NASCaps:神经架构搜索框架,用于优化神经网络的准确性和硬件效率卷积胶囊网络

阿尔贝托·马尔基西奥*1 ,安德烈·马萨*2 ,Vojtech Mrazek3 ,比阿特丽斯布索里诺2 ,毛里齐奥·玛蒂娜2 ,穆罕默德·沙菲克1,4
[1]计算机工程研究所,维也纳技术大学,维也纳,奥地利
2都灵理工大学电子与电信系,都灵,意大利
[3]信息技术学院,布尔诺科技大学,布尔诺,捷克4 纽约大学阿布扎比工程系,阿联酋
alberto.marchisio@tuwien.ac.at, a.massa@studenti.polito.it, mrazek@fit.vutbr .cz, {beatrice.bussolino, maurizio.martina}@polito.it, muhammad.shafique@nyu.edu

抽象的

深度神经网络 (DNN) 已做出重大改进,以达到在各种机器学习 (ML) 应用程序中采用的所需精度。最近,Google Brain 的团队展示了胶囊网络 (CapsNets) 编码和学习不同输入特征之间的空间相关性的能力,从而获得比传统(即非基于胶囊的)DNN 更优越的学习能力。

然而,使用传统方法设计 CapsNets 是一项乏味的工作,并且需要大量的训练工作。最近的研究表明,为一组给定的应用程序和训练数据集自动选择最佳/最佳 DNN 模型配置的强大方法基于神经架构搜索 (NAS) 算法。此外,由于对计算和内存的极端要求,DNN在 IoT-Edge/CPS 设备中使用专门的硬件加速器。

在本文中,我们提出了 NASCaps,这是一种用于不同类型 DNN的硬件感知 NAS 的自动化框架,涵盖传统的卷积 DNN 和 CapsNet。我们研究部署多目标遗传算法(例如,基于 NSGA-II 算法)的功效。

所提出的框架可以联合优化网络准确性和相应的硬件效率,以执行 DNN 推理的给定硬件加速器的能量、内存和延迟来表示。

除了支持传统的 DNN 层(例如卷积层和全连接层),我们的框架是第一个建模并支持 NAS 流中的专用胶囊层和动态路由的框架。我们在不同的数据集上评估我们的框架,生成不同的网络配置,并展示不同输出指标之间的权衡。我们将在 https://github.com/ehw-fit/nascaps 上开源Pareto 最优架构的完整框架和配置。

关键词

深度神经网络、DNN、胶囊网络、进化算法、遗传算法、神经架构搜索、硬件加速器、准确性、能效、内存、延迟、设计空间、多目标、优化。

*这些作者的贡献相同。

ICCAD 20,2020 年 11 月 2-5 日,虚拟活动,美国 © 2020 Association for Computing Machinery。
美国计算机学会书号978-1-6654-2324-3/20/11。..
15.00 美元https://doi.org/10.1145/3400302.3415731

ACM 参考格式:
阿尔贝托·马尔基西奥*1 ,安德烈·马萨*2 ,Vojtech Mrazek3 ,比阿特丽斯布索里诺2 ,毛里齐奥·玛蒂娜2 ,穆罕默德·沙菲克 1,4 。 2020. NASCaps:用于优化卷积胶囊网络的准确性和硬件效率的神经架构搜索框架。在 IEEE/ACM计算机辅助设计国际会议 (ICCAD 20),2020 年 11 月 2 日至 5 日,虚拟活动,美国。 ACM,美国纽约州纽约市,9 页。 https://doi.org/10。 1145/3400302.3415731

1 简介深度神经网络 (DNN) 是

基于机器学习的高级算法,在对大型数据集进行广泛训练时,声称在特定任务集(例如图像分类、对象识别、检测和跟踪)中的准确性超过人类水平 [4] [7][26]。为一组给定的应用程序和给定的数据集设计最佳 DNN 是一项极具挑战性的工作。

高精度的 DNN 需要大量的硬件资源,这在资源受限的 IoT Edge 设备上可能不可行 [2][18][28][32]。称为胶囊网络 (CapsNets) [27] 的高级 DNN 模型能够以更接近我们目前对人脑功能的理解的方式学习输入特征的层次和空间信息。

然而,由胶囊构成的层相对于传统卷积神经网络 (CNN) 的卷积层和全连接层的矩阵引入了额外的维度,除了动态路由计算之外,还给底层硬件带来了极其繁重的计算和通信工作量[15]。在图 1 中,我们将 CapsNet [27] 与 LeNet [13] 和 AlexNet [12] 进行了比较,并分析了它们的内存占用和执行推理过程所需的乘法和累加 (MAC) 操作总数。请注意,MAC/内存比率是显示网络计算复杂性的良好指标,从而证明了与传统 CNN 相比,CapsNet 具有更高的计算密集型特性。

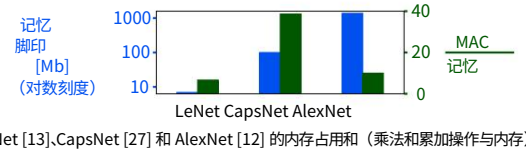


图 1:LeNet [13]、CapsNet [27] 和 AlexNet [12] 的内存占用和(乘法和累加操作与内存)比率(MAC/内存)。

1.1 研究问题和相关挑战在文献中,已经提出了许多 DNN 模型/架构 [6] [12][29][33][34]。在深度学习的开拓时代,DNN 架构是手动设计的。

然而,它们的结构变得非常复杂。因此,神经架构搜索 (NAS) 方法成为一种有吸引力的程序,可以为给定的应用程序集和训练数据集选择最佳 DNN 模型 [37]。提出了基于进化算法 (EA) 的方法 [31] 来学习小型 DNN。然而,对于更复杂的 DNN,EA 已用于改进单个/单独 DNN 层 [22] 的某些参数,以及全局 DNN 超参数 [21] 或子模块之间的连接 [36]。

文献中提出的大多数基于 NAS 算法的自动化工具只专注于优化 DNN 精度 [31][38]。他们中只有少数人最近在优化问题中引入了硬件约束,例如,考虑可用于执行 DNN 推理的硬件资源（例如,#FLOP、内存要求等)[9][10][14] [30]。据我们所知,它们都没有在设计空间中包含使用胶囊层和动态路由的可能性,这对于自动设计 CapsNets 是不可避免的。

为此,我们提出了NASCaps,这是一个 DNN NAS 的框架,它不仅包含最常见类型的 DNN 层（例如卷积层、全连接层）,而且还首次包含了不同类型的胶囊层。我们的框架支持多目标硬件感知优化,因为它调查网络准确性,并考虑了对嵌入式 DNN 推理加速器至关重要的不同硬件效率参数（例如内存使用、能耗和延迟）。

然而,为了获得详尽的帕累托最优解决方案而应该探索的大量可能配置可能会急剧增加。除此之外,尽管采用了最先进的学习策略并使用了高端 GPU 集群,但复杂的 CapsNet 和 CNN 通常需要很长的训练时间 [5][19]。由于仿真时间长,完整详细的综合后硬件测量对于此搜索是不可行的。上述限制挑战了这种探索在实际案例 HW/SW 协同设计搜索中的适用性,具有严格的上市时间限制。

1.2 我们的主要研究贡献为了应对上述挑战,我们设计了不同的优化并将它们集成到我们的 NASCaps 框架中（图 2）。以下新颖的贡献总结了这些步骤:

- 我们提出了一个名为NASCaps 的框架,用于基于卷积层和胶囊层自动搜索 DNN 模型架构配置。（第3 节） · 我们以参数化方式对 CapsNet 架构中涉及的操作进行建模,包括不同类型的胶囊层和动态路由。（第 3.1 节）

- 我们在高层对给定的专用 CNN 和 CapsNet 硬件加速器的功能行为进行建模,以便在执行不同的 DNN 架构模型时快速估计内存使用量、能耗和延迟。（第 3.2 节） · 基于 NSGA-II 方法 [3],我们开发了一种专门的多目标遗传算法来解决本文针对的优化问题,即 DNN 架构的多目标帕累托边界选择,同时优化神经网络的准确性、能量消耗、内存使用和延迟。（第 3.3 节） · 为了减少探索不同解决方案的训练时间,我们提出了一种方法来评估部分训练的 DNN 的准确性。训练时期的数量是基于训练时间和 Pearson 相关系数 wrt 完全训练的 DNN 之间的权衡选择。（第 4.2 节） · 在探索阶段,我们训练和评估了 600 多个候选 DNN 解决方案,这些解决方案在配备四个高端 Nvidia V100-SMX2 GPU 的 GPU HPC 计算节点上运行。我们的 NASCaps 框架生成的 Pareto 最优解决方案与之前 CapsNet 的 SoA 精度值（即 DeepCaps [25])相比具有竞争力,同时提高了相应的硬件效率,从而为在边缘。（第 4.4 节） · 为了鼓励 DNN 研究社区的快速进步,我们将在 https://github.com/ehw-fit/nascaps 上开源完整的 NASCaps 框架和帕累托最优架构的配置。

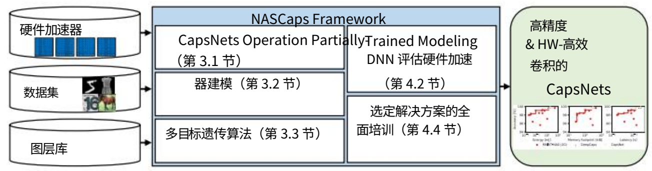


图 2:我们的NASCaps框架概览。

在进入技术部分之前,我们简要概述了 CapsNet 和执行 CapsNet 推理的硬件加速器。（第2 节）

2 背景:胶囊网络

将神经元分组形成胶囊的想法首先由 G. Hinton 等人提出。在[8]中。胶囊的目的是保留实体或特征的实例化概率,以及有关其实例化参数的信息,例如位置、旋转或宽度。相反,传统的神经元只能检测特征而不知道它们的空间特征和相互关系。

第一个带有胶囊的 DNN,即胶囊网络 (CapsNet),由 Google Brain 团队在 [27] 中提出。在这个模型中,胶囊内的神经元排列成向量的形状。

每个神经元编码与胶囊相关联的实体的空间参数,向量的长度表示对象存在的概率。 [27] 中的 CapsNet 由三层组成,如图 3.a 所示: (1) Conv 层:一个卷积 (Conv) 层,它将步幅为 1 的 9x9 内核应用于输入图像并产生 256 个输出通道。

(2) PrimaryCaps 层:一个 Conv 层,它应用步幅为 2 的 9x9 内核并产生 256 个输出通道。输出通道分为 32 个通道的 8-D 胶囊。由于胶囊的长度编码概率,因此应用挤压函数（等式 1)以强制长度在 [0,1] 范围内。 |x| (1 + |x|)2 |x| (3) DigitCaps 层:具有 10 个输出 16-D 胶囊的全连接 (FC) 胶囊层（对于 10 类数据集）。 DigitCaps 层执行动态路由,这是一种将耦合系数与从 PrimaryCaps 层获得的预测相关联的迭代算法。该算法的迭代次数最大化了胶囊的实例化概率)预测相同的结果（相似的空间参数）。

$$y = \frac{\|x\|^2}{(1+\|x\|^2)} \frac{x}{\|x\|} \quad (1)$$

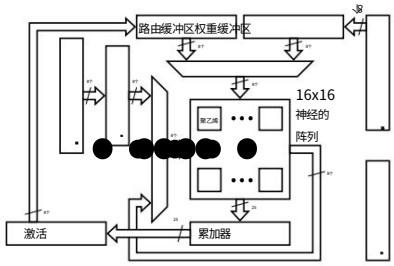
CapsNets 属于特定类别的 DNN,即自动编码器,可以重建它们作为输入接收的图像。

因此,上面列出的三层之后是一个解码器,由三个 FC 层组成,它们从 DigitCaps 层的输出重建输入图像。在训练期间,计算了两个损失,即分类网络的损失和重建网络的损失。当出于分类目的执行推理时,可以删除重建网络。

为了进一步减少处理时间和能量,量化 [20] 和近似 [17] 可以应用于 CapsNets 的推理,而不会降低准确性。

最近在 [25] 中提出了 DeepCaps,一种新颖的更深的 CapsNet 架构。如图 3.b 所示,该架构由传统的 Conv 层、FC 胶囊层 (cFC) 和 Conv 胶囊层 (cConv) 组成。后者按块排列,这里称为 CapsCells（图 3.c），其中一个 cConv 层与两个 cConvs 层并行运行。用作重建网络的解码器由一系列反卷积层组成。DeepCaps 还在最后一个 CapsCells 之间引入了一个跳跃连接,以减轻影响更深层网络的梯度消失效应。

胶囊层涉及传统 DNN 不执行的操作,因此,它们不受现有 DNN 硬件加速器的支持。

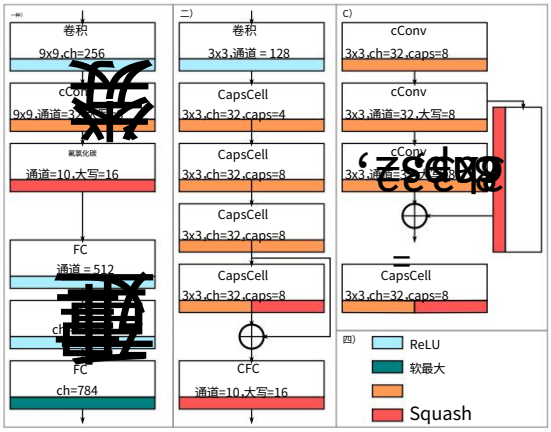


此外,CapsNets 需要修改数据图 4:映射的架构视图。

CapsAcc，

CapsAcc [15] 加速器。[15]

中提出了一个针对 CapsNets 加速的硬件平台,如图 4 所示。CapsAcc 的计算核心由一组处理元素 (PE) 组成,后面是一个适当添加部分和的累加器。然后有一个激活单元可以将 ReLU、softmax 或 squash 函数应用于累加器的输出。激活和权重分别存储在数据和权重内存中,在计算过程中使用了三个缓冲区,以利用数据重用并最大限度地减少对较大内存的访问。特别是,数据和权重缓冲区存储激活和权重,以及路由缓冲区



Dynamic Routing图 3:(a) CapsNet 模型 [27] 和用于图像重建的解码器； (b) DeepCaps模型[25]（解码器省略）； (c) DeepCaps模型中使用的CapsCell； (d) 每层中使用的激活函数的图例。

用于存储动态路由迭代的部分结果。控制单元选择用于将不同层映射到PE阵列的路径。高效的片上内存管理 [16] 可以进一步降低其能耗。

3 NASCAPS:神经结构

搜索卷积 CAPSNET

我们的多目标 NASCaps 框架通过执行多目标 NAS 来生成和评估基于卷积和胶囊的 DNN,以找到一组准确但资源高效的 DNN 模型,即联合考虑 DNN 验证准确性、能耗、延迟和内存占用。该搜索基于我们对遗传 NSGA-II 算法 [3] 的专门化,以实现在生成的候选 DNN 中进行多目标比较和选择的搜索。

NASCaps 框架的整体结构和工作流程如图 5 所示。作为输入,它接收底层硬件加速器的配置（将在真实场景中执行生成的 DNN)和用于 DNN 训练的给定数据集,以及可用于形成不同候选 DNN 的可能层类型的集合。首先,我们创建一个层库,其中包括卷积层、胶囊层（如 [27] 中定义的)以及 [25] 中定义的 CapsCell 和 FlatCaps 层。我们设想,由于我们框架的模块化结构,其他类型的层可以很容易地集成到其未来版本中以进一步扩展搜索空间,这也归功于使用依赖于候选网络的简单模块化表示单层描述符的组合,如第 3.1 节所述。

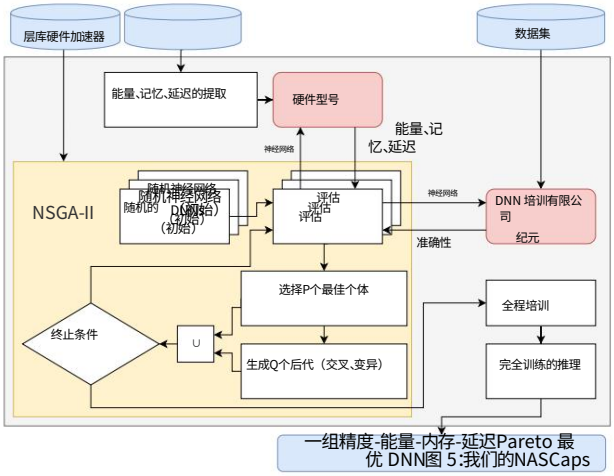


图 5:我们的NASCaps框架概述,显示了定义工作流的不同组件及其互连。

自动搜索使用随机生成的 DNN 进行初始化,用作输入以启动进化搜索过程。每个候选 DNN 在经过有限数量的 epoch 训练后,根据其验证准确性进行评估。正如我们将在第 4.2 节中讨论的那样,这种优化旨在减少计算成本并减少搜索所需的计算时间,同时保持良好的相关性 wrt 全训练精度,用 Pearson 相关系数测量。此外,通过考虑在专用 DNN 硬件加速器上执行生成的 DNN 的最终真实世界用例,对其推理处理进行建模,每个被测 DNN 还具有能耗、延迟和内存占用的特征。在这个评估点,遗传

算法继续下一步,在每次迭代中找到一个包含最佳候选 DNN 解决方案的新帕累托边界。在此选择过程结束时,帕累托最优 DNN 解决方案经过全面训练 1，以进行准确的准确性评估。在以下小节中,我们将详细讨论框架的关键组件。

3.1 胶囊网络层和架构的参数化建模所提出的基于遗传的 NASCaps 框架

依赖于候选 DNN 每一层的显式基于位置的表示。这种表示允许唯一地定义每一层的关键参数。

DNN 层是使用层描述符构建的,它以非常紧凑的形式对构建和建模给定候选网络所需的信息进行编码。每个层描述符都是一个基于位置的 9 元素结构，从而保证了构建任何给定候选 DNN 架构的模块化。

层描述符的元素如下： (1)层的类型，(2)输入特征映射的大小，(3)输入通道数 (4)输入胶囊的数量 (5)内核大小 (6) stride size (7) 输出特征图的大小 (8) 输出通道数 (9) 输出胶囊数 这种表示允许通过简单地定义新的层类型来描述更复杂的结构。例如,一个层描述符可以定义一个更复杂的重复结构,由多个元素组成,就像 DeepCaps 架构中的 CapsCell。这样,DeepCaps 架构已经用六层描述符进行了描述。第一个用于单个卷积层、四个 CapsCell 块和一个最终的 Class Capsule 层。

完整的 DNN 架构描述然后由两个非层术语完成,允许对跳过连接的位置进行编码,以及一个称为调整大小标志的指示符,以明确指示是否需要调整输入的大小。图 6 显示了为描述候选 DNN 架构而提出的格式,从现在开始,它被称为基因型。

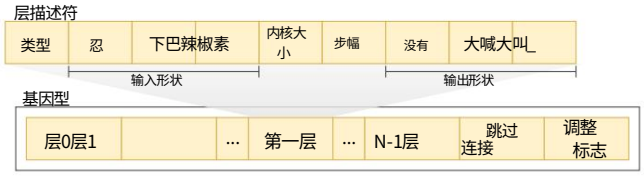


图 6:基因型的拟议结构。

3.2 在硬件加速器中对 CapsNet 的执行进行建模

NASCaps 框架可以接收任何给定的执行 DNN 推理的硬件加速器作为输入。为了便于说明,我们展示了 CapsAcc 加速器的建模;这个选择与它支持所有胶囊层的执行有关。从 CapsAcc 架构的 RTL 级描述开始,我们提取和建模不同的微架构更高抽象级别的配置,构成我们模型的输入。首先,介绍了层的操作特定参数的描述。然后,讨论了与 CapsAcc 加速器严格相关的全局参数。

3.2.1 不同层的特定操作建模。

可以从硬件中不同操作的执行中提取的特定于操作的参数如下： :层中的权重数， :要为输出值添加的项数， :特征图的数量乘以相同的权重。

对于每个操作,这些参数由不同的方程式计算,由于各自类型的计算的性质不同,请参见表 1。请注意,通过设置为 1,ConvCaps 和 ClassCaps 层成为传统的卷积层和全连接层层,分别。

3.2.2 全局参数建模。

对于给定的 CapsNet,我们的模型估计了一个输入的推理的延迟和能量消耗，而内存占用被计算为每一层的权重数量的总和。它们以模块化方式（即自下而上)为每个操作建模。首先,必须将权重加载到 PE 数组中,然后只要它们需要与其他输入相乘即可重复使用。之后,加载下一组权重,直到完成所有层的计算（参见等式 2-4）。

通过将结果与 CapsAcc [15] 加速器的硬件实现进行比较,该模型得到了验证。采用的模型参数如下：

:将权重加载到 PE 阵列所需的时钟周期数，

: 加载到 PE 数组上的权重组数，

( ): 执行层所需的周期数, : 内存访问次数, : 单次内存访问的能耗, : PE阵列的功耗。

$$= 16 \tag{2}$$

$$= \frac{}{16 \cdot \text{分钟}(16, \quad)} \tag{3}$$

$$( ) = \cdot + \tag{4}$$

然后将总延迟计算为各层贡献的总和（等式 5）。

$$= \sum_{\in} ( ) \cdot \tag{5}$$

在等式中。 6,通过区分操作是否为卷积层来计算内存访问次数。

这种区别是通过分析 的值来实现的,它对于卷积层大于 1,否则为 1。

$$= \begin{cases} 256, & \text{如果 } = 1 \\ 16 \cdot \text{最大}(- 15, 1), & \text{否则} \end{cases} \tag{6}$$

加速器的能量（公式 7)估计为内存访问能量和 PE 阵列中处理的每一层的功耗之和乘以其延迟（周期和周期数）。请注意,我们的模型中使用了 PE 阵列的平均功耗。

$$= \frac{\cdot 8}{128} \cdot + \sum_{\in} ( ) \cdot \cdot \tag{7}$$

1对 MNIST、Fashion-MNIST 和 SVHN 数据集进行了第 100 个训练，对 CIFAR-10 数据集进行了第 300 个训练。

表 1:CapsNets 特定操作建模的方程式。

手术			
ConvCaps 层(·	+ 1) · · ·	+ 1) · ·	) · ·
ConvCaps3D 层(·	+ 1) · · ·	((+ 1) · ·	(( ) · ·
ClassCaps 层(·	+ 1) · ·	(+ 1) · ·	
动态路由	· · · ·		

3.3 多目标NSGA-II 算法NASCaps 框架的帕累托最优解的选择是基于进化算法NSGA-II [3]。

它有一个主循环（算法 1 的第 2-14 行），其迭代代表初始种群的单代整体进化过程。初始种群（大小为 n)是随机生成的,可以称为1 （算法 1 的第 1 行）。

这组解代表算法的初始父代。属于（第 3 行)的解决方案之间的交叉允许生成一组新的后代个体。

此时,种群 ∪ 根据非支配标准进行排序。对于内循环的每次迭代（第 6-13 行），候选解决方案被分组到不同的前沿。包含在第一个前面1中的那些代表了总体中找到的最佳解决方案。每个后续前沿 (2, 3, . .) 都是通过从种群中移除所有前面的前沿然后找到一个新的帕累托前沿来构建的。由于第一个前沿始可能包含少于 n 个个体组成,因此后续前沿的解决方案也将被选为下一代父代的一部分。

为了在输出集中恰好有 n 个父母,属于最后一个前沿的解决方案使用拥挤距离比较方法（第 11 行)进行排序,该方法包括根据每个目标函数值按升序对该前沿的人口进行排序.这些步骤如图 7 所示。只有一半的种群成为下一代父代的一部分,而另一半则被丢弃。

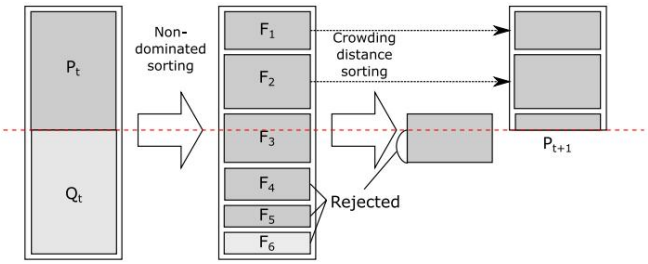


图 7:人口排序。

这些步骤重复一定数量的世代。
算法 1 中报告了完整的伪代码,其中使用了以下过程：

- ( ) 随机生成属于搜索空间的配置。(,) 通过交叉和变异从父母那里产生新的后代（在 3.3.1 节中描述）。( ) 评估新的候选解
- 从一组。
- ( ) 从集合中选择帕累托最优解,并将这些解从集合中移 ,除。
- (,) 返回集合中的解 (作为 [3] 中描述）。

多目标算法的优势在于它在每一代都重新构建 Pareto 前沿,旨在覆盖所有可能的解决方案。该算法的输出是一组非支配解。

算法 1:我们的NASCaps框架中使用的遗传 NSGA-II 算法。

要求:搜索空间,人口规模|几代人 |, | |, 数量

确保:帕累托集 ⊆ 1 × 2 × · · · ×
1: 1 ← 2:对 (| |)
于= 1 3: ← . . . 做
(, | |) (∪)
4: ←
5: +1 ← ∅ 6:
同时| +1 | < | |做( )
7: = 如
果| +1 | + | | ⩽ | |然后+1否则
8:9:10:11: +1 ← +1 ∪如 (, | | - | +1 |)
12: 果结束
13:结束 while 14:结束
15:返回 ( )

3.3.1 交叉和变异操作。

遗传算法进程中的两个关键算子是交叉和变异。标准的单点交叉操作允许生成后代解决方案,给定两个父解决方案并且之前已经在当前种群候选者中随机挑选。两个亲本个体的基因型分别分为两部分。分裂点是伪随机选择的。最初,随机选择一个切割点。然后,执行一系列检查以验证输出基因型的有效性。已应用以下标准来正确选择拆分点：
· 选择切点以确保生成的 DNN 由至少一个初始卷积层和至少 2 个胶囊层组成， · 未放置卷积层在两个胶囊层之间。

请注意,第二个约束背后的原因是胶囊旨在通过卷积层获得更高级别的信息。
至此,进行了真正的交叉操作。如图 8 所示,亲本基因的最后部分和被交换。

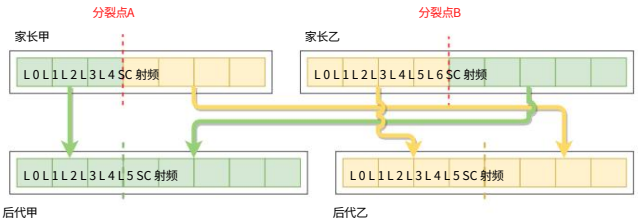


图 8:两种基因型之间的交叉示例。

该算法执行的第二个关键操作是变异。由于它已在我们的 NASCaps 框架中实现,操作员通过从输入候选网络的基因型中随机选择一个层描述符,并以概率随机修改所选层的主要参数之一来执行突变。

特别是,可能受突变影响的参数是内核大小,

步幅、输出胶囊的数量和跳过连接的位置。

在这两个操作之后,执行进一步的步骤以确保输出基因型的有效性,在大量情况下,这将代表无效的 DNN。此校正步骤允许为源自突变或交叉操作的基因型的每一层正确调整输入和输出张量维度,这可以随机修改或将不同的父基因型连接在一起。

4 评估我们的 NASCAPS 框架

4.1 实验设置我们的实验设置和

工具流程概览如图 9 所示。候选 DNN 的准确性训练和测试是使用 TensorFlow 库 [1] 进行的,同时使用 GPU-配备四个 NVIDIA Tesla V100-SXM2 GPU 的 HPC 计算节点。我们提出的 NASCaps 框架已经针对 MNIST [13]、Fashion MNIST (FMNIST) [35]、SVHN [23] 和 CIFAR-10 [11] 数据集进行了评估。 HW 模型的实现基于开源 SoA 胶囊网络加速器 (CapsAcc) [15],如第 3.2 节所述。使用具有 45nm 技术节点和 3ns 时钟周期的 Synopsys Design Compiler 综合了核心处理元件。

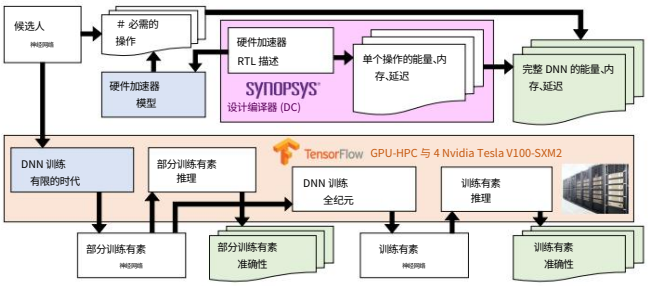


图 9:进行实验的设置和工具流程。

实验分为三个步骤。 (1) 在第一步中,执行了基本的随机搜索,以调查训练候选 DNN 需要多少个训练周期,并评估它们在遗传 NSGA II 算法循环中的准确性。 (2) 在第二步中,执行用于为能量、内存、延迟和精度目标寻找帕累托最优 DNN 架构的搜索算法。 (3) 最后,对选定的 Pareto 最优 DNN 进行了充分训练。为了评估所选 DNN 对不同数据集的可迁移性,所选 DNN 也针对其他数据集进行了全面训练。

此外,以下设置已用于进行实验:

· 初始父母种群大小 | | = 10 · 后代种群大小: | | = 10 · 遗传环的最大世代数 突变概率: ∈ {3 × 3, 5 × 5, 9 × 9} ∈ {1, 2} · ∈ {1, 2, . . . , 64} ∈ {1, 2, . . . , 64} = 10%

-
-
-

这些值以及训练超参数(例如,批量大小、时期数和学习率)是通过进行一组初步实验并考虑合理的运行时间来选择的。

4.2 全训练精度估计减少训练时期的结果

与 NAS 问题相关的最关键方面之一在于其高计算探索成本。这是由于构成种群的候选网络数量众多,以及评估准确性所需的耗时训练步骤。

为了限制执行完整搜索所需的时间及其计算成本,我们提出了一种两阶段评估方法。 (1) 第一步包括用有限的 epoch 训练候选网络的数量,产生一组部分训练的 DNN。部分训练的 DNN 获得的验证精度已用于评估 NSGA-II 算法中的帕累托前沿,如第 3.3 节所述。通过分析不同时期大小对不同数据集实现的准确性的影响,仔细确定了时期数量的选择。 (2)

之后,在 Pareto-front 中显示其准确性和硬件效率的候选网络经过全面训练以评估其实际验证准确性。

因此,这种方法允许仅使用减少数量的训练时期来预测 DNN 的完整训练准确性。

这种方法已经使用 66 个随机生成的 DNN(除了 CapsNet 和 DeepCaps 架构)进行了测试,并对它们进行了全面训练,同时记录了每个训练时期获得的验证准确性。已计算 Pearson 相关系数 ( ) [24],以分析完全训练的 DNN 的准确性与相同 DNN 在中间步骤的准确性之间的相关性。

表 2 显示了值 ,在训练时期后的 DNN 准确度与完整训练后的准确度之间进行计算。还报告了执行时代训练所需的中位数累积训练时间。正如预期的那样,这项研究允许确定更复杂的数据集需要更多的训练时期才能正确区分最有希望的网络和其他网络。对于 MNIST 数据集的情况,5 个时期足以达到等于 0.9999。相反,对于 CIFAR-10 数据集,在前几个时期内从未达到如此高的置信度值。在本例中,选择了 10 个训练周期,确保 a 等于 0.9334。这种选择利用了相关系数和所需训练时间之间的权衡。当然,也可以选择更多的训练时期,但由于 DNN 训练,这会大大增加探索时间,这是 NASCaps 框架探索大量种群和/或世代[illegible]的关键步骤。的选择允许丢弃比 5 个时期后丢弃的更多的 Pareto 支配的候选网络。对于 Fashion-MNIST 和 SVHN 数据集,选择阶段也在 5 个训练周期后进行。

表 2:MNIST、Fashion-MNIST (FMNIST)、SVHN 和 CIFAR-10 数据集的 Pearson 相关系数 (PCC) 和以秒表示的中位累积训练时间 (MCTT)。

纪元				10	15	20
MNIST	PCC 0.8407 0.9998 0.9999 1.0000 1.0000 1.0000 277.0 831.0 1108.0					
	MCTT 55.4	166.2	PCC 0.8305 [illegible]			
FMNIST	862.3 1293.4 1724.6 PCC 0.6812 0.8733 0.9518 0.9531 0.9667					
	MCTT 86.2	258.7 431.1		0.9876 MCTT 128.3		
SVHN		385.0	641.6 1283.3 1924.9 2666.6			
CIFAR-10	PCC 0.2969 0.4259 0.7279 0.9334 0.9518 0.9879					
	MCTT 61.6	184.7	307.9	615.8	923.6 1231.5	

请注意,一组特定的网络可以相对较早地被丢弃,即在几个训练时期之后,因为它们没有改进它们

准确率多。通过选择阶段的候选网络可以完成训练。第二个选择阶段有利于对候选网络进行更细粒度的选择,并避免对 Pareto 支配的 DNN 进行繁琐且计算量大的全面训练。

4.3 部分训练的 NASCaps 结果

深度神经网络

我们的 NASCaps 框架首先应用于 MNIST 数据集,以评估其效率和正确行为。世代数设置为 20,但 MNIST 和 Fashion-MNIST 数据集的最大超时时间为 12 小时,而 CIFAR-10 和 Fashion-MNIST 数据集的最大搜索时间为 24 小时 SVHN 数据集。

对 MNIST-NAS 的搜索持续了 20 个完整的世代,单个候选网络被训练了 5 个 epoch。此设置导致训练和评估总共 210 个 DNN。将生成的单个解决方案与两个参考 SoA 解决方案（即 CapsNet 和 DeepCaps 架构)进行比较。

在图 10a 中,每个单独的 DNN 架构都代表了搜索的四个目标。

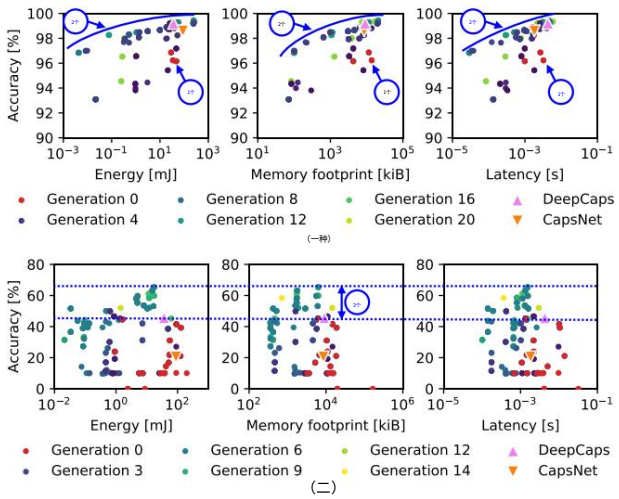


图 10:针对 (a) MNIST 数据集和 (b) CIFAR-10 数据集的部分训练的 DNN NAS。颜色显示解决方案首先出现在哪一代。

Fashion-MNIST 搜索在其第 19代（12 小时内)结束,并评估了总共 200 个候选架构。对 SVHN 数据集的搜索持续了 12 代,它允许评估 130 个架构。对于 CIFAR-10 数据集,搜索达到了第 14代,共有 150 个测试架构。

图 10 显示了 MNIST 和 CIFAR-10 数据集的进化搜索算法的进展情况。请注意,红点,即第 0 代的初始种群（参见图 10a 中的指针1 ）,代表随机生成的 DNN。当我们的进化算法迭代地使用交叉和变异操作找到更好的候选 DNN 架构时,目标在随后的迭代中显着改进（参见指针2 ）。

减少的 epoch 训练允许评估大量基于卷积层和胶囊层的候选网络（总共近 700 个架构）。这种方法导致找到多个候选架构,这些架构能够达到比部分训练的 SoA 解决方案中最好的高出 30.86% 的准确度,即在大大降低的限制范围内

训练时间。例如,CIFAR-10 数据集的 NAS 在 10 个 epoch 后生成的网络精度为 76.46%,而 DeepCaps 架构在相同的训练间隔内仅达到 45.60% 的精度（见图 10b 中的指针3 ）。这证实了这样一个事实,即当受限于较短的训练时间时,与类似 DeepCaps 的结构相比,我们的 NASCaps 可以生成精度更高的网络。

4.4 选定的完全训练的 DNN 的 NASCaps 结果在第一个选择

阶段之后,属于帕累托最优子集的候选 DNN 已经过完全训练以评估其最终准确性。图 11 显示了完整训练过程结束时的帕累托最优解。

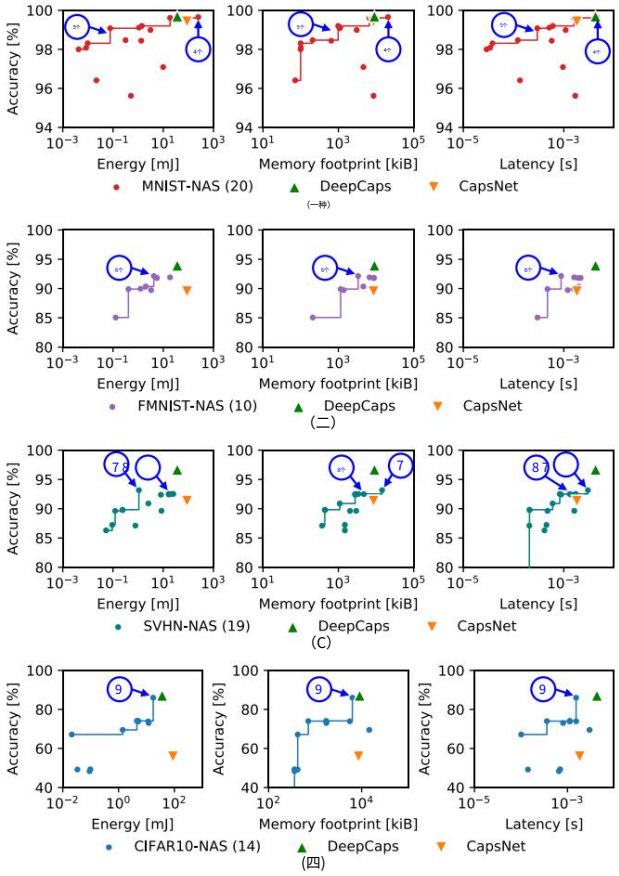


图 11:(a) MNIST、(b) Fashion-MNIST、(c) SVHN 和 (d) CIFAR-10 数据集的完全训练 DNN 结果。

4.4.1 MNIST 数据集的 NASCaps。

在 MNIST 搜索期间发现的最高精度架构（参见图 11a 中的指针4)允许在 93 个训练周期中达到 99.65% 的精度。然而,与 CapsNet 架构相比,该特定解决方案需要多 2.8 倍的能量、多 2.5 倍的时间和多 2.4 倍的内存。红色前沿（见指针5)还突出显示了属于派生帕累托最优前沿的其他有趣解决方案,精度略低,但我们确定的解决方案实现的能量、内存和延迟降低了几个数量级。

4.4.2 Fashion-MNIST 数据集的 NASCaps。

最佳解决方案之一（参见图 11b 中的指针6)在 51 个时期内实现了 92.15% 的准确率。该解决方案改进了

与 CapsNet 和 DeepCaps 架构相比,延迟（-79.38%）、能量（-88.43%)和内存占用（-63.05%）,准确率与最后一个几乎相同,为 93.94%。

4.4.3 SVHN 数据集的 NASCaps。
SVHN 数据集的一组实验产生了一个解决方案（参见图 11c 中的指针7 ）,在 56 个训练周期中,该解决方案的精度达到 93.17%,即比 DeepCaps 低 3.52%。

与 DeepCaps 相比,该解决方案还显着降低了 97.05% 的能量和 29.56% 的延迟,但它需要多 1.6 倍的内存。另一方面,与 DeepCaps 相比,另一个有趣的解决方案（参见指针8)达到了 92.53% 的准确率,同时需要低 30.59% 的能量、低 59.63% 的延迟和 62.70% 的内存。

4.4.4 CIFAR-10 数据集的 NASCaps。
与 DeepCaps 架构相比,CIFAR10-NAS 找到的解决方案（参见图 11d 中的指针9)在 300 个 epoch 的训练后达到了 85.99% 的准确率,同时显着改善了所有其他目标。与在 CapsAcc 加速器上执行的 DeepCaps 相比,这个特定的解决方案（NASCaps-C10- 表 3 中最好的)降低了 52.12% 的能耗,64.34% 的延迟和 30.19% 的内存占用,同时遇到了轻微的精度下降大约 1%,同时使用相同的训练设置。表 3 还报告了我们的 NASCaps 框架为 CIFAR-10 数据集发现的其他帕累托最优 DNN 架构。

表 3:经过 300 个纪元训练后选定的 CIFAR-10 架构。

建筑学	Accuracy Energy Latency Memory 87.10%2 36.30 mJ
深帽 [25]	4.29 ms 9,052 kiB 85.99% 17.38 mJ 1.53 ms 6,319
NASCaps-C10-最佳9	kiB 74.11% 4.53 mJ 1.12 ms 1,718 kiB 74.00%
NASCaps-C10-a0d	5.11 mJ 0.36 ms 713 kiB 73.91% 5.06 mJ 1.54 ms
NASCaps-C10-9fd	5,573 kiB 55.85%2 88.80 mJ 1.82 毫秒 8,573 KB
NASCaps-C10-658	
胶囊网络 [27]	

4.4.5 所选 DNN 跨不同数据集的可迁移性。
为了测试我们的 NASCaps 框架发现的 DNN 解决方案的可转移性,特定数据集发现的 DNN 也在其余考虑的数据集上进行了训练和测试。表 4 报告了通过此可转移性分析获得的最高精度解决方案的矩阵。

表 3 的 NASCaps-C10 最佳架构对于其他数据集的结果也特别准确。对于 MNIST 数据集,它在 37 个 epoch 的训练中达到了 99.72% 的准确率,这也高于 MNIST-NAS 找到的解决方案。对于 Fashion-MNIST 数据集,它在 32 个 epoch 的训练中达到了 93.87% 的准确率,这在训练 100 个 epoch 后甚至高于 DeepCaps。当在 SVHN 数据集上进行测试时,它的准确率达到了 96.59%,因此优于 SVNH-NAS 期间发现的最高精度 DNN。 NASCaps-C10-best 架构类似于 DeepCaps,但它有两个初始卷积层和三个 CapsCell 块,没有跳跃连接。 MNIST-NAS 发现的最高精度架构也被证明可以很好地与 Fashion-NMIST 数据集配合使用,经过 91 个 epoch 的训练后,准确率达到 93.34%。

表 4 中报告的结果显示了解决方案 NASCaps C10 如何成为在执行的四次搜索中找到的最佳整体架构。这是由于多种原因:进化

2注:DeepCaps 和 CapsNet 报告的准确度与 [25] 中报告的准确度并非 100% 匹配。这可以归因于训练超参数设置的差异,因为他们的论文没有披露有关训练的完整深入信息,以确保其结果的可重复性。

表 4:数据集特定 NAS 发现的最高精度 DNN,然后针对其他数据集训练 100 个时期。

建筑学	MNIST FMNIST SVHN CIFAR-10			
NASCaps-MNIST-最佳4	99.65%	93.34%	96.36%	71.44%
NASCaps-FMNIST-最佳6	99.49%	92.15%	93.12%	68.34%
NASCaps-SVHN-最佳7	99.51%	63.72 %	91.43%	93.17%
NASCaps-C10-最佳9	99.72%	93.87%	96.59%	78.46%

过程基于在每次搜索时新生成的随机初始父种群。此外,初始父群体的规模较小可能导致已执行的四个特定于数据集的搜索不收敛。此外,在实验结束时,并非四次搜索中的每一次都达到了同一代。

4.5 关键结果总结以上结果显示了我们的
NASCaps 框架如何能够探索具有不同权衡的多个解决方案,这要归功于使用进化算法进行多目标搜索。

已经可以为四个特定于数据集的搜索生成和测试 690 个候选网络。使用四个高端 NVIDIA Tesla V100-SXM2,我们的 NASCaps 框架需要 90 个 GPU 小时来测试部分训练的候选网络。新的 64 Pareto 最优架构已经过全面训练,总共需要额外的 682 GPU 小时（即 28 天）。尽管对单个搜索应用了严格的时间限制,但在执行全面训练时,我们的方法允许超越 SoA 解决方案的许多目标。总之,我们的框架允许： · 派生出一些有趣的架构,例如上面讨论的 NASCaps-C10-best,它达到了与 SoA 几乎相似的准确度,同时显着改进了搜索的所有其他目标,即能量、内存和延迟。 · 执行早期候选人选择,同时在执行完整训练后仍然实现高精度。 · 在不同数据集之间实现良好的可迁移性,事实证明,针对 CIFAR10 特定搜索发现的 NASCaps-C10-best DNN 在其他数据集上也优于其他特定于数据集的搜索。

5 结论

在本文中,我们介绍了 NASCaps,这是一种用于卷积胶囊网络 (CapsNets) 的神经架构搜索 (NAS) 的框架。当在专用硬件加速器上执行时,我们框架的一组优化目标是网络准确性和硬件效率,以能耗、内存占用和延迟表示。我们使用具有多个 Tesla V100 GPU 的 GPU-HPC 节点执行了一个大型 NAS,并发现了有趣的 DNN 解决方案,与 SoA 解决方案相比,这些解决方案硬件效率高且准确度高。当设计时间较短、设计中心的培训资源有限且 DNN 设计的培训时间较短时,我们的框架更为有利。我们的 NASCaps 框架可以简化基于胶囊层的 DNN 在资源受限的物联网/边缘设备中的部署。我们将在 https://github.com/ehw-fit/nascaps 开源我们的框架。

致谢

这项工作得到了由 TU Wien 信息学学院和 FH-Technikum Wien 联合运行的博士学院弹性嵌入式系统的部分支持,部分得到了捷克科学基金会项目 GJ20-02328Y 的支持。计算资源得到了教育、青年和体育部大型研究、实验开发和创新基础设施项目“e-Infrastructure CZ – LM2018140”的支持。

参考

[1] M. Abadi 等人。 Tensorflow:一个用于大规模机器学习的系统。
在 OSDI,2016 年。

[2] M.卡普拉等人。用于加速深度卷积神经网络的高效硬件架构的最新调查。未来互联网，
2020.

[3] K. Deb、A. Pratap、S. Agarwal 和 T. Meyarivan。一种快速的精英多目标遗传算法:Nsga-ii。 IEEE进化计算汇刊,2002 年。

[4] S. Grigorescu、B. Trasnea、T. Cocias 和 G. Macesanu。自动驾驶深度学习技术调查。野外机器人学报,2019。

[5] A. Harlap 等人。 Pipedream:快速高效的流水线并行 dnn
训练。 ArXiv,2018 年。

[6] K. He, X. Zhang, S. Ren, and J. Sun.图像深度残差学习
认出。 CVPR,2015 年。

[7] K. He, X. Zhang, S. Ren, and J. Sun.深入研究整流器:在 imagenet 分类上超越人类水平的表现。在ICCV,2015 年。

[8] GE Hinton、A. Krizhevsky 和 SD Wang。改造自动编码器。在 ICANN,2011 年。

[9] W.江等。准确性与效率:通过 fpga实现感知神经架构搜索来实现。在 DAC 中,2019 年。

[10] W.江等。内存计算神经加速器的设备-电路-架构协同探索。 IEEE计算机交易,2020 年。

[11] A.克里热夫斯基。从微小图像中学习多层特征。
技术报告,计算机科学系,多伦多大学,2009 年。

[12] A. Krizhevsky、I. Sutskever 和 GE Hinton。使用深度卷积神经网络的Imagenet分类。公社。美国计算机学会,2017 年。

[13] Y. Lecun、L. Bottou、Y. Bengio 和 P. Haffner。基于梯度的学习应用于文档识别。 IEEE 会议记录,1998 年。

[14] Z. Lu、K. Deb 和 VN Boddeti。 Muxconv:卷积神经网络中的信息复用。 ArXiv，2020 年。

[15] A. Marchisio、MA Hanif 和 M. Shafique。 Capsacc:一种高效的胶囊网络硬件加速器,具有数据重用性。日期,2019 年。

[16] A. Marchisio、MA Hanif、MT Teimoori 和 M. Shafique。 Capstore:用于capsulenet 推理加速器的片上存储器的节能设计和管理。 CoRR,abs/1902.01151,2019 年。

[17] A. Marchisio、V. Mrazek、MA Hanif 和 M. Shafique。 Red-cane:一种在近似下进行胶囊网络弹性分析和设计的系统方法。日期,2020 年。

[18] A.马尔基西奥等。边缘计算的深度学习:当前趋势、跨层优化和开放研究挑战。在 ISVLSI 中， 2019 年。

[19] A.马尔基西奥等。 Fastrcaps:用于快速而准确地训练胶囊网络的集成框架。在 IJCNN,2020 年。

[20] A.马尔基西奥等。 Q-capsnets:量化胶囊网络的专门框架。在 DAC,2020 年。

[21] R. Miikkulainen 等人。不断发展的深度神经网络。 arXiv,2017 年。

[22] V. Mrazek 等人。 Alwann:深度神经网络加速器的自动逐层逼近,无需重新训练。在 ICCAD 中,2019 年。

[23] Y. Netzer 等人。使用无监督特征学习读取自然图像中的数字。国家知识产权局，2011 年。

[24] K. 皮尔逊。注意在两个父母的情况下回归和继承。伦敦皇家学会会刊,1895 年。

[25] J. Rajasegaran 等人。 Deepcaps:深入了解胶囊网络。
在 CVPR,2019 年。

[26] J. Redmon、SK Divvala、RB Girshick 和 A. Farhadi。您只看一次:统一的实时对象检测。 CVPR,2016 年。

[27] S. Sabour、N. Frosst 和 GE Hinton。胶囊之间的动态路由。在 NIPS 中,2017 年。

[28] M. Shafique 等人。机器学习的下一代架构概述:物联网时代的路线图、机遇和挑战。日期,2018 年。

[29] K. Simonyan 和 A. Zisserman。用于大规模图像识别的非常深的卷积网络。在 ICLR,2015 年。

[30] D. Stamoulis 等人。单路径 nas:在不到 4 小时内设计硬件高效的卷积网络。在 ECML/PKDD,2019 年。

[31] Y. Sun、B. Xue、M. Zhang 和 GG Yen。使用用于图像分类的遗传算法自动设计 cnn 架构。 IEEE控制论汇刊,2020 年。

[32] V. Sze, Y.-H.陈,T.-J。杨和 JS Emer。深度神经网络的高效处理:教程和调查。 IEEE 论文集，
2017.

[33] C. Szegedy、S. Ioffe、V. Vanhoucke 和 AA Alemi。 Inception-v4、 inception-resnet 以及残差连接对学习的影响。
在 AAAI 中,2017 年。

[34] C. Szegedy 等人。通过卷积更深入。在 CVPR,2015 年。

[35] H. Xiao、K. Rasul 和 R. Vollgraf。 Fashion-mnist:一种用于对标机器学习算法的新型图像数据集。 ArXiv,2017 年。

[36] L. Xie 和 AL Yuille.遗传CNN。中国计算机科学研究院,2017 年。

[37] B. Zoph 和 QV Le。带有强化的神经结构搜索
学习。在 ICLR,2017 年。

[38] B. Zoph、V. Vasudevan、J. Shlens 和 QV Le。学习可迁移
用于可扩展图像识别的体系结构。在 CVPR,2018 年。