高技术通讯 2022 年 第 32 卷 第 11 期:1143-1152

doi:10.3772/j.issn.1002-0470.2022.11.005

面向高能效场景的神经网络结构和加速器协同设计①

陈维伟②*** 王 颖③* 张 磊*

(*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 神经网络算法和深度学习加速器已成为推动深度学习方法应用最重要的两股力量,但目前的神经网络结构设计主要围绕模型精度、计算量等指标,忽略了不同模型在目标加速器上计算效率的差异;而加速器设计一般针对既定的神经网络基准程序进行优化,往往难以覆盖到未来不断迭代进化的神经网络模型,这就容易导致加速器在新的网络架构上表现不佳。本质上,神经网络架构与加速器相对独立的设计流程,导致了两者的设计和优化不匹配,从而无法达到最优的深度学习推理性能。为此,本文提出了一种针对图像分类任务的网络结构和加速器软硬件协同设计的框架,将网络结构和加速器设计融合到统一的设计空间中,并针对设计约束,自动搜索最优协同设计方案,实现了端到端的深度学习推理定制和优化。实验表明,在真实的图像分类数据集和脉动阵列架构上,相对于传统的网络结构和加速器分别独立优化的方法,本文提出的协同设计方法实现了平均 40% 的能耗降低。

关键词 神经网络结构设计;加速器设计;软硬件协同设计;设计空间探索

0 引 言

近年来,深度神经网络(deep neural network,DNN)在计算机视觉、自然语言处理等领域都取得了巨大的突破[1]。算法研究人员针对不同的应用设计了大量的网络结构和对应的训练方法,如在图像分类领域,从最初的 AlexNet 到目前的 EfficientNet,人们对神经网络结构的改进不断提升其精度和性能表现[2]。同时针对深度学习这类计算密集和数据密集应用,业界关注开发特定领域的架构(如硬件加速器),此类硬件加速架构多采用单指令多数据流、用户控制与缓存等技术,使其在并行性及带宽利用率上比通用架构更加高效,通过结合专用领域的编程语言以继续提高计算能力[3-5],如 Google 公司研发的张量处理单元(tensor processing unit,TPU)在神经网络应用中平均比图形处理器(graphics processing unit,GPU)(Nvidia K80 GPU)或中央处理器(central processing unit,CPU)(Intel Haswell CPU)快 15~30 倍,能效比(Tops/Watt)高出约 30~80 倍[3-4]。

深度学习软硬件的兴起为构建高能效的人工智能系统提供了可能,然而,目前深度学习算法与硬件专用架构设计之间依然存在着很大的鸿沟。在算法设计方面,DNN 模型设计者通常专注于设计具有更高精度、更低参数和更小每秒浮点运算次数(floating-point operations per second,FLOPs)的 DNN 模型,但是其较少考虑到 DNN 在具体硬件上的执行特性,使得仅设计具有较低参数和较少 FLOPs 的 DNN 模型并不能直接提升性能[2,7],如 NasNet 与分层网络 MobileNet 相比,其有更低的参数但使用多分支结

① 国家自然科学基金(61902375)和中国科学院战略性先导科技专项(C 类)(XDC05030201)资助项目。

② 男,1994 年生,博士生;研究方向:计算机体系结构,软硬件协同设计优化;E-mail:chenweiwei@ict.ac.cn。

③ 通信作者,E-mail:wangying2009@ict.ac.cn。

(收稿日期:2021-08-27)

构,其碎片化的单元结构使得硬件运算效率较低[8]。在硬件设计方面,目前,对应用广泛的、具有代表性的程序进行泛化是硬件设计和优化的传统方法,其通常先构建一组基准测试程序(Benchmark,如 SPEC),包含了一组特定的工作负载程序,硬件设计的目标之一就是优化这组特定程序的平均性能。例如,CPU 是为串行程序(如桌面应用程序)而优化,GPU 是为大规模并行工作负载(如图形渲染、科学计算等)而设计。深度学习硬件设计评估也沿用此方式,当前 MLPerf 已成为评估深度学习加速器设计的标准 Benchmark,深度学习硬件加速器在 MLPerf 上的得分成为衡量硬件性能的重要指标。然而 Benchmark 所选定的应用程序可能在相当长的一段时间内固定不变,这容易使得专用硬件设计对新算法的优化不够充分。特别是在深度学习领域,由于其快速发展在短时间内催生了大量精度更高、泛化能力更强、计算效率更高的模型,使得最新的 DNN 模型和加速器协同度不高。例如,最新的神经网络 EfficientNet,其使用了全局池化和 SiLU/Swish 非线性激活函数来提升精度,但最新的加速器不能很好支持此类计算[9-10],导致系统不能工作在最佳状态。

因此,针对目前软硬件设计分离问题,神经网络模型和加速硬件的协同设计对进一步提升人工智能(artificial intelligence,AI)系统的整体性能尤为重要[11]。为此,本文针对图像分类任务,提出了一种新的网络结构和加速器软硬件协同设计框架,本方法可以在统一的 DNN 结构和加速器设计空间中,针对不同目标自动进行协同设计搜索,进一步提升 AI 系统性能,并减少人工开销。本文的主要贡献点如下。

(1) 通过在统一的联合搜索空间中将 DNN 模型结构和加速器搜索参数化,可以针对不同设计场景和设计约束,自动搜索 DNN 结构和加速器协同设计方案,进一步扩展帕累托边界。

(2) 针对协同设计搜索空间巨大的问题,提出了一种 LSTM + RL 的方案,使用长短时记忆(long short-term memory,LSTM)来生成设计序列,同时使用强化学习(reinforcement learning,RL)方法指导搜索方向。实验结果表明,此方法可有效降低协同设计搜索开销。

(3) 针对神经网络精度评估和硬件设计评估开销巨大问题,提出了基于 HyperNet 的网络精度评估方法,候选 DNN 模型可直接从 HyperNet 中继承权重以避免昂贵的训练开销;同时采用高斯过程模型对硬件设计性能进行预测,减少了整体的评估开销。

1 研究现状

为进一步提升 AI 系统的性能和服务质量(quality of service,QoS),神经网络软硬件协同设计已经成为近年来的研究热点[12]。当前研究工作从神经网络结构设计到硬件加速器的多个优化方面展开,主要存在 3 类提升神经网络运算能效的方法。第 1 类是硬件感知神经网络结构设计方法,其面向特定硬件设计更高效的 DNN 模型。具体而言,硬件感知神经网络结构搜索方法根据应用设定 DNN 结构设计空间,在设计过程中,依赖性能评估模型来估计候选神经网络模型在目标硬件上的性能,然后在迭代设计过程中决定是否应该保留或更新所发现的网络,最后筛选出针对目标硬件和约束的最优 DNN 模型。如 FbNet[13] 和 ProxylessNAS[14] 都是针对智能手机设计的 DNN 模型,这 2 个 DNN 模型都旨在实现较高模型精度的同时,实现较高的推理速度。然而此类硬件感知神经网络结构搜索方法需要针对每一个任务专门设计网络结构搜索空间,需要丰富的专家知识,极大影响了其适应不同场景的能力。

第 2 类提升神经网络计算效率的方法是针对特定任务的神经网络硬件加速方法,其需要根据 DNN 的计算模式来设计新的指令集架构、编程模型、编程工具,甚至是新的编程语言,来实现最大化的性能提升。如 Kiseok 等人[6]针对嵌入式视觉任务,提出了一种 DNN 模型和加速器的协同设计方法,其首先针对 SqueezeNet,专门设计了一个 DNN 加速器,然后调整 SqueezeNet 的超参数,使得修改后的 SqueezeNet 在定制的加速器上运行更高效。然而针对不同场景设计专用硬件加速器耗时耗力,还需要针对性地设计专用的编译部署方法,这使得部署成

本高昂。

此外,第 3 类提升神经网络计算效率的方法是 DNN 模型结构和硬件协同设计。如 Yang 等人[15]在嵌入式现场可编程门阵列(field-programmable gate array,FPGA)上采用软硬件协同设计方法来加速 ConvNet 模型。其采用 1 × 1 的卷积块来构建 DNN 模型,然后基于 FPGA 设计高度定制化的 1 × 1 卷积计算单元,以提高前向推理速度。然而,该方法由于定制化程度较高,不能移植到其他应用程序和平台。Ji 等人[16]提出了工具链 Neutrams,该工具链将现有神经网络模型进行转换(如算子融合),以满足神经形态芯片的硬件约束要求,然后进行硬件适配并将转换后的神经网络模型映射到神经形态芯片上。Hao 等人[1]则提出了一种 FPGA/DNN 协同设计方法,其主要由两部分构成:一个是针对硬件自底向上的神经结构块构建,用于高精度网络结构搜索;另一个是针对每一个神经网络块,自顶向下的构建专用知识产权核(intelligent property,IP)实例,并在 FPGA 上将这些 IP 组织起来,从而实现专用的神经网络计算加速。

2 研究动机

当前的 DNN 模型和加速器协同设计工作证明了其提升神经网络计算系统性能的潜力,然而它们基本都遵循着两阶段式的设计流程。图 1 对当前的两阶段设计流程进行了简要描述。首先,此类设计要么修改现有的 DNN 模型,要么使用人工或网络结构搜索(neural architecture search,NAS)方式设计一个高精度的 DNN 模型,然后设计一个针对此 DNN 模型的硬件加速器。此两阶段式的设计方式限制了其只能在局部空间中进行搜索,导致可能不会收敛到最优解,特别是当需要考虑到真实部署环境中有更多实际的设计约束目标(如带宽、能耗、QoS 等)[17]时,针对最高精度的 DNN 模型进行专用硬件化并不一定能保证最好的系统性能[18]。此外,由于这些硬件设计流程都是针对特定的应用和加速器架构,如文献[1]中人工定制的神经操作运算 IP,当移植到其他应用上时,还需要大量的人工改动从而导致高额的人工设计开销。因此,本文针对目前两阶段设计方法的缺点,提出了一种新的网络结构和加速器软硬件协同设计方法并将其命名为单阶段协同设计方法。图 2 展示了此单阶段设计方式的流程图,相较于两阶段设计方式,本方法可以在统一的 DNN 结构和加速器设计空间中,针对不同目标自动进行协同设计搜索,进一步提升 AI 系统性能。

本文通过实验也证明了当前两阶段设计在面向不同应用时能效提升有限的问题。图 3 展示了使用 3 种不同的脉动计算阵列(processor element,PE)大小,在 Once-for-All[2]的网络设计空间中进行神经网络结构搜索获得的帕累托曲线。可以发现,在不同的设计约束下,最优的网络结构和硬件配置组合是不同的,如在延迟约束为 8 ms 时,加速器配置 3 的结果最优,而当延迟约束为 6 ms 时,加速器配置 2 的结果最优。这也验证了,针对特定的应用场景和设计约束,需要在统一的设计空间中进行网络结构和加速器的协同设计,来进一步提升 AI 系统效率。

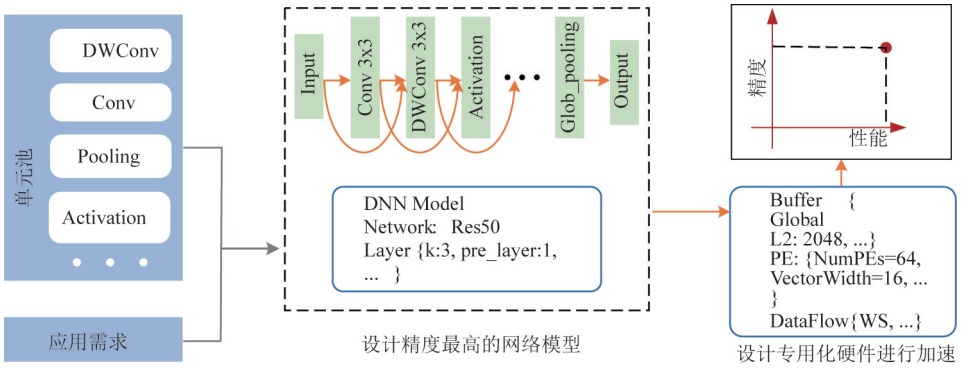


图 1 传统的两阶段设计流程示意

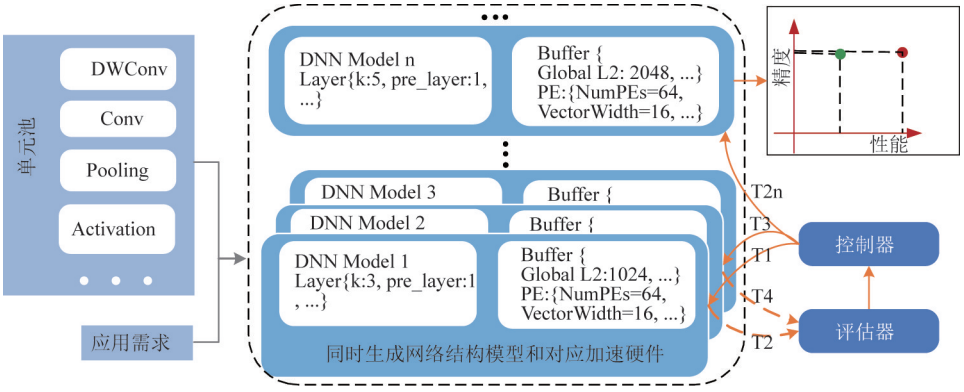


图 2　单阶段网络结构/加速器协同设计流程示意

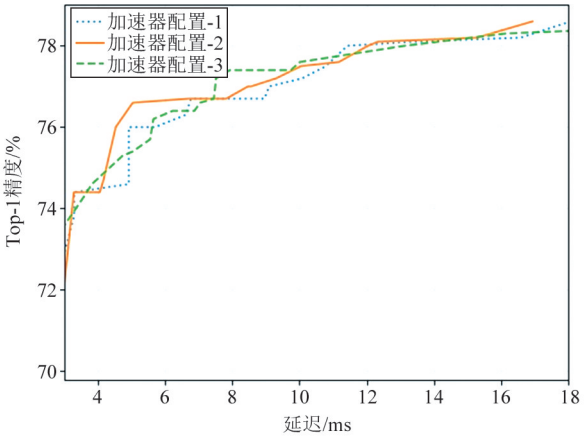


图 3　在同一网络结构设计空间中,3 种不同硬件参数配置下搜索获得的帕累托曲线图

3　单阶段协同设计搜索方法

3.1　问题定义

在给定神经网络结构的搜索空间 $\boldsymbol{\eta}=\{\eta^1,\cdots,\eta^m\}$(包含 m 个神经网络模型结构)和加速器配置搜索空间 $\boldsymbol{C}=\{c^1,\cdots,c^n\}$(总共有 n 种配置选择),同时在给定用户设计约束 *thres* 情况下,此工作的设计目标是自动化生成最优网络结构 $\eta*$ 和相对应的加速器配置 $c*$,因而能在满足用户约束下,提供最优的系统性能和精度,其形式化描述如下:

$$(\eta*,c*)\in\underset{\eta*\in\boldsymbol{\eta},c*\in\boldsymbol{C}}{\operatorname{argmax}}A(\eta,c)\quad \text{s.t.}\quad Perf(\eta*,c*)<thres \tag{1}$$

其中 A 代表 DNN 模型精度;*Perf* 代表了系统的性能约束,在具体实验中,可代表延迟或能耗等具体指标。

3.2　方法概述

在融合后的统一协同设计空间中直接进行搜索主要面临着 2 个挑战:(1)巨大的协同搜索空间;(2)高额的评估开销。由于没有成熟的理论来解释网络结构和性能之间的准确性关系,就需要对每一个网络进行训练以获得精度,以及对硬件设计进行模拟来获得性能指标,这造成了较高的网络训练以及性能模拟开销成本。基于此,本文设计了 2 种技术来解决上述问题,主要为:(1)基于 LSTM 的序列生成器,用来生成协同设计点(包含 DNN 结构和硬件配置),同时采用强化学习进行搜索;(2)快速的 DNN 精度和硬件性能评估器。图 4 给出了系统的整体流程图,其主要分为 3 步:

(1) DNN 精度和硬件性能评估器构建。此步骤中,系统根据目标任务(如分类、检测)构建 HyperNet,候选 DNN 模型可视为 HyperNet 的子网,其可直接从 HyperNet 中进行抽取并继承权重,以便避免单独训练从而实现快速的 DNN 模型精度评估。同时从硬件模拟器中采样,获取不同硬件配制下的性能数据,进而构建高斯过程模型直接预测硬件性能来替代高耗时的硬件模拟。

(2) 针对目标约束的高效搜索。根据设计约束(如精度、能耗、延迟等),采用 RL 进行迭代搜索。在每一步迭代中,使用 LSTM 输出设计序列,然后送入第一步中构建的精度和性能评估器中进行评估,获得 reward 反馈来指导 RL 控制器进行下一次的迭代。

(3) 确定最优设计点。当搜索迭代到一定次数,选择搜索过程中综合得分最高的前K个点,进

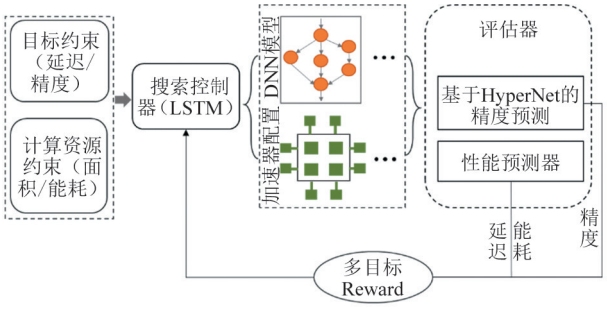


图 4　整体方法设计流程概览图

行 DNN 模型重新训练和真实的硬件评估,并选择重训后得分最高的作为最优设计点。

3.3　强化学习搜索策略

在此协同设计空间中,将每一个候选设计点当做一个序列: $\boldsymbol{\lambda} = (\boldsymbol{\eta}, \boldsymbol{c}) = (d_1, \cdots d_S, d_{S+1}, \cdots, d_{S+L})$。此序列包含两部分,分别是网络结构超参数(长度为 S)和硬件架构超参数(长度为 L),可以从每一个候选 $\boldsymbol{\lambda}$ 构建出唯一的 DNN 模型和硬件加速器组合。其中,对于 $\boldsymbol{\lambda}$ 中的每一个参数的选择可看做强化学习中的一个 action(动作选择),所以整个 $\boldsymbol{\lambda}$ 可被视为一个动作序列。因此,本文使用 LSTM(采用 120 个隐层单元)来生成此序列,图 5 列出了生成序列的示意图。在生成 $\boldsymbol{\lambda}$ 的第 i 个参数时,LSTM 接收第 $i-1$ 个参数作为输入,并将 0 作为第一个参数的输入。为提高搜索效率,本文使用强化学习来捕捉动作序列之间的相关关系来指导序列生成具有较高的奖励(reward),此 reward $R(\boldsymbol{\lambda})$ 的形式化表示如下:

$$R(\boldsymbol{\lambda}) = A(\boldsymbol{\lambda}) + \alpha_1[l(\boldsymbol{\lambda})/t\_lat]^{\omega_1} + \alpha_2[e(\boldsymbol{\lambda})/t\_eer]^{\omega_2} \tag{2}$$

$R(\boldsymbol{\lambda})$ 由多个目标构成,其中 $A(\boldsymbol{\lambda})$ $l(\boldsymbol{\lambda})$, $e(\boldsymbol{\lambda})$ 分别表示生成序列 $\boldsymbol{\lambda}$ 的精度、延迟和能耗。t_lat, t_eer 分别表示有用户设定的延迟和能耗约束, α_1、ω_1、α_2、ω_2 为控制系数。因此,强化学习的目标是找到最优的超参 $\boldsymbol{\lambda}^*$ 来最大化 reward $R(\boldsymbol{\lambda})$, 即:

$$\boldsymbol{\lambda}^* = \underset{\lambda \in \boldsymbol{\Lambda}}{\operatorname{argmax}}(E_{p(\lambda;\varphi)}[R(\boldsymbol{\lambda})]) \tag{3}$$

φ 代表 LSTM 中需要被更新的参数,通过更新 φ 来生成具有更高 reward 的序列 $\boldsymbol{\lambda}$。同时,使用 REINFORCE 算法[19]来更新 LSTM 参数 φ, 其具体更新方法表示如下:

$$\nabla_{\varphi} L(\varphi) = -E[(R(\boldsymbol{\lambda}) - b)\nabla_{\varphi} \log p_{(\lambda;\varphi)}(\boldsymbol{\lambda})] \tag{4}$$

其中 b 代表了采用当前 φ 的生成序列的期望 reward,也即平均 reward。通过引入平均 reward 值可以降低参数更新的方差,从而提高了搜索效率。

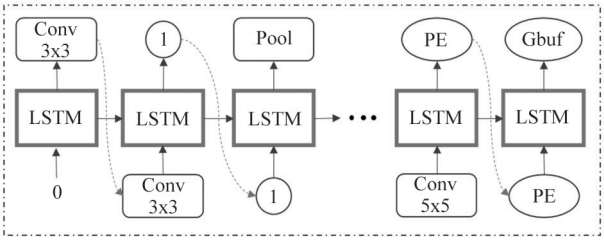


图 5　基于 LSTM 的序列生成流程示意图

3.4　基于 HyperNet 的精度评估

为了快速评估 DNN 模型的精度,本文利用权重共享方法[5]构建了一个超网(HyperNet)来对每一个候选 DNN 直接赋予权重值。每一个候选 DNN 模型可被视为 HyperNet 中的一条路径,并直接从 HyperNet 中继承权重。从而,只需要训练这个 HyperNet 一次,避免了之前对每一个网络进行单独训练的高额开销。

HyperNet 结构。图 6 给出了 HyperNet 的结构示意图。与之前网络结构设计方法[14,18,20]相同,本文使用 2 种不同的 cell 结构:规约单元(reduction cell)和正常单元(normal cell)来构建 HyperNet,其中 reduction cell 和 normal cell 有着相似的结构,但 reduction cell 的步长值(stride)为 2。对于每一个 cell 结构,可视为由 B 个节点(node)组成的无向图,其中每一个 node 代表着 DNN 处理过程中的特征图。对于第 i 个 node I_i, 其需要选择 2 个前向 node (j 和 k)作为输入。Node 之间的连接可视为图的边,对于每一个边,需要选择一个具体运算操作(如 conv3×3、conv5×5、pooling 等),其形式化描述如下:

$$I_i = \theta_{(i,j)}(I_j) + \theta_{(j,k)}(I_k) \quad \text{s.t. } j < i \,\&\, k < i \tag{5}$$

$\theta_{(i,j)}$ 和 $\theta_{(i,k)}$ 分别为 (i, j) 和 (i,k) 之间选择的运算操作, I_0 和 I_1 为前 2 个 cell 的输出。在输出选择上,本文将所有 node(除去 I_0 和 I_1)进行拼接之后作

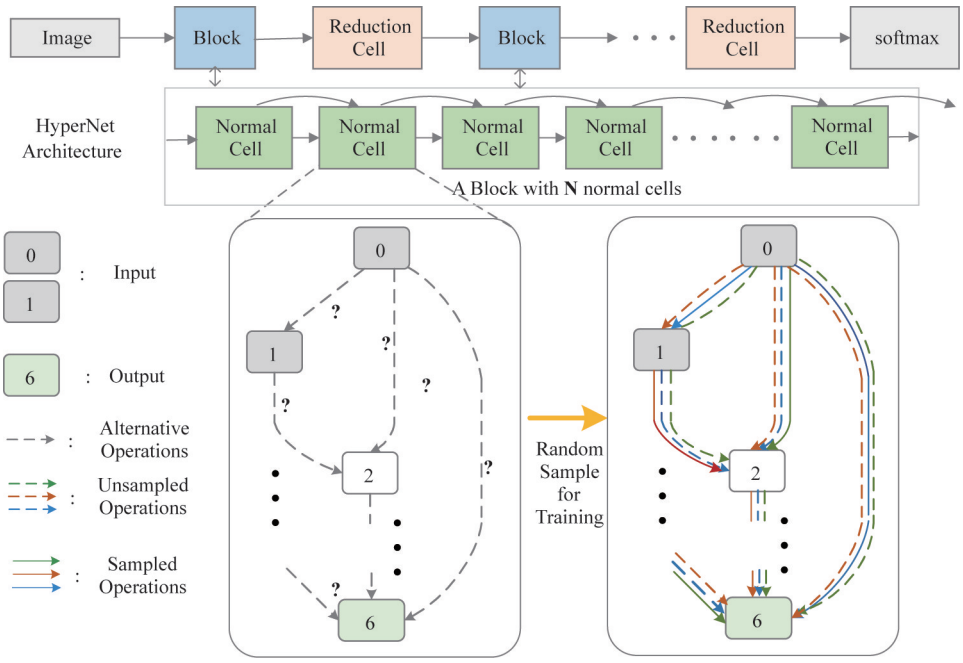


图 6　HyperNet 结构与训练策略示意

为此 cell 的输出。因此，每一个 node 通过选择不同的前向 node 作为输入以及选择不同的运算操作，从而确定不同的 cell 结构，又通过不同的 cell 组合构成不同的 DNN 模型。因而每一个候选 DNN 模型可视为整个 HyperNet 中的一条路径，其可直接从 HyperNet 中继承权重。在实验中，本文选取了 6 种运算操作和 7 个 node 组成 HyperNet，因此，整个 HyperNet 包含$(6\times(B-2)!)4\approx5\times10^{11}$种不同的候选 DNN 模型结构。

HyperNet 训练策略。为了公平地评估每一个子网 DNN 模型的精度，防止子网选择偏好对精度评估造成影响，本文对 HyperNet 采取了一种公平的训练策略，即在 HyperNet 的每一次训练迭代中，随机采样子网并进行参数更新，可表示为

$$p(I_0)=\cdots=p(I_{i-1})=2/i \ \& \ p(\theta_{(i,j)})=1/V \tag{6}$$

对于第 i 个 node，其选择前向第 $i-1$ 个 node 的概率为 $p(I_{i-1})=2/i$。对于每一个备选运算操作，其被选概率为 $1/V$，V 为运算操作的个数，实验中选用了 6 个不同的运算，所以此处为 6。对比之前工作[7,18,21]，此随机采样策略有效避免了采样偏差导致的精度评估不准确问题，从而对候选 DNN 模型结构进行更有效的精度评估。

3.5　基于高斯过程的硬件性能评估

在传统的加速器设计中，通常需要构建一个精确的模拟器来对硬件进行时钟级的行为仿真，以反映硬件的真实运行状态，但这同时导致模拟时间开销巨大，使得在协同设计的超大空间中直接使用模拟器来进行硬件性能评估不可行。因此，为了进行快速的硬件性能评估，本文采用机器学习技术来构建硬件性能评估器，实现了近 2000 × 的速度提升，同时预测精度误差小于 4%。在实验中，本文只构建了延迟和能耗预测器，但此方法可以扩展到其他硬件性能预测。

能耗预测器。图 7 比较了 6 种不同的机器学习模型对能耗的预测能力。在选择具体的机器学习评估模型前，本文首先从模拟器中收集了 3600 个训练样本，并使用 600 个样本作为预测样本，然后使用这 6 种不同的机器学习方法在此样本上进行训练，并选取预测能力最优的机器学习模型作为性能预测器。通过实验发现，基于高斯过程（采用 RBF 核）的预测模型具有最高的预测精度，因此采用高斯过程模型作为能耗预测器。

延迟预测器。同构建能耗预测方法一样，本文比较了这 6 种不同的预测模型，然后根据预测精度来筛选预测器，最后也同样使用高斯过程来构建延

迟预测器。

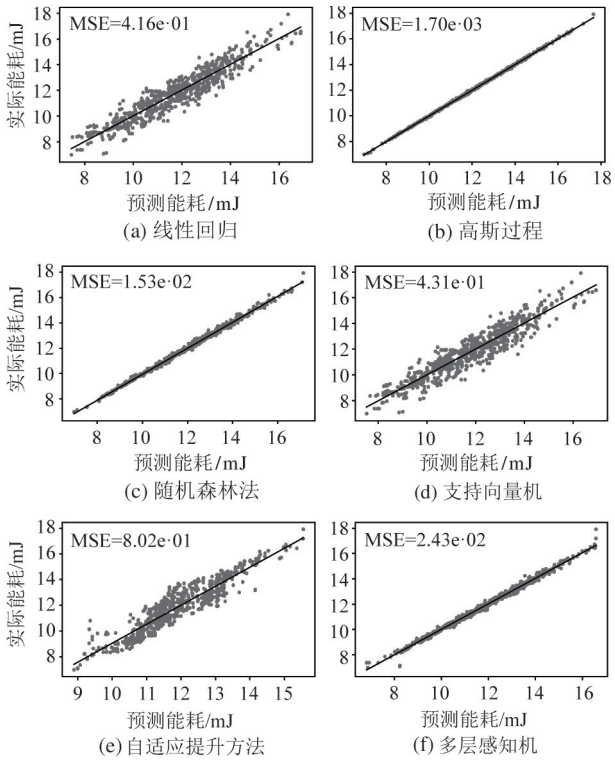


图 7　不同机器学习方法的能耗预测结果对比

4　实验与分析

4.1　实验设置

为验证此协同设计框架性能,本文在图像分类任务数据集 Cifar10 和 Eyeriss(脉动阵列架构)[22]加速器上进行实验验证。根据设计流程,此搜索框架根据用户设计约束,自动搜索 DNN 模型结构和加速器配置。在加速器可配置参数中,假设可配置 PE 的大小、全局缓冲器大小、寄存器大小和 Dataflow 的方式。表 1 列出了需要搜索的一些关键参数。本文所有实验运行在单台 Linux 主机上,采用 Intel Xeon 8163 2.5 GHz CPU 和单卡 NVIDIA V100 GPU(16 GB HBM2)配置。在实验中,使用 Cuda 10.0 和 TensorFlow 1.12 实现了 RL 控制器、LSTM 和 HyperNet。

4.2　HyperNet 验证

DNN 精度预测器是协同搜索中精度评估的关键,为此,实验验证了本文提出的基于 HyperNet 的 DNN 模型精度预测器。也就是说,通过从 HyperNet 继承的权重来获得 DNN 模型的精度的方法,与直接单独训练 DNN 模型的方法相比,是否有正相关性?

表 1　DNN 加速器协同设计参数

搜索参数	解释
$\langle N\_cell \rangle$	组成一个 block 所用 normal cell 的个数
$\langle I_j, I_k \rangle$	每一个 node 需选择的前向 node
$\langle \theta_{(i,j)}, \theta_{(i,k)} \rangle$	每一个 node 所选运算操作
PE_x	PE 阵列 X 轴维度大小(8 ~ 32)
PE_y	PE 阵列 Y 轴维度大小(8 ~ 32)
G_buf	全局 buffer 大小(128 ~ 2048 kB)
R_buf	每一个 PE 寄存器大小(128 ~ 1024 byte)
Dataflow	数据流选择:权重静置(weight stationary,WS),输出静置(output stationary,OS),行静置(row station-ary,RS),和无局部重用(no local reuse,NLR))

因此,实验中使用 6 个 cell(4 个 normal cell 和 2 个 reduction cell)来构成 HyperNet 并将其训练 300 个 epoch。在 HyperNet 的训练过程中,本文采用动量为 0.9 的随机梯度下降优化器和数据增强策略,同时采用余弦学习率衰减策略并控制学习率在 0.05 ~ 0.0001 之间。此外,使用 L2(4×10^{-5})正则化对权重进行优化。

图 8(a)展示了 HyperNet 在训练过程中的精度变化。具体而言,在训练过程中从 HyperNet 随机抽出子网模型,并以此精度来代表 HyperNet 的精度。可以看出,HyperNet 的精度随着训练迭代不断提升。图 8(b)展示了 HyperNet 训练完成后,从 HyperNet 直接继承权重的 DNN 模型精度与单独训练的实际精度的相关关系。由于协同设计空间中DNN模型

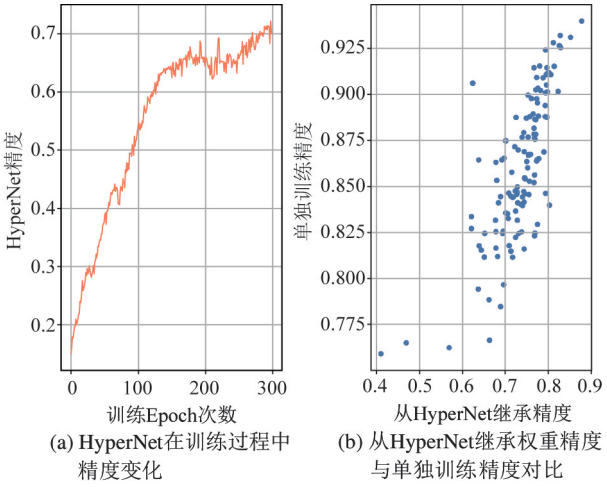


图 8　HyperNet 精度验证

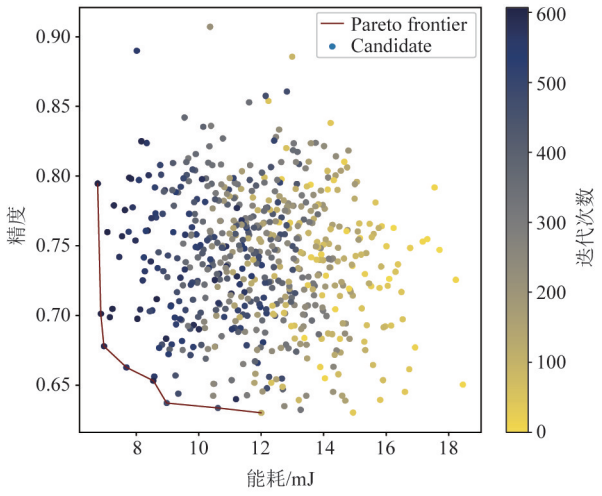
数量巨大,不能对全部 DNN 模型进行验证,本文只从 DNN 设计空间中随机选取 130 个模型进行了单独训练(每个模型单独训练 70 个 epoch)。从图 8(b)中可以明显看出,直接从 HyperNet 继承权重与单独训练的精度具有较高的正相关性,从而表明从 HyperNet 生成权值可以用于预测 DNN 模型的真实精度,进而避免了高额的单独训练开销。

4.3　搜索策略

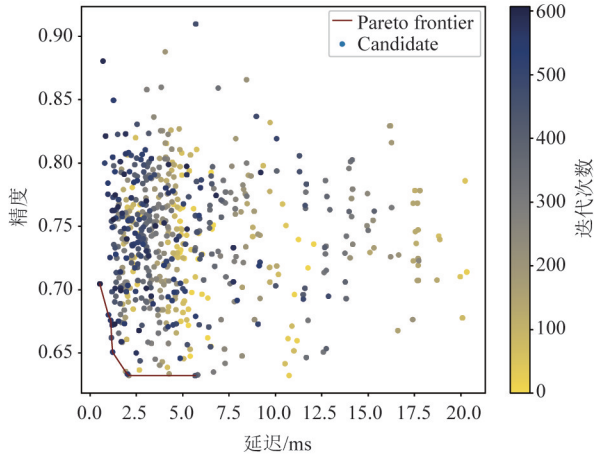
在实验中,本文对比了使用强化学习与随机搜索在协同设计中的搜索过程,此两种方法都以寻找更高的网络结构和对应硬件配置为目标。此强化学习搜索算法采用了 REINFORCE 算法[19]进行更新,参数更新采用 Adam 优化器,其学习率为 0.0035。由于此协同设计为多目标优化,在式(2)中,可以设定不同的 α_1、ω_1、α_2 和 ω_2 系数使得搜索结果偏向不同的优化目标,如精度最高、延迟最低或精度/能耗综合得分最高。因此,本文设计了 2 种不同的 reward 函数,以使搜索结果偏向不同的设计目标。图 9(a)针对 DNN 模型精度和能耗进行权衡的场景,也就是需要更高的精度/能耗综合得分目标下,使用强化学习方法进行搜索获得的结果。通过设定式(2)中的参数,具体为 $\alpha_1 = 0.6$、$\omega_1 = -0.4$、$\alpha_2 = 0.3$ 和 $\omega_2 = -0.2$。可以看出,随着搜索迭代,其搜索区域逐渐向 Pareto 曲线靠近。同样,在图 9(b)中,根据 DNN 模型精度/延迟进行权衡的场景(设定参数为 $\alpha_1 = 0.3$、$\omega_1 = -0.3$、$\alpha_2 = 0.6$ 和 $\omega_2 = -0.4$),其搜索结果也不断朝着精度/延迟 Pareto 曲线区域靠近,证明了基于强化学习的搜索策略能根据用户需求设定,自动朝着具有较高 reward 的多目标区域移动,具有自适应搜索能力。从图 9(a)和(b)可以看出,相比于随机搜索,基于强化学习的搜索策略能有更好地适应此大空间搜索问题,在多次迭代过程中逐渐找到具有最高的 reward 的协同设计方案。

4.4　与两阶段设计方法结果对比

最后,本文将此单阶段协同设计方案与之前的两阶段设计方案进行了对比。为了进行公平比较,选择现有的具有较高精度的 DNN 模型[7,18,21-24]。对每一个对比模型,在统一的加速器设计空间中,枚举



(a) 设置 reward(α_1: 0.6 ω_1: −0.4 α_2: 0.3 ω_2: −0.2)
使得搜索结果偏好于有更高的精度/能耗综合得分



(b) 设置 reward(α_1:0.3 ω_1: −0.3 α_2: 0.6 ω_2: −0.4)
使得搜索结果偏好于有更高的精度/延迟综合得分

图 9　不同目标设定下的搜索过程示意

(实验中假设能耗约束为 9 mJ,延迟约束为 1.2 ms)

所有可能的加速器配置,并选择最优配置下的结果作为对比。对于单阶段协同设计方案,由于精度和性能预测器存在一些偏差,对最终选择结果有一定影响,为此本实验首先用 5×10^6 次迭代完成搜索,然后选择搜索过程中 reward 最高的前 10 个协同设计方案结果,对其中每一个网络单独训练获得准确精度,最后从这 10 个协同设计方案中选取综合结果最优的一组作为代表。表 2 展示了协同设计搜索相对之前两阶段设计方案的性能提升结果。作为对比,ACC/Latency 代表了本方案在精度/延迟权衡情况下的搜索结果,与之前两阶段方案相比,在同精度

情况下,本文提出的方案实现最低 0.77 ms 的延迟。同样,ACC/Energy 代表了本方案在精度/能耗权衡情况下的搜索结果,与之前方案相比,在同精度情况下,本文提出的方案实现了最低 7.5 mJ 的能耗。图 10展示了与之前两阶段设计方案相比,在同精度情况下,其平均能耗大约降低 40%,最高延迟降低 60% 左右。可以证明,更大的联合设计空间,可以进一步提升 AI 系统的性能;同时通过缩减搜索空间、提升搜索效率方法,对针对不同设计约束实现自动化搜索可以进一步降低人工开销。

表 2　不同设计方法结果对比

模型	搜索时间开销(GPU * Day)	精度误差	能耗/mJ	延迟/ms	加速器配置(PEs/g _ buf/r _ buf/data _ flow)
NasNet-A[6]	1800	3.41	15.24	2.11	16×32/196kb/256b/OS
Darts _ v1[5]	0.38	3.0	10.63	1.38	16×32/512kb/512b/OS
Darts _ v2[5]	1	**2.82**	11.01	1.62	14×16/256kb/128b/OS
AmoebaNet-A[7]	3150	3.12	13.67	1.76	16×32/256kb/1024b/OS
EnasNet[11]	1	2.89	16.65	2.25	16×32/196kb/512b/OS
PnasNet[8]	150	3.63	17.17	2.37	16×20/512kb/256b/OS
ACC/Latency	0.45	3.12	8.26	**0.76**	32×32/512kb/512b/OS
ACC/Energy	0.45	3.05	**7.47**	0.98	16×16/256kb/256b/OS

注:GPU * Day 代表一个 GPU 运行一天的时间开销

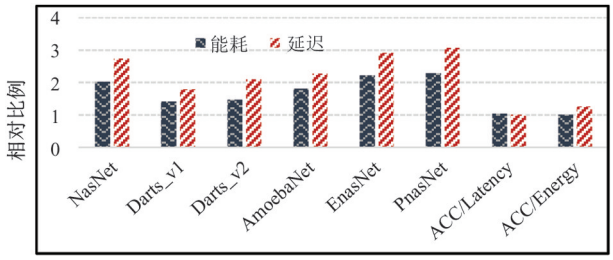


图 10　在同精度设计约束下,不同设计方案结果与最优延迟和最优能耗结果的相对比例

5　结　论

本文在已有的两阶段协同设计方法基础上,提出了一种直接在 DNN/加速器协同设计空间中进行自动化搜索的框架,以进一步提升 AI 系统的性能,并减少人工开销。针对候选协同设计点评估开销巨大的问题,设计了基于 HyperNet 的网络精度评估方法,每一个候选 DNN 模型可直接从 HyperNet 中继承权重以避免昂贵的训练开销;同时采用高斯过程模型对硬件设计性能进行预测,减少了整体的评估开销。针对联合设计搜索空间巨大问题,分析设计序列的相关性,利用 LSTM 来生成协同设计序列并使用 RL 来进行搜索。这些方法能加快性能评估并显著提高搜索效率。实验证明,在基于 Eyeriss 的加速器结构和 Cifar10 数据集上采用此协同设计方法,能在单卡 GPU 上 12 h 内完成整个搜索。与之前的两阶段设计方法相对比,此协同设计方法在同精度设计要求下,能实现平均近 40% 的能耗降低。

参考文献

[1] HAO C, ZHANG X F, LI Y H, et al. FPGA/DNN co-design: an efficient design methodology for IoT intelligence on the edge[C]// Proceedings of the 56th ACM/IEEE Design Automation Conference, Las Vegas, USA, 2019:1-6

[2] CAI H, GAN C, WANG T Z, et al. Once-for-all: train one network and specialize it for efficient deployment[C]// Proceedings of the 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, 2020:845-857

[3] GAO M Y, PU J, YANG H, et al. TETRIS: scalable and efficient neural network acceleration with 3D memory[C]// Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems, Xi'an, China, 2017:751-764

[4] SONG L, WANG Y, HAN Y, et al. C-Brain: a deep learning accelerator that tames the diversity of CNNs through adaptive data-level parallelization[C]// Proceedings of the 53rd Annual Design Automation Conference, Austin, USA, 2016:1231-1236

[5] WANG Y, HAN Y H, LI X W, et al. A retrospective evaluation of energy-efficient object detection solutions for embedded devices[C]// Proceedings of Design, Automation and Test in Europe Conference, Dresden, Germany, 2018:709-714

[6] KISEOK K, ALON A, AMIR G, et al. Co-design of deep neural nets and neural net accelerators for embedded vision applications[C]// Proceedings of the 55th ACM/IEEE Design Automation Conference, San Francisco, USA, 2018:1-6

[7] REAL E, AGGARWAL A, HUANG Y P, et al. Regularized evolution for image classifier architecture search[C]//IEEE Conference on Artificial Intelligence, Honolulu, USA, 2019:4780-4789
[8] HOWARD A, PANG R, MARK S, et al. Searching for MobileNetV3[C]//IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 2019:1314-1324
[9] ZHANG Y G, FU Y G, JIANG W W, et al. DNA: differentiable network-accelerator co-search[EB/OL]. http://arxiv.org/abs/2010.14778: arXiv, (2020-10-28), [2021-06-27]
[10] 王佩琪. 神经网络软硬件协同加速关键技术[D]. 北京:清华大学计算机科学与技术系, 2021:10-38
[11] SOTOUDEH M, BAGHSORKHI S S. C3-Flow: compute compression co-design flow for deep neural networks[C]//Proceedings of the 56th Annual Design Automation Conference, Las Vegas, USA, 2019:88-96
[12] KANG J, DOWHAN C, KWANGHYUN C, et al. Fast performance estimation and design space exploration of manycore-based neural processors[C]//Proceedings of the 56th ACM/IEEE Design Automation Conference, Las Vegas, USA, 2019:72-78
[13] WU B, DAI X, ZHANG P, et al. FBnet: Hardware-aware effcient ConvVet design via differentiable neural architecture search[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, USA, 2019:10734-10742
[14] CAI H, ZHU L, HAN S. ProxylessNAS: direct neural architecture search on target task and hardware[EB/OL]. https://arxiv.org/pdf/1812.00332.pdf: arXiv, (2019-02-23),[2021-06-27]
[15] YANG Y F, HUANG Q J, WU B C, et al. Synetgy: algorithm-hardware co-design for ConvNet accelerators on embedded FPGAs[C]//Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Program-mable Gate Arrays, Seaside, USA, 2019: 23-32
[16] JI Y, ZHANG Y H, LI S C, et al. Neutrams: neural network transformation and co-design under neuromorphic hardware constraints[C]//IEEE/ACM International Symposium on Microarchitecture (MICRO), Taibei, China, 2016:1-13
[17] LI Y H, HAO C, ZHANG X F, et al. Edd: efficient differentiable DNN architecture and implementation co-search for embedded AI solutions[C]//Proceedings of the 57th ACM/IEEE Design Automation Conference, San Francisco, USA, 2020:1-6
[18] LIU H X, KAREN S, YANG Y M, et al. DARTS: differentiable architecture search[C]//The 7th Interna-tional Conference on Learning Representations, New Orleans, USA, 2019:218-226
[19] SUTTON R S, MCALLESTER D, SINGH S, et al. Policy Gradient Methods for Reinforcement Learning with Function Approximation[M]. Cambridge:MIT Press,1999
[20] 罗人千. 高效神经网络结构搜索算法及应用[D]. 合肥:中国科学技术大学计算机科学与技术学院, 2021:32-65
[21] TAN M X, CHEN B, PANG R M, et al. MnasNet: platform-aware neural architecture search for mobile[C]//IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, USA, 2019:2820-2828
[22] CHEN Y H, KRISHNA T, JOEL E, et al. Eyeriss: an energy-efcient reconfgurable accelerator for deep convolutional neural networks[J]. *IEEE Journal of Solid State Circuits*, 2017, 52:127-138
[23] ZOPH B, VIJAY V, JONATHON S, et al. Learning transferable architectures for scalable image recognition[C]//IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018:8697-8710
[24] LIU C X, ZOPH B, NEUMANN M, et al. Progressive neural architecture search[C]//Proceeding of the European Conference on Computer Version, Munich, Germany, 2018:19-35

Neural network architecture and accelerator co-design for high energy-efficient scenarios

CHEN Weiwei[*,**], WANG Ying[*], ZHANG Lei[*]
([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
([**] University of Chinese Academy of Sciences, Beijing 100049)

Abstract

Neural network architecture and hardware accelerators have been two driving forces for the rapid progress in deep learning. However, previous work has optimized either neural architectures given fixed hardware, or hardware give fixed neural architectures. The design of neural network structure algorithm focuses on the accuracy, and does not take the characteristics of accelerator hardware into consideration. The accelerator design is generally aimed at specific Benchmark and does not support the new network structure, which makes the hardware design lag behind the algorithm update. At the same time, deep learning has a variety of application scenarios, and different scenarios have different software and hardware requirements. Therefore, special design of software and hardware is required for special scenarios, which requires a lot of labor costs and expert knowledge. This paper studies the importance of co-designing neural architectures and hardware accelerators. To this end, an automatic framework that jointly searches for the best configuration for both neural network architecture and accelerator is proposed. This framework combines the network architecture and accelerator design space, then searches the co-design solution given the design constraints automatically, thus providing better performance opportunities than previous approaches that design the network and accelerator separately. The experiments show that, compared with previous method, joint optimization can reduce the average energy consumption by 40% in a real image classification task under the some level of accuracy constraints.

Key words: neural network architecture search, accelerator design, hardware/software co-design, design space exploration