高技术通讯 2022 年 第32 卷 第9 期:881-893

doi:10.3772/j.issn.1002-0470.2022.09.001

基于位串行计算的动态精度神经网络处理器①

郝一帆②*** 支 天③* 杜子东*

(*中国科学院计算技术研究所处理器芯片全国重点实验室 北京 100190)

(**中国科学院大学 北京 100049)

摘 要 针对当前神经网络动态精度计算系统在周期性的模型重训练和动态精度切换的过程中会引入大量的计算和访存开销问题,提出了基于串行位计算的动态精度神经网络处理器(DPNN),其可支持任意规模、任意精度的神经网络模型;支持以非重训练的方式对模型数据精度进行细粒度调整,并消除了动态精度切换时因权值 bit 位重叠造成的重复计算与访存。实验结果表明,相较于自感知神经网络系统(SaNNs)的最新进展之一 MinMaxNN,DPNN 可使计算量平均降低 1.34~2.52 倍,访存量降低 1.16~1.93 倍;相较于代表性的 bit 串行计算神经网络处理器 Stripes,DPNN 使性能提升 2.57 倍、功耗节省 2.87倍、面积减少 1.95 倍。

关键词 神经网络处理器;动态精度计算;位串行计算

0 引 言

近年来,人工神经网络作为最具代表性的深度学习模型,在图像识别、自然语言处理、游戏 AI 和智能驾驶等领域取得了瞩目的成就。因此,神经网络模型在硬件设备中的算法部署也引起了学术界和工业界的广泛关注。为了实现低功耗的目标,在实际应用中神经网络计算系统往往需要根据输入变化,实时动态切换或调整模型的数据精度,即动态精度计算。例如,在安防摄像头设备中部署的用于目标检测的神经网络模型,当画面中出现大量移动的人或物体时,设备往往需要调用高精度版本的模型以保证识别准确率;当画面中物体较少或静止时,只需要运行低精度版本的模型以节省能耗。由于上述策略能大幅节省设备在实际应用中的能耗,有利于神经网络算法应用在能耗受限的不同场景中实现广泛应用部署。因此,如何以更高效的硬件系统支撑神经网络模型的动态精度计算成为近年的研究热点。

针对动态精度计算场景,研究者们提出了自感知神经网络系统(self-aware neural network systems, SaNNs)[1],一种根据从执行环境中连续的感知信息作出动态反应的神经网络处理系统,其基本设计思路遵循以下两点原则。(1)高能效:系统对输入数据进行识别难易度的判别,动态调用不同精度的神经网络模型以节省能耗。(2)可靠性:系统定期对低精度模型进行正确性校验与精度调整,以保证其识别准确率不低于预设阈值,从而确保整个系统的可靠性。然而,现有的 SaNNs[2-5] 主要面临如下两个问题。(1)对模型精度的调整往往依赖于周期性的重训练,造成了额外的计算开销。(2)在每次切换模型版本时,都需要完全重新加载与重新计算新的模型,这造成了大量的数据访问与累积乘法(multiply and accumulate, MAC)操作数,进而损害了能效。

① 国家重点研发计划(2018AAA0103300),国家自然科学基金(62222214,U20A20227,U19B2019,U22A2028),北京智源人工智能研究院,中国科学院稳定支持基础研究领域青年团队计划(YSBR-029)和中国科学院青年创新促进会资助项目。

② 男,1996 年生,博士生;研究方向:计算机系统结构,深度学习算法;E-mail:haoyifan@ict.ac.cn。

③ 通信作者,E-mail:zhitian@ict.ac.cn。

(收稿日期:2021-05-10)

根据对动态精度计算过程的分析，本文发现一组相同结构、不同位宽的神经网络模型间存在大量由权值比特位重叠导致的重复数据访问与计算。基于此，本文提出了一种可高效支持动态精度计算场景的动态精度神经网络处理器（dynamic-precision neural-network processor，DPNN），通过按位次处理权值比特参与的 MAC 计算，消除上述重复数据访问与计算；通过非重训练的方式对权值位宽做逐比特动态调整，使模型在高能效与可靠性之间取得最优平衡。

1　动态精度计算中的模型间数据冗余

本节首先对动态精度计算场景进行整体介绍，并详细介绍其中最重要的两个子过程，即关键帧处理与动态精度校验；然后详细分析该过程中模型间的数据冗余，即权值比特位重叠导致的重复数据访问与计算。

1.1　动态精度计算简介

在动态精度计算场景下，硬件需要计算不同精度版本的模型。例如在安防摄像头设备中部署的用于物体检测的神经网络，如图 1 所示，动态精度计算的过程可划分为两部分。（1）当画面突然出现较大变化或识别难度较高时（例如突然出现大量移动物体），此时的输入为关键帧（key frame），设备往往需要调用高精度（长位宽）版本的模型，以保证识别准确率。（2）当画面处于正常状态时（例如画面静止），此时的输入为普通帧（normal frame），设备只需要运行低精度（短位宽）版本的模型，以节省能耗。

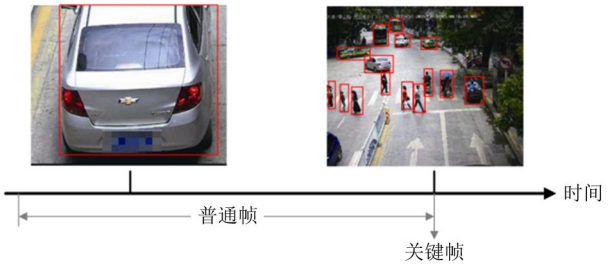


图 1　安防摄像头设备中的动态精度计算

此外，每经过一定数量的普通帧，将自行转入一帧关键帧的处理，即关键帧的出现不仅是在输入画面突变或识别难度较高时，而且呈周期性。并且在每次对关键帧处理完毕后，高精度模型的计算结果也保留下来，用于后续对低精度模型正确性的校验，以及根据校验结果对低精度模型中的数据位宽作动态调整。通过了正确性校验的低精度模型用于对普通帧的处理，这一过程并无特殊之处，硬件仅需对低精度模型进行计算。下面本文分别介绍对关键帧的处理和动态精度校验过程，以及其中的数据冗余问题。

1.2　关键帧处理与动态精度校验

动态精度计算对每帧输入的处理如图 2 所示。在计算关键帧时，设备需要同时运行高精度模型与低精度模型。其中高精度模型用于输出对关键帧的执行结果；低精度模型的计算结果用于验证与高精度模型的差距是否在预设阈值内，并对模型数据位宽作出调整。低精度模型的精度由均方误差（mean square error，MSE）度量，如图 2 虚线框中内容所示。

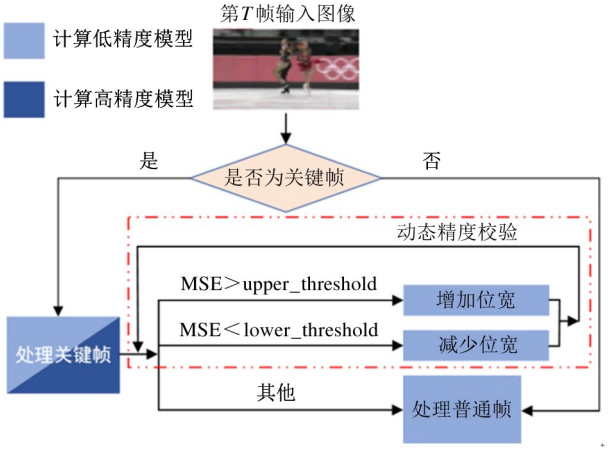


图 2　动态精度计算对每帧输入的处理过程

当 MSE 小于预设偏差下限时，表明低精度模型输入数据的识别难度或变化程度相对偏低，需减少数据位宽以节省能耗。

当 MSE 大于预设偏差上限时，表明低精度模型输入数据的识别难度或变化程度相对偏高，需增加数据位宽以提高精度。

当 MSE 处于预设偏差的上限和下限之间时，表明低精度模型在可靠性与高能效之间取得了平衡，无需调节数据位宽。

此外，本文在进行动态精度校验时，依据 MSE

的结果可实现对权值位宽逐层、逐比特位的细粒度调整,从而在高能效与可靠性之间取得最优平衡。

1.3 模型间的数据冗余

在动态精度计算场景中,同一网络结构的高精度模型和低精度模型的权值数值存在极高的重合度(如图3中灰色部分所示),在当前的计算和访存过程均存在数据的冗余处理。如图3所示,低精度模型的3比特权值由高精度模型的6比特权值以"四舍五入"的方式规约得到。高位宽权值的数值由两部分拼接组成,即高比特位 same _ bits 和规约截断比特位 cut _ bits。设 cut _ bits 的长度为 k,于是高精度模型中的 MAC 计算可以分解为两部分,如式(1)所示。低位宽权值的数值由两部分的加和组成,即高位宽权值的高比特位 same _ bits 和1比特规约进位 round _ bit。于是低精度模型中的 MAC 计算也可以分解为两部分,如式(2)所示。上述规律同样适用于采用"随机规约"、"向上取整规约"、"向下取整规约"等精度调节方法。特别地,当使用"向下取整规约"时,等同于对长位宽权值进行直接截断,此时所有权值的 round _ bit 都为0。

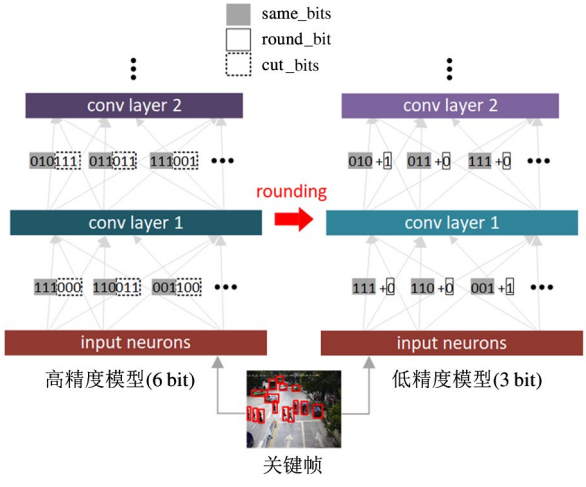


图3 动态精度计算中的权值 bit 位重叠(same _ bits)

$$\begin{aligned}\sum_i \text{neuron}_i^{\text{high}} \cdot \text{synapse}_i^{\text{high}} &= \sum_i \text{neuron}_i^{\text{high}} \cdot \langle \text{same\_bits}, \text{cut\_bits} \rangle \\ &= 2^k \cdot \sum_i \text{neuron}_i^{\text{high}} \cdot \text{same\_bits} \\ &\quad + \sum_i \text{neuron}_i^{\text{high}} \cdot \text{cut\_bits} \end{aligned} \tag{1}$$

$$\begin{aligned}\sum_i \text{neuron}_i^{\text{low}} \cdot \text{synapse}_i^{\text{low}} &= \sum_i \text{neuron}_i^{\text{low}} \cdot (\text{same\_bits} + \text{round\_bit}) \\ &= \sum_i \text{neuron}_i^{\text{low}} \cdot \text{same\_bits} \\ &\quad + \sum_i \text{neuron}_i^{\text{low}} \cdot \text{round\_bit} \end{aligned} \tag{2}$$

结合式(1)和式(2)可以发现,对于高精度模型与低精度模型共享的 same _ bits,即权值比特位重叠的部分,并不需要在计算不同精度模型时加载多次。并且当输入神经元相同时,即 $\text{neuron}_i^{\text{high}} = \text{neuron}_i^{\text{low}}$,same _ bits 参与的 MAC 计算也是重复的。

上述计算过程的数据计算和访存冗余,不仅存在于关键帧处理和普通帧处理的模型切换过程,还大量存在于动态精度校验中使用的网络结构相同、权值位宽不同的模型中。

2 基于位次的权值数据 MAC 计算方法与多精度 MAC 运算单元

本节主要介绍基于位次的权值数据 MAC 计算方法,并提出了一种适配动态精度计算的专用 MAC 运算单元。

2.1 基于位次的权值数据 MAC 计算

在执行卷积计算时,将所有卷积核中的权值从高位到低位逐比特展开,并统一计算相同位次的权值比特参与的 MAC 运算。如图4所示,设该卷积运算中神经元向量 $\vec{x}$ 的长度为 N,将3个卷积核 $\vec{w}_1$、$\vec{w}_2$、$\vec{w}_3$ 中的3比特权值按比特位次划分,分别计算每个比特位次对应的二进制权值矩阵 $\boldsymbol{W}_i(i=1,2,3)$ 与 $\vec{x}$ 的乘积。当这3个位次的权值比特用于多个模型的计算时,例如作为图3中的 same _ bits,硬件仅需加载 cut _ bits 和 round _ bit;当输入神经元也一致时,硬件可将 same _ bits 参与的 MAC 计算结果反复使用,而无需重新计算。

由于在固定位宽的乘法器中一个 p 比特乘法由 $p-1$ 个移位加法完成,故可近似地将一个 p 比特乘法等效为 $p-1$ 个加法。于是图4示例的原计算量为 $3N+(32-1)\cdot 3N = 96N$。设 $\vec{x}$ 的位宽为32比特,那么使用这种计算方式共需要约 $9N$ 个加法;而

原本的计算量是 $3N$ 个 32 比特乘法与 $3N$ 个加法。故按位次处理权值比特的计算模式，不仅不会增加 MAC 计算量，而且避免了固定位宽 MAC 单元[6-7]处理混合精度神经网络模型中的不同位宽数据时的算力浪费问题。

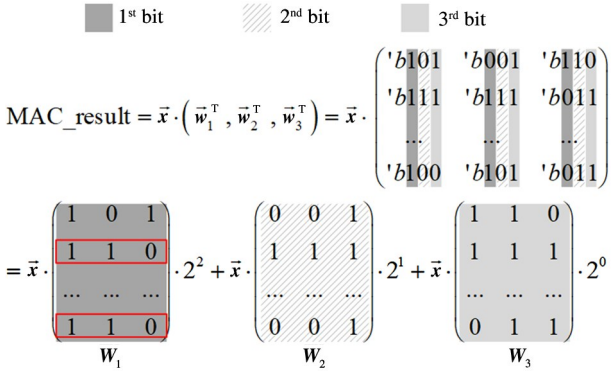


图 4　按权值比特位次划分的 MAC 计算

此外，在单个权值矩阵中也可能存在比特位的重叠带来的重复计算，可采用“合并同类项”的思路去除这种重复计算。如图 4 中的线框所示，以 $\vec{x}\cdot\boldsymbol{W}_1$ 为例，其中神经元 x_2 和 x_N 与相同的行向量 $\vec{v}=(1,1,0)$ 相乘，所以可“合并同类项”为 $x_2\cdot\vec{v}+x_N\cdot\vec{v}=(x_2+x_N)\cdot\vec{v}$。如此，原先需要计算 2 次的 x_2+x_N 便只需计算 1 次。

$$\vec{v}_i=\begin{cases}\vec{0} & index_i=0\\ \vec{k}_{index_i} & index_i\neq 0\end{cases} \tag{3}$$

应用上述思路，如图 5 所示。$\boldsymbol{W}_1$ 中的 N 个行向量共有 $2^3-1=7$ 种非零取值，按从小到大的顺序记为 $\vec{k}_1,\vec{k}_2,\cdots,\vec{k}_7$。而后将 $\boldsymbol{W}_1$ 记作标量序列 $index_1$，$index_2,\cdots,index_N$，其中 $index_i\in\{0,1,2,\cdots,7\}$，对应第 i 个 $\boldsymbol{W}_1$ 行向量 $\vec{v}_i$ 的取值，如式(3)所示。于是基于位次的权值数据 MAC 计算可以分为两步。

步骤 1　根据标量序列 $index_1,index_2,\cdots,index_N$，将神经元 $\vec{x}=(x_1,x_2,\cdots,x_N)$“合并”(累加)到 2^a-1 个“同类项”(部分和) $\vec{z}=(z_1,z_2,\cdots,z_{2^a-1})$ 中，如式(4)所示。

步骤 2　根据向量序列 $\vec{k}_1,\vec{k}_2,\cdots,\vec{k}_{2^a-1}$，将上述 2^a-1 个“同类项”(部分和)整合为向量 $\vec{o}=(o_1,o_2,\cdots,o_a)$，如式(5)所示。

$$\forall k\in\{1,2,\cdots,2^a-1\},\ z_k=\sum_{i=1}^{N}I(index_i==k)\cdot x_i \tag{4}$$

$$\vec{o}=\vec{z}\cdot(\vec{k}_1^{\mathrm{T}},\vec{k}_2^{\mathrm{T}},\cdots,\vec{k}_{2^a-1}^{\mathrm{T}})^{\mathrm{T}}=\sum_{i=1}^{2^a-1}z_i\cdot\vec{k}_i \tag{5}$$

其中 N 为单个卷积核的大小；a 为一次性合并处理的卷积核个数(本例中 $a=3$)；向量序列 $\vec{k}_1,\vec{k}_2,\cdots,\vec{k}_{2^a-1}$ 的取值仅与 a 相关；$I(\cdot)$ 为示性函数。最后，在本例中设 $\vec{o}_i=\vec{x}\cdot\boldsymbol{W}_i(i=1,2,3)$，那么最终计算结果 $\mathrm{MAC\_result}=\sum_{i=1}^{3}2^{3-i}\cdot\vec{o}_i$。

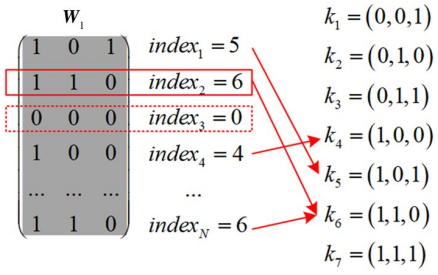


图 5　“合并同类项”法提取重复的 MAC 计算

效果上，相较于此前的比特串行 MAC 单元[8-11]仅利用了数据比特的稀疏性，即跳过权值或神经元数据中比特位取 0 导致的含零操作数的乘加，“合并同类项”法考虑了权值比特位重叠带来的计算重复，进一步降低了 MAC 计算量。

总之，按位次的权值数据 MAC 计算方法在保证不改变计算结果的同时，具有更低的计算量。

2.2　多精度 MAC 运算单元设计

本节提出了一种高效的多精度 MAC 运算单元。该 MAC 运算单元的整体结构如图 6 所示，由 ACC 单元与 Convert 单元组成。其中 ACC 单元完成式(4)所示的部分和向量 $\vec{z}$ 的计算(步骤 1)；Convert 单元完成式(5)所示的整合向量 $\vec{o}$ 的计算(步骤 2)。

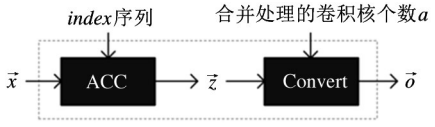


图 6　多精度 MAC 运算单元的整体结构

具体地，ACC 单元的结构如图 7 所示。当 $index_i\neq 0$ 时，ACC 单元根据其取值将神经元 x_i 分

发给部分和 z_{index_i}，并与之累加，即 $z_{index_i} = z_{index_i} + x_i$。其中神经元数据 x_1，x_2，…，x_{N_0} 以比特流的方式逐比特输入 ACC 单元，即每个时钟周期输入 N_0 比特；进位保留在运算单元内，并随着比特流更新。值得注意的是，一个 ACC 单元的吞吐量 N_0 是固定的，而卷积核大小 N 是一个变量。在实际部署时，往往需要多个 ACC 单元并行处理同一个卷积运算。

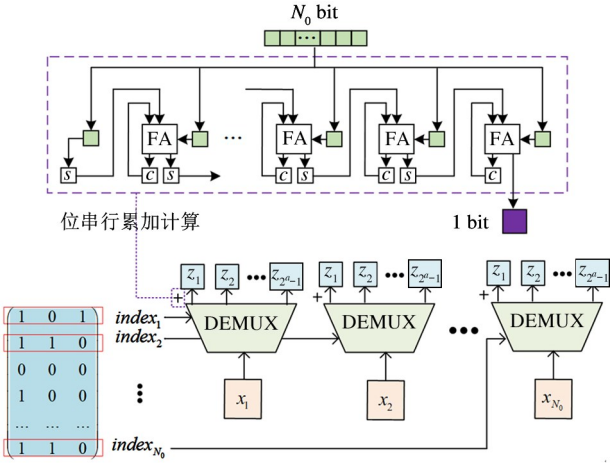


图 7　ACC 单元的结构

Convert 单元的结构如图 8 所示。图中以一次性合并处理的卷积核个数 $a = 3$ 为例，长度为 $2^a - 1 = 7$ 的部分和向量 $\vec{z}$ 由 ACC 单元逐比特输出，并逐比特流入 Convert 单元。bit 串行累加的进位保留在 Convert 单元内，并随输入比特流迭代更新。值得注意的是，Convert 单元总是与 ACC 单元成对工作，即一个 ACC 单元的输出对应一个 Convert 单元的输入。

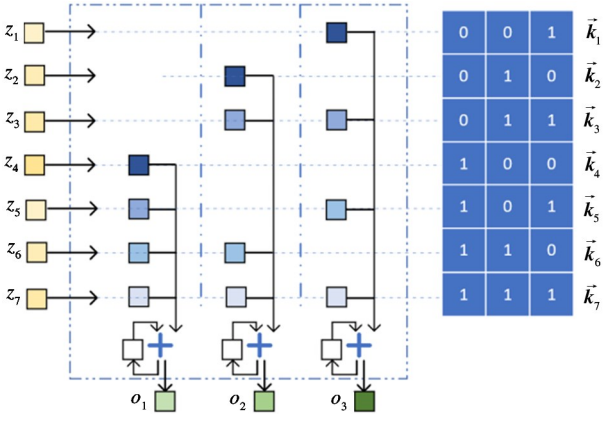


图 8　Convert 单元的结构

不仅如此，如式(4)和式(5)所示，其中的 MAC 运算都可以写成向量内积的形式，故将神经元和权值切分后得到的部分子块依然参与内积运算，即局部与整体一致。这种类似数学上“分形”的特性保证了多个 MAC 单元的可并行性，从而使得硬件支持任意规模的运算。

3　基于位串行计算的动态精度神经网络处理器 DPNN

本节主要介绍动态精度计算处理器 DPNN 的架构设计，并重点介绍其中基于上述多精度 MAC 运算单元的邻接互联结构高效计算阵列以及用于降低片外数据访问的游程编解码器。

3.1　整体架构

图 9 显示了处理器 DPNN 的总体结构，它主要由存储访问模块 DMA(direct memory access)、片上存储 SRAM(static random-access memory)、矩阵处理单元(matrix process unit，MPU)、向量处理单元(vector process unit，VPU)、标量处理单元(scalar process unit，SPU)、游程编解码器 Codec 和中心控制单元(control unit，CU)组成。

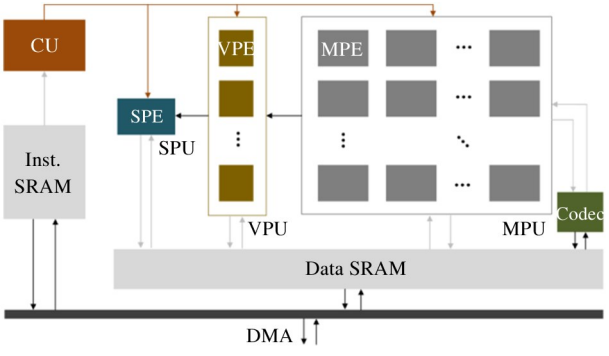


图 9　动态精度神经网络处理器 DPNN 的总体结构

MPU、VPU 和 SPU 分别是用于矩阵、向量和标量运算的计算处理单元。MPU/VPU 内部的计算元件 MPE/VPE 以一种邻接互联结构的形式组织成 2D/1D 计算阵列，即允许相邻计算元件间的直接数据传输。MPE/VPE 内负责 MAC 计算的元件即为上一节所述的多精度 MAC 运算单元。这些处理单元的输入数据通过 DMA 从数据 SRAM(Data SRAM)

中获取,例如执行卷积计算时,MPU 由 SRAM 分别为其内部的计算阵列提供水平和垂直方向的数据,即神经元和权值。SPU 的结构类似于一个 MPE/VPE,主要的区别在于 SPU 功能更丰富。因为 SPU 需要支持神经网络模型中池化、正则化、激活函数等操作中的各种标量计算,例如开平方函数 sqrt、三角函数 sin/cos 和指数函数 exp 等。

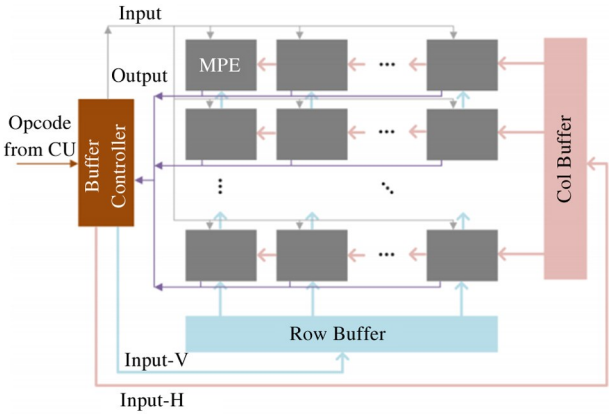
Codec 负责对片内稀疏率较高的数据以游程编码的方式进行压缩以及对编码的解压缩还原。

CU 读取存放在 SRAM(Inst. SRAM)中的指令并通过译码器解析成对应的控制流,完成对其他模块的调度,从而控制整个处理器的运行。

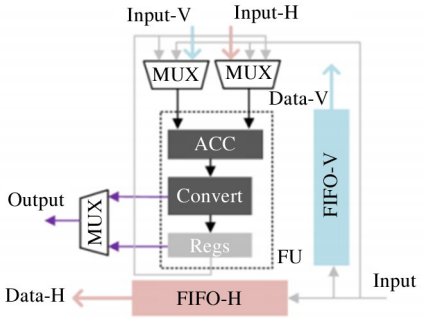
3.2　矩阵处理单元 MPU

MPU 主要负责神经网络中的卷积计算,并且出于可扩展性和通用性的考虑,MPU 也支持各种矩阵操作,例如矩阵转置、矩阵乘加和矩阵乘向量等。图 10(a)显示了 MPU 的具体结构,它包含 $P_x \times P_y$ 个矩阵处理元件 MPE 构成的 2D 计算阵列、1 个缓存控制器(buffer controller)和 2 个数据缓存(row buffer 和 Col buffer)。该 2D 计算阵列采用邻接互联结构,允许相邻的 MPE 之间直接通信,从而实现在 MPU 内部共享数据或迭代更新中间计算结果。阵列中每行有 P_x 个 MPE,每列有 P_y 个 MPE,所有的 MPE 由上述缓存控制器控制。缓存控制器将来自 Data SRAM 的垂直与水平方向的数据,即图示 Input-V 和 Input-H,分别提供给 MPE 计算阵列的右方和下方边界。缓存控制器也可以直接向所有 MPE 提供输入数据,即图示 Input。此外,所有 MPE 计算的输出数据,即图示 Output,可以由缓存控制器收集并存储在 SRAM 中。MPE 2D 计算阵列采用邻接互联结构的关键原因是:(1)支持 MPE 之间积累中间结果,便于进行内积、矩阵乘法等操作。(2)利用神经元与权值数据的重用带来的重复计算与数据访问,在卷积等操作中提升数据利用率,进而降低数据访存。(3)避免所有 MPE 相互之间的完全连接,从而保持有效的硬件实现(布局布线)。

图 10(b)显示了每个 MPE 的具体结构。MPE 的中心组件是 MAC 功能单元(functional unit, FU),由2.2节所述的多精度MAC运算单元和用于记录



(a) 矩阵处理单元 MPU 的结构



(b) MPU 中的一个矩阵处理元件 MPE 的结构

图 10　矩阵运算单元结构示意图

计算结果的寄存器组成。该 FU 可以接收来自其右方和下方相邻 MPE 的 Input-V 和 Input-H 数据作为输入,以 bit 串行的方式完成 MAC 计算,并将计算结果记录在内部寄存器中。此外,FU 还可以使用存储在其内部寄存器中的历史数据作为输入之一,从而能比较方便地进行 MAC 运算中部分和的累积。MPE 的另一个输入来源是从 MPU 的缓存控制器中直接获取的数据 Input。该 Input 数据可以存放于先入先出存储 FIFO-V 和 FIFO-H 中,用于向上方和左方的 MPE 提供数据 Data-V 和 Data-H。当 FU 完成计算,MPE 将从 FU 或直接从内部寄存器输出计算结果数据 Output。以上 MPU/MPE 中的所有操作统一由 CU 中的指令解析出的控制流调度完成。

VPU 中的 VPE 结构与 MPE 类似,其中的 1D 计算阵列也同样是邻接互联结构。由于篇幅限制,在此不作赘述。

3.3　游程编解码器 Codec

游程编码是一种仅记录序列中的非零取值,以及两个非零取值之间 0 的个数的压缩编码方式,适

合对稀疏率较高的数据进行压缩。游程编码的格式为〈data, count〉,其中 data 为 m bit 非零数据;count 为相邻两个非零数据之间“0”的个数,设位宽为 k bit。值得注意的是,当非零数据间隔中“0”的个数超过 2^k 时,将第 2^k+1 个“0”当作非零值记录下来。设数据的稀疏率为 λ,则上述游程编码的数据压缩率 compress _ rate(原始数据的大小/编码后数据的大小)满足式(6)。

$$\text{compress\_rate} = \frac{m}{(1-\lambda)\cdot(m+k)} \tag{6}$$

对于神经网络模型计算,如 2.1 节所述,权值可完全由标量序列 $index_1$, $index_2$,…, $index_N$ 和向量序列 $\vec{k}_1$, $\vec{k}_2$,…, $\vec{k}_{2^a-1}$ 代替。而 $\vec{k}_1$, $\vec{k}_2$,…, $\vec{k}_{2^a-1}$ 的取值仅由一次性合并处理的卷积核个数 a 确定,故权值的访存开销在于标量序列 $index_1$, $index_2$,…, $index_N$。当权值数据较为稀疏时,上述 $index$ 序列中将出现较多的“0”。可在片外预先对该序列进行游程编码,再输入到数据 SRAM 中。此后 Codec 读取 SRAM 上的编码值,将其解码后发送到相应的处理单元中。另一方面,当激活数据较为稀疏时,也同样可通过 Codec 进行游程编码,将编码值存储在片上 Data SRAM 上,再输出到片外 DRAM 中。相较于直接读取片外的原始权值或直接输出片内的原始激活值,这种方式将片外数据访问量降低了 compress _ rate 倍。

4 基于 DPNN 的动态精度计算

本节首先介绍神经网络中的卷积计算在处理器 DPNN 上的映射,主要体现如何利用神经元和权值数据重用降低访存;然后分别介绍动态精度计算中的两个关键过程,即关键帧处理与动态精度校验,在 DPNN 上的映射,主要体现如何利用位次处理权值 bit 的计算消除权值 bit 位重叠造成的数据冗余。

4.1 卷积计算映射

如 2.1 节所述,对于 a 个包含 p 比特权值的卷积核,可按照权值比特位次将其划分为 p 个二进制矩阵 $\boldsymbol{W}_1$, $\boldsymbol{W}_2$,…, $\boldsymbol{W}_p$,例如 $\boldsymbol{W}_1$ 包含的是所有权值的最高比特位。设每个卷积核的大小为 N,则 dim $\boldsymbol{W}_i = N \times a$,以及单个滑动窗口内的神经元个数为 N。进一步地,每个二进制矩阵 $\boldsymbol{W}_i$ 可转为长度为 N 的 $index$ 序列和长度为 2^a-1 的 $\vec{k}$ 序列,根据式(4)和式(5)由 $\lceil \frac{N}{N_0} \rceil$ 个专用 MAC 运算单元并行处理(其实也对应相同个数的 MPE)。其中 N_0 为一个 MAC 运算单元每个时钟周期输入的最大比特数,并且 $\vec{k}$ 序列的取值仅与 a 相关,即不需要占用存储。

和 2.1 节的用例保持一致,设 $a = p = 3$;设每个二进制矩阵 $\boldsymbol{W}_i$ 由 $\lceil \frac{N}{N_0} \rceil = 3$ 个 MAC 单元并行处理;设滑动窗口的个数为 3,展开成的神经元向量记作 $\vec{x}_1$, $\vec{x}_2$, $\vec{x}_3$;设 MPU 中的计算阵列大小为 $P_x \times P_y = 3 \times 3$。整个卷积计算在 MPU 中计算阵列上的映射如图 11 所示。图中 $\boldsymbol{W}_1$、$\boldsymbol{W}_2$、$\boldsymbol{W}_3$ 表示的是其对应的 $index$ 序列。可以看到,MPU 中每列的 3 个 MPE 负责一个滑动窗口展开成的神经元向量的计算,并且每行的 3 个 MPE 负责一个位次的权值 bit($index$ 序列)的计算。于是在 3×3 的计算阵列上,每行的 MPE 之间局部共享权值,并且每列的 MPE 之间局部共享神经元。由于不同的滑动窗口内往往存在大量重合的神经元,Row _ Buffer 可一次性取多个滑动窗口中的全部神经元,并根据滑动步长将神经元分发到相应的 MPE 中。这种神经元和权值数据的共享大幅降低了对片外 DRAM 的访存。

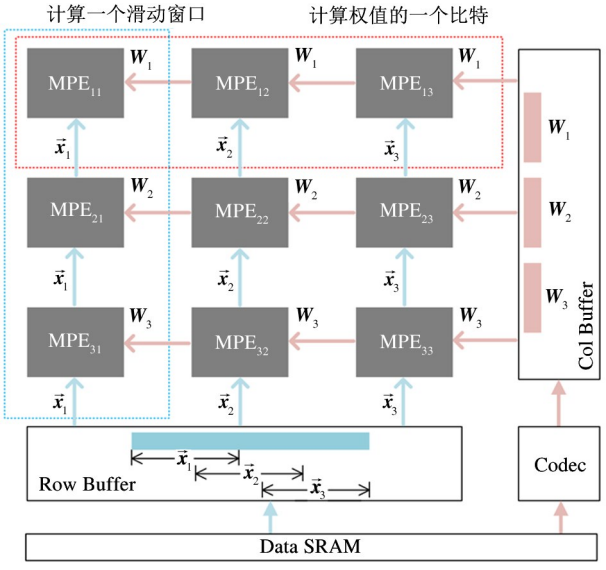


图 11 卷积计算在 DPNN 内矩阵处理单元 MPU 中 2D 计算阵列上的映射示例

此外，若从片外 DRAM 中加载的是已经过游程编码后的权值，编解码器 Codec 将其解码后送入 Col _ Buffer，而后也分发到相应的 MPE 中。最后，每行或每列的 MPE 计算结果经 VPU 或 SPU 完成整合，得到最终卷积计算结果。

在实际使用中，往往根据卷积计算的规模进行拆分，允许出现多行 MPE 处理来自同一个 $\boldsymbol{W}_i$ 的 *index* 序列，或多列 MPE 处理同一个滑动窗口中的神经元。即神经元和权值数据的共享不局限于一行或一列 MPE 之间，由实际卷积计算的规模决定。

4.2　关键帧处理映射

根据 1.2 节所述，在进行关键帧的处理时，处理器需要计算权值比特部分重叠的高、低精度模型的结果，并计算二者结果的差距度量 MSE。其中重叠的权值比特记作 same _ bits，是高精度权值的高位；权值降低精度时规约舍弃的比特位记作 cut _ bits，是高精度权值的低位；根据规约规则（例如“四舍五入”）出现的 1 比特进位记作 round _ bit，低精度权值由 same _ bits 和 round _ bit 的加和得到。

整个计算过程在 DPNN 上的映射如图 12 所示。DPNN 同时计算高、低精度的卷积，一方面输出高精度模型的计算结果，主要由 MPU 完成；另一方面输出高、低精度模型计算结果间的 MSE，由 MPU、VPU 与 SPU 协同完成。具体地，MPU 中的 MPE 阵列分 3 部分使用，分别用于计算 round _ bit、same _ bits 和 cut _ bits。对于计算 round _ bit 的 MPE 子阵列，神经元数据来自低精度模型；对于计算 same _ bits 的 MPE 子阵列，神经元数据一半来自低精度模型，另一半来自高精度模型（当神经元相同时，神经元也可以完全共享）；对于计算 cut _ bits 的 MPE 子阵列，神经元数据来自高精度模型。MPU 对高、低精度卷积的计算结果传入 VPU 中，由 VPU 计算 MSE。与处理向量内积时相同，每个 VPE 负责计算 MSE 中的一小段部分和，最后累加所有 VPE 的结果。VPU 对 MSE 的计算结果传入 SPU 中，SPU 负责将 MSE 与预设偏差阈值比较。

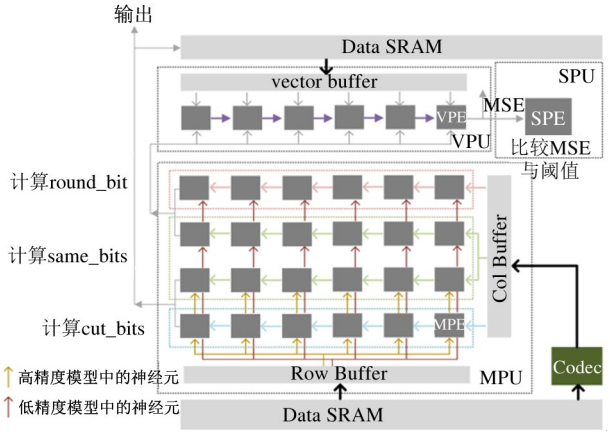


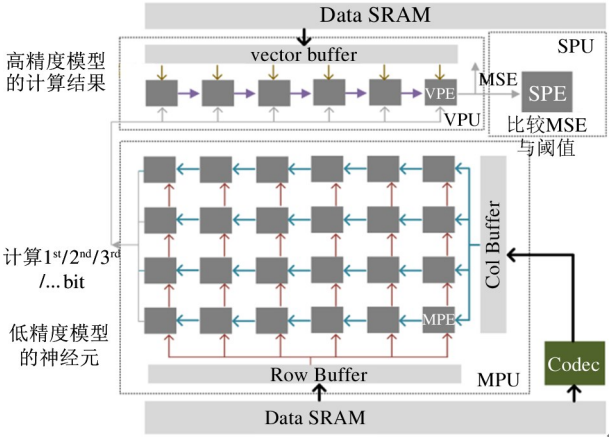
图 12　关键帧处理时高、低精度模型的卷积计算在 DPNN 上的映射

值得注意的是，当 Data SRAM 的容量相对于卷积输出的数据量较为宽裕时，高精度模型的计算结果除了直接输出到片外，也放置于 Data SRAM 中，以方便后续动态精度检验中计算 MSE 时对该计算结果的调用。

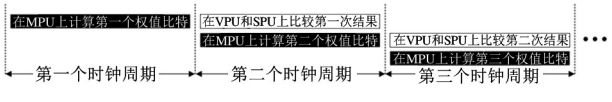
4.3　动态精度校验映射

根据 1.2 节所述，在进行动态精度校验时，处理器需要计算一组结构相同、位宽不同的模型，使得低精度模型以最低的权值位宽（最高的能效）满足可靠性要求。

整个计算过程在 DPNN 上的映射如图 13（a）所示。与关键帧处理十分类似，整个计算过程中，按位次处理权值 bit 参与的 MAC 操作，即分别计算 $\boldsymbol{W}_1$，$\boldsymbol{W}_2$，$\boldsymbol{W}_3$，…。并且每当 MPU 完成 $\boldsymbol{W}_i$ 参与的 MAC 计



（a）计算过程映射



（b）ping-pong 式流水处理过程

图 13　动态精度校验时按位次处理权值 bit 的卷积计算在 DPNN 上的映射

算后,VPU与SPU紧接着计算MSE并比较,此时计算 MSE 的两方是位宽为 i 的权值的卷积结果及此前存于 Data SRAM 中高精度卷积结果(当卷积输出规模过大时,需要从片外 DRAM 读入)。整个计算过程如图 13(b)所示的 ping-pong 式流水,即 VPU 与 SPU 根据位宽为 i 的权值的卷积结果计算 MSE 的同时,MPU 中同时进行 $\boldsymbol{W}_{i+1}$ 参与的 MAC 计算。这种 ping-pong 式流水处理保证了 DPNN 中算力资源的高效并行。设第 i^* 次比较时,MSE 处于预设偏差的上限和下限之间,那么权值位宽为 i^* 的低精度卷积通过校验。

效果上,整个计算过程中相同的权值 bit 位只加载了一次,其参与的 MAC 计算也只计算了一次,从而彻底消除了模型间 bit 位重叠带来的数据冗余。反复使用上述计算过程逐层执行动态精度校验后,低精度模型每层的权值位宽得以确定,此时的低精度模型在高能效与可靠性之间取得了最优平衡,可用于后续普通帧的计算。

5 实 验

本节对 DPNN 进行硬件实现,并通过实验从两个角度验证 DPNN 的优势:(1)作为一款基于 bit 串行架构的、支持任意规模、任意位宽神经网络计算的处理器,DPNN 具备一定的通用性。与代表性的 bit 串行架构神经网络处理器 Stripes[8] 相比,DPNN 具有更高的能效、更低的面积开销。(2)作为一款支持动态精度计算的处理器,DPNN 为 SaNNs 提供了更高效的硬件支撑。与 SaNNs 的最新进展之一 MinMaxNN[2] 相比,DPNN 具有更低的计算量与访存量。

5.1 实验工具与硬件实现

实验用 Verilog RTL 实现硬件加速器,Synopsys Design Compiler 进行综合,IC Compiler 进行布局布线,以及用 Synopsys PrimeTime PX 估计能耗。硬件综合所使用是 TSMC 45 nm 工艺,主频为 1 GHz。实现处理器 DPNN 所使用的硬件参数如表 1 所示,硬件实现后的面积与功耗信息如表 2 所示。

5.2 和 bit 串行神经网络处理器 Stripes 的比较

5.2.1 Benchmark

实验使用了一组常用的神经网络模型作为测试用例集。测试用例包括 AlexNet[12]、VGG-11[13]、VGG-16[14]、GoogleNet[15]、ResNet-18[13] 和 ResNet-50[13],且它们执行的推理计算是在数据集 ILSVRC12[16] 上进行图像分类。文献[17-19]将这些模型的权值位宽量化为 2 ~7 bit 不等,如表 3 所示。

表 1 处理器 DPNN 的硬件参数

硬件参数	数值	注释
P_x	16	MPU 的 2D 计算阵列一行中 MPE 的个数
P_y	16	MPU 的 2D 计算阵列一列中 MPE 的个数
P_z	16	VPU 的 1D 计算阵列中 VPE 的个数
N_0	32	MAC 运算单元输入 bit 流的宽度
a	4	MAC 运算单元输出 bit 流的宽度
Inst. SRAM	32 kB	指令 SRAM 的存储大小
Data SRAM	736 kB	数据 SRAM 的存储大小

表 2 处理器 DPNN 的功耗与面积

硬件模块	面积/mm^2	面积/%	功耗/mW	功耗/%
整个 MPNN	7.71	100.00	1472.31	100.00
FU 总计	4.20	54.47	1072.02	72.82
MPU 中的 FU	3.48	45.14	964.12	65.48
VPU 中的 FU	0.10	1.30	16.69	1.13
SPU 中的 FU	0.02	0.25	3.64	0.25
CU	0.23	2.98	83.00	5.64
Codec	0.30	3.89	125.62	8.53
SRAM 总计	2.98	38.65	191.68	13.02
Inst. SRAM	0.18	2.33	12.64	0.86
Data. SRAM	2.27	29.44	179.04	12.16

5.2.2 非定量分析

实验的比较对象为神经网络加速器 Stripes,该加速器同样基于 bit 串行架构,支持任意位宽的数据计算。首先对 DPNN 与 Stripes 进行非定量地比较。

(1)在执行相同模型的计算量上,DPNN 预计小于 Stripes。原因在于 DPNN 中的专用多精度 MAC 运算单元不仅过滤了 bit 位取“0”的权值参与的计算,而且消除了权值 bit 位重复造成的计算重复。

(2)在执行相同模型的访存上,DPNN 预计小于 Stripes。原因有两方面,一方面,DPNN 中邻接互联结构的计算阵列支持数据共享与中间结果累积/传

表 3 常见神经网络模型及其权值位宽

网络模型	AlexNet	VGG-11	VGG-16	GoogleNet	ResNet-18	ResNet-50
权值位宽	4	7	6	3	4	2

递,充分利用了卷积计算中的数据重用;另一方面,DPNN 通过 Codec 模块对权值与激活进行编解码,利用数据稀疏性对片内 SRAM 与片外 DRAM 交互的数据进行压缩,从而进一步降低了访存。

5.2.3 定量验证

功耗与面积 对于同为 bit 串行计算架构的 Stripes,调整 Stripes 中的硬件参数(PE 的数量)重新在 TSMC 45 nm 工艺与 1 GHz 主频下将其实现,并且保证与 DPNN 的峰值带宽一致,如式(7)所示。使用相同的工具得到 Stripes 的面积为 15.07 mm^2,功耗为 4231.12 mW。于是 DPNN 相较于 Stripes,在面积上节省了 1.95 倍,在功耗上节省了 2.87 倍。

$$peak\_bandwidth = N_0 \cdot P_x \cdot P_y = 32 \times 16 \times 16 \text{ bit/cycle} \tag{7}$$

性能与能效 如图 14 所示,在这些神经网络模型中,相较于 Stripes,DPNN 的性能平均提升了 2.57 倍。结合功耗数据,DPNN 的能效提升了 7.38 倍。

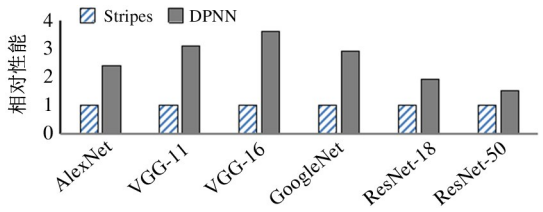


图 14 DPNN 与 Stripes 的性能比较

此外,如 2.1 节所述,DPNN 中按位次处理权值数据的 MAC 运算单元并不改变计算结果,故 DPNN 的精度与 Stripes 相同。本文抽查了数据集 ILSVRC12 中 1000 张图片的分类结果,发现 DPNN 与 Stripes 的输出结果完全一致。

5.3 和自感知神经网络系统 MinMaxNN 的比较

5.3.1 Benchmark

本节实验的目的是检验 DPNN 在动态精度计算场景中的效果。实验选取的数据、模型与相关度量如下。

(1) 数据集选用 VOT2018(Visual Object Tracking dataset 2018)[20],共包含 60 个用于物体追踪的短视频。

(2) 神经网络模型选用适合时间序列处理的 ResNet-50[13]。高精度 ResNet-50 中的权值数据为 32 bit 定点数(INT32),初始低精度 ResNet-50 中的权值数据为 INT16(后续计算过程中低精度模型的权值位宽将通过动态精度校验进行实时调整)。选用高精度权值到低精度权值的规约方式为“向下取整”。

(3) 选用 EAO[21] 度量模型为短视频数据的识别准确度。

(4) 使用 confidence score[2] 度量当前输入的识别难度,当 confidence score 小于给定阈值时,表示当前帧识别难度过高,将当前帧视为关键帧;此外,当帧数 T 为给定周期 T_0 的倍数时,也将此帧视为关键帧。其余情况均视为普通帧。

5.3.2 非定量分析

实验的比较对象为 SaNNs 的最新进展之一 MinMaxNN,首先对 DPNN 与 MinMaxNN 进行非定量地比较。

(1)在硬件支撑上,DPNN 更加高效。原因在于 MinMaxNN 中的硬件部分使用的是稀疏神经网络处理器 Cambricon-X[2,6]。而文献[8]中已经讨论了 Stripes 相较于 MinMaxNN 的优势,即在数据稀疏的基础上加入 bit 稀疏降低计算量;本文 5.2 节的实验则证明了 DPNN 相较于 Stripes 的优势,即在数据与 bit 稀疏的基础上进一步消除权值 bit 位重复带来的重复 MAC 计算。不仅如此,Cambricon-X 使用的是固定位宽的运算器,在处理多种位宽的模型时存在算力浪费问题。也就是说,在计算的高效性方面,DPNN > Stripes > Cambricon-X。

(2)在执行相同动态精度计算过程的计算量与访存量上,DPNN 预计小于 MinMaxNN。原因在于:MinMaxNN 以反复加载高、低精度模型,且重训练调

整模型精度的方式完成动态精度计算;而 DPNN 不仅避免了重训练,而且利用了一组结构相同、位宽不同模型间的权值 bit 位重叠带来的计算冗余,进一步降低了计算与访存。

上述两点 DPNN 的优势中,第 1 点优势主要由第 2 节所述专用 MAC 运算单元带来,而这种优势已经在 5.2 节的实验中得到了验证。本文突出的是 DPNN 在动态精度计算中创新性的调度(如 4.2 节和 4.3 节所述)带来的第 2 点优势,将 MinMaxNN 中的硬件直接替换为 DPNN,保留此前反复加载不同精度模型的计算模式,即仅使用普通的卷积计算(如 4.1 节所述)逐个处理不同精度的模型,从而抹平专用 MAC 单元的效果。

5.3.3 定量验证

以 VOT2018 中的视频数据 iceskater2 为例,整个计算过程中的识别准确度、每帧计算量、每帧访存量的变化如图 15 所示。图中上述 3 个指标分别由相对 EAO 值、相对 MAC 操作数(MAC-op)和相对访存大小(data access)度量(设高精度模型 INT32 ResNet-50 的 3 种度量值均为单位 1);关键帧的计算过程记作MP_CONV;动态精度校验的计算过程记作 DP _ MSE;普通帧的计算过程记作 CONV。设置关键帧出现的周期 $T_0 = 100$。

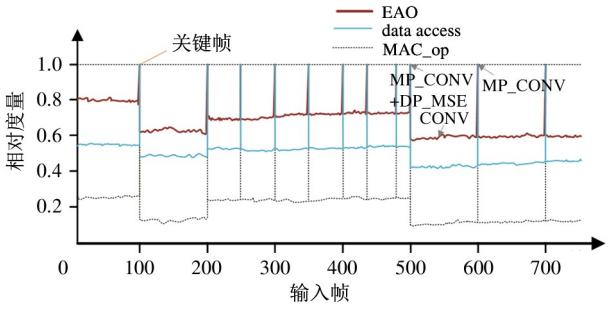
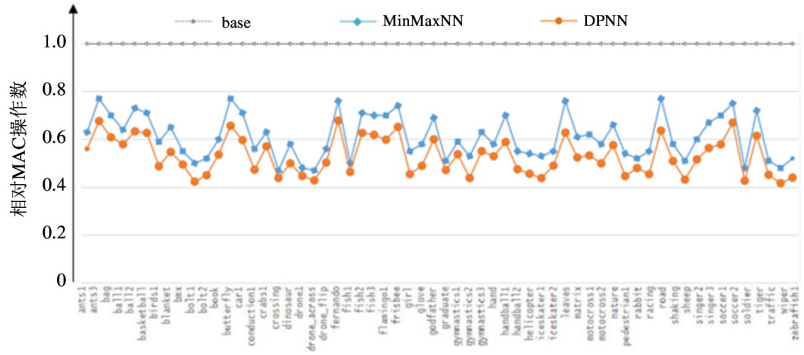


图 15 DPNN 对 iceskater2 的动态精度计算过程

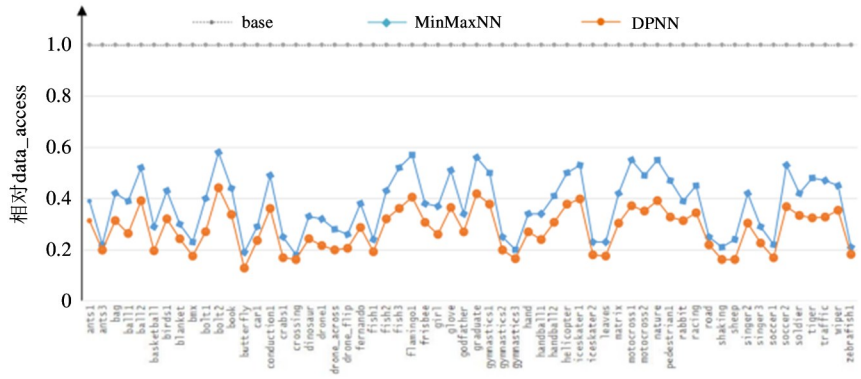
从图中可以看出,从起始到第一个关键帧之前,以 INT16 ResNet-50 作为低精度模型执行前 99 帧普通帧;在 iceskater2 视频中运动员动作变化较大的第 200 帧到第 500 帧之间,关键帧频繁出现,此时低精度模型的识别准确度维持在相对较高水准(于第 200 帧时经动态精度校验调整权值位宽);在其余运动员动作较收敛的帧数内,关键帧仅呈周期出现,且低精度模型的识别准确度相对较低(分别于第 100 帧和第 500 帧经动态精度校验调整权值位宽)。这种识别准确度的变化由低精度模型权值位宽的调整带来。此外,数据访存量与 MAC 计算量的变化趋势基本与识别精度保持一致,只是变化幅度有所不同。

在 iceskater2 视频的处理过程中,相较于全程使用高精度模型(INT32 ResNet-50)处理,DPNN 使计算量降低了 5.31 倍,访存量降低了 2.05 倍;相较于 MinMaxNN 的高精度模型与低精度模型(INT16 ResNet-50)互相切换的处理过程(也不进行重训练),DPNN 使计算量降低了 1.27 倍,访存量降低了 1.13 倍。

在 VOT2018 中所有 60 个视频的处理中,如图 16 所示,相较于全程使用高精度模型处理(base),DPNN 平均使计算量降低了 3.75 倍,访存量降低了 1.92 倍;相较于 MinMaxNN,DPNN 平均使计算量降低了 1.34 倍,访存量降低了 1.16 倍。进一步地,当关键帧周期 T_0 缩短为 50 时,相较于 MinMaxNN,DPNN 的计算量降低了 1.63 倍,访存量降低了 1.31 倍;当关键帧周期 T_0 缩短为 10 时,计算量降低了 2.52 倍,访存量降低了 1.93 倍。



(a) 计算量比较



（b）访存量比较

图 16　DPNN 对数据集 VOT2018 的动态精度计算

由此可见，在动态精度计算场景中，支持逐层、逐比特细粒度调整低精度模型位宽的 DPNN，相较于一般自感知神经网络系统中高、低精度模型切换的计算模式，更具有计算量与访存量的优势。而且对于更加复杂多变的场景，即关键帧处理与动态精度校验更频繁出现的场景，DPNN 节省计算与访存的优势更加明显。

6　结　论

本文提出了基于比特串行架构的、支持任意规模、任意位宽神经网络计算的处理器 DPNN，可以高效支持动态精度计算场景。DPNN 支持以非重训练的方式对模型数据位宽的逐层、逐比特的细粒度调整；并在调整模型精度或切换模型时消除了不同精度的模型间的权值 bit 位重叠造成的重复 MAC 计算与数据访问。

参考文献

[1] KOUNEV S, ZHU X, KEPHART J O, et al. Model-driven algorithms and architectures for self-aware computing systems [J]. *Dagstuhl Reports*, 2015, 5(1):164-196

[2] DU Z D, GUO Q, ZHAO Y W, et al. Self-aware neural network systems: a survey and new perspective[J]. *Proceedings of the IEEE*, 2020, 108(7): 1047-1067

[3] HEGDE K, AGRAWAL R, YAO Y, et al. Morph: flexible acceleration for 3d CNN-based video understanding [C]//2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Fukuoka, Japan, 2018: 933-946

[4] RIERA M, ARNAU J M, GONZÁLEZ A. Computation reuse in DNNs by exploiting input similarity[C]//2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), Los Angeles, USA, 2018: 57-68

[5] 陈壮. 基于动态精度的可扩展高能效 CNN 加速器设计[D]. 南京：东南大学微电子学院, 2018: 13-18

[6] ZHANG S, DU Z, ZHANG L, et al. Cambricon-X: an accelerator for sparse neural networks [C]//2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Taibei, China, 2016: 1-12

[7] 王婷,陈斌岳,张福海. 基于 FPGA 的卷积神经网络并行加速器设计[J]. 电子技术应用, 2021, 47(2):81-84

[8] JUDD P, ALBERICIO J, HETHERINGTON T, et al. Stripes: bit-serial deep neural network computing[C]//2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Taibei, China, 2016: 1-12

[9] 王明钊, 程华, 王宇泽, 等. 基于精度可变乘法器的脉动阵列[J]. 南京大学学报(自然科学), 2020, 255(6): 128-134

[10] DELMAS LASCORZ A, JUDD P, STUART D M, et al. Bit-tactical: a software/hardware approach to exploiting value and bit sparsity in neural networks[C]//Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Providence, USA, 2019: 749-763

[11] 谭曼琼, 徐成, 刘彦. 位串行 SVD 处理器的设计[J]. 小型微型计算机系统, 2012, 33(6):1358-1362

[12] 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

[13] HE K M, ZHANG X Y, REN S Q, et al. Deep residual learning for image recognition[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, 2016: 770-778

[14] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[EB/OL]. https://arxiv.org/pdf/1490.1556v1.pdf: arXiv, (2014-09-04),[2021-05-10]

[15] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, USA, 2015: 1-9

[16] RUSSAKOVSKY O, DENG J, SU H, et al. Imagenet large scale visual recognition challenge[J]. *International Journal of Computer Vision*, 2015, 115(3): 211-252

[17] HE Y, ZHANG X, SUN J. Channel pruning for accelerating very deep neural networks[C]// Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017: 1389-1397

[18] MIGACZ S. 8-bit inference with TensorRT[C]// GPU Technology Conference, San Jose, USA, 2017: 5

[19] YANG J W, SHEN X, XING J, et al. Quantization networks[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Los Angeles, USA,2019: 7308-7316

[20] KRISTAN M, MATAS J, LEONARDIS A, et al. A novel performance evaluation methodology for single-target trackers[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016, 38(11): 2137-2155

[21] KRISTAN M, MATAS J, LEONARDIS A, et al. The visual object tracking vot2015 challenge results[C]// Proceedings of the IEEE International Conference on Computer Vision workshops (ICCV), Santiago, Chile, 2015: 1-23

Bit-serial-based dynamic-precision neural network processor

HAO Yifan[*,**], ZHI Tian[*], DU Zidong[*]

([*]State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

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

Abstract

Aiming at the problem that the existing neural network dynamic-precision-computing system introduces a lot of computing and data access overhead in the process of periodic model retraining and switching, this paper proposes a dynamic-precision neural-network processor (DPNN) based on bit-serial-computing, which can support neural networks of any scales and bit-widths. DPNN supports fine-grained adjustment of model data accuracy without retraining, and eliminates repeated operands and data access caused by bits-of-synapses overlap during dynamic-precision-computing. The experimental results show that, compared with MinMaxNN, one of the latest advances in self-aware neural network systems (SaNNs), DPNN could reduce operands by 1.34 – 2.52 times and data access by 1.16 – 1.93 times on average. Compared with Stripes, the representative bit-serial-computing neural network processor, DPNN improves performance by 2.57 times, saves power-consumption by 2.87 times, and reduces area by 1.95 times.

Key words: neural network processor, dynamic precision computing, bit-serial