智能物联网终端自适应模型量化方法

王羽展，郭斌，王虹力，刘思聪

引用本文

王羽展，郭斌，王虹力，刘思聪. 智能物联网终端自适应模型量化方法[J]. 计算机科学，2023，50(11): 306-316.

WANG Yuzhan, GUO Bin, WANG Hongli, LIU Sicong. Adaptive Model Quantization Method for Intelligent Internet of Things Terminal [J]. Computer Science, 2023, 50(11): 306-316.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230300078

智能物联网终端自适应模型量化方法

王羽展　郭　斌　王虹力　刘思聪
西北工业大学计算机学院　西安 710129
(wyz_yy@mail.nwpu.edu.cn)

摘　要　随着深度学习与万物互联的快速发展,将深度学习与移动终端设备结合已经成为了一大研究热点。深度学习给终端设备带来性能提升的同时,将模型部署在资源受限的终端设备时也面临诸多挑战,如终端设备计算和存储资源受限,深度学习模型难以适应不断变化的设备状态等。基于此,研究了资源自适应的深度学习模型自适应量化问题。提出资源自适应混合精度模型量化方法,利用门控网络和骨干网络进行模型构建,以层为粒度寻找模型最佳量化策略,结合边端设备降低模型资源消耗。为了寻找最优模型量化策略,采取基于FPGA的深度学习模型部署。需要将模型部署在资源受限的边端设备上时,根据资源约束进行自适应训练,采取量化感知方法降低模型量化带来的精度损失。实验结果表明,该方法能够在保留78%的准确率的同时,降低50%的存储空间;同时,在FPGA设备上模型精度下降不超过2%,而能源消耗降低60%。

关键词:智能物联网;深度学习;模型量化;资源自适应;FPGA
中图法分类号　TP391

Adaptive Model Quantization Method for Intelligent Internet of Things Terminal

WANG Yuzhan,GUO Bin,WANG Hongli and LIU Sicong
College of Computer Science,Northwestern Polytechnical University,Xi'an 710129,China

Abstract　With the rapid development of deep learning and the Internet of Everything,the combination of deep learning and mobile terminal devices has become a major research hotspot. While deep learning improves the performance of terminal devices,it also faces many challenges when deploying models on resource-constrained terminal devices,such as the limited computing and storage resources of terminal devices,and the inability of deep learning models to adapt to changing device context. We focus on the adaptive quantization of deep models with resource adaptive. Specifically,a resource-adaptive mixed-precision model quantization method is proposed,which firstly uses the gated network and the backbone network to construct the model and partitioned model at layer as the granularity to find the best quantization policy of the model,and combines the edge devices to reduce the model resource consumption. In order to find the optimal model quantization policy,FPGA-based deep learning model deployment is adopted. When the model needs to be deployed on resource-constrained edge devices,adaptive training is performed according to resource constraints,and a quantization-aware method isadopted to reduce the accuracy loss caused by model quantization. Experimental results show that our method can reduce the storage space by 50% while retaining 78% accuracy,and reduce the energy consumption by 60% on the FPGA device with no more than 2% accuracy loss.

Keywords　AIoT,Deep learning,Model quantization,Resource adaptation,FPGA

1　研究背景

近年来,随着移动设备和可穿戴设备的普及,基于深度学习(如DNN,CNN和RNN等深度学习网络模型)的智能技术迅速发展,实现了无处不在的智能应用和服务。面向智能物联场景,将深度学习模型(简称模型)部署到资源受限且环境多变的物联网终端设备逐渐成为一种趋势。然而,强大的深度学习模型总是以巨大的计算和存储资源为代价。

随着硬件加速器(如GPU,DPU和TPU等硬件加速器[1])的发展,深度学习模型得到迅速发展,在追求更高准确率和更广应用面的同时,深度学习模型的复杂度也呈现爆炸式增长[2]。例如,在对图像数据进行处理时,经典的AlexNet[3]对于1 000分类、输入图像为224×224×3的任务有6 000万的参数量,需要进行7.29次浮点数计算,同时消耗240 MB的内存存储;VGG-16更是有1.44亿个网络参数,需要150亿次浮点型运算和528 MB内存开销;之后更加优秀的

到稿日期:2023-03-10　返修日期:2023-07-20
基金项目:国家杰出青年科学基金(62025205);国家自然科学基金(62032020,61725205,62102317)
This work was supported by the National Science Fund for Distinguished Young Scholars (62025205) and National Natural Science Foundation of China (62032020,61725205,62102317).
通信作者:郭斌(guob@nwpu.edu.cn)

GoogLeNet(Inception V1)，虽然在参数量、计算量和规模上有很大优化，但是它也有 500 万的参数量，计算量也到达 15 亿次浮点运算。深度学习模型虽能提高准确率，但在部署到智能物联网边端设备时也面临着挑战：1)移动端设备的计算、存储资源受到严格限制，往往无法直接在移动设备上部署深度学习模型；2)移动端设备应用场景十分复杂，终端设备计算环境具有动态变化的特性；3)深度学习模型的训练过程是基于特定数据集的知识学习过程，对终端复杂应用场景的适应能力差。

面对上述挑战，深度学习领域的科研人员提出了大量关于神经网络压缩的方法，例如剪枝、蒸馏、量化等模型轻量化方法，以将深度学习模型部署在资源受限的边端设备上。然而，上述方法仅在深度学习模型本身进行优化，未考虑因为硬件设备差异带来的影响，因此难以应对边端设备资源变化以及不同硬件设备带来的差异性。

本文提出了智能物联网边端设备深度学习模型自适应量化方法，旨在寻找资源限制条件下的深度学习混合精度量化策略。分别利用自适应的模型量化方法以及具有可编程性器件的 FPGA 进一步寻找混合精度量化配置，实现混合精度量化的最佳方案。本文的工作主要包括以下几个方面：

1)利用深度学习模型量化思想提出基于资源自适应的混合精度量化方法，综合利用资源自适应和模型量化设计，使得深度学习模型能够在不同资源约束下给出合理的量化方案。

2)为进一步实现模型所提出的“混合精度”方案，采取量化感知方法对 1)中所建立的模型进行优化，并在 FPGA 上实现混合精度量化的方法，利用 FPGA 对神经网络的加速和硬件可编程性来探索混合精度带来的模型存储、推理精度的优化，从而找到将模型应用到智能边端设备的最优方案。

3)为验证所提方法的有效性，分别在树莓派和 FPGA 上进行实验，结果表明该方法能够在树莓派保留 78%的准确率，同时降低 50%的存储空间，且能够在 FPGA 设备上保证模型精度下降不超过 2%，并降低 60%以上的能源消耗。

2 相关工作

2.1 深度学习模型轻量化相关技术

为了实现更为复杂的任务，深度学习模型规模不断扩大，随之而来的是计算量和推理时间的剧增，深度学习模型的压缩和加速方法也不断涌现。深度学习模型的压缩，往往指的是采取一定的方法减小神经网络结构和模型中的参数规模。下面介绍一些常用的深度学习模型压缩方法。

1)参数剪枝(Parameters Pruning)：基于神经网络具有不重要或不必要参数的假设，通过剔除这些冗余参数而保留对模型精度影响较大的参数，来实现模型参数量和计算量的有效缩减。Luo 等[4]提出了一种过滤器级剪枝方法——ThiNet，使用下一层的特征映射来指导当前层中的过滤器修剪，通过贪婪算法和最小化特征映射的重构误差来选择通道。

图 1 直观地展示了突触剪枝和神经元剪枝这两种方式。相比于非结构化剪枝，结构化的剪枝更受底层和硬件的支持。目前，参数剪枝是一种最为简单、有效和广泛利用的模型压缩方法。

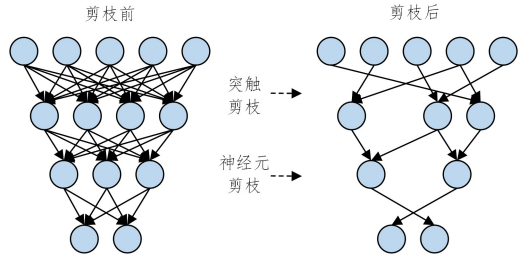


图 1 突触剪枝和神经元剪枝

Fig. 1 Synaptic pruning and neuron pruning

2)参数量化：通过减少表示每个权重所需的比特数来压缩原始网络，实现模型复杂度的降低。Gupta 等[5]的工作在基于随机取整的 CNN 训练中使用了 16 位表示，显著减少了内存使用和浮点运算，且精度损失很小。Vanhoucke 等[6]的研究表明，在不显著降低分类精度的情况下，对参数进行 8 位量化可以显著加快速度。如图 2 所示，在一些模拟量化中，量化后的模型参数以低精度存储，但操作(如矩阵乘法和卷积)是以浮点运算进行的，所以量化后的浮点运算之前需要进行反量化。因此，这种推理方式不能完全获得低精度推理带来的收益(因为其过程包含了部分全精度推理)。

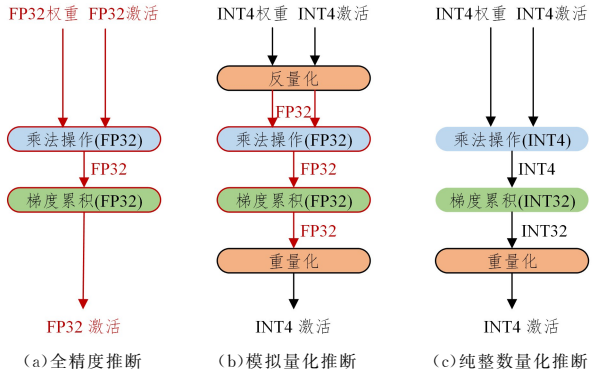


图 2 不同推断模型示意图

Fig. 2 Schematic diagram of different training methods

除了上述在深度学习模型上采取措施来减少推理时间外，还可以采取硬件加速的方式：使用支持并行计算的设备，如 GPU 和 FPGA 等深度学习加速器；采取稀疏矩阵、稀疏计算来减少模型参数，从而加速深度学习推理；等。这些方法虽然可以快捷、方便地将轻量化方法应用在以往工作之中，但前人的工作通常是有上限的，需要设计更加有效、轻量的神经网络结构。

2.2 基于 FPGA 的深度学习技术

深度学习模型是以数据为核心，包含了大量的数据计算，对以顺序计算方式的 CPU 提出了巨大的挑战。在深度学习领域，很多应用场景对功耗、延迟等性能有着一定的要求，传统的 CPU 难以满足这些要求。因此，研究者在应用深度学习技术时，通常会使用其他硬件加速。目前，主流的硬件加速器有 3 类：GPU(Graphics Processing Unit)，ASIC(Application-Specific Integrated Circuits)和 FPGA(Filed Programmable Gate Array)。本文主要围绕 FPGA 加速特点以及其上的深度学习技术进行介绍。

FPGA 又称现场可编程门阵列，它允许无限次编程，即

开发人员可以根据自己的需要，通过可编程的连接将 FPGA 内部的逻辑单元连接起来，并使用小型查找表来实现组合逻辑等各种功能[7]。

FPGA 的加速设计属于硬件结构适应算法，而 GPU 属于算法适应硬件结构，由于 GPU 硬件结构是固定的，因此在算法设计时，往往需要根据 GPU 硬件结构来调整深度学习算法的设计。同时，与 GPU 相比，FPGA 通常拥有更加有效的能源效率[8-9]；GPU 通常更适合深度学习训练阶段，推理阶段要求硬件加速器拥有低延迟、高性能、低功耗等特性，因此 FPGA 更适合深度学习的推理阶段。ASIC 加速在性能上通常优于 FPGA，这是因为 ASIC 是针对特定算法的专用电路。但是 ASIC 开发周期往往更长，需要经过诸多物理设计和验证，面对正处于高速发展的深度学习模型，其不能够满足广泛的应用场景需求。综上所述，FPGA 有着独特特性，与 GPU 和 ASIC 相比，在深度学习加速方面拥有着自己独特的优势。

近年来，关于面部图像识别的应用场景越来越多，如人脸识别[10]、字符识别[11]和自然场景识别[12]等。得益于 FPGA 作为硬件载体的稳定性和速度、产品低价格和低功耗等特点，上述图像识别场景都成功地部署在 FPGA 上。这些应用和技术正在为生活中的各方面带来便利。除此之外，在语音处理方面，往往需要产品拥有灵活性和移动性，因此常在 FPGA 上部署语音识别模型来满足需求；在文本处理方面，以 NLP (Natural Language Processing) 语义计算框架为典型代表，将自然语言处理结合深度学习应用在 FPGA 已然成为一大研究热点，被广泛应用在各种翻译机器、情感分析等产品中。

除上述研究方向以外，FPGA 还能用于网络安全、智能控制、目标跟踪等现代化智能场景。在广泛的应用背景下，也有很多应用对硬件设备的性能、功耗和鲁棒性等方面有着严格的要求。因此，在将深度学习模型应用到 FPGA 时对其进行轻量化，将使得深度学习部署到智能边端设备时更加现实。

3 资源感知自适应模型量化方法

针对深度学习模型压缩后仍可能无法部署在资源受限的终端设备上，同时在受限的设备资源约束下难以适应多变的环境的问题，提出了一种面向智能物联网终端设备的资源自适应的深度学习方法，旨在采取策略满足不同资源约束下的深度学习任务。系统框架如图 3 所示，整体可分为两个模块：结合智能终端设备的不同资源约束以及硬件支持，选择合适的量化方法，并调整模型结构；根据资源状态进行动态自适应训练，对于不同终端设备，寻找最优量化配置，从而最大程度地节约资源，加速推理。

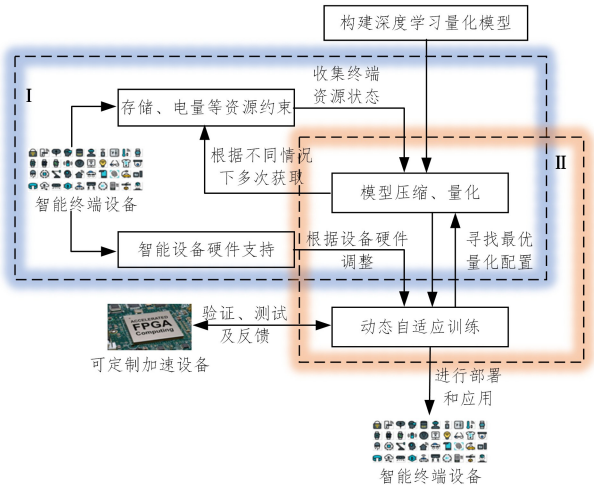


图 3 智能物联网终端设备资源自适应量化方法

Fig. 3 Adaptive model quantization method for intelligent Internet of Things terminal

本文将在通用的深度学习模型 ResNet 以及 MobileNet 的基础上进行设计，使其能够针对不同资源约束训练出与之对应的深度学习模型，并进行自适应训练和混合精度量化等方法，减少重训练占用的计算资源。

3.1 资源自适应量化模型的设计流程

基于深度学习的自适应量化方法的相关框架主要由 3 个模块组成，分别是主网络(骨干网络)模块、量化选择门控网络模块和自适应训练模块。

ResNet 和 MobileNet 分别是以残差块和 Bottleneck 块构建而成的网络，主要计算操作为卷积操作、激活以及全连接(线性操作)。本文的系统框架由 ResNet 和 MobileNet 构成(因为两种网络各具代表性，为后续实验而设)，如图 4 所示。

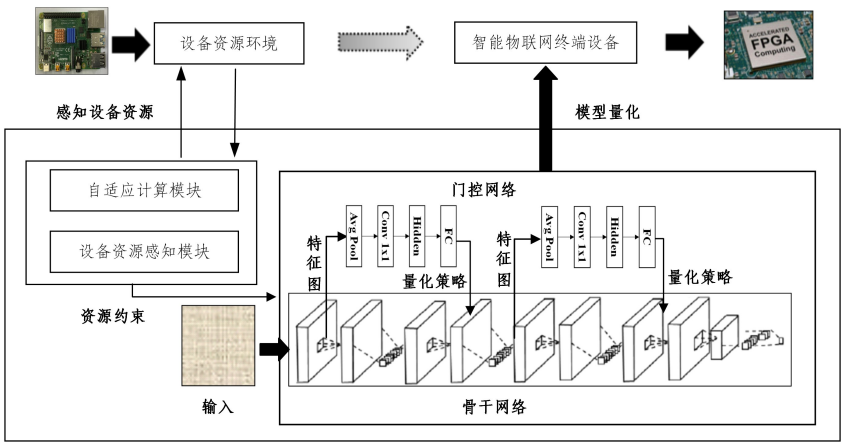


图 4 系统框架图

Fig. 4 System framework diagram

1）骨干网络模块

骨干网络由 ResNet 和 MobileNet 两种网络构成。ResNet 网络残差单元能够以跳层连接的执行实现，即将一个残差块的输入直接与输出加在一起，然后激活。因此，残差网络可以很好地使用主流的微分深度学习框架实现；同时，残差网络很好地解决了深度学习网络退化的问题。Mobile-Net 最初即为移动端和嵌入式端深度学习应用设计的网络，在 CPU 上也能达到理想的速度要求，其最大的特点即为轻量化，非常符合本文量化的目标，能更好地部署在智能物联网终端设备中。这两种网络符合本文设计与实现的特性，故将其作为主干网络，主要使用 ResNet38，ResNet74 以及 MobileNetV2。

2）门控量化网络模块

门控量化网络需要一个能够根据输入特征信息生成量化策略的网络，首先需要该网络是轻量级的，因为本文设计方案是面向资源受限的物联网终端设备；其次，由于特征图在网络层间的传输是顺序进行的，因此需要一个能够捕获这种层间传递的序列化信息。根据文献[13-14]的描述，长短期记忆网络（Long Short-Term Memory，LSTM）能够捕获系列化冗余信息，并且由于其参数共享设计，LSTM 的计算成本仅占其对应残差块的 0.04%。因此，本文采用 LSTM 为门控网络进行量化策略的生成。

3）自适应训练模块

该模块包含两部分：网络动态学习自适应和设备资源学习自适应。

（1）网络动态学习自适应：神经网络模型在训练阶段中，往往会采取一定的学习率，根据模型的误差逐渐调整参数，从而达到一个最优的知识表征效果。但学习率的设定往往需要大量的实验，不同深度学习任务所采取的学习率也存在差异。若学习率过大，在训练中后期将难以学习到更细粒度的信息，从而达不到最佳收敛效果；若学习率过小，则会耗费大量计算资源，同时收敛速度也将降低，甚至会导致过拟合等问题。

（2）设备资源学习自适应：由于资源受限的物联网设备往往处于一个动态变化的资源环境下，如果能够采取一定的措施，使得网络在训练阶段学习到相应的信息，将在模型部署环节获得更好的模型效果。因此本文采用自适应训练策略，根据网络训练过程以及资源动态调整学习率，从而达到最优的效果。

3.2 基于 LSTM 的量化门控网络

基于 LSTM 的量化门控网络在训练过程中学习并建立了一个网络控制器，用来控制网络以何种方式（何种策略）进行推理。经典残差网络 ResNet 由多个残差块连接而成，为了得到较高的特征学习率以及准确性，其网络结构通常十分巨大。图 5 为 ResNet 残差块示意图。

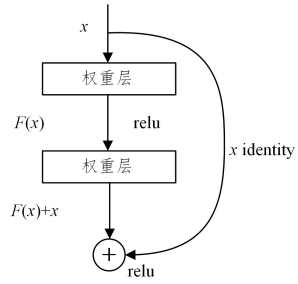


图 5 ResNet 残差块示意图

Fig. 5 ResNet residuals block diagram

ResNet 通过使用多个有参数的层来学习输入输出的残差表示，与传统的 CNN 网络（如 VGG 和 AlexNet 等）使用有参数层来直接学习输入和输出之间的映射不同。ResNet 由多个残差块组成，然而在真实情况下，只有当输入样本比较复杂时，才需要如此深的网络结构。当输入样本较为简单时，往往在识别初期就获得了非常高的准确率，但由于网络结构的设定，仍需进行所有层推理，这就会导致网络冗余。文献[13-14]分别提出了 SkipNet 和细粒度跳变网络，验证了 LSTM 能够获得一个较好的网络控制效果。受其启发，本文采取一个单门控 LSTM 网络进行网络控制，并生成量化策略。

如图 6 所示，LSTM 门控网络由一个平均池化层、一个 1×1 卷积层、一个隐藏层（即 LSTM 结构层）以及一个全连接层构成，其中隐藏层和全连接层采取参数共享形式。全连接层输出一个概率向量，即为量化位宽的选择概率：根据预设（资源获取方式将在 3.3 节详细描述）的待选位宽（具体选择方式将在第 4 节详细说明）向量生成一个对应的概率向量，为每一层网络选择不同的量化位宽，即混合精度量化，从而实现资源效率和准确率之间的最佳权衡。

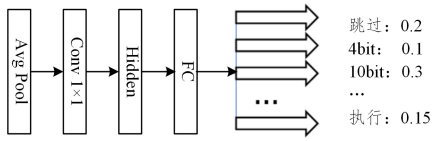


图 6 LSTM 门控网络示意图

Fig. 6 LSTM gated network diagram

3.3 资源自适应训练方法

自适应训练方法根据智能物联网终端设备资源预算水平进行设计，在门控网络的基础上使得在训练时能够及时根据资源预设调整主干网络的量化策略，以满足资源的限制。资源自适应训练方法的框架如图 7 所示。

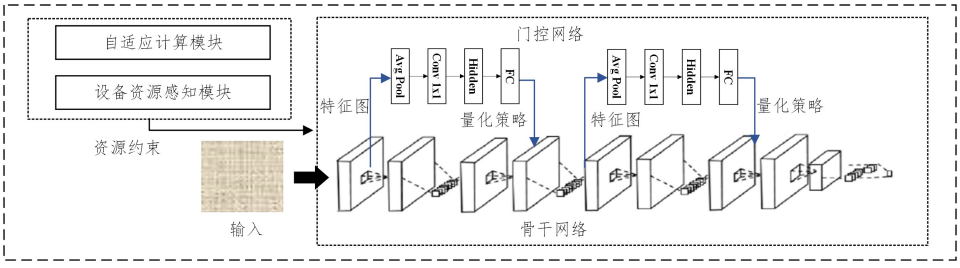


图 7 资源自适应训练方法

Fig. 7 Resource adaptive training methods

1)量化网络训练

如 3.1 和 3.2 节所描述,量化网络是由主干网络和门控网络构成,主干网络在同一量化级别上使用一套参数,而门控网络需要根据主干网络反馈回来的量化效果快速调整量化策略。具体做法是:在训练阶段,同一批数据在通过门控网络的同时,将其得到的损失值传递给主干网络,一旦某一门控网络的向前传播导致资源预算较低,就会立刻向后传播,这可能影响到其他层的量化策略。训练目标是提高预测精度(即损失最小),同时最小化资源成本。损失函数如式(1)所示:

$$\min \sum_{i=1}^{n} \mu_1 Acc_{loss_i} - \mu_2 E_i \tag{1}$$

其中,n 表示主干网络层数,μ_1 和 μ_2 分别表示计算精度和资源预设权重系数,E_i 表示资源效率。可以看出,对每一层网络进行误差累积,然后计算总误差与资源预设的差,当误差趋近于 0 时,表示当前模型对资源的适应性很高。

2)骨干网络的量化感知训练

共享骨干网络也是非常复杂的,因为骨干网络需要考虑门控网络在不同量化精度下的所有训练。如果训练方案不考虑量化精度不同导致精度损失的多样性,将导致骨干网络的性能下降。而传统的骨干网络量化后微调方案[15]对量化推理精度的校准也存在不足,而且量化后微调方案需要大量的重训练计算资源。因此,有必要将训练策略与量化的前向传播协同设计,以最大限度地减少骨干网络的精度损失。具体地,训练过程中的反向传播过程使用全精度权重来计算规则梯度下降。在前向传播过程中,逐步利用不同的量化级别(例如,在一个或多个层上,2 位、4 位、6 位、8 位、…、32 位)来量化骨干网络的权重和激活,用于训练一个量化感知骨干网络。正向传递中的量化操作模拟了不同粒度的量化推理效果,此环节遵循了课程学习思想,进行了具体的递进训练[16],加入不同噪声强度的不同量化噪声,提高了整体精度。具体地,将从弱(如 2 位)到强(如 32 位)的量化噪声添加到每个训练迭代的前向传递中。训练结束后,只需要存储训练好的全精度权重,即可在运行时应用不同的量化级别。

3)上下文资源感知

本文利用上下文感知块捕捉给定物联网平台的资源可用性变化,然后在部署阶段选择合适的策略进行推理。预定义的资源指定期检查资源的可用性。具体来说,上下文感知模块通过操作系统指令获取平台资源信息,包括 CPU/GPU 使用情况、内存使用情况、缓存、缓冲区、能源供应等。例如,使用指令(例如"cat /proc/stat")来获取设备的 CPU 使用率,然后据此输出规范化的资源级别。

在训练阶段,设备资源往往是充足的,这就需要根据真实物联网设备资源状况预设一系列可能的存在。为处理给定物联网平台的不可知和动态资源约束,需将资源可用性规范化为 4 个离散级别的缓存可用性(即缺少 25%、中等 50%、足够 75%、富裕 100%),原因是内存预算是硬约束,而缓存可用性是影响深度模型执行性能的关键因素,它主要影响缓存命中率,从而影响可达到的内存带宽和峰值计算性能。

本文利用著名的屋顶线模型[17](Roofline)来表示量化模型的性能和资源效率。如图 8 所示,对应于不同的资源级别(例如缓存可用性)。具体而言,某一平台屋顶的峰值计算性能和内存带宽由智能物联网设备的配置决定。

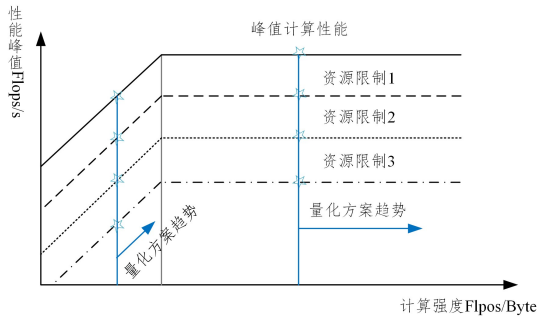


图 8 资源带宽下的屋顶线模型

Fig. 8 Roof line model under resource bandwidth

量化深度学习模型的计算强度度量了处理器与缓存之间的通信流量,反映了资源效率。当量化模型的计算强度 C/S 遇到顶板平坦部分时,模型性能受到计算限制,量化策略需要减少更多的计算量。如果它击中屋顶的对角线部分,模型的性能最终会受到资源(内存)的限制,需要更多地减小参数/激活大小。因此,本文建立自适应函数,如式(2)所示:

$$y = G(f(n, x, E), \eta) \tag{2}$$

其中,y 为网络量化策略,是一个从输入 x 到量化策略 y 的映射;G 为整个模型结构的状态,包含骨干网络和门控网络所有信息;f 为当前层(或残差块)的状态;n 为网络层信息;x 为输入特征;E 表示当前资源约束状态;η 表示当前学习率。在推理过程中,门控网络通过 Softmax 函数投影到待选位宽,获得对应的概率向量,基于该向量,指导每层做出相应的量化选择。

4 基于 FPGA 的深度学习模型的设计

深度学习模型自适应量化方法在物联网智能终端设备上部署时,其性能往往会因为受到终端设备的支持而发生变化。特别是对于混合精度量化方法,通常设备对任意位混合精度量化策略的支持度较低,这给自适应混合精度量化方法的真实部署带来了困难,而 FPGA 具有可编程性,能够很好地解决此问题。

基于 ResNet 网络的混合精度量化策略方法确定后,需要将网络结构和生成的模型应用到 FPGA 上,并评定量化策略对模型在智能终端设备上的真实性能产生的影响。由于 FPGA 设备的特殊性,将深度学习模型应用到 FPGA 上需要经过 3 个环节:1)将深度学习网络结构代码重构成能被 FPGA 应用的 Verilog 程序;2)将深度学习模型参数文件(如 pth,onnx 等)转换成 2 进制数格式,以供 FPGA 读取;3)将深度学习量化策略转换成可以被 FPGA 执行的数字逻辑电路程序。本节对整个网络到 FPGA 构建框架进行说明和建模,对设备选择、整体框架设计、衡量指标等进行完整的阐述,最后通过实例验证方法的可用性。

4.1 基于 FPGA 的混合精度量化的构建

不同于基于 Linux 或者 Windows 操作系统的终端设备,FPGA 拥有其专用开发程序语言——Verilog HDL 或 VHDL 寄存器传输级(RTL)等硬件描述语言。使用硬件描述设计程序时,往往需要一定的数字逻辑电路基础,并要求开发人员较好地掌握计算机组成原理等相关知识,开发门槛较高,难度较大,但对设备资源利用率高。因此催生了一种从 C/C++到硬件描述语言的工具 HLS(High-level Synthesis),其将高级

程序语言转化到 RTL 实现，并将其综合到现场可编程门阵列（FPGA）上，以实现软件的硬件加速效果。本文采用 Verilog HDL 硬件描述语言来完成深度学习的构建，通过框架设计、量化策略选择来进行混合精度量化。

1）框架设计

第 2 节中所提出的自适应混合精度模型量化策略首先在 GPU 中进行模型预训练和自适应训练，得到一个基于 ResNet 以及应对 Cifar-10 数据集的模型，将其应用于 Raspberry Pi 上进行测试。由于树莓派是基于 Linux 系统，有其自身操作系统，可以对其资源进行约束（如使用 Docker 或 C 语言进行资源限制），从而得到不同资源约束下的量化策略以及模型性能。而对于 FPGA，无法使用程序进行资源约束，其适合高性能、低延迟推理，而不适合高计算量的训练阶段，因此将采取第 3 节所训练得到的模型及其量化策略进行 RTL 级验证，具体包括从深度学习框架到 Verilog HDL、仿真验证以及 FPGA 测试。基于 FPGA 的模型量化框架如图 9 所示。

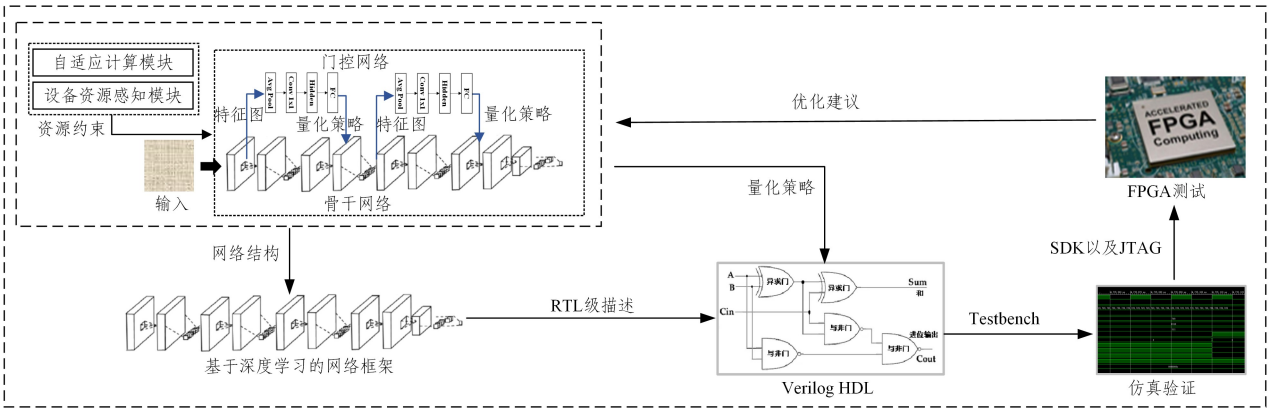


图 9 基于 FPGA 的模型量化框架

Fig. 9 Model quantization framework based on FPGA

2）基于 FPGA 的图像识别

由于 FPGA 相当于大量逻辑电路的集成，因此 FPGA 中通常不缓存数据，而是存储推理阶段需要的程序（指令）以及模型当前需要处理的数据。若将数据集 Cifar-10 或 Cifar-100 使用缓存、连接 DDR 或其他存储器进行存储，将带来设备无法承受的存储开销以及寻址访存开销，从而导致 FPGA 性能大幅降低甚至无法正常工作。因此本文采用 MINIST 数据集，该数据集被广泛用于衡量轻量级模型的性能，即进行手写体数字识别任务。

除此之外，FPGA 的数据输入需要外界输入（连接摄像头等设备进行输入，或使用程序进行初始化控制输入等）；同时，FPGA 不像 Linux 或 Windows 操作系统那样拥有自己的图形化界面以及终端环境，需要使用集成逻辑分析器（Vivado 的在线分析仪——Integrated Logic Analyzer，ILA）监控程序或 LCD/HDMI 显示屏获取其数据信息，信号（即数据信息，在逻辑电路中任何数据都以信号的方式呈现）在 FPGA 中以高低电平信息呈现，因此需要将其转换为直观数据或图像，以便获得测试结果。

具体框架如图 10 所示，采取 OV5640 系列摄像头以及 AN430 和 RGBLCD 进行数据采集和结果显示。设备的具体信息将在 4.2 节进行阐述。

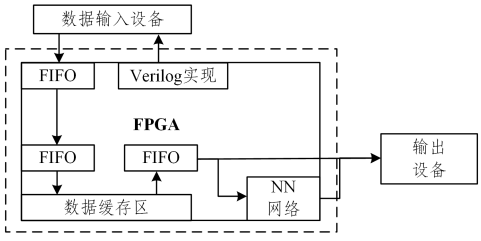


图 10 基于 FPGA 的图像识别框架

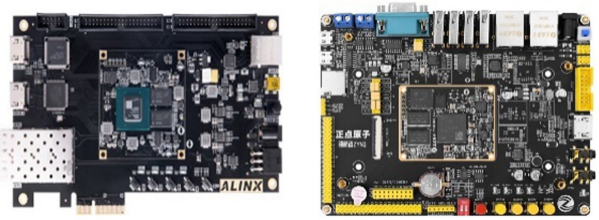
Fig. 10 Image recognition framework based on FPGA

4.2 FPGA 以及设备配置

本小节将围绕 FPGA 实验所用设备以及设备配置等方面进行阐述。

1）FPGA 硬件设备

本文采用 Artix-7 200T 核心板以及 ZYNQ-7020 开发板，其实物如图 11 所示。Artix-7 200T 和 Zynq-7020 均属于高性能开发板，拥有大量逻辑单元以及运算单元，能够应对较为复杂的深度学习任务。



（a）Artix-7 200T （b）Zynq-7020

图 11 两种常用 FPGA 器件

Fig. 11 Two kinds of commonly used FPGA devices

两种设备的具体参数如表 1 所列。两种设备各有优势，Artix-7 200T 拥有更多的逻辑单元，对于复杂的逻辑运算有着巨大的优势。

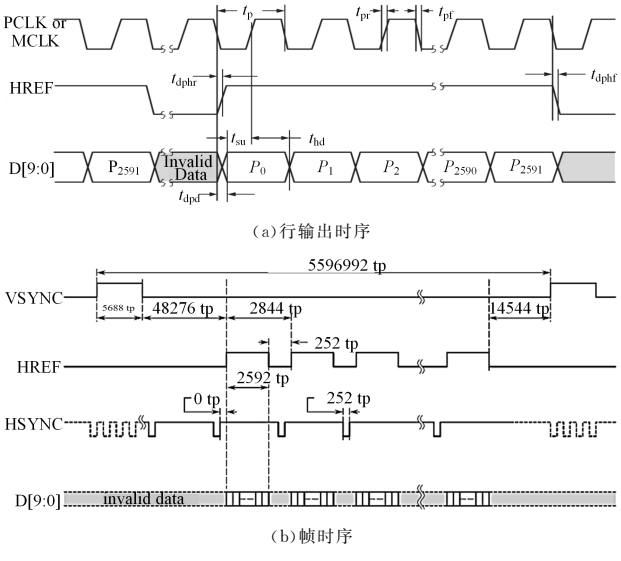
表 1 FPGA 参数

Table 1 FPGA parameters

核心板参数	Artix-7 200T	Zynq-7020
逻辑单元	215 360	85 000
查找表	33 650	53 200
触发器	269 200	—
寄存器	—	106 000
Block Ram	105.12 Mbit	4.9 Mbit
CPU 内核	无 PS 端	双核 ARM Cortex-A9
工作温度	−40℃～85℃	−40℃～85℃

2)数据输入、输出设备

对于 Artix-7 200T 开发板,使用 AN5640 摄像头以及 AN430 LCD 显示屏;对于 Zynq-7020,使用 OV5640 双目摄像头以及 ATK-7016 显示屏。AN5640 与 OV5640 双目摄像头均属于 OV5640 系列 CMOS 图像传感器,其为 500w 像素级摄像头,支持更高的分辨率,其时序如图 12 所示[18]。



(a)行输出时序

(b)帧时序

图 12 OV5640 摄像头时序图

Fig. 12 OV5640 camera sequence diagram

OV5640 摄像头的输出时序共有 3 种数据线路:1)像素时钟 PCLK 以固定频率进行置位、复位(置位为 1,复位为 0),像素时钟以时钟下降沿为触发条件;2)HREF 表示行参考信号,用于指示采取像素的有效时刻,以上升沿为有效触发条件;3)数据线路即 D[9:0]共 10 路数据,分别为 D0 到 D9,其具体引脚信息根据不同设备而确定。因此当 PCLK 和 HREF 分别为下降沿和上升沿时,下一刻像素数据开始为有效数据,在此期间需要进行数据收集、缓存、转换以及向显示设备进行输送,直到 PCLK 和 HREF 均为下降沿时,数据线路数据再次为无效数据。

帧时序共有 4 种数据线路:1)VSYNC 脉冲信号,该信号在每一帧数据之前拥有一个脉冲信号,代表这帧数据开始传输,一帧则表示 LCD 显示的一个画面;2)每一行数据都会有一 HREF 脉冲,其中 HERF 高电平期间传输这一行数据;3)HSYNC 为行同步信号,表示扫描一行的开始,在 HERF 高电平以及 HSYNC 同为高电平时,数据是有效的;4)数据线路即 D[9:0]与行输出时序中数据信号一致。

数据采集有多种方式,如 RGB565,RGB555 和 RGB888 等。RGB565 和 RGB555 为 16 位像素,而 RGB888 为 24 位像素。如图 13 所示,本实验中采取 RGB565 像素编码格式,OV5640 每个摄像头拥有 8 位像素数据输入,因此需要将两个相邻时钟周期读取到的数据进行拼接,得到 Data[15:0]数据。以 Data[15:11]为 R(red)数据,Data[10:5]为 G(green)数据,Data[4:0]为 B(blue)数据,进行图像生成。输出设备 AN430 和 ATK-7016 均属于 LCD 显示屏,其标准分辨率分别为 480×320 和 1 024×600,也可根据用户自定义分辨率显示。

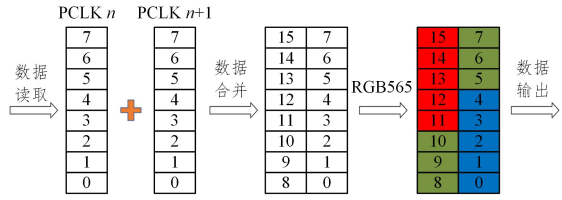


图 13 RGB565 读取示意图

Fig. 13 Schematic diagram of RGB565 read

5 实验验证

5.1 资源自适应量化模型设计验证

本文将使用两种广泛使用的模型即 ResNet 和 MobileNet-V2 进行训练,采用两个广泛使用的数据集 Cifar-10 和 Cifar-100 进行模型测试。实验包含混合精度量化验证和自适应感知验证。

1)混合精度量化验证

该部分使用 ResNet20 和 ResNet38 构建主干网络,训练阶段使用 NVIDIA GeForce RTX 3080 GPU 平台。在智能物联网终端设备 Raspberry Pi 4B 上进行测试,该设备基于 Linux 操作系统,是一款常用的终端设备,其实物如图 14 所示。

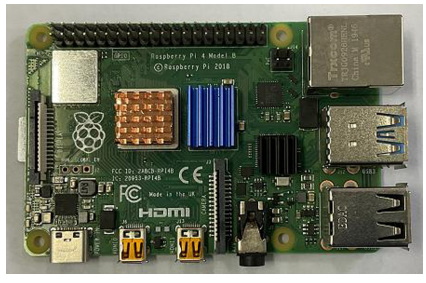


图 14 Raspberry Pi 4B 实物图

Fig. 14 Physical image of Raspberry Pi 4B

Raspberry Pi 4B 拥有 1.5 GHz 主频的 Arm Cortex-A72 处理器,能够处理大多数深度学习任务。本实验将对其计算资源(主要为 CPU 计算资源)进行限制,采取 C++语言执行"空取指"操作,从而实现资源约束的效果。具体方法如算法 1 所示。

算法 1 资源约束

输入:取指向量大小 v

输出:输出剩余 CPU 可用资源

```
1. //初始化向量
2. std:vector<int>v;
3. //初始化睡眠时间 t
4. while(true)
5.     //获取当前 cpu 使用率
6.     get.os("/proc/stat cpu")
7.     for(auto &a:v)
8.       {//进行"空取指",从而占用 CPU 资源}
9.     endfor
10.    //睡眠 tμs
11.    usleep(t)
12. endwhile
```

通过“空取值”与“睡眠”交替执行，实现计算资源约束。以 Raspberry Pi 4B 为例，当取指向量大小为 1×10^7 时，计算资源占比约 25%；取指向量大小为 1×10^5 时，计算资源占比约 10%。因此对于获得更高的资源约束时，可采取多线程执行资源约束程序，从而无需多次修改取值向量，避免因修改后编译程序导致额外的资源占用。

首先对两种模型在 Raspberry Pi 上的性能进行测试，测量其在模型运行期间占用 CPU 资源的取值范围，从而为自适应量化模型的资源约束范围提供参考。该部分使用 ResNet20 和 ResNet38 构建主干网络，训练阶段使用 NVIDIA GeForce RTX 3080 GPU 平台。

如图 15 所示，整体上 ResNet38 资源使用率要高于 ResNet20，因为其网络更加复杂。其资源使用范围均在 50%～95%之间，但是为了使得模型更具有代表性，对 ResNet38 采取资源约束[18%，38%，54%，77%，82%，95%]（表示设备剩余资源）、ResNet20 采取资源约束[18%，26%，49%，67%，85%]混合精度量化策略验证。

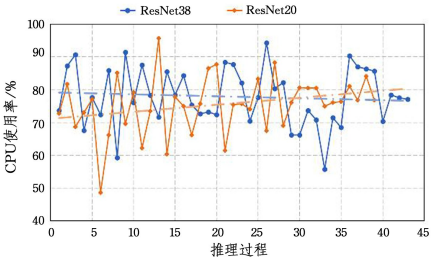
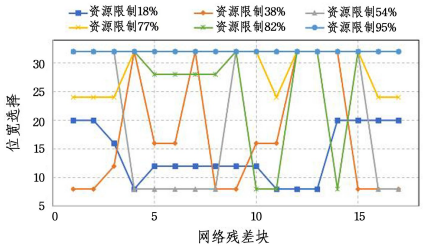


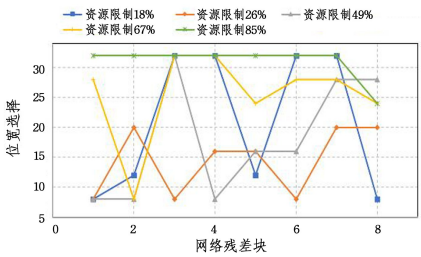
图 15 ResNet20 和 ResNet38 推理过程中的 CPU 使用率

Fig. 15 CPU usageduring ResNet20 and ResNet38 reasoning

如图 16 所示，在不同资源约束下自适应量化模型选择出混合精度，可以得出如下结论：(1)位宽选择策略随着资源约束的提高，更倾向于选择高 bit 位方案；(2)在不同残差块分别呈现出一定的规律性，如在模型残差块前期（即开端），ResNet38 和 ResNet20 分别在第四残差块和第三残差块呈现出保留全精度的现象，在残差块后期（即末尾）呈现出普遍下降趋势。



(a)ResNet38 位宽选择



(b)ResNet20 位宽选择

图 16 ResNet 位宽选择

Fig. 16 ResNets bit wide selection

考虑上述特性有如下原因：(1)在模型初期，为了保持预测精度，往往会保留高位层来提高模型的整体性能；(2)为了满足模型性能约束，即式(1)所示优化函数，模型在残差块后期选择了较低位宽，保留了低位层来降低模型计算量，从而减少资源需求。除此之外，还采取了 ResNet101 与 ResNet74 模型作为主干网络进行测试，得到了一致的结论。总之，实验结果证明该方法能够根据资源约束生成应对其合适的量化策略，为后续部署到智能物联网终端设备以及 FPGA 上提供了一定的量化指导方案。

2)自适应量化模型效果

上述混合精度量化方法能够根据设备资源状态选择不同的量化策略，从而降低模型计算量，使得深度学习模型能够更好地部署在智能物联网设备上。为了验证模型性能和量化方案的有效性，根据 3.4 节所属缓存可用性 4 个离散方案（缺少 25%、中等 50%、足够 75%、富裕 100%），分别在 Cifar-10 和 Cifar-100 数据集上对深度学习模型 ResNet38 和 ResNet74 的性能进行测量，实验结果如表 2 所列。

表 2 ResNet38 和 ResNet74 量化模型参数

Table 2 ResNet38 and ResNet74 quantized model parameters

模型	资源约束/%	参数量（权重）/MB	参数量（激活）/KB	推理时间/ms	Top-1 准确率/%
ResNet38	100	2.3	11.4	31.2	93.8
	75	1.9	8.6	30.9	83.2
	50	1.2	6.2	31.8	81.9
	25	0.8	3.5	31.2	79.2
ResNet74	100	4.8	22.7	62.1	92.7
	75	3.5	17.2	61.8	89.9
	50	2.1	12.9	61.2	84.8
	25	1.5	8.5	61.1	78.7

根据表 2 和图 16 结果可知，当训练阶段资源约束降低时，深度学习模型倾向于选择更低的量化位宽，从而降低参数所占内存量，以适应设备低资源状态；参数量降低，推理时间理应缩短，但由于设备计算资源匮乏时，计算能力也会相应下降，因此推理时间呈现一个几乎不变的状态；而随着计算资源的降低，ResNet38 和 ResNet74 准确率最高下降了 14.6%和 14.0%，最低还保留有 78%以上的准确率。除此之外，还对比了门控网络相较于骨干网络的开销，如图 17 所示。门控网络相对于骨干网络，内存开销占比分别为 4.11%和 8.11%，同样地，MobileNet-V2 上的门控网络仅占主干网络的 5.56%。

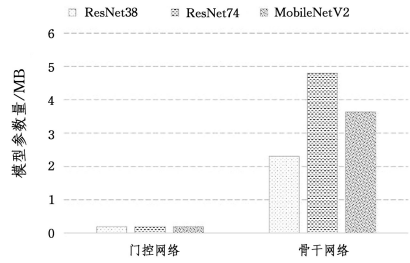


图 17 门控网络和骨干网络的内存开销

Fig. 17 Memory overhead of gated network and backbone network

为了更好地体现自适应计算给模型带来的计算时间上的

优化，将不采取自适应计算的 Resnet 网络在 Raspberry Pi 4B 上进行运行时间的测量，并记录其在不同计算资源限制下的运行时间，结果如图 18 所示。

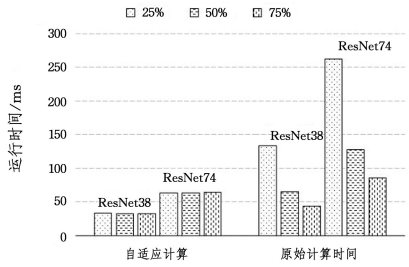


图 18 不同 ResNet 网络在不同资源限制下的运行时间

Fig. 18 Runtime of different ResNet networks with different resource constraints

综上所述，门控网络的设计为整个模型设计仅仅带来了微不足道的存储和延迟增加，但是却带来了巨大的模型性能提升。

5.2 基于 FPGA 的混合精度量化验证

本小节实验验证部分将对第 2 节混合精度量化策略进行验证，主要分为相关配置、仿真验证、FPGA 测试。

1)实验相关配置

为了获得更加真实的量化效果，以及考虑到 Verilog 对卷积操作的计算优势，本实验采用 CNN 作为骨干网络，通过自适应混合精度量化策略来对不同层进行量化。由于 FPGA 对浮点数计算性能较差，更适用于定点数运算，因此首先将深度学习模型从浮点数模型转化为定点数模型，如式(3)所示：

$$M_i' = \frac{M_i \times Coeff_i}{\max\{|\Omega_i|\}} \tag{3}$$

其中，M 表示原始模型参数，$Coeff$ 为定点缩放系数，Ω_i 表示模型第 i 层网络参数的最大和最小值。当把模型转化为 1 位定点数(即定点缩放系数为 1)时，参数范围为[−1,1]，其模型结构未发生改变，逐层量化。以 CNN 为例，模型存储占用减少到原始模型的 48.2%，是由于在训练时模型文件中保存了训练阶段的相关信息，进行定点量化时保存数据只保存有效数据，因此降低了模型存储量。

将模型转换为定点数，是为了降低存储和方便 FPGA 计算。但以小数的形式无法使用 Verilog HDL 语言进行描述，需要将其量化(Rescale)到整数形式，以二进制数字形式存储。

2)仿真验证

设置相关配置后需要进行仿真验证，即通过仿真、时序分析等手段检验设计正确性。在 FPGA 开发流程中，其作为硬件描述语言设计的一大环节，需要尽可能考虑到上板测试可能发生的所有可能，从而保护设备不会因为人为操作而损坏。

仿真验证具体可分为功能验证和时序验证两个部分。功能验证即为前仿真，可利用 Modelsim 和 Vivado 等仿真工具进行仿真，检验设计功能是否正确。而时序仿真是指在 FPGA 进行引脚约束、综合、布局布线之后，根据生成的时序报告进行分析，找出不符合约束的路径。本实验中，进行功能验证仿真，具体设计如图 19 所示。

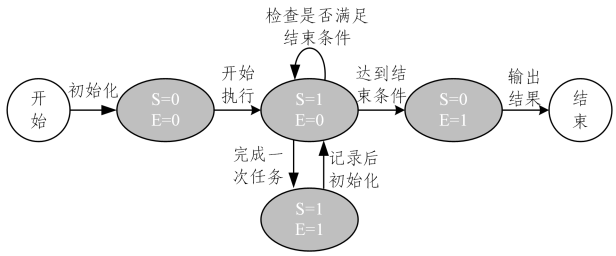


图 19 仿真状态机

Fig. 19 Simulation state machine

仿真设计为状态机的形式，由于 FPGA 以及 Verilog HDL 与传统的程序设计语言 C/C++和 Python 等高级程序设计语言不同，其采用并行设计，无法根据程序指令的先后顺序来控制程序的执行，因此需要有限状态机(也称为有限自动机，Finite Automaton，FA)来进行程序执行控制。其中，S 表示 Start，即标识程序为运行状态；E 表示 End，即标识程序为暂停或中止状态，有限状态机的具体状态信息如 2 所列。

表 2 有限状态机状态含义

Table 2 Finite state machine state meaning

Start	End	含义
0	0	最初状态，程序未进行，进行数据初始化
1	0	初始化结束，程序进行中，时刻判断检查点
1	1	完成一次任务，程序暂停，记录结果并更新数据
0	1	输出结果，程序结束

3)混合精度量化准确率

功能仿真验证完成后，分别将模型从 1 bit 到 32 bit 进行量化并测量其准确率，实验结果如图 20 所示。

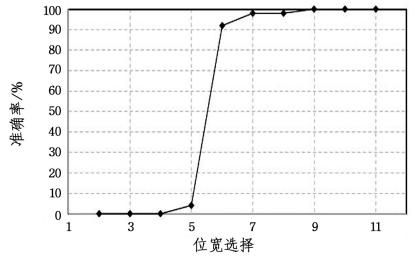


图 20 量化到不同位宽的准确率

Fig. 20 Accuracy of quantization to different bit widths

实验结果显示，模型准确率从 8bit 开始下降，量化到 5bit 时精确率呈断崖式下降，直到完全失去准确率。针对此模型，结合以上结果，对于第 3 节中的自适应感知量化策略，将待选位宽向量设置为[6,7,8,9,10]，从而进行资源约束下的自适应感知训练，最终对寻找出不同资源约束下的量化策略起到了先验精确度的指导性意义。

对于混合精度量化策略，采取了[5,6,7,8,9]，[6,7,8,9]和[7,8,9,10]等方案进行混合量化，发现当有单位宽精度较低的 bit 位出现在混合精度中时，模型呈现出极低的准确率，如[5,6,7,8,9]量化方案准确率只有 5%，并不会出现“平均精确率”的情况；当单位宽精度较高的 bit 位进行混合精度量化时，模型呈现出较高的准确率，如[7,8,9,10]方案的准确率达到 98%；同时根据功能及仿真显示，完成一次图像识别仅需 4.73 ms，与人类相比，其仅仅占用了大脑识别一次图像用时的 1/3[19]。

4）能耗评测

将模型分别量化到不同精度，测量模型在 Aritx-7 200T 核心板的能耗，如图 21 所示。模型能耗随着量化位宽的增高而上升，在 8 bit 以下时，模型能耗基本保持在 20 W 左右，而在 24 bit 以上，模型能耗大幅上升；同时在 Zynq-7020 开发板上测量，发现其能耗相近。

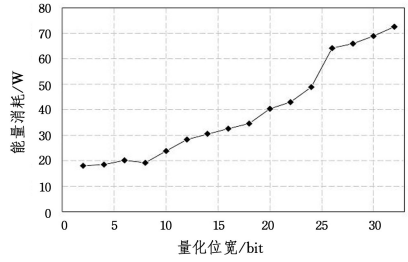


图 21 不同量化位宽下的能耗

Fig. 21 Energy consumption at different quantization bit widths

最后分别在 Zynq-7020 开发板上对全精度模型、低精度（4bit）以及混合精度（[7，8，9，10]）策略进行能耗测量，结果如图 22 所示。混合精度量化方法相对于全精度模型降低了 68.15%的能量消耗，而准确率仅仅下降了 2%。

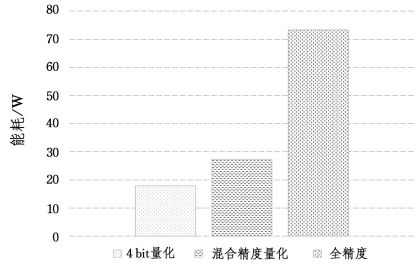


图 22 不同量化策略的能耗

Fig. 22 Energy consumption of different quantization strategies

结束语 面向资源约束的自适应模型量化方法，能够有效地降低模型大小和参数量，使得在智能物联网终端设备上部署深度学习模型成为可能；通过 FPGA 这种可编程器件，利用其灵活的并行性和可编程性，为部署在拥有独立操作系统的终端设备量化方案起到了指导作用，对部署轻量化模型到终端设备有着重要的发展意义。基于此，本文主要研究了资源自适应混合精度量化策略问题。1）首先，本文提出了一种基于资源限制的自适应量化模型方案，根据不同资源约束得出不同量化策略，对常用的卷积神经网络进行分层量化，并在 Raspberry Pi 上测试了不同资源约束下对模型参数量以及开销的衡量，验证了方法的有效性。2）利用 FPGA 的灵活可编程性，本文采取了从深度学习模型网络结构到 RTL 级硬件描述语言的转换，并将其应用在 FPGA 上，获得了针对混合精度量化待选位宽具有指导性的范围；衡量了不同量化位宽下模型能源的消耗程度，以及将图像识别任务部署在 FPGA 上的反应速度。

实验结果表明，本文采取的混合精度量化策略能够在 FPGA 硬件设备上降低 60%以上的能源消耗，并获得了可观的准确率，一定程度上减少了深度学习对资源受限的终端设备计算能力和资源的需求，且可根据具体需求灵活选取不同的量化策略，具有很高的任务扩展性。

本工作还有很多可以改进的地方：首先，目前混合精度量化方案在可定制化器件之外的设备上支持度不足，深度学习框架对其支持度也很匮乏，因此将混合精度量化方案完全应用在智能物联网终端设备上仍需软硬件的发展；其次，目前有精确度更高、更加轻量化的模型或更细粒度的人工神经网络设计，比如结合处理器调度的计算方案，能够更好地利用终端设备上的资源；最后，目前从深度学习网络到硬件描述语言的转换，仍是一个比较复杂并且任务量大的项目，还需要结合如 HLS、Zynq-Python 的桥梁 PYNQ（Python Productivity for Zynq）等方式降低深度学习到 FPGA 的难度，从而更好地将深度学习部署在此类可定制化的终端设备之中。

参 考 文 献

[1] LI W，LIEWIG M. A survey of AI accelerators for edge environment[C]// World Conference on Information Systems and Technologies. Cham：Springer，2020：35-44.

[2] ALOM M Z，TAHA T M，YAKOPCIC C，et al. A state-of-the-art survey on deep learning theory and architectures[J]. Electronics，2019，8(3)：292.

[3] KRIZHEVSKY A，SUTSKEVER I，HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Advances inNeural Information Processing Systems，2012，25.

[4] LUO J H，WU J，LIN W. Thinet：A filter level pruning method for deep neural network compression[C]// Proceedings of the IEEE International Conference on Computer Vision. 2017：5058-5066.

[5] GUPTA S，AGRAWAL A，GOPALAKRISHNAN K，et al. Deep learning with limited numerical precision[C]// International Conference on Machine Learning. PMLR，2015：1737-1746.

[6] VANHOUCKE V，SENIOR A，MAO M Z. Improving the speed of neural networks on CPUs[C]// NIPS 2011. 2011.

[7] JIAO L C，SUN Q G，YANG Y T，et al. Progress，implementation and prospect of deep neural network FPGA design [J]. Chinese Journal of Computers，2022，45(3)：441-471.

[8] CHEN D，SINGH D. Fractal video compression in OpenCL：An evaluation of CPUs，GPUs，and FPGAs as acceleration platforms [C]// 2013 18th Asia and South Pacific Design Automation Conference(ASP-DAC). IEEE，2013：297-304.

[9] NURVITADHI E，SIM J，SHEFFIELD D，et al. Accelerating recurrent neural networks in analytics servers：Comparison of FPGA，CPU，GPU，and ASIC[C]// 2016 26th International Conference on Field Programmable Logic and Applications (FPL). IEEE，2016：1-4.

[10] AHMED M T，SINHA S. Design and Development of Efficient Face Recognition Architecture Using Neural Network on FPGA [C]// 2018 Second International Conference on Intelligent Computing and Control Systems(ICICCS). IEEE，2018：905-909.

[11] RICE K L，BHUIYAN M A，TAHA T M，et al. FPGA implementation of Izhikevich spiking neural networks for character recognition[C]// 2009 International Conference on Reconfigu-

rable Computing and FPGAs. IEEE, 2009: 451-456.
[12] MA Y, CAO Y, VRUDHULA S, et al. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks[C]// Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2017: 45-54.
[13] SHEN J, WANG Y, XU P, et al. Fractional skipping: Towards finer-grained dynamic cnn inference[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2020: 5700-5708.
[14] WANG X, YU F, DOU Z Y, et al. Skipnet: Learning dynamic routing in convolutional networks[C]// Proceedings of the European Conference on Computer Vision(ECCV). 2018: 409-424.
[15] JACOB B, KLIGYS S, CHEN B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 2704-2713.
[16] WU Z, NAGARAJAN T, KUMAR A, et al. Blockdrop: Dynamic inference paths in residual networks[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 8817-8826.
[17] WILLIAMS S, WATERMAN A, PATTERSON D. Roofline: an insightful visual performance model for multicore architectures [J]. Communications of the ACM, 2009, 52(4): 65-76.
[18] Dual Lens Camera Module AN5642 User Manual[EB/OL] https://alinx.com/public/upload/file/AN5642_User_Manual.pdf.
[19] POTTER M C, WYBLE B, HAGMANN C E, et al. Detecting meaning in RSVP at 13 ms per picture[J]. Attention, Perception, & Psychophysics, 2014, 76(2): 270-279.

WANG Yuzhan, born in 2000, master. His main research interests include mobile computing, model compression, and middleware for the Internet of things.

GUO Bin, born in 1980, Ph. D, professor. His main research interests include ubiquitous computing, mobile crowd sensing, and HCI.

(责任编辑:柯颖)