doi:10.3772/j.issn.1002-0470.2021.05.001

面向多任务处理的神经网络加速器设计①

许浩博②*** 王 颖* 王郁杰* 闵 丰*** 韩银和③*

(*中国科学院计算技术研究所 北京 100190)

(**中国科学院大学 北京 100049)

摘 要 本文从多任务系统以及算法硬件协同设计的角度出发,提出了一种面向多任务处理的神经网络加速器架构。在算法方面,提出了一套多任务处理算法,能够有效挖掘多任务间的特征重复性和计算重用性;在架构设计方面,设计了能够利用网络参数共享实现多任务间计算重用的加速器架构。该架构采用基于时间片的调度方法,实现了多视觉任务的高效并行处理。实验结果表明,与目前主流的移动图形处理器(GPU)相比,本文提出的加速器能够在多任务推理阶段减少65.80%的乘法运算量,同时获得了11倍的性能提升。

关键词 神经网络;多任务;加速器;参数共享;计算复用

0 引 言

在错综复杂、瞬息万变的终端场景中,常常要求嵌入式设备和终端设备能够处理多种基于深度学习的感知任务,例如对于移动机器人来说,需要实时处理场景感知、行人检测、交通标志识别等多种视觉任务[1-2],并通过这些基础任务实现环境观察、实时定位和人机交互等多种应用。但是,嵌入式设备和终端设备往往受到计算资源和功耗的限制,难以保证在低延时下高效处理多个深度学习任务。

专用神经网络加速器是处理深度学习算法的有效方法,然而大多数用于嵌入式设备或终端设备的神经网络加速器是单任务加速器[3-7],在这样资源受限的设备中处理多个视觉任务时,通常采用以下两种任务部署方案。(1)多任务串行执行。加速器以时分复用的方式将各个任务部署在设备上,各个任务之间没有协同调度策略,执行完所有任务的总时间至少是各个任务执行时间的总和,这对于执行队列后部的任务来说,处理延时很大。(2)多任务并行执行。加速器为各个并发任务分配计算资源和存储资源,各个任务进程利用所分配的资源处理相应任务。在运算资源丰富的设计中[8],并行执行多个任务可以充分利用计算资源,提高多任务执行效率,然而对于资源受限的硬件,由于各个任务进程往往无法获得足够的计算资源,资源的均摊导致执行时间延长,这对于延迟敏感的实时性任务来说是灾难性的。

为了解决这一问题,本文采用算法与硬件协同设计的方法设计了一款面向多任务处理的神经网络加速器。首先,提出了一套通过共享神经网络参数处理多个视觉任务的算法,该算法能够充分利用多任务间的重复特征和计算重用,建立了多任务参数共享和特征重用机制。接着,在上述算法基础上设计一款面向多任务处理的神经网络加速器,该加速器充分考虑了多任务之间的负载分时复用,并从多任务系统的角度,设计了基于时间片的调度机制,有

① 国家重点研发计划(2017YFB1301100),北京市自然科学基金(4194092),国家自然科学基金(61834006, 61874124),北京市科技计划(Z171100000117019,Z181100008918006)和中国科学院战略重点研究计划(XDPB12)资助项目。

② 男,1990年生,博士;研究方向:计算机体系结构,神经网络加速器设计;E-mail: xuhaobo@ict.ac.cn

③ 通信作者,E-mail: yinhes@ict.ac.cn

(收稿日期:2020-04-03)

效减少了多任务间重复且冗余的数据调度，保证了加速器能够在时间约束和资源约束下高效顺畅完成多任务处理。实验结果表明，本文提出的加速器与目前主流移动图形处理器（graphics processing unit，GPU）和神经网络处理器相比性能平均提升 11 倍。

1 相关工作

在深度学习技术的实际应用中，将预训练模型应用于某一新任务或数据集时通常采用“微调”（fine-tune，FT）模型参数的方法，但是采用“微调”的方法实现模型的多任务扩展会导致“灾难性遗忘”的现象[9-10]，即随着新任务的学习，原有任务的处理精度会显著下降。为保证在不显著增加参数量的条件下，各个任务均能取得较高处理精度，Mallya 等人[11]提出了一种利用二值化掩码（mask）在多任务间共享模型参数的方法，使单个神经网络模型具有处理多个视觉任务的能力。在该方法中，除了共享的权重参数外，每个特定任务还包含一组由数值 0 和 1 组成的二值化掩码矩阵 $\boldsymbol{m}_b$，其表达形式为

$$\boldsymbol{m}_b = Binarize(\boldsymbol{m}) = \begin{cases} 1 & \boldsymbol{m}_\tau \geqslant \tau \\ 0 & \text{其他} \end{cases} \tag{1}$$

其中 $\boldsymbol{m}$ 表示由实数组成的原始掩码矩阵，$Binarize(\boldsymbol{m})$ 表示二值化函数，τ 表示阈值。二值化掩码以逐元素相乘（element-wise multiplication）的形式作用在原始权重上，因此神经网络中的点积乘法运算可以表示为

$$\boldsymbol{y} = (w \odot \boldsymbol{m}_b) \cdot \boldsymbol{x} \tag{2}$$

其中 w 代表权重，$\boldsymbol{x}$ 代表输入特征图，$\boldsymbol{y}$ 是乘积结果，$\odot$ 代表逐元素乘法。在权重更新阶段，通过 $\boldsymbol{m}_b$ 的梯度来更新由实数组成的权重矩阵 $\boldsymbol{m}$，训练结束后矩阵 $\boldsymbol{m}$ 不再参与运算，而采用二值掩码矩阵 $\boldsymbol{m}_b$ 代替。基于这种方式，通过预训练得到原始模型的权重参数后，只需为特定的任务训练得到二值掩码矩阵 $\boldsymbol{m}_b$，即可将同一个网络模型扩展至不同任务中。由于不同任务间参数模型的差异仅为二值掩码矩阵 $\boldsymbol{m}_b$，因此权重能够在不同任务间共享，大幅减少了多个任务模型的参数总量。更为重要的是，各个任务相对独立，学习新任务时不会导致网络模型“忘记”先前获得的能力。

在这项通过掩码实现权重参数共享的技术启发下，本文进一步挖掘了多个视觉任务之间的重复性特征，提出了一套面向多任务处理的软硬件协同设计方案，能够有效减少各个任务间的重复计算，大幅提高多任务处理效率。

2 多任务处理算法

2.1 多任务间提取重复特征

在同一时间段，不同任务的输入特征图具有很强的相似性，为进一步强化这种相似性，本文提出了一种提取多任务间的重复特征的方法。首先，利用第 1 节描述的权重共享方法，完成了训练二值化掩码实现多任务扩展的技术，在此基础之上，进一步对激活值进行低比特量化，增强任务之间特征的相似性。本文采用线性分段量化方法，量化函数为

$$Quant(x_q, n) = x \quad x \in (p_t, p_{t+1}] \tag{3}$$

在式（3）中，n 定义为量化因子，代表将激活值量化为 n 比特位；$(p_t, p_{t+1}]$ 是量化区间，代表每一个比特能够表示的数值范围；量化函数 $Quant(x_q, n)$ 将激活值映射到相应的量化区间，每个量化区间均由固定的二进制数表示。与文献[12]一致，本文在实现过程中将网络模型内除第一层和最后一层外其余层量化为低比特，量化因子为 3。采用这样的方法，原本多变的激活值被限定采用特定的低比特数表示，激活值的特征大幅减少，于是不同任务之间的重复特征能够被充分挖掘，为不同任务间的计算复用和数据复用提供了前提条件。

2.2 多任务处理算法整体描述

传统方法在实现多任务处理时，往往采用“微调”模型参数的方式得到一组全新的权重参数值，任务越多计算装置需要存储的权重参数就越多。本文基于权重共享和重复特征提取的方法，提出一套致力于挖掘多任务间权重重用和乘积重用的多任务处理算法。算法 1 详细描述了本文提出的多任务处理算法在训练阶段的流程。首先针对任务 1 对原始模型开展预训练；其次采用上文提到的低比特量化激活值的方法重训练原始模型，得到任务 1 最终的

模型参数;接着针对每一个特定任务,将掩码与任务1的权重参数逐元素相乘,并在训练过程中不断学习更新掩码数值,同时通过多次迭代将激活值逐渐量化至需要的低比特。

在算法1中,通过施加掩码,即使不更新预训练模型的权重,也能够获得大量不同的权重滤波器,因此在原始模型参数基础上仅增加单比特表示的二值化掩码就能够实现多任务扩展。与为每一个特定任务均"微调"出一组全新参数的方法相比,本文提出的算法大幅减少了处理多任务的参数总量,并且权重能在多任务间重用;更为重要的是,通过提取不同任务间网络特征图中的重复特征,卷积运算中的乘法计算结果能够在多个任务间重用,大幅减少了多任务间冗余的重复计算。图1举例说明了通过提取重复特征消除冗余计算的机制,此例采用3×3卷积核对4×4输入特征图进行卷积,处理的任务数为3,量化因子为2,即激活值仅有4个不同的值。在这一例子中,多个任务间有效的乘法运算只涉及预训练得到的权重值和4个固定的激活值,因此在网络模型推理阶段能够通过重用"有效乘积"大幅降低运算量。

算法1 本文提出的多任务处理算法

输入:X_i:训练集中每一个特定的任务,$i \in [1, 2, \cdots, N]$;W_i:初始化的权重;量化函数 $Quant(x, n)$;掩码二值化函数 $Binarize(\boldsymbol{m})$。

输出:W:多任务共享的权重;M_i:每一个任务的二值化掩码。

步骤

1:预训练任务1的网络模型;
2:对步骤1得到网络模型引入 $Quant(x, n)$ 重训练;
3:for $i \in [2, 3, \cdots, N]$ do
4:　　将运算 $\{w \cdot x\}$ 中的乘法用式(2)代替;
5:　　通过 $Quant(x, n)$ 将激活值量化为 n 比特;
6:　　计算 Loss 损失并更新二值化掩码;
7:end for

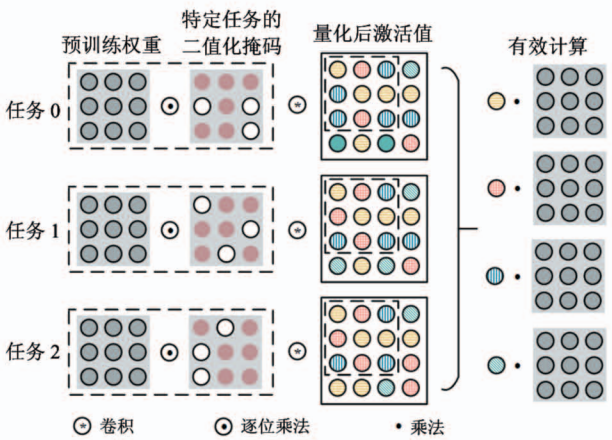


图1 多任务处理算法示意图

3 架构设计

本节将详细介绍如何基于所提出的多任务处理算法设计一款面向多任务处理的神经网络加速器,实现多任务处理的软硬件协同设计。

3.1 整体结构

本文提出的加速器架构如图2所示,加速器由片上存储器、计算引擎(computing engine,CE)和控制逻辑等部分组成。片上存储器包括存储预训练权重参数的权重缓存(weight buffer)、存储二值化掩码的掩码缓存(mask buffer)、存储激活值以及中间计算结果的激活值缓存(activation buffer)。计算部件包括乘法单元(M unit)、查找表引擎(look-up engine,LE)阵列和加法树(adder tree,AT)。乘法单元完成权重和量化后激活值的乘法运算,计算结果存储在查找表引擎中,并且完成模型第一层以及最后一层中必要的乘法操作;在多任务的前向推理过程中,加速器能够通过查找表引擎提供可重复使用的乘积结果,而不需要激活乘法单元。权重和激活值的乘积在加法树中累加,生成下一层的激活值或中

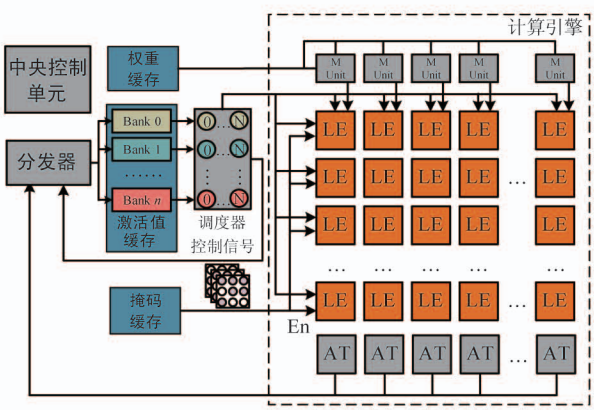


图2 加速器整体架构

间计算结果。控制逻辑包括中央控制单元(central control unit)、分配器(dispatcher)和调度器(scheduler)。中央控制器和调度器配合完成全局的数据调度,分配器保证计算引擎生成的激活值或中间结果能够正确存储在相应的缓存中。激活值从查找表引擎阵列的最左列载入,能够从左向右流动,实现输入激活值复用;不同输出特征图的卷积核载入至计算引擎的不同列中,能够实现输出特征图多通道并行计算。

3.2　计算重用机制

本文提出的加速器能够利用第 2 节中的多任务处理算法实现权重重用和乘积重用,下面具体描述两种重用机制。

权重重用　根据第 2 节提出的算法,每个特定任务均通过训练得到了不同的二值化掩码{0,1},在任务执行过程中掩码与权重值进行逐元素相乘,根据这样的计算特征,加速器设计实现了任务间和任务内两个层次的权重重用。对于任务间权重重用,加速器将预训练得到权重参数应用于多个任务,无需反复载入权重值,对于不同的任务仅需载入特定的二值化掩码,并且仅当掩码数值为 1 时,激活查找表引擎,同时通过查表得到相应乘积,当掩码为 0 时,不激活查找表引擎,直接将数值 0 输出至加法树,降低查找表引擎的动态功耗,如图 3 所示;对于任务内权重重用,本文采用权重复用数据流,每个权重滤波器能够在输入特征图中滑动重复使用多次,同时不破坏任务间权重的数据局部性,减少了权重值在计算单元和存储器之间的数据搬运。

乘积重用　多任务处理算法能够将激活值量化为低比特位宽,仅采用特定的几种数值形式就能够表达所有激活值。基于这样的机制,权重和激活值的乘法运算能够分为两个阶段。首先,构造查找表,一组权重与具有特定表达形式的低比特激活值相乘,乘积暂存在查找表引擎中;接着,在计算阶段,采用查表的方式代替乘法运算,输入激活值动态载入查找表单元中,激活值作为索引从查找表获得对应的乘积结果,从而无需再度激活乘法单元,如图 3 所示。由于权重重用保证了一组权重能够在任务间和任务内高效重用,因此查找表无需频繁更新,能够获得较高的乘积重用效果。

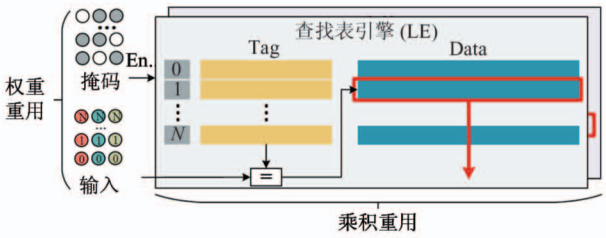


图 3　权重重用和乘积重用示意图

3.3　基于时间片的数据调度机制

为充分利用多任务间的相似特征并最大限度重用权重参数,本文从系统设计的角度提出了一种基于时间片的数据调度机制,能够在消除多任务间冗余操作的同时保证各个任务能够及时处理,图 4 展示了基于时间片的数据调度机制示意图。根据中央控制单元生成的地址,来自不同任务的输入特征图动态加载到 LE 中,由于各个任务能够共享同一组预训练得到的权重参数且网络模型相同,因此各个任务能够遵循统一的数据流。具体来说,首先采取循环切片(loop tiling)的策略,将输入网络层划分为多个子层;接着根据权重重用策略将计算单元的执行时间划分为多个时间组(time group),每个时间组处理一个网络子层,在每个时间组内,参与运算的激活值来自不同任务中相同像素位置,并与相同的权

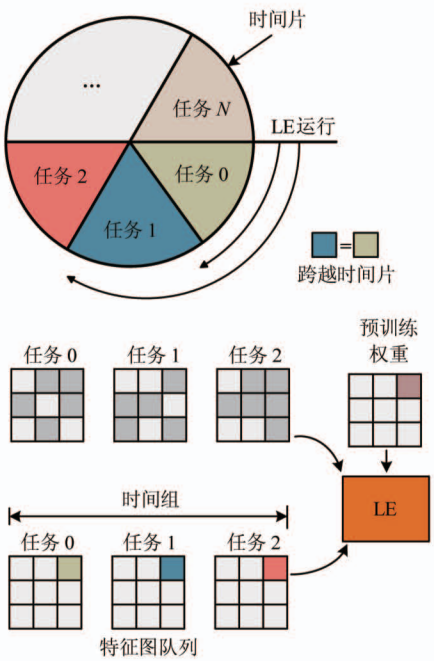


图 4　基于时间片的数据调度机制示意图

重值发生乘法运算;最后将每个时间组进一步划分为多个时间片,每个任务占用一个时间片,多个任务构成一个执行队列,计算单元在多个任务间不断切换执行。由于不同任务的数据调度方式是一致的,中央控制器只需执行一次全局控制指令,计算单元在任务间的切换由调度器完成,因此从系统角度看多个任务的处理是"同时"进行的。这种基于时间片的数据调度方式消除了多个任务的重复调度,并且能够充分利用权重重用和乘积重用,提升了多任务计算效率。

激活值通过低比特量化后能够由特定的比特数表示,这就增加了各个任务间同一像素位置特征相同的概率。基于这个计算特征,在上述数据调度机制基础上,本文设计"跨越时间片"(through time-slice)策略,当任务 N 的特征与执行队列中前一个任务特征一致时,即任务所涉及的激活值数据相同时,计算引擎能够消除当前任务的操作,减少了任务的执行时间。相同特征的匹配也由调度器完成。

4 实验和结果

本节从性能、能耗以及运行效率等方面评估本文提出的面向多任务处理的神经网络加速器。

4.1 实验方法

利用 Verilog 语言在 RTL 级实现了本文提出的面向多任务处理的神经网络处理器,采用 Synopsys Design Compiler 综合工具在 65 nm 工艺下进行综合,使用 Synopsys VCS 仿真工具对所提出的加速器进行了仿真与验证,使用 Synopsys PrimeTime 对功耗进行了评估。片上静态存储器的容量为 354 kB,片外存储器功耗由 CACTI 模拟器评估。加速器包含 256 个查找表引擎,其中包含有 4 kB 的寄存器堆。电路工作在 500 MHz 时,平均功耗为 342.5 mW。

参考视觉十项全能挑战赛[13],本文以嵌入式终端设备中常见的视觉任务数据集作为测试基准,包括 ImageNet[1]、Places365[14]、German traffic signs (GTS)[15], Daimler pedestrian classification (DPC)[16] 和 UCF101[17]。Place365 是场景分类数据集,GTS 是交通标识数据集,DPC 是行人检测数据集,UCF101 是人类动作识别数据集。通过 ImageNet[1] 数据集对 VGG-16 和 Resnet-50 两个网络类型进行了预训练。各项任务分别在主流移动端 GPU NVIDIA TX1、深度神经网络加速器 Eyeriss[3] 和本文提出的面向多任务的神经网络加速器上做了前向推理性能评估。对于 Eyeriss,本文根据文献[3]开发了时钟周期粒度的模拟器,执行过程中采用串行执行策略执行多任务队列,并且为了公平比较处理性能,在评估过程中本文调整 Eyeriss 的工作频率与本文工作一致。

4.2 评估结果

性能评估 为了评估加速器的多任务处理能力,本文设计了两个测试场景。第一场景(场景Ⅰ)包括目标识别[1]、场景识别[14]和行人检测[16]3 个任务。第二场景(场景Ⅱ)在第一场景的基础上添加了交通标志分类[15]和人类行为识别[17]两个任务,共包括 5 个任务。这两个场景测试得到的归一化加速比如图 5 所示。由图 5 可知,本文提出的加速器在两个场景均比 TX1 和 Eyeriss 表现出更好的性能,加速器的前向推理速度与 TX1 和 Eyeriss 相比分别平均提高了 11 倍和 1.86 倍。进一步分析可以发现,随着任务数量的增加,虽然片外访问带宽和片外访问时延有所增加,但是本文提出的加速器性能并不会下降,这是由于加速器能够利用任务间的重复特征补偿片外带宽带来的性能损失。同时评估了单任务处理性能,本文方法与 TX1 和 Eyeriss 相比分别平均提高了 8.8 倍和 1.51 倍。

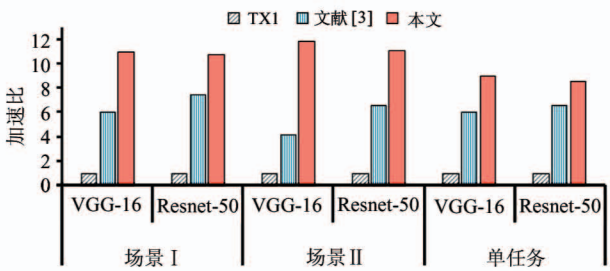
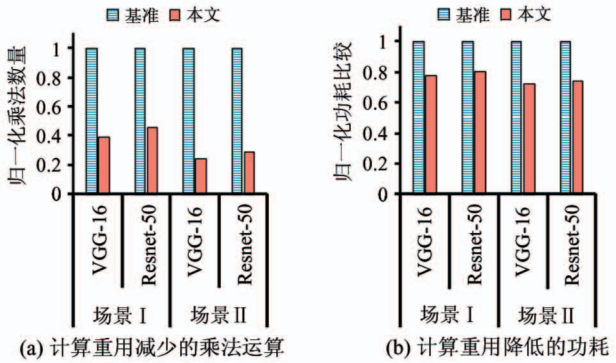


图 5 加速比评估

计算重用评估 与常规的网络前向推理方法相比,本文提出的加速器由于具有乘积重用机制,能够平均减少 65.80% 的乘法运算。此外,采用查找运算代替乘法运算,能够有效降低计算单元的动态功耗。评估显示,乘法运算的动态功耗可以降低88.74%,

加速器的整体功耗平均降低 23.94%（见图 6）。随着任务数量的增加，加速器能够在更大范围利用乘积重用，但是片上存储资源是有限的，无法无限增加并行任务的数量。



(a) 计算重用减少的乘法运算　(b) 计算重用降低的功耗

图 6　计算重用评估

能耗评估　在能耗评估中，本文考虑了访问片外存储器所需要的能量。与传统加速器相比，本文设计的加速器增加了查找表引擎、调度器和分配器等部件来实现计算复用和基于时间片的数据调度机制，因此有额外的硬件面积开销和功耗开销，但是本文提出的加速器能够从以下方面弥补这些损耗。首先，本文提出的加速器与对比工作相比具有更短的执行时间，能够从时间维度减少能量损耗；其次，由于消除冗余计算的机制，计算单元的动态功耗大幅减少；最后，多任务间共享权重技术和重复特征提取技术大幅降低了权重和激活值的数据带宽，能够明显降低片外存储器的访问能耗。由于 TX1 的能耗包括中央处理器（central processing unit，CPU）等其他计算部件能耗，为公平横向比较，本文在能耗评估中仅与神经网络加速器 Eyeriss[3] 比较，结果如图 7 所示。运行相同的任务，本文提出的加速器与神经网络加速器 Eyeriss[3] 相比，平均能耗减少了 75.54%。

精度评估　图 8 展示了本文提出的多任务处理算法和“微调”方法的模型准确性对比结果。由于二值化掩码和重复特征提取技术的使用，与“微调”方法相比本文提出的方法存在 3.6% ~9% 的精度损失，但是整体接近由“微调”方式得到的最佳结果。尽管存在一定程度的精度损失，但是本文提出的方法能够在仅增加部分参数量的情况下使原有模型更具通用性，能够有效解决在计算资源和存储资源受限的嵌入式设备和移动设备中高效处理多任务的问题。

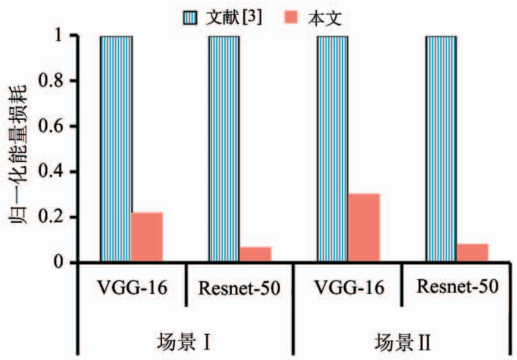


图 7　能量损耗评估

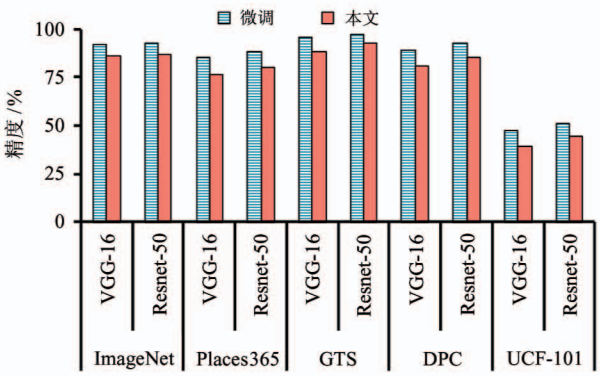


图 8　精度评估

5　结　论

本文提出了一款面向多任务处理的新型神经网络加速器，该加速器在挖掘多任务间重复特征的基础上实现了模型参数和重复特征的共享，并通过基于时间片的数据调度方法，充分利用了多任务间的权重重用和乘积重用，大幅减少了多任务间的重复运算。实验结果表明，与现有的多任务解决方案相比，本文提出的加速器在性能和能耗方面具有优势，为在计算资源和存储资源受限的嵌入式设备和移动设备中高效处理多任务提供了有效的解决方案。

参考文献

[1] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. *Advances in Neural Information Processing Systems*, 2012 (25): 1097-1105

[2] Jafari O H, Groth O, Kirillov A, et al. Analyzing modu-

lar CNN architectures for joint depth prediction and semantic segmentation[C]// International Conference on Robotics and Automation, Singapore, 2017: 4620-4627

[3] Chen Y H, Krishna T, Emer J S, et al. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks[C]// IEEE International Solid State Circuits Conference, San Francisco, USA, 2016: 262-263

[4] Du Z D, Fasthuber R, Chen T S, et al. ShiDianNao: shifting vision processing closer to the sensor[C]// Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, USA, 2015: 92-104

[5] Yin S Y, Peng O Y, Tang S B, et al. A high energy efficient reconfigurable hybrid neural network processor for deep learning applications[J]. *IEEE Journal of Solid-State Circuits*, 2017, 53(4): 968-982

[6] Lee J, Kim C, Kang S, et al. UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision[C]// IEEE International Solid-State Circuits Conference, San Francisco, USA, 2018: 218-220

[7] Yuan Z, Yue J S, Yang H R, et al. Sticker: A 0.41-62.1 TOPS/W 8Bit neural network processor with multi-sparsity compatible convolution arrays and online tuning acceleration for fully connected layers[C]// IEEE Symposium on VLSI Circuits, Honolulu, USA, 2018: 33-34

[8] Shao Y S, Clemons J, Venkatesan R, et al. Simba: scaling deep-learning inference with multi-chip-module-based architecture[C]// International Symposium on Microarchitecture, Columbus, USA, 2019: 14-27

[9] Li Z, Hoiem D. Learning without forgetting[C]// European Conference on Computer Vision, Munich, Germany, 2018: 614-629

[10] Rebuffi S A, Vedaldi A, Bilen H. Efficient parametrization of multi-domain deep neural networks[C]// IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 8119-8127

[11] Mallya A, Davis D, Lazebnik S, et al. Piggyback: adapting a single network to multiple tasks by learning to mask weights[C]// European Conference on Computer Vision, Munich, Germany, 2018: 72-88

[12] Zhou S, Ni Z, Zhou X, et al. DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients[J]. *arXiv*:606.06160, 2016

[13] Rebuffi S, Bilen H, Vedaldi A, et al. Learning multiple visual domains with residual adapters[C]// Neural Information Processing Systems, Long Beach, USA, 2017: 506-516

[14] Zhou B, Lapedriza A, Khosla A, et al. Places: A 10 million image database for scene recognition[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017, 40(6): 1452-1464

[15] Stallkamp J, Schlipsing M, Salmen J, et al. Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition[J]. *Neural networks: the Official Journal of the International Neural Network Society*, 2012, 32:323-332

[16] Munder S, Gavrila D M. An experimental study on pedestrian classification[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006, 28(11):1863-1868

[17] Bilen H, Fernando B, Gavves E, et al. Action recognition with dynamic image networks[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018, 40(12): 2799-2813

The neural network accelerator design for multi-task processing

Xu Haobo[*,**], Wang Ying[*], Wang Yujie[*], Min Feng[*,**], Han Yinhe[*]

([*]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

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

Abstract

An accelerator architecture from the perspective of multi-task systems and algorithm-and-hardware co-design is proposed. First, a simultaneous multi-task algorithm is proposed to exploit the repetitive feature and the computational reuse among the tasks. Then a compact accelerator that fully utilizes the proposed algorithm is designed to process various vision tasks simultaneously. A time slice-based scheduling method is further proposed to conduct the multi-vision task in parallel and the network parameter sharing is utilized to achieve significant computational reuse. It is evaluated in experiments that the proposed accelerator removes 65.80% of the repetitive operations from the multi-simultaneous inference tasks and achieves 11 × performance speedup over the state-of-the-art mobile graphics processing unit (GPU).

Key words: neural network, multi-task, accelerator, parameter sharing, computing reuse