出风险的,而且溢出主要发生在加和部分。

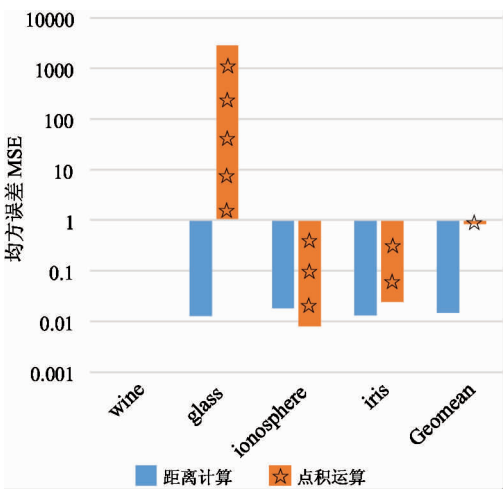

图 4　UCI 数据集的利用 16 位浮点数进行距离计算和点积运算相比于 32 位浮点数的均方误差(未归一化)

当浮点数位数进一步缩小的时候,即采用 8 位浮点数进行运算。本文分别测试了 1 位符号位、2 位指数位和 5 位尾数位的格式,以及 1 位符号位 、3 位指数位和 4 位尾数位的格式,无论数据是否进行归一化处理,都在运算过程中出现了数据溢出的现象,这是由于数据位宽过小,限制了数据表示的范围和精度。所以,这里不考虑采用 8 位浮点数格式来实现运算单元。

(2) 硬件开销

本文采用 Synopsys Design Compiler 工具综合了台积电公司(Taiwan Semiconductor Manufacturing Company, TSMC)的 65 nm 标准库中的浮点数乘法器和加/减法器,测得的面积开销和功耗开销如表 1 所示。其中,16 位浮点数的乘法器和加/减法器相比于 32 位浮点数的分别节省约为 2.20 倍和 3.49 倍的面积开销,而功耗开销也仅为采用 32 位浮点数时的 52.79% 和 24.24% 。显然,如果运算器采用 16 位浮点数,可以大大减少面积和功耗。同时,由于 16 位浮点数和 32 位浮点数相比减少了一半的数据位宽,也可以进一步降低存储器接口的负担和数据传输所消耗的功耗。

表 1　运算器特性

运算器类型	面积(μm^2)	功耗(mW)
16 位浮点数加/减法器	927.72	0.2501
32 位浮点数加/减法器	2044.80	0.4738
16 位浮点数乘法器	1312.20	0.3825
32 位浮点数乘法器	4576.68	1.5777

(3) 运算单元结构设计

本文的运算单元的设计针对其运算共性特点,分为由加法器和乘法器组成的前两个阶段,并利用加法树和累加器来加速后续的加和和累加运算。3 个阶段可以采用流水线的方式同时进行运算。另外,本文在流水过程中采用了不同的浮点数格式。如图 5 所示,用 M 和 N 表示运算器位数。前 2 个阶

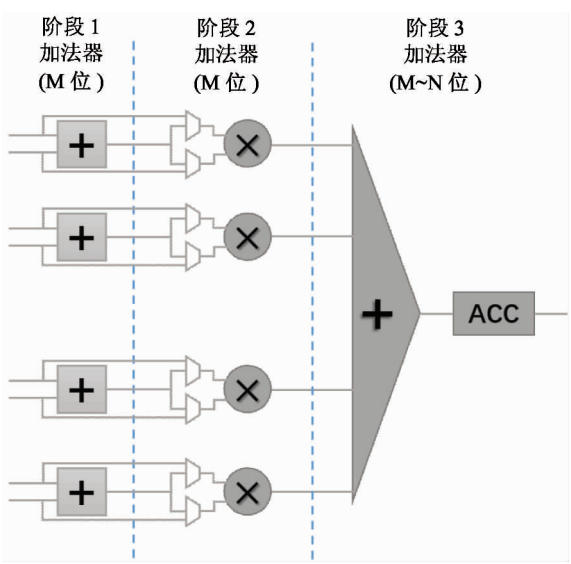

图 5　运算单元的结构

段的运算器均是 M 位浮点数,而第 3 阶段的前面的加法树部分的运算器采用 M 位浮点数,最后一级的运算器,即累加器,使用 N 位浮点数。在本设计中,M 为 16,N 为 32。

第 1 阶段由 2^n(n 为整数)个加法器组成,每个加法器待运算浮点数的位宽是 M。加法器可以并行计算。在此阶段,将输入和输出数据格式优化为 16 位浮点数(M = 16)。与使用 32 位浮点数(M = 32)相比,该方案减少了近一半的面积和功耗。

第 2 阶段由乘法器组成,如图 5 所示,其数量与第 1 阶段的加法器的数量相同,每个乘法器的位宽都是 M 位浮点。同样的,乘法器可以并行运算,且输入和输出数据的数据格式使用 16 位浮点数(M = 16)。在计算距离时,每个乘法器的两个待运算数据为同一个数据,即前一阶段加法器的输出,用于计算该数据的平方值。在计算点积时,输入数据是两个样例中对应的特征值。

运算单元的第 3 阶段是一个加法树和一个累加器,如图 5 所示。它用于将乘法器的输出值进行加和操作。该阶段由不同位数的运算器组成,即加法树中的各级加法器均使用 16 位浮点数(M = 16)进行运算,最后一级累加器使用 32 位浮点数(N = 32)。

在运算单元中使用不同的数据格式有两个原因。一方面,基于前述分析,采用 16 位浮点数可以大大降低运算器的面积和功耗开销,并同时减轻存储接口的负担。另一方面,在累加阶段可能存在数据溢出的风险,特别是当输入样例是高维向量或参数数值较大的时候。而 32 位浮点数的数据表示的上限为 $(2-2^{-23})\times 2^{127}\approx 3.4402823\times 10^{38}$,16 位浮点数的数据表示的上限为 $(2-2^{-10})\times 2^{15}=65\ 504$,即 32 位浮点数大大扩大了数据的可表示范围。所以,这里在加法树阶段依然采用 16 位浮点数以减少面积和功耗开销,而在最后的累加部分采用 32 位浮点数,以避免运算过程中发生数据溢出的情况。

其中,如果无需第 1 阶段的加法或减法操作,数据可以跳过该阶段而直接进行相乘。譬如,在点积运算的时候,只需要先得到相应维度的乘积,而后对乘积进行加和,如图 2 所示。所以数据可以跳过第 1 阶段的加法器,而直接送入第 2 阶段的乘法器中进行运算。

3 实验和结果

本节将展现有关实验的详细信息。首先,描述实验中所使用的测试集;而后从算法的角度,对比分析了使用不同位数浮点数对正确率的影响;最后对比了采用不同位宽的运算器对面积开销和功耗开销的影响。

3.1 测试集

本实验使用 MNIST 和 ImageNet 这两种广泛使用的规模较大的数据库作为机器学习算法的基准。

MNIST 是一种手写图像数据库,包含 6 万张训练图片和 1 万张测试图片,每张图片是一张尺寸为 28 像素 ×28 像素的黑白手写数字图[9]。

ImageNet 是根据 WordNet 层次结构组织的图像数据库。图像数量和图像种类数量远远大于任何其他图像数据库。本文选择的是 2012 年 ImageNet 大规模视觉识别挑战(ILSVRC2012)提供的数据集作为实验数据集,其中包含 120 万个训练样本,其中有 1 000种标签和 5 万个验证样本[10]。

3.2 正确率

本实验使用 ImageNet 和 MNIST 来评估使用不同位浮点数对整个算法运算的正确率的影响。这里,设置运算单元在第 1 阶段有 16 个加法器,在第 2 阶段有 16 个乘法器,在第 3 阶段的加法树阶段有 16 +8 +4 +2 +1 =31 个加法器,即每个始终周期可以完成 16 个样例特征(维度)的计算。

实验结果如表 2 所示。“16 位”表示从输入到输出均使用 16 位浮点数。即在图 5 中,M = N = 16。“16-32 位”表示混合位的方式,输入 16 位浮点数,并在第 1、2 阶段和第 3 阶段的加法树均采用 16 位浮点数进行运算,第 3 阶段的最后一级累加器采用 32 位浮点数进行运算,即 M = 16,N = 32。

实验结果如表 2 所示,使用 16 位浮点数进行运算时很容易产生数据溢出的情况。然而,使用混合位浮点数进行运算时,正确率损失很少,甚至某些算法的准确度反而会有所提高。例如,将逻辑回归算

表 2　使用不同位浮点数的正确率比较(归一化到 32 位)

算法	实验结果		
	测试集	16 位	16-32 位
k-NN	MNIST	99.9%	100%
	ImageNet	57.3%	100%
k-Means	MNIST	93.9%	100.1%
	ImageNet	100%	100%
SVM	MNIST	52.93%	100.16%
	ImageNet	65.79%	100%
LR	MNIST	100%	100.01%
	ImageNet	71.43%	105.71%

法应用于 ImageNet 数据集。采用 32 位浮点数进行运算时正确率为 70%,而使用混合位浮点数时,正确率为 74%,提高了 4%。然而,当采用所有 16 位浮点数进行运算时,正确率仅有 50%,也就是运算过程中发生了数据溢出。因为这是一个二类分类问题,且每类中样例数量相同,故理论正确率即为 50%。

3.3　面积和功耗开销

本文比较了运算单元的面积开销和功耗开销。其中,运算单元由 Synopsys Design Compiler 以标准 VT 中的台积电公司(Taiwan Semiconductor Manufacturing Company, TSMC)的 65nm GP 工艺进行综合,利用 Synopsys ICC Compiler 进行布局和布线工作,使用 Synopsys VCS 工具进行模拟和验证设计,基于模拟的 Value Change Dump(VCD)文件使用 Synopsys Prime-Time PX 来评估功耗。

面积/功耗开销的比较结果如图 6 和图 7 所示。每组数据包含 2 列,其中,左列是使用 32 位浮点数表示时运算单元的面积/功耗,即图 5 中的 M = N = 32 的情况;右列是使用混合位(16 位和 32 位)浮点数表示时的情况,即图 5 中的 M = 16 和 N = 32。右列的百分比表示的是面积/功耗的比例,即当运算单元使用混合位浮点数时的面积/功耗开销与使用 32 位浮点数的面积/功耗开销之比。

如图 6 中的第 3 组数据所示,当使用 32 位浮点数时,总面积消耗为 60606.38 μm^2,而当使用混合位浮点时,总面积消耗为 18303.12 μm^2,仅为前者的

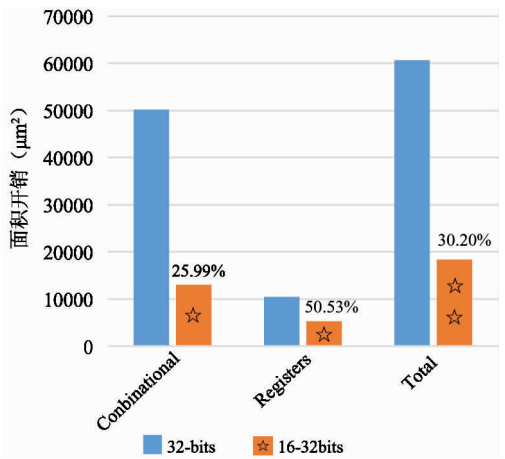

图 6　采用 32 位和混合位浮点数时面积开销的比较

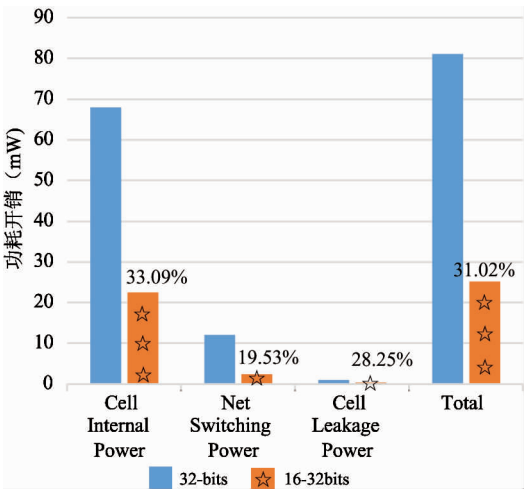

图 7　采用 32 位和混合位浮点数时功耗开销的比较

30.20%。类似地,图 7 中的第 4 组代表总功耗开销。本文可以看到右列的功耗开销仅为左列的 31.02%。面积开销和功耗开销的降低幅度高达近 70%,这主要是因为门和连接的复杂性都降低了。

从上面的实验结果可以看出,本文的运算单元设计在正确率上几乎没有损失,并且可以大幅度地节省面积开销和功耗开销,这对加速器设计来说是非常有意义的。

4　结　论

在这个数据暴增的时代,机器学习算法依然是数据挖掘、数据处理最为常用的算法之一,扮演着极为重要的角色。而面对着海量的数据,提高数据处理的速度是极为关键的,对于机器学习加速器设计

而言,其运算单元的设计是极为关键的。与此同时,运算单元的面积和功耗也是一个不容忽视的问题。针对上述问题,本研究选取了 4 种具有代表性的机器学习算法,包括 k-近邻算法,k-均值算法,支持向量机和逻辑回归等,并对这些算法的运算特点和最耗时的步骤进行了分析,提取出其运算共性;同时,对数据位宽对运算精度和硬件特性的影响进行了进一步的研究和讨论,从而提出了一款可用于机器学习加速器中的运算单元,该运算单元能够支持包括上述 4 种算法在内的多种机器学习算法。特别地,该运算单元使用了多种浮点数的运算器,即同时包括 16 位浮点数和 32 位浮点数的运算器,从而使得在几乎没有损失正确率的情况下,其面积开销和功耗开销分别仅为全部使用 32 位浮点数的运算单元的面积和功耗的 30.20% 和 31.02%,从而大幅度降低了运算单元的硬件开销,为进一步降低机器学习加速器的整体的功耗和面积提供了可能。

参考文献

[1] Gantz J, Reinsel D. Extracting value from chaos[J]. *IDC iView*, 2011, 1142(2011): 1-12

[2] Chen T, Du Z, Sun N, et al. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning[J]. *ACM Sigplan Notices*, 2014, 49(4): 269-284

[3] Chen Y, Luo T, Liu S, et al. Dadiannao: a machine-learning supercomputer[C]. In: Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, 2014. 609-622

[4] Liu D, Chen T, Liu S, et al. Pudiannao: a polyvalent machine learning accelerator[C]. In: Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems, Istanbul, Turkey, 2015. 369-381

[5] Manolakos E S, Stamoulias I. IP-cores design for the kNN classifier[C]. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, Paris France, 2010. 4133-4136

[6] Papadonikolakis M, Bouganis C S. A heterogeneous fpga architecture for support vector machine training[C]. In: Proceedings of the 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, Charlotte, USA, 2010. 211-214

[7] Majumdar A, Cadambi S, Becchi M, et al. A massively parallel, energy efficient programmable accelerator for learning and classification[J]. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2012, 9(1): 6

[8] Dua D, Taniskidou K E. UCI machine learning repository [EB/OL]. http://archive. ics. uci. edu/ml. Irvine: School of Information and Computer Science University of California, 2017

[9] LeCun Y, Cortes C, Burges C J C. The MNIST database of handwritten digits [EB/OL]. http://yann. lecun. com/exdb/mnist/: LeCun, 1998

[10] Deng J, Dong W, Socher R, et al. Imagenet: a large-scale hierarchical image database[C]. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, USA, 2009. 248-255

An area and power-efficient machine learning functional unit

Zhou Shengyuan[*,**,***], Du Zidong[*], Liu Daofu[*,***], Zhi Tian[*], Chen Yunji[*,**]

([*]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]University of Chinese Academy of Sciences, Beijing 100049)

([***]Cambricon Tech. Ltd, Shanghai 201203)

Abstract

With the increasing popularity of machine learning (ML) techniques, many dedicated ML accelerators have been proposed. However, such accelerators are still limited by their narrow scope for their specified purposes. Moreover, despite restricted improvements from silicon technology, the expanding scale of problems exacerbates the inefficiency of proposed ML accelerators. In this paper, we thoroughly analyze the most time-consuming parts of four popular ML algorithms, i. e., k-nearest neighbors (k-NN), k-Means, support vector machine (SVM) and logistic regression (LR). In addition, in order to achieve higher area and power efficiency, we further study the effect of data-width on accuracy and hardware overheads. Based on the analysis, we propose a functional unit accommodating various ML algorithms, which consists of mixed-bit floating point operators——including both 16-bit floating point operators and 32-bit floating point operators. The results of the experiments show that the proposed functional unit can reduce 69.80% of total area consumption and 68.98% of total power consumption with little accuracy loss.

Key words: machine learning (ML), functional unit, accelerator, area-efficient, power-efficient