DOI:10.3969/j.issn.1003-5060.2024.09.002

面向深度强化学习自动驾驶决策算法的硬件加速器

冉敬楠, 倪 伟, 陈世宇
(合肥工业大学 微电子学院,安徽 合肥 230601)

摘 要:针对自动驾驶决策计算低功耗、低延时、高精度的需求,文章设计一种支持混合精度运算的深度强化学习自动驾驶决策算法的硬件加速器。通过多运算单元重构方式设计乘累加单元(multiply-and-accumulate unit,MAC),支持多种精度模式的计算,提高加速器的灵活性,降低量化模型的部署成本;通过多层次优化数据流,提高复用程度,优化加速器能耗比。在随机潜在演员评论家(stochastic latent actor-critic,SLAC)自动驾驶决策算法上测试该硬件加速器,结果表明:有效算力达到 18.3 GOPS,是 CPU 的 10.7 倍,GPU 的 3.3 倍;能效比达到 2.197 GOPS/W,是 CPU 的 104 倍,GPU 的 28 倍。同时提出一种高位数据编码(most significant bit data coding,MSB-DC)方法实现层内混合精度特征图计算,实验结果表明,该方法能以较少的延迟成本有效降低量化所带来的误差。

关键词:深度强化学习;自动驾驶;混合精度;神经网络量化;硬件加速
中图分类号:TN47 **文献标志码**:A **文章编号**:1003-5060(2024)09-1159-11

Hardware accelerator for deep reinforcement learning based autonomous driving decision algorithm

RAN Jingnan, NI Wei, CHEN Shiyu
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: In order to meet the requirements of low power consumption, low delay and high precision of autonomous driving decision calculation, a hardware accelerator for deep reinforcement learning based autonomous driving decision algorithm supporting mixed precision operation was designed. Multiply-and-accumulate unit(MAC) designed by multiple operation units reconstruction can support multiple precision mode calculation, thus improving the flexibility of accelerator and reducing the deployment cost of quantitative model. The multi-level optimization of the data flow improves the reuse degree and optimizes the accelerator energy consumption ratio. The effective computing power of the hardware accelerator for stochastic latent actor-critic(SLAC) based autonomous driving decision algorithm is 18.3 GOPS, which is 10.7 times that of CPU and 3.3 times that of GPU. The energy efficiency ratio is 2.197 GOPS/W, which is 104 times that of CPU and 28 times that of GPU. At the same time, the most significant bit data coding(MSB-DC) method is proposed to realize the calculation of intra-layer mixed precision feature map. Experiments show that this method can effectively reduce the error caused by quantization with less delay cost.

Key words: deep reinforcement learning; autonomous driving; mixed precision; neural network quantization; hardware acceleration

收稿日期:2022-04-18;**修回日期**:2022-05-02
基金项目:国家重点研发计划资助项目(2018YFB2202604)
作者简介:冉敬楠(1996—),男,安徽蚌埠人,合肥工业大学硕士生;
倪 伟(1977—),男,安徽合肥人,博士,合肥工业大学副教授,硕士生导师,通信作者,E-mail:ni.wei@hfut.edu.cn.

0 引 言

近年来,深度学习越来越广泛地应用于各种图像处理、语音识别等场景,然而在自动驾驶这种训练数据难以获得、极端场景难以覆盖的复杂场景下,仅通过监督学习是难以应付的。基于深度强化学习的端到端自动驾驶结构克服了深度学习依赖先验信息的问题,实现了感知到控制的直接映射[1]。但是,任务的复杂导致网络规模趋于庞大,对计算和存储的需求大幅增加。自动驾驶所要求的低延迟、高可靠性使其不可能将网络部署在云端,功耗限制也使得 GPU 等高能耗计算设备难以适用[2]。相较之下,领域定制硬件加速器(domain-specific hardware accelerator,DSA)具有性能高、功耗低、灵活性强等优势[3-4]。

为了克服边缘设备有限的计算量和存储量,越来越多的研究转向深度神经网络的量化部署[5]。利用神经网络对噪声容忍度很大的特性,将操作数由传统的单精度浮点数量化至 16、8 bit 整型甚至二值化计算,在消除高资源、高功耗浮点运算的同时成倍地降低存储和访存的需求[6-7]。然而各种各样的网络特性不同,导致网络之间、层与层甚至单层内部权值和特征图的分布差异,单一精度的神经网络加速器难以获得精度和资源的最优平衡。

因此,本文设计一种应用于深度强化学习的自动驾驶决策加速器,采用计算位宽可重构的乘累加单元(multiply-and-accumulate unit,MAC),能够实现不同精度的神经网络运算;引入存储块和计算块的可配置映射,优化不同网络结构下的数据流;为了解决单层网络内精度与计算位宽难以双赢的情况,提出一种高位数据编码(most significant bit data coding,MSB-DC)方法,在单层网络内部实现混合精度运算,避免特征图分布不均情况下量化所带来的截断误差。

1 深度强化学习自动驾驶决策

传统基于规则[8]和基于监督学习[9]的自动驾驶解决方案本质上都是通过人为数据"教会"机器如何开车,这种方式会带来一些问题:① 人为启发方式的自动驾驶策略过于保守;② 不同场景和任务需要对整个系统重新设计,且很多极端场景难以覆盖;③ 训练数据难以获得。

深度强化学习将深度学习的感知能力和强化学习的决策能力相集成,以整体形式同时解决感知、决策和控制等问题,为自动驾驶提供了一种端到端的解决方案。

2013 年,首个深度强化学习算法深度 Q 网络(deep Q network,DQN)被提出,它通过引入深度神经网络解决了强化学习中动作-状态空间过于庞大的问题[10];文献[11]将 DQN 应用在自动驾驶任务上,但由于 DQN 只支持离散动作空间,车辆仅有 5 种不同的动作;文献[12]在自动驾驶中采用了深度确定性策略梯度(deep deterministic policy gradient,DDPG)算法[13],使得模型能够决策出连续控制信号。

但基于深度强化学习的自动驾驶决策还是存在一定的局限性:① 端到端方案从感知直接映射至控制,系统黑盒内部原理无法解释,可靠性难以分析;② 道路情况的千变万化和多传感器融合导致感知数据复杂,感知至决策的映射跨度大、难度高。

文献[14]将序列潜在变量(sequential latent variables)[15]和最大熵(maximum entropy)强化学习[16]结合的随机潜在演员评论家(stochastic latent actor-critic,SLAC)算法[17]应用于自动驾驶,将高维感知信息编码至低位序列潜在空间,降低了复杂感知到决策的映射难度,并为端到端模型提供了可解释性。SLAC 算法使用概率图模型(probabailistic graphical model,PGM)对序列潜在空间进行建模,如图 1 所示。图 1 中:x_t 为 t 时刻的高维观测;z_t 为 t 时刻由高维观测编码得到的潜在变量;a_t 为 t 时刻的动作;m_t 为根据由 z_t 解码还原的观测。

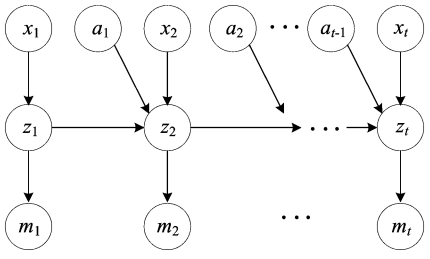

图 1 PGM 序列潜在环境模型

标准强化学习框架的学习目标是得到一个最大化期望收益的最优策略:

$$\pi^* = \arg\max_{\pi} E_{(s_t,a_t)} \sum_t r(s_t,a_t) \tag{1}$$

其中,s_t、a_t 分别为 t 时刻的状态和动作。最大熵强化学习框架在此基础上引入了熵的概念,在累计奖励上额外增加一个熵项,于是式(1)变为:

$$\pi^* = \arg\max_{\pi} \underset{(s_t,a_t)\sim\rho_\pi}{E} \sum_t [r(s_t,a_t) + H(\pi(a_t \mid s_t))] \tag{2}$$

$$H(\pi(a_t \mid s_t)) = -\mathrm{lb}\ \pi(a_t \mid s_t) \tag{3}$$

其中：熵 $H(\pi(a_t \mid s_t))$ 表示随机策略 π 概率分布的随机程度，策略的随机性越大，则熵越大。

将最大熵强化学习引入自动驾驶任务的 PGM 中，最终基于 SLAC 深度强化学习算法的自动驾驶结构如图 2 所示。

参照文献[18]变分自编码器(variational auto-encoder，VAE)中的自回归潜在变量，在实现时将潜在状态拆分为 2 个随机变量 z_t^1 和 z_t^2，并加入中间状态 f_t 作为高维观测的低维特征，通过神经网络实现状态转移函数。

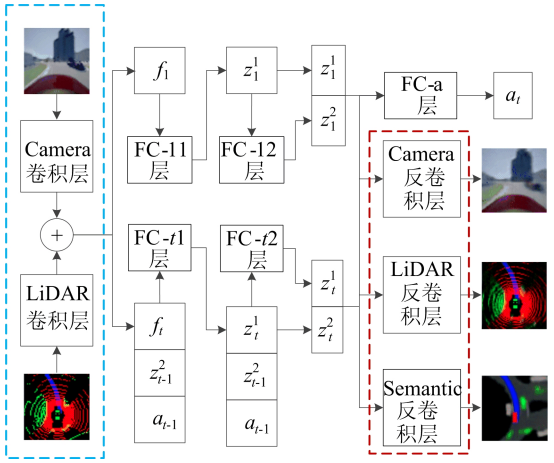

图 2　基于 SLAC 深度强化学习的自动驾驶决策实现

图 2 中：蓝色虚线框为 $f_t \sim p(f_t \mid x_t)$，x_t 为前视摄像头和 2D 激光雷达点云的 RGB 图片，实现形式为分别通过 2 个卷积神经网络后得到 $f_{t\text{-Camera}}$ 和 $f_{t\text{-LiDAR}}$，最终相加得到 f_t；红色虚线框为 $m_t \sim p(m_t \mid z_t^1, z_t^2)$，表示对感知信息的还原解码器，所有解码器均由转置卷积神经网络组成；FC-11 层为 $z_1^1 \sim p(z_1^1 \mid f_t)$，表示初始潜在状态 1 的概率分布；FC-12 层为 $z_1^2 \sim p(z_1^1 \mid z_1^1)$，表示初始潜在状态 2 的概率分布；FC-$t1$ 层为 $z_t^1 \sim p(z_t^1 \mid f_t, z_{t-1}^2, a_{t-1})$，表示潜在状态 1 的概率分布；FC-$t2$ 层为 $z_t^2 \sim p(z_t^2 \mid z_t^1, z_{t-1}^2, a_{t-1})$，表示潜在状态 2 的概率分布；FC-a 层为 $a_t \sim \pi(a_t \mid z_t^1, z_t^2)$，表示策略的概率分布；以上各层均采用全连接神经网络实现。

2　加速器设计

2.1　整体结构

本文所设计的混合精度神经网络加速器应用于自动驾驶的决策系统，加速器整体结构如图 3 所示。

主控块负责控制整个系统，主控制器从指令存储器中取指译码后将控制信息发送至其他模块；存储块中的权值缓存模块组和特征图缓存模块组用于缓存权值和特征图；存储控制块中的权值缓存地址控制器和特征图缓存地址控制器负责计算时权值和特征图的数据搬运，高维数据编码索引表用于存储层内混合计算信息；计算块负责数据的计算和写回，其中，乘累加阵列负责卷积、全连接计算，量化器组对乘累加阵列的计算结果执行量化过渡和激活函数后写回特征图缓存模块组。

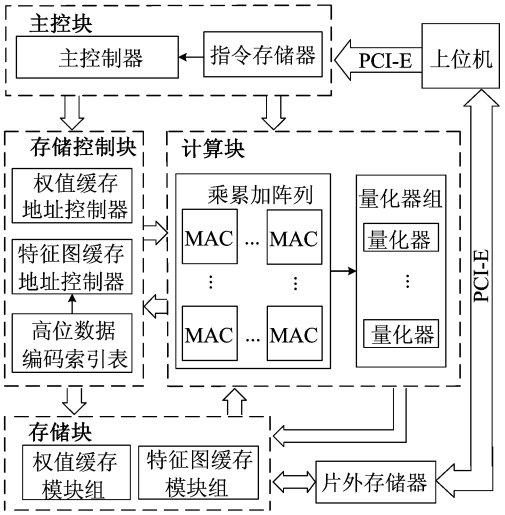

图 3 混合精度神经网络加速器整体结构

2.2　混合精度乘累加单元

神经网络中的大部分运算(如卷积、全连接)都由基本的乘累加操作来完成。为了实现混合精度的神经网络部署，设计了混合精度 MAC 单元，其结构如图 4 所示。

单个 MAC 单元包含 4 个乘法器、4 个加法器、3 个多路选择器、2 个移位逻辑、多个输出寄存器和若干控制逻辑，可配置为 8 bit×8 bit、16 bit×8 bit、16 bit×16 bit 3 种精度模式。当精度配置为 16 bit×16 bit 时，4 个 8 位乘法器将特征图和权值拆分后分别计算，得到的乘积通过选通移位逻辑后相加得到正确的结果；当精度配置为 8 bit×8 bit、16 bit×8 bit 时，单个 MAC 可同时计算 4 路、2 路乘累加操作，为了避免额外加法器树的资源消耗，MAC 内部的多路结果最终会累加在一起，这降低了 MAC 的输出带宽和资源使用。根据神经网络的原理，复用的输入不会累加在一

起[19],这种设计略微提高了输入带宽并牺牲了MAC内部的数据共享[20-21]。

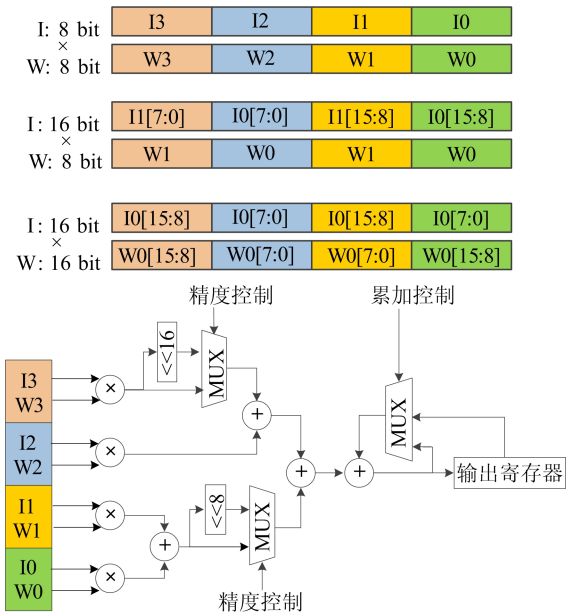

图 4 混合精度 MAC 结构

2.3 数据流设计

神经网络计算中输入特征图、权值和输出特征图均存在着不同程度的数据复用,如何最优化设计数据复用的数据流是众多研究中所探讨的重点。神经网络加速器中,数据的搬运所需要的功耗占系统功耗的绝大部分,而功耗优化正是边缘设备设计的难点[22]。

不同的网络层根据其输入特征图、输出特征图、卷积核、通道的形式和数量上的区别,使用单一的复用方案和数据流难以实现整个网络的整体优化。

因此本文通过计算块和存储块的互联结构、计算层内部的复用缓存和计算单元内部的复用缓存来优化加速器的数据流。

2.3.1 MAC 阵列与存储块的映射重构

MAC 阵列与存储块的映射重构如图5所示。

整个MAC阵列包含64个混合精度MAC,能够配置为 $m\times n$ 分组形式。其中:m 表示权值复用组个数,每个权值复用组中包含 n 个MAC单元,共享相同的权值;n 表示输入特征图复用组个数,每个输入特征图复用组中包含 m 个MAC单元,共享相同的输入特征图。整个权值缓存模块组包含64个Block RAM,可配置为 m 个权值缓存模块(图5中W RAM);整个特征图缓存模块组包含16个Block RAM,可配置为 n 个特征图缓存模块(图5中A RAM)。其中:$m=4,16,64$;$n=1,4,16$,且 $m\times n=64$。通过调整 m 和 n 的数量,对MAC阵列和缓存模块进行不同的分组模式后,重构MAC阵列和缓存模块组的映射连接。根据不同层的计算结构选择合适的 m、n 组合,能够获得偏向于特征图复用或偏向于权值复用或更均衡的复用数据流。

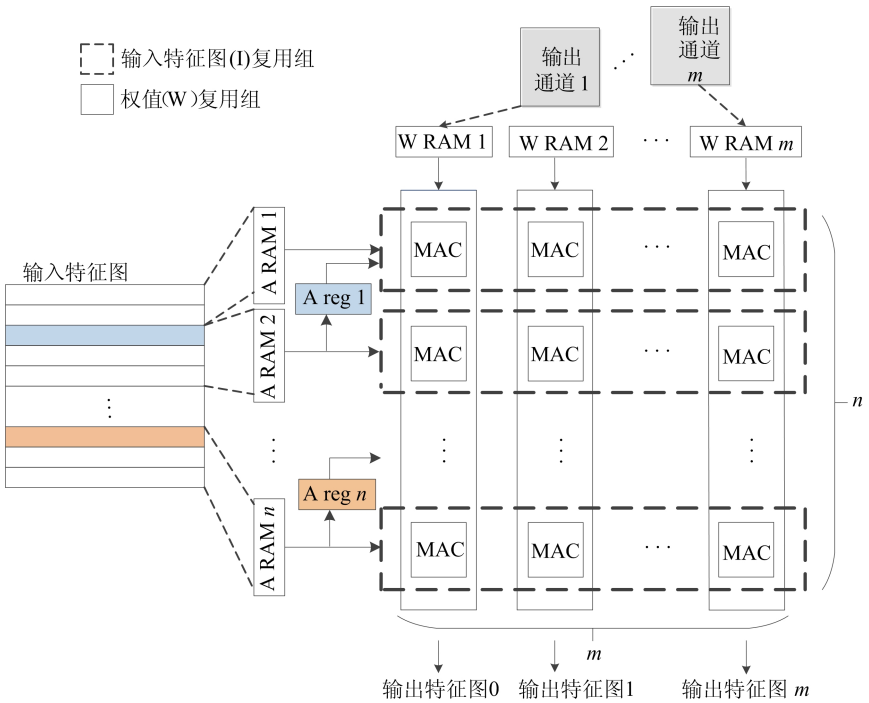

图 5 MAC 阵列与存储块的映射重构

同一特征图在卷积运算中存在大量的重叠部分计算，为避免重复访问这些数据，在数据缓存和计算单元重构互连映射的基础上，插入特征图行间缓存寄存器 A reg。图 5 中，每个特征图复用组负责一行输出特征图的计算，卷积操作中滑动窗口的存在，使得相邻的 2 个特征图复用组之间存在输入特征图的行间复用（图 5 中输入特征图中的彩色部分），当下方的特征图复用组接收到复用行后会对其进行计算并存入 A reg，当上方的特征图复用组计算至该复用行时，会直接从A reg 中取出数据，这避免了多次从 Block RAM 中访问复用行，在 MAC 阵列内部提高了同一特征图的数据复用。

2.3.2　MAC 内部的数据流及数据复用

上文从空间复用的角度通过 MAC 阵列的多组并行实现了数据共享，本节将从时间角度在 MAC 内部进一步挖掘数据复用的机会。

设计时，数据流将 2D 卷积以 1D 卷积方式展开，所有 1D 卷积的部分和通过输出特征图复用寄存器累加至完成一个完整的 2D 卷积，所有 2D 卷积的部分和通过输出特征图复用寄存器累加至完成一个完整的输出通道。在此仅以 MAC 内部的一路乘累加通道的 1D 卷积操作进行说明，其中卷积步长为 2。MAC 内部数据流如图 6 所示。

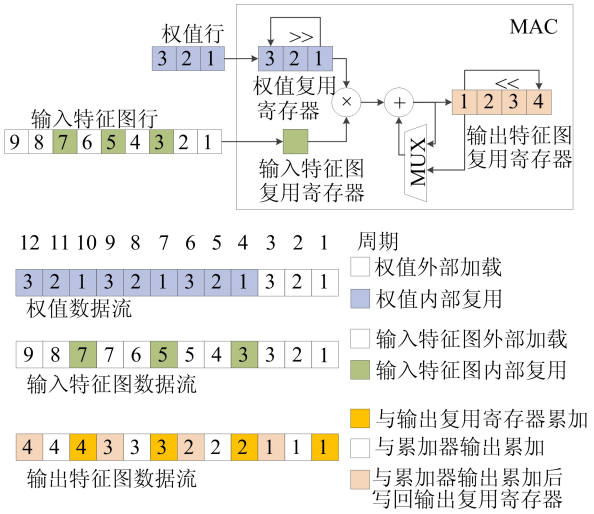

图 6　MAC 内部数据流

任意 MAC 内部为输入特征图、权值和输出特征图，均设置了复用寄存器。每次计算，权值缓存模块将卷积核中将要计算的一行发向 MAC，MAC 按顺序对权值逐个计算并存入权值复用寄存器，之后权值复用寄存器通过循环移位为计算单元提供复用的权值；同时特征图缓存模块将对应的一行输入特征图发向 MAC，其中重叠部分存储在输入特征图复用寄存器以供下一次卷积操作使用；一个卷积窗口内，第 1 个部分和与输出特征图复用寄存器内的值累加，之后的部分和与累加器输出进行累加，当最后一次累加结束后，结果写回输出特征图复用寄存器。这避免了多输入通道计算时输出结果的多次读写。

本文采用这种权值固定、输入特征图固定和输出特征图固定的方式提高了卷积操作中重叠的输入和输出的复用，降低了 MAC 单元与存储之间的数据搬运，通过高维至低维的卷积拆分，降低了这种复用操作所需要的寄存器成本。

2.4　层内混合精度计算

在神经网络算法中，合适的量化方案能够将其对精度的影响控制在一定范围内。然而，神经网络模型千变万化，同一网络中的不同层都可能出现完全不同的参数分布[23-24]。目前普遍采用的逐通道量化能使权值获得较好的量化效果，而对于特征图，逐层量化则是更多时候的选择。但是整层特征图按照同一个量化标准进行量化会带来以下问题[25]：当分布范围较大但并不均衡时，采用低位宽量化会导致因截断选取过高造成量化分辨率太低而带来的精度损失，或因截断选取过低造成截断误差太大而带来的精度损失；而采用高位宽量化又会因分布不均衡而造成大量存储和计算资源的浪费。

针对不均衡分布的特征图逐层量化方案，本文提出一种层内混合精度计算，通过更平衡的量化截断位置选取，尽可能地降低截断误差。由于加速器将高维卷积拆分为一维卷积，因此在实现层内混合精度计算时，将每个特征图行划分为一个区块。层内混合精度计算过程如图 7 所示，实现了 8～16 bit 层内混合精度计算。特征图缓存模块的位宽采用 32 bit，能够向 MAC 同时提供 4 输入通道 8 bit 特征图或 2 路的高位数据编码，其中 MSB-DC 位宽为 16 bit，[15∶14]表示此编码所属的输入通道，[13∶12]表示此编码所对应的权值，[11∶8]表示此编码所对应的输出寄存器，[7∶0]为高 8 bit 数据。

对于一个区块的计算，特征图缓存地址控制器会先将当前行的低 8 bit 数据搬运至 MAC 进行常规的 8 bit 乘累加，累加结束后，所有的部分和暂存在 MAC 内部的输出特征图复用寄存器内。之后特征图缓存地址控制器从 MSB-DC 索引表中获取当前区块中 MSB-DC 的数量，根据此

数量将特征图缓存模块中的 MSB-DC 搬运至 MAC,MAC 解析 MSB-DC 内的编码信息后进行计算并累加至相应的输出特征图复用寄存器。

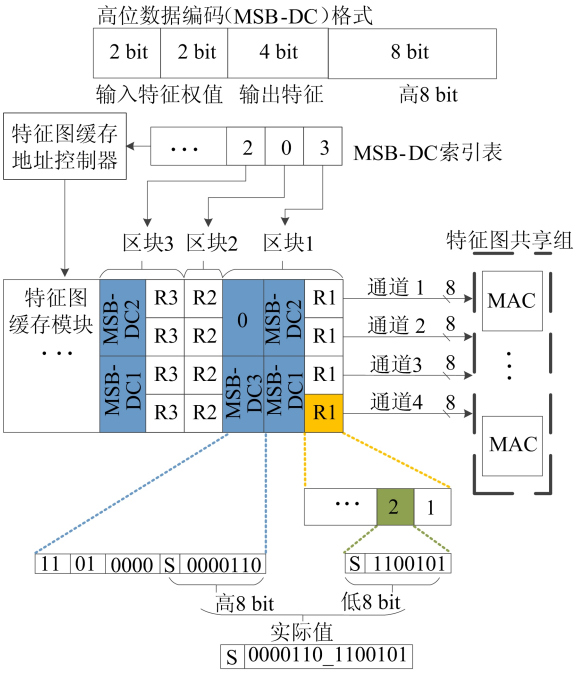

图 7 层内混合精度计算过程

图 7 以区块 1 中的 MSB-DC3 为例做具体描述。MAC 接收到 MSB-DC3,根据输入通道信息 $(11)_2$ 和输入通道信息 $(01)_2$ 解析得到此 MSB 对应第 4 输入通道中第 2 个权值后,将此权值从权值复用寄存器中取出与 MSB 部分进行乘法运算后并左移 8 位,解析输出信息 $(0000)_2$ 得到计算结果,对应 MAC 中输出复用寄存器中的第 1 个寄存器,然后将结果累加至此寄存器中。

通过这种层内混合精度计算方法,在输入特征图动态范围大但分布集中在部分范围的情况下,能够在 8 bit 位宽运算下以增加极小的运算时间和访存的代价获得更接近 16 bit 位宽的精度。

2.5 量化器

在神经网络的量化计算中,对于特征图普遍采用层间量化,这导致前层的输出特征图在作为后层的输入特征图前需进行两层之间的量化过渡处理,若当前层输出特征图的量化系数为 q_{O_i},而下一层的输入特征图量化系数为 $q_{I_{i+1}}$,则当前输出特征图在作为下一次输入特征图计算前需要进行量化系数转换,即

$$O_{i+1} = I_i\left(\frac{q_{O_i}}{q_{I_{i+1}}}\right) \tag{4}$$

所部署的 SLAC 网络模型各层均采用 ReLU/Leaky-ReLU 激活函数,即

$$f(x) = \begin{cases} x, & x > 0; \\ \alpha x, & x \leqslant 0 \end{cases} \tag{5}$$

其中,α 为 ReLU 类激活函数系数。利用这一特性可将量化过渡和激活函数合并操作,即

$$O_{i+1} = \begin{cases} I_i\dfrac{q_{O_i}}{q_{I_{i+1}}}, & I_i > 0; \\ I_i\dfrac{\alpha q_{O_i}}{q_{I_{i+1}}}, & I_i \leqslant 0 \end{cases} \tag{6}$$

根据以上描述设计了包含激活函数功能的量化器,量化器电路结构如图 8 所示。主控制器将本次操作的正量化-激活系数($q_{O_i}/q_{I_{i+1}}$)和负量化-激活系数($\alpha q_{O_i}/q_{I_{i+1}}$)发送至量化器,数据格式为单精度浮点数;之后 MAC 发送输出特征图,根据输出特征图的符号位选择匹配的量化-激活系数,进行输出特征图与量化-激活系数尾码的乘法运算,再根据量化-激活系数阶码进行移位,得到量化-激活结果的整型格式。

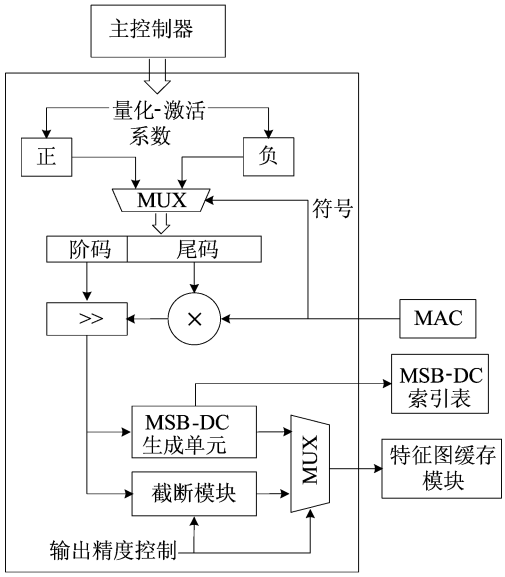

图 8 量化器电路结构

为了匹配 MAC 的多种特征图精度模式,量化器也提供了相应的处理方案。若下一层的输入特征图量化方案为 16 bit 或 8 bit,则通过截断模块对结果进行 16 bit 或 8 bit 截断后写回特征图缓存模块;若下一层的输入特征图量化方案为 16 bit与 8 bit 混合(下文简称 16-8 混合),则截断模块会先将数据中的 L8b 部分提取,同时 MSB-DC 生成单元判断数据的溢出情况,若 L8b 无法完全表示,则生成相应的 MSB-DC,当完成一整个区块的输出特征图处理后,MSB-DC 生成单元将

统计此区块所产生的 MSB-DC 数量并写入 MSB-DC 索引表。

3　实验分析

3.1　实验设置和性能分析

加速器采用 Xilinx UltraScale-XCVU440 实现，主频 125 MHz；CPU 对照组采用 Intel Corei 9-10900K@3.70 GHz，10 核 20 线程，128 GiB DDR4 内存；GPU 对照组采用 Nvidia GTX 1050Ti@1 354 MHz，CUDA 核心数量 768 个，4 GiB显存。采用 TensorFlow 2.5 和自动驾驶仿真平台 CARLA 0.9.11，CARLA 能够虚拟道路环境并为加速器提供传感器输入，同时接收加速器决策结果并反馈至车辆控制。实验软硬件环境如图 9 所示。

图 9　实验的软硬件环境

测试模型使用 SLAC 自动驾驶决策网络，参数量为 3.86×10^6 个，网络结构如图 10 所示，具体参数见表 1 所列。

所设计加速器通过 Vivado 2018.3 进行综合及布局布线，资源见表 2 所列。BRAM 使用率较高，这是由于为了优化加速器数据流，增加数据的复用，设置了较多的存储资源以避免出现带宽瓶颈。

在同样的实验条件下，本文平台与 CPU 和 GPU 的性能对比结果见表 3 所列。

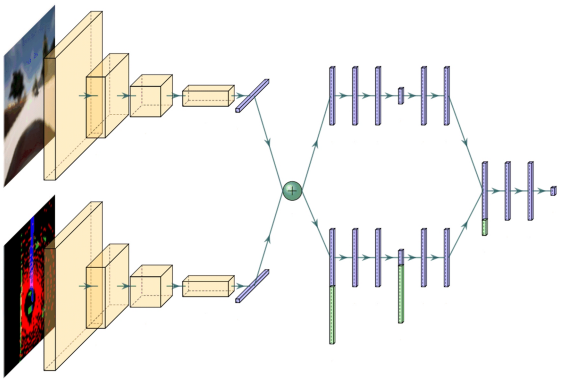

图 10　SLAC 自动驾驶决策网络结构

表 1　SLAC 网络参数

层	输入	卷积核	激活函数
Conv1	64×64×3	3×5×5×32	Leaky-ReLU
Conv2	32×32×32	32×3×3×64	Leaky-ReLU
Conv3	16×16×64	64×3×3×128	Leaky-ReLU
Conv4	8×8×128	128×3×3×256	Leaky-ReLU
Conv5	4×4×256	256×4×4×256	Leaky-ReLU
Latent1	256/514	256×256/514×256	Leaky-ReLU
Latent2	256	256×256	Leaky-ReLU
Latent3	256	256×64	None
Latent4	32/290	32×256/290×256	Leaky-ReLU
Latent5	256	256×256	Leaky-ReLU
Latent6	256	256×512	None
Actor1	288	288×256	ReLU
Actor2	256	256×256	ReLU
Actor3	256	256×4	None

表 2　加速器资源使用情况

资源	使用	可用	利用率/%
LUT/个	355 003	2 532 960	14.02
FF/个	122 223	5 065 920	2.41
BRAM/KiB	962	2 520	38.17
DSP/个	608	2 880	21.11

表 3　多平台性能比较

性能参数	本文平台	CPU	GPU
每帧推理时间/ms	1.983	20.081	6.557
帧率/(帧/s)	504	49	152
有效算力/GOPS	18.3	1.7	5.5
功耗/W	8.33	80.78	70.12
能效比/(GOPS/W)	2.197	0.021	0.078

由表 3 可知：本文所设计的 SLAC 自动驾驶决策网络能实现 504 帧/s 的高帧率，有效算力是 CPU 的 10.7 倍，是 GPU 的 3.3 倍；能效比是 CPU 的 104 倍，是 GPU 的 28 倍；满足自动驾驶决策低延迟、低功耗的边缘部署需求。

SLAC 网络运算量约为 36.3 MOPS，在卷积层配置为 8 bit×8 bit、全连接层配置为 16 bit×8 bit的精度模式下，本文所设计的加速器进行 1 帧推理大约需要 1.983 ms，有效算力为 18.3 GOPS，功耗为 8.33 W。

3.2　精度分析

在实验组中网络权值固定为 8 bit，特征图以 16 bit 为基准，对比 8 bit 特征图和 16-8 混合特征图的精度差异。目前对神经网络量化误差的分析普遍采用的方法是对比识别准确率，然而在深度强化学习中，最终的推理结果是智能体的决策行为，难以量化地评估哪种行为更优。因此本文从量化分析和仿真实验 2 个角度进行精度比较。

3.2.1 量化分析

由于 SLAC 的全连接层分布范围均很大,在实际加速器部署时采用特征图 16 bit-权值 8 bit 的量化方案,重点对卷积层进行 16-8 混合精度的性能测试。考虑到第 1 层卷积层输入为 RGB 图像,8 bit 整型完全能够在不损失精度的情况下表示,最终确定实验层为 Camera 的 Conv2～Conv5 和 LiDAR 的 Conv2～Conv5,其特征图分布详情见表 4 所列和图 11 所示,可以看出,每层特征图分布范围均很大,但主要部分还是集中在 0 点附近。

表 4 各层特征图范围

Camera	Min	Max	LiDAR	Min	Max
Conv2	−0.20	0.85	Conv2	−0.32	0.54
Conv3	−0.11	0.41	Conv3	−0.08	0.42
Conv4	−0.16	0.57	Conv4	−0.16	0.57
Conv5	−0.16	0.86	Conv5	−0.21	1.51

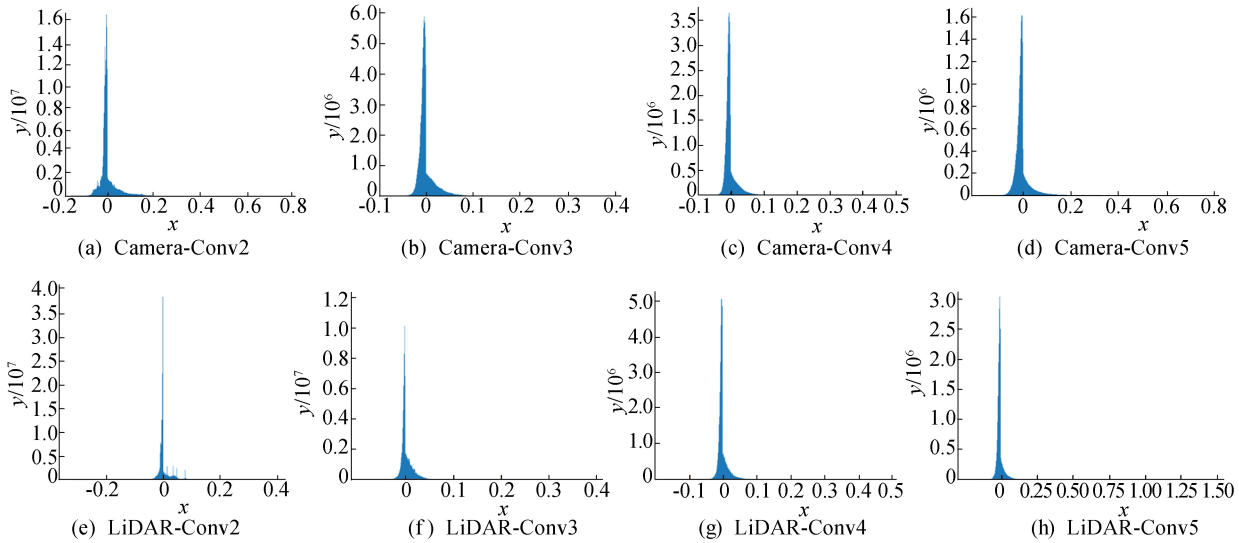

图 11 各层特征图参数分布

对 16 bit 量化,本文采用非饱和量化,将[−Max,Max]映射至[−32 767,32 767]。对于 8 bit 和 16-8 混合量化,采用 3 组不同的量化方案进行对比:

1) 饱和量化,将[−Min,Min]映射至[−127,127];

2) 欠饱和量化,将[−1.5Min,1.5Min]映射至[−127,127];

3) 过饱和量化,将[−0.5Min,0.5Min]映射至[−127,127]。

对以上对照组设置进行推理,比较运行的时间和推理精度。当使用 8 bit 精度时,加速器的数据流是规则的,因此 Camera 层和 LiDAR 层的运行时间是相同的;而在 16-8 混合精度下,因为输入特征图差异会导致计算时间动态变化,所以本文的运行时间以 8 bit 为基准,测试引入 16-8 混合计算对 Camera 层和 LiDAR 层的速度影响;由于深度神经网络中间特征图的实际意义还没有统一说法,为了比较单层推理输出特征图的精度,本文采用欧式距离量化 8 bit、16-8 混合与 16 bit 基准的误差。

欧式距离的缺陷是变异性较大的分量起着更重要的作用,但是类比神经网络中绝对值较大的值对结果的影响也更大,因此本文没有对欧式距离标准化。

不同量化截断下运行时间与精度如图 12 所示。从图 12 可以看出,在引入 16-8 混合精度后,大部分层的运算时间相较于 8 bit 增量很少,但是精度获得了较大提升。在不同层分不清的情况下不同截断点的选取也导致了精度和速度上的差异。

1) 选取较大的截断点会降低截断误差,但也会导致量化点间隔太大而降低计算精度,同时由于溢出较少,引入混合精度计算所带来的精度提升也较有限。

2) 选取较小的截断点会由于需要被截断的特征图太多而导致 8 bit 的精度损失很大,引入 16-8 混合计算在规避特征图截断的同时享受了更小量化间隔所带来的精度提升,但同时也增加了更多的计算时间。

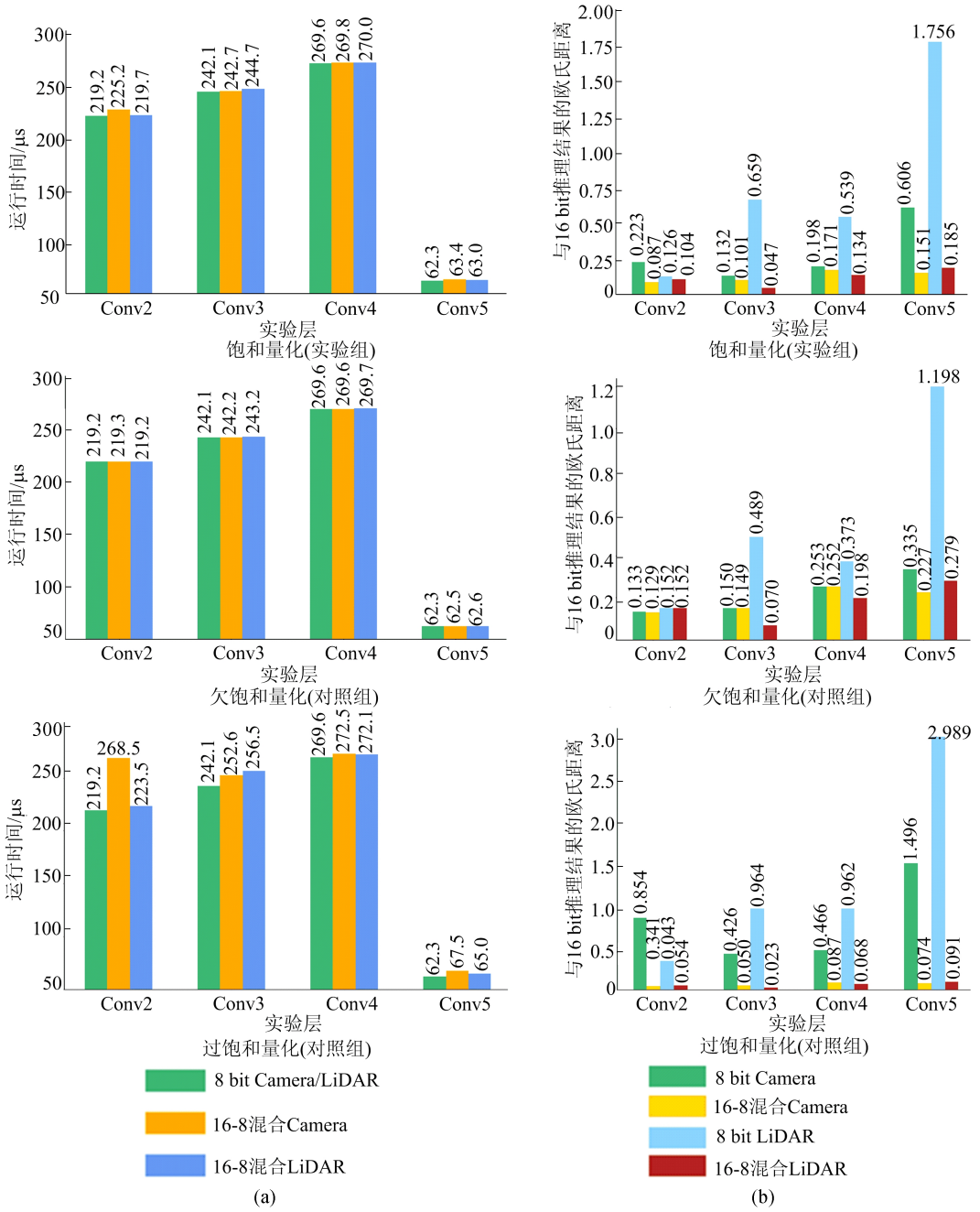

图 12　不同量化截断下运行时间与精度对比

3.2.2　仿真实验

本节通过软硬件结合进行实际的决策性能比较，上位机运行自动驾驶仿真环境 CARLA，通过 PCI-E 接口将虚拟的摄像头和激光雷达点云图像发送至现场可编程逻辑门阵列(field programmable gate array，FPGA)，FPGA 完成决策后反馈给上位机进行车辆的控制。在强化学习研究中，普遍的衡量方法是智能体的实验 Step 与获得 Reward。本实验设置参照文献[8]，Step 为从实验开始到发生碰撞或偏离车道时所经历的帧数，Reward 定义为：

$$R_{\text{reward}} = 200r_{\text{c}} + v + 10r_{\text{f}} + r_{\text{out}} - 5\alpha^2 + 0.2r_{\text{l}} - 0.1,$$

其中：v 为车辆速度大小；α 为转向角度；r 为车辆横向速度大小；发生碰撞时 r_{c} 为 -1，否则为 0；车速超过 8 m/s 时 r_{f} 为 -1，否则为 0。

实验对照组设置为：① 权值和特征图均采用单精度浮点数，通过 GPU 实现；② 权值 8 bit 量化，特征图 16 bit 量化；③ 权值 8 bit 量化，全连接层特征图 16 bit 量化，卷积层特征图 8 bit 量化；④ 权值 8 bit 量化，全连接层特征图 16 bit 量化，卷积层特征图 16-8 混合量化。

仿真实验测试如图 13 所示,由上至下分别为实验组 1～实验组 4,由左至右分别为每个实验组中的不同时刻,每张图上方分别为加速器所输入的感知数据,下方为仿真器运行情况。

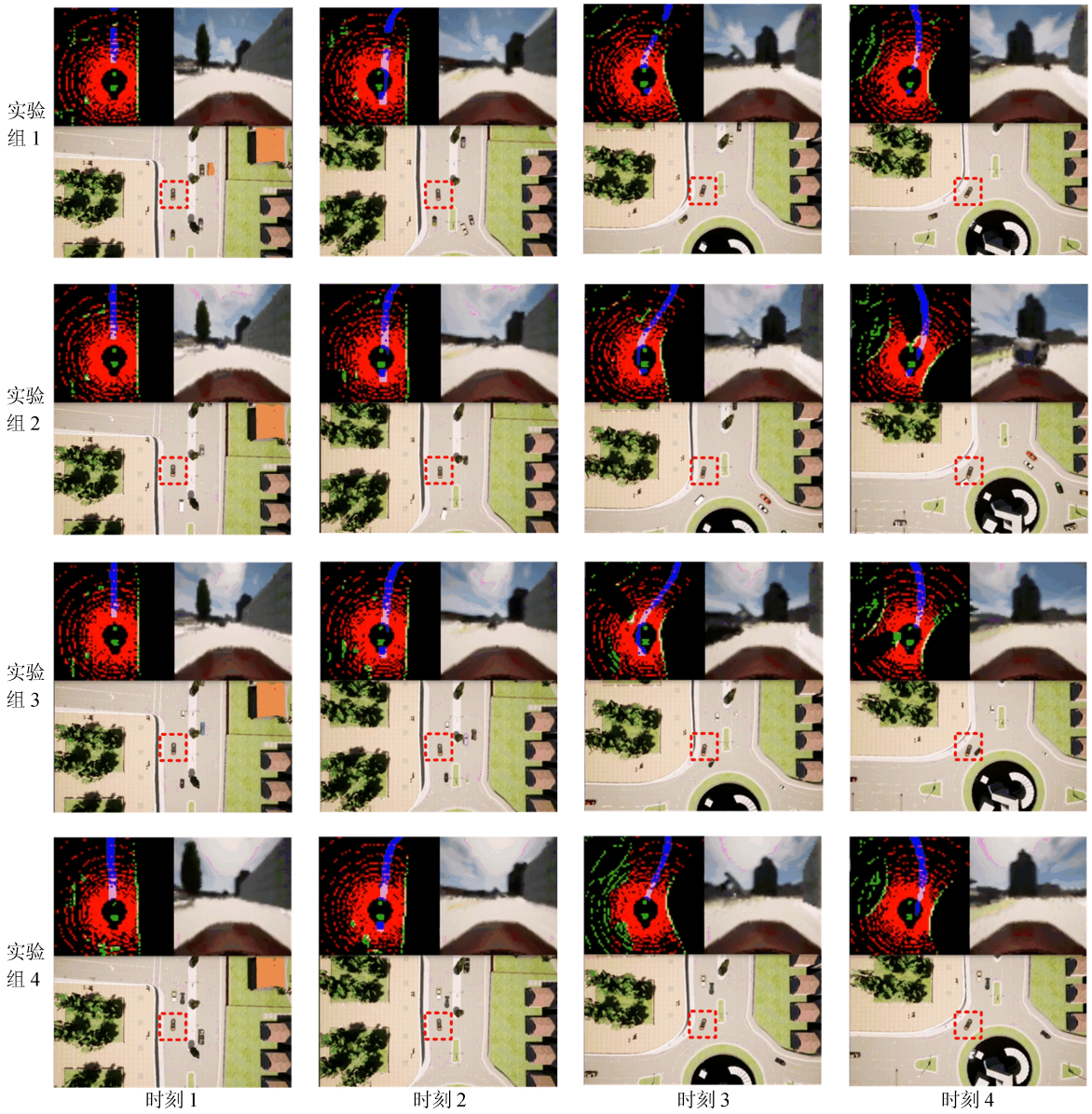

图 13 仿真实验测试

由于仿真器运行效率限制,每组运行 500 次并取均值,结果见表 5 所列。根据表 5 中数据难以直观地看出各实验组的计算精度,这是由于在每次实验中虚拟的其他车辆和行人行为有着很大的随机性,给实验路况带来较大的不确定性。

表 5 自动驾驶仿真实验结果

实验组	Step	Reward
1	314	636
2	317	359
3	160	385
4	219	408

4 结 论

本文从多个角度优化神经网络计算的数据流,降低访存量以优化功耗,提高计算单元的利用率来获得更低的计算延迟。本文设计可以达到 18.3 GOPS 的有效算力,是 CPU 的 10.7 倍、GPU 的 3.3 倍,能效比为 2.197 GOPS/W,是 CPU 的 104 倍、GPU 的 28 倍,适用于自动驾驶场景下的低延迟、低功耗的深度强化学习决策;同时通过计算单元的重构实现不同位宽的计算,提高了加速器的灵活性,降低了模型部署成本;并引

入特征图的层内混合精度计算方法，以较少的计算成本获得较高的精度收益，能够有效解决自动驾驶任务中高精度与低延迟、低功耗之间的冲突。

［参　考　文　献］

[1] KIRAN B R, SOBH I, TALPAERT V, et al. Deep reinforcement learning for autonomous driving: a survey[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(6): 4909-4926.

[2] CANZIANI A, PASZKE A, CULURCIELLO E. An analysis of deep neural network models for practical applications [EB/OL]. (2016-05-24)[2017-04-17]. https://doi.org/10.48550/arXiv.1605.07678.

[3] DALLY W J, TURAKHIA Y, HAN S. Domain-specific hardware accelerators[J]. Communications of the ACM, 2020, 63(7): 48-57.

[4] ROTHMANN M, PORRMANN M. A Survey of domain-specific architectures for reinforcement learning[J]. IEEE Access, 2022, 10: 13753-13767.

[5] LIM J S. Two-dimensional signal and image processing [M]. Englewood Cliffs: Prentice-Hall, 1990.

[6] VANHOUCKE V, SENIOR A, MAO M Z. Improving the speed of neural networks on CPUs[C]//Deep Learning and Unsupervised Feature Learning Workshop. [S. l.]: NIPS, 2011: 15196840.

[7] RASTEGARI M, ORDONEZ V, REDMON J, et al. Xnor-net: imagenet classification using binary convolutional neural networks[C]//European Conference on Computer Vision. Cham: Springer, 2016: 525-542.

[8] 朱冰，张培兴，赵健，等. 基于场景的自动驾驶汽车虚拟测试研究进展[J]. 中国公路学报，2019，32(6)：1-19.

[9] CODEVILLA F, MLLER M, LPEZ A, et al. End-to-end driving via conditional imitation learning[C]//2018 IEEE International Conference on Robotics and Automation. [S. l.]: IEEE, 2018: 4693-4700.

[10] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[EB/OL]. (2013-12-19)[2019-04-19]. https://doi.org/abs/10.48550/arXiv.1312.5602.

[11] WOLF P, HUBSCHNEIDER C, WEBER M, et al. Learning how to drive in a real world simulation with deep q-networks[C]//2017 IEEE Intelligent Vehicles Symposium (IV). [S. l.]: IEEE, 2017: 244-250.

[12] WANG S, JIA D Y, WENG X S. Deep reinforcement learning for autonomous driving[EB/OL]. (2018-11-28)[2019-05-19]. https://doi.org/10.48550/arXiv.1811.11329.

[13] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[EB/OL]. (2015-09-09)[2019-07-05]. https://doi.org/10.48550/arXiv.1509.02971.

[14] CHEN J Y, LI S E, TOMIZUKA M. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning[J]. IEEE Transactions on Intelligent Transportation Systems, 2022, 23(6): 5068-5078.

[15] MURPHY K P. Machine learning: a probabilistic perspective[M]. Cambridge, Mass.: MIT Press, 2012.

[16] HAARNOJA T, ZHOU A, ABBEEL P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor[C]//International Conference on Machine Learning. [S. l.: s. n.], 2018: 1861-1870.

[17] LEE A X, NAGABANDI A, ABBEEL P, et al. Stochastic latent actor-critic: deep reinforcement learning with a latent variable model[J]. Advances in Neural Information Processing Systems, 2020, 33: 741-752.

[18] RAZAVI A, VAN DEN OORD A, VINYALS O. Generating diverse high-fidelity images with VQ-VAE[EB/OL]. (2019-06-02)[2019-07-05]. https://doi.org/10.48550/arXiv.1906.00446.

[19] CAMUS V, MEI L, ENZ C, et al. Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019, 9(4): 697-711.

[20] MOONS B, DE BRABANDERE B, VAN GOOL L, et al. Energy-efficient convnets through approximate computing [C]//2016 IEEE Winter Conference on Applications of Computer Vision (WACV). [S. l.]: IEEE, 2016: 1-8.

[21] SHIN D, LEE J, LEE J, et al. 14.2 DNPU: an 8.1 TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks[C]//2017 IEEE International Solid-State Circuits Conference (ISSCC). [S. l.]: IEEE, 2017: 240-241.

[22] HOROWITZ M. 1.1 computing's energy problem (and what we can do about it)[C]//2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC). [S. l.]: IEEE, 2014: 10-14.

[23] HAN S, MAO H, DALLY W J. Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding[EB/OL]. (2016-02-15)[2016-12-15]. https://doi.org/10.48550/arXiv.1510.00149.

[24] MIYASHITA D, LEE E H, MURMANN B. Convolutional neural networks using logarithmic data representation [EB/OL]. (2016-03-03)[2016-03-17]. https://doi.org/10.48550/arXiv.1603.01025.

[25] MIGACZ S. 8-bit inference with tensorRT[C]//GPU Technology Conference. [S. l.: s. n.], 2017: 7.

（责任编辑　胡亚敏）