http://www.acabice.cn acabice@163.com

引用格式： *梅继林，杨隆兴，孙自浩，等. 面向资源受限无人系统的深度神经网络轻量化软件设计与应用[J]. 空间控制技术与应用，2021，47(6)：09-18. MEI J L，YANG L X，SUN Z H，et al. Design and application of a lightweight deep neural network software for resource constrained unmanned systems[J]. Aerospace Control and Application，2021，47(6)：09-18 (in Chinese). doi：10.3969/j.issn.1674-1579.2021.06.002*

面向资源受限无人系统的深度神经网络轻量化软件设计与应用

梅继林[1,2]，杨隆兴[1,2]，孙自浩[1,2]，陆 顺[1,2]，邢 琰[3,4]，姜甜甜[3,4]，胡 瑜[1,2*]

1. 中国科学院计算技术研究所，北京 100190
2. 中国科学院大学，北京 100049
3. 北京控制工程研究所，北京 100094
4. 空间智能控制技术重点实验室，北京 100094

摘 要： 地外探测无人系统具有存储、算力和能量等资源受限的特点. 以深度学习为基础的感知、定位和决策算法可有效提升无人系统的智能化水平，而这类算法通常需要高算力，难以直接应用于地外探测无人系统. 首先针对剪枝和量化的深度神经网络模型轻量化方法，在公开数据集上对多种算法进行定量分析. 其次，提出基于剪枝、量化的轻量化计算方案，实现了基于模块化配置的轻量化计算软件 StarLight，对深度神经网络进行快速轻量化和性能评估，解决了模型难以直接应用到计算资源受限系统的问题. 最后，基于StarLight，对应用于火星车实验系统中的多种任务模型进行轻量化，在计算功耗≤15 W、计算处理主频≤1.2 GHz和计算存储容量≤1TB 的受限资源条件下，实现了深度神经网络模型部署. 实验表明，该软件能够满足计算资源受限系统的深度神经网络模型轻量化需求，为进一步提升地外探测无人系统的智能化水平奠定了基础.

关键词： 地外探测无人系统；深度神经网络；轻量化计算

中图分类号： TP18 **文献标志码：** A **文章编号：** 1674-1579(2021)06-0009-10

0 引 言

随着我国深空探测任务的逐步开展，地外探测无人系统承担着越来越多的任务[1]，对系统的智能化水平提出更高的要求. 以深度学习为基础的感知、定位和决策算法可有效提升无人系统的智能化水平，而这类算法对算力、存储和功耗有着较高的需求[2]，难以直接应用于探测系统. 深度神经网络轻量化技术旨在保持模型精度的基础上减少模型参数量和复杂度，加速推断速度，使模型在资源受限条件下高效运行，具有重要的研究意义和实用价值.

当前，深度神经网络轻量化技术已有大量的研究，可分为轻量化网络设计与轻量化网络优化. 轻量化网络设计从网络的拓扑结构出发，旨在设计出参数更少、效果更好的网络. 常见的网络有手工设计的 MobileNet[3]、LightDet[4]等，基于神经网络架构搜索技术[5]自动化搜索出的 DU-DARTS[6]、DD-SAS[7]等. 轻量化网络优化则是在已有的网络基础上，利用神经网络的冗余性质进行网络压缩和加速，常见的方法有剪枝、量化、知识蒸馏和低秩分

收稿日期：2021-11-01；录用日期：2021-12-06
基金项目： 国家重点研发计划资助项目(2018AAA0102700)
* **通信作者：** E-mail：huyu@ict.ac.cn

解. 上述轻量化技术中,剪枝和量化算法通用性强、成熟度较高,具备大规模应用的前景.

深度神经网络的参数量远远超过其训练数据量,却具有良好的泛化能力,表明神经网络是过参数化的[8],这是剪枝的理论基础. 剪枝方法可分为非结构化和结构化 2 种. 以神经元连接为粒度的剪枝方法称为非结构化剪枝. 韩松等[9]提出使用权重作为神经元显著度的指标,对 AlexNet、VGG 等网络进行压缩,在没有精度损失的情况下,分别实现了 9 倍、13 倍的参数压缩量,这种剪枝的特点是能够实现显著的参数压缩比,但却不能实现有效的加速比. 因此,结构化剪枝应运而生,其特点是对特征图的通道或卷积核的滤波器进行剪枝,这样无需特定硬件或软件支持就可以实现高加速比. 结构化剪枝有启发式方法,如基于卷积核幅值[10]、基于批归一层尺度因子[11]、基于卷积核的中值[12];另外,结构化剪枝还可与贝叶斯概率理论结合[13]以及与自动化搜索方法结合[14].

深度神经网络使用单精度浮点数(FP32)训练和推理,32 比特的存储要求和庞大的参数量对设备的存储空间和计算能力提出极高的要求. 量化方法从压缩参数比特数的角度实现网络的轻量化. 针对神经网络训练和推断中的权重、激励和梯度,量化方法可以用不同的比特数来表示,如 8 比特或 1 比特[15-16],从而有效地压缩网络参数并提升训练或推断速度. 半精度浮点数(FP16)相较单精度浮点数只占用一半的存储,GUPTA 等[17]的工作对其进行了研究,实现不错的效果. 在韩松提出的 DeepCompression[18]方法中,量化与霍夫曼编码的组合显著地压缩了网络的参数,但精度损失严重. 在工业界量化算法也同样受到关注,英伟达开发的 TensorRT[19]使用标定集对特征图激励的分布进行分析,结合英伟达硬件设备,能够实现 4 倍以上的加速压缩比,同时几乎没有精度上的损失.

本文首先在公开数据集上对多种剪枝、量化算法进行了定量分析,提出基于剪枝、量化的轻量化计算方案,以此为基础,实现了一种基于模块化配置的轻量化计算软件 StarLight,对深度神经网络模型进行快速轻量化和性能评估,解决了模型难以直接应用到计算资源受限系统的问题. 相比于已有的开源方案主要针对公开数据集的分类任务,StarLight 不仅在公开数据集上对多种轻量化方法进行评估,还针对火星模拟数据集实现目标跟踪、目标检测和语义分割等多种算法的轻量化并部署到低算力嵌入式平台;同时提供可视化界面易于使用和调试.

1 轻量化方法的建立

1.1 剪枝方法

本文主要讨论结构化剪枝,即对特征图的通道或卷积核的滤波器进行剪枝. 假设一个具有 L 层卷积的神经网络,N_i 和 N_{i+1} 分别表示第 i 个卷积层的输入和输出的通道. $\quad_{ij}$ 表示第 i 个卷积层的第 j 个滤波器. 通常,结构化剪枝的目标是定义一个度量函数 $\delta(\quad_{ij})$,判断 $\quad_{ij}$ 是否被剪去. 如表 1 所示,本文选取近期具有代表性的 5 种算法进行剪枝效果的对比:基于启发式的方法 Taylor[20] 和 FPGM[12]、基于渐进式的方法 AGP[21]、基于约束的优化方法 NetAdapt[22] 和基于自动化参数搜索的方法 AutoCompress[23].

表 1 剪枝方法的选取

Tab. 1 Selection of pruning methods

方法	分类
Taylor(CVPR2019)	启发式
FPGM(CVPR2019)	
AGP(ICLR2018)	渐进式
NetAdapt(ECCV2018)	带约束优化
AutoCompress(AAAI2020)	自动化参数搜索

Taylor 算法假设网络权重在反向传播过程中的一阶导数可以作为滤波器的重要性因子,剪枝的度量函数定义为 $\delta_{ij} = \|g_{ij} \quad_{ij}\|_2$,其中 g_{ij} 为第 i 层中第 j 个滤波器的反向传播梯度. 对于训练好的网络模型,进行参数调优(fine-tuning)后,剪去 K 个 δ_{ij} 值最小的滤波器.

FPGM 算法采用几何中值作为度量因子,指出靠近几何中值的滤波器可由其他滤波器线性近似表示,几何中值处的滤波器冗余性较大,可以剪去. 该算法首先计算几何中值:

$$x^{GM} = \arg\min_x \sum_{j\in[1,N_{i+1}]} \|x - \quad_{ij}\|_2 \tag{1}$$

对于第 i 层,选择距离几何中值最近的一个或多个滤波器:

$$\quad_{ij^*} = \arg\min_{ij} \| \quad_{ij} - x^{GM}\|_2,\ s.t.\ j \in [1, N_{i+1}] \tag{2}$$

$_{ij^*}$ 可由其他滤波器表示,剪去它们对模型的精度影响较小.

AGP 算法采用滤波器参数的 $L1$ 范数作为度量因子即 $\delta_{ij} = \| \ _{ij}\|$,提出一种渐进式的剪枝算法,自动调整每一步的剪枝稀疏度.解决了直接按照目标稀疏度剪枝导致精度损失过大的问题.为自动调整稀疏度,采用式(3)实现渐进策略:

$$s_t = s_f + (s_i - s_f)\left(1 - \frac{t - t_0}{n\Delta t}\right)^3$$
$$s.t.\ t \in \{t_0, t_o + \Delta t, \cdots, t_0 + n\Delta t\} \tag{3}$$

其中,t 为训练步数,Δt 表示剪枝频率,s_t 表示第 t 步的稀疏度,s_i, s_f 表示初始化和最终的稀疏度.式(3)中,剪枝稀疏度的变化与 $\left(1 - \frac{t - t_0}{n\Delta t}\right)^3$ 成正比,该三次方项随着时间的变化逐渐趋向于 1,且趋势越来越平缓,从而同时实现稀疏度的逐步增大与增大比例的逐渐减少.

相比于上述以滤波器的参数为标准设计度量因子,NetAdapt 直接将运算时间和算力消耗作为约束条件,以经验测量值建立搜索空间,自动的优化预训练模型,直到满足约束条件,该过程中还可获得不同折衷度的简化网络.优化过程如下:

$$\text{maximize}_{Net_i} Acc(Net_i)$$
$$s.t.\ Res_j(Net_i) \leqslant Res_j(Net_{i-1}) - \Delta R_{ij}, j \in [1, m] \tag{4}$$

$Acc(\cdot)$ 为精度,$Res_j(\cdot)$ 为第 j 个资源消耗,ΔR_{ij} 为在第 i 次迭代中第 j 个资源消耗和约束的差值.该方法需要根据部署设备构建经验测量值.

AutoCompress 提出一种自动化的剪枝算法,设计基于启发式的搜索算法对参数进行采样、评估,作为决定每层剪枝率和剪枝策略的依据.流程包含动作采样、快速评估、策略选取和实际剪枝等 4 个步骤.采用交替方向乘子法 ADMM(alternating direction multiplier method)作为优化算法,ADMM 的特点是动态正则化,在每轮迭代都会调整、惩罚不同的参数.

1.2　量化方法

本文主要讨论 FP16 和 INT8 量化,这 2 种方式已在研究界和工业界得到了广泛的验证.相比于神经网络中常用的基于单浮点精度的推理方式,量化后的网络模型对存储的需求降低,并能提升推理速度.量化是神经网络模型在实际部署时所需的一项重要技术,如表 2 所示,目前已有多种量化解决方案.本文首先对表 2 中的方案进行调研,总结出以下 4 个方案选择原则,1)支持 Nvidia Jetson Xavier 平台,Xavier 属于低算力嵌入式设备,并针对 INT8 的张量计算提供了硬件加速,是常用的量化算法评估平台;2)易用性好,量化方案的选取需要考虑使用成本;3)文档完善度高,加快方案的学习和推广;4)社区活跃,对于实际使用中遇到的问题,可在社区进行讨论,以快速解决.对比英特尔、微软、百度、英伟达以及其他多种方案,结合提出的 4 个选择原则,确定了 Nvidia TensorRT 满足需求.需要注意,TensorRT 仅支持常见网络层的量化,对于新的网络层或者自定义操作则需要手动编写量化插件.

TensorRT 支持 FP16 和 INT8 量化,其中 INT8 量化需要标定数据,仅适用于网络推理.FP32 转 INT8 的过程中,如何选取缩放因子是影响量化效果的关键因素.有非饱和与饱和 2 种映射方法.非饱和方法将张量的最大范围线性映射为 ±127,这种方法忽略了数值分布的不连续性,会导致大量无用数值被量化.饱和方法则是选取阈值 $|T|$,将阈值 $\pm|T|$ 之内的张量映射为 ±127,为降低量化带来的精度损失,TensorRT 使用 INT8 分布和原始 FP32 分布的 KL 散度作为优化目标,KL 散度越小表示 INT8 量化后的信息损失越小.具体实现是选取验证集的一个子集作为校准集,计算每一层 FP32 的激活值分布,使得 INT8 精度的激活值尽可能接近 FP32 的激活值.

表 2　量化方法的选取

Tab. 2　Selections of quantization method

参考方案	Intel Distiller
	微软 NNI
	百度 PaddlePaddle
	Nvidia TensorRT
	其他[24]
选择原则	支持 Nvidia Xavier 平台
	易用性好
	文档完善度高
	社区活跃
最终方案	Nvidia TensorRT + 自定义层

1.3　轻量化方案选取

从上述 2 小节可看出,目前存在多种剪枝和量化方法可供选择,这些方法的实际使用效果如何,以及如何搭配剪枝和量化实现高效的轻量化方案,本小节以公开数据集和网络模型为基础,定量分析各种方法的优缺点,并尝试给出高效的网络压缩方案.

各种方法的测试结果如表 3 所示，采用 ResNet50 作为基础模型，TinyImageNet 为数据集进行分类任务测试. 评价指标包括 Top1 分类精度、运算量(FLOP)、参数量(param)、存储量(storage)以及推理时间(latency). 网络模型轻量化的目标是在保证精度不降或者小幅降低的条件下，尽可能减少计算资源的消耗.

本文在 TinyImageNet 上重新训练 ResetNet50，其精度为 60.4%，该结果小于当前的 SOTA 方法，本文更关注轻量化前后精度的变化；选择重新训练的原因在于，部分算法在剪枝过程中需要进行模型的参数更新，为保证对比的公平性，尽可能保持该过程的超参数和原始模型训练的一致.

针对剪枝的 5 种算法评估结果如表 3 所示，在原始模型上进行剪枝，可看到精度均有不同程度的下降，FPGM 精度降低最小，AutoCompress 降幅最大接近 20%，另外 3 个剪枝模型的精度基本一致；无论哪种方法，剪枝后的计算量、参数、存储和推理时间指标均有降低，这也验证了剪枝可作为轻量化计算的一条重要途径. 在实验过程发现，AutoCompress 方法过于复杂、超参数多，并对超参的调整敏感，需要较多时间进行参数调试，在实际使用中不推荐该方法. NetAdapt 算法在参数量和存储上有优势，但运算量上要高于其他方法，这也使得它的推理时间明显变长. FPGM 算法在精度、计算量和推理时间上均有优势. Taylor 和 AGP 算法压缩效果接近，使用难度较低，也可以作为备选的剪枝算法. 综合的各方面指标，推荐 FPGM、Talyor 和 AGP 作为候选的剪枝算法.

在量化的定量分析中，选取 TensorRT 作为工具，对 FP16、INT8 和混合精度量化(Mix)进行测试. 表 3 中的 Mix 表示根据每个网络层的特点自动选择 FP16 或 INT8. 对原始模型直接进行量化，表 3 显示，量化的特点是不改变计算量(FLOP)和参数量(param)，但可以大幅降低存储和推理时间，以 FP16 为例，存储减少约 50% 且推理时间获得超过 20 倍提升. 从精度来看，3 种量化方法相差不多，FP16 量化没有对精度造成影响. 在存储上，INT8 最低，FP16 最高，Mix 处于两者之间，这符合我们的预期. 在推理时间上，3 种方法的效果基本一致. 结合以上分析，建议在实际使用中，这 3 种均可作为候选的量化方法.

表 3 基于公开数据集的轻量化算法定量对比

Tab. 3 Comparison of different lightweight algorithms on public dataset

	算法	Top1/%	计算量/M	参数量/M	存储/MB	推理时间/ms
原始模型	ResNet50	60.4	334.1	23.9	92	23.7
剪枝	FPGM[12]	58.3	244.0	17.3	67	13.8
	Taylor[20]	57.3	248.6	17.6	68	14.0
	AGP[21]	57.3	247.9	17.5	68	14.2
	NetAdapt[22]	57.4	301.0	16.5	64	15.0
	AutoCompress[23]	40.7	292.5	18.1	70	14.3
量化[19]	FP16	60.4	334.1	23.9	47	0.98
	INT8	60.2	334.1	23.9	25	0.99
	Mix	60.2	334.1	23.9	36	0.89
剪枝 + 量化	FPGM + FP16	58.3	244.0	17.3	35	0.92
	FPGM + INT8	57.7	244.0	17.3	19	1.65
	FPGM + Mix	58.3	244.0	17.3	35	0.90
	Taylor + FP16	57.4	248.6	17.6	36	0.90
	Taylor + INT8	57.0	248.6	17.6	19	1.53
	Taylor + Mix	57.4	248.6	17.6	36	0.92
	AGP + FP16	57.3	247.9	17.5	36	0.91
	AGP + INT8	57.1	247.9	17.5	19	1.77
	AGP + Mix	57.3	247.9	17.5	36	0.92

从上述剪枝、量化的单独测试中,可发现以下规律,剪枝算法可有效降低计算量、参数量、存储和推理时间;而量化算法不改变计算量和参数量,但能大幅降低存储和推理时间. 基于这个现象,表 3 中给出了剪枝 + 量化的结果,先对原始模型进行剪枝,再对剪枝后的模型进行量化. 我们对候选的 3 种剪枝和 3 种量化算法进行组合,共有 9 组测试结果. 以 FPGM + FP16 为例,2 种轻量化算法结合后,相比于仅用剪枝,存储和推理时间得到了大幅降低. 从表 3 的数据中可发现,先剪枝后量化的策略可结合 2 类算法的优势,进一步提升模型轻量化的效果. 表 3 中剪枝后的模型和 INT8 组合,其推理时间反而高于仅使用 INT8,原因有 2 方面:1)表 3 的实验是在 Nvidia Tesla V100 GPU 上进行,该 GPU 属于 Volta 架构,没有 INT8 的硬件张量加速核,使得 INT8 量化推理时间和 FP16 相比几乎没有提升;缺少硬件张量加速核仅影响推理时间,对其他指标没有影响. 2)剪枝之后网络层的通道数不再是 2 的指数幂,这对 GPU 的并行计算是不友好的方式,可能导致推理时间增加 0.5 ~ 0.8 ms;量化过程使用了 TensorRT 方案,其核心代码闭源,无法进行深入的调试,该现象的深层次原因目前无法定位.

2　轻量化软件 StarLight

以上述剪枝、量化等算法为核心,设计了基于模块化配置的轻量化软件 StarLight,提供用户友好的使用界面,降低轻量化算法的调试难度并提高易用性.

2.1　系统功能描述

StarLight 支持从原始模型到轻量化模型嵌入式平台部署的整个流程,如图 1(a)所示,目前 StarLight 可处理物体分类、目标检测、语义分割和目标跟踪等多种类型的神经网络模型. 在获取轻量化模型后,可直接部署到低算力嵌入式平台,比如 Nvidia Jetson Xavier,以下简称为 Xavier. StarLight 的剪枝部分可在 Intel CPU、Nvidia GPU 上正常运行,剪枝算法的策略是一边剪枝一边微调模型权重,对并行算力有较大的需求,采用 Nvidia GPU 可大幅降低运行时间,且在高算力 GPU 下会进一步降低. 在本文的实验中,使用了 Nvidia Tesla V100 GPU,以下简称为 V100 GPU. 另外,StarLight 的量化部分采用了 Nvidia TensorRT 方案,目前只支持 Nvidia GPU 平台.

如图 1(b)所示,整个软件包含输入界面、输出界面、轻量化算法和数据回放等 4 个模块. 软件共有 4 个线程. 主线程中有参数配置、指标设置和数据回放等功能;轻量化算法、推理可视化和训练/模型可视化各自对应一个子线程. 输入/输出界面的各模块可根据需求打开或者关闭.

在功能上,StarLight 包含在线和离线 2 种模式. 在线模式指通过软件实时监测轻量化的过程,并对中间结果进行可视化. 离线模式指轻量化已完成,对中间过程进行数据回放,以检查结果的合理性和正确性. 2 种模式可在 UI 界面中随时切换.

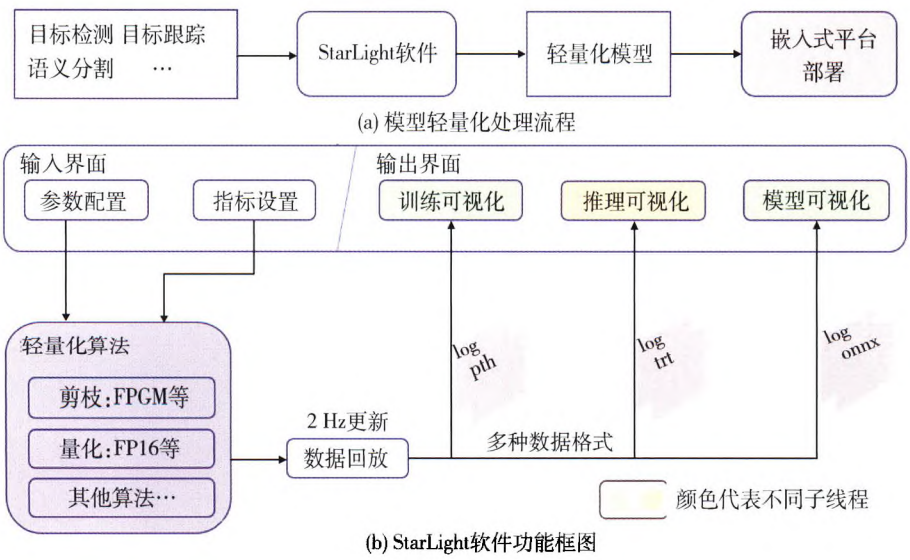

图 1　轻量化计算软件系统流程图

Fig. 1　The system framework of lightweight computing software

在软件设计上,采用前端-可视化界面和后端-轻量化算法分离的策略. 后端保存运行过程中的日志和中间结果,通过数据回放模块以 2Hz 的频率分发给输出界面. 这种方式实现了前后端的分离,具体优势体现在,轻量化算法的运行时间较长,可随时打开或关闭主界面而不影响算法的运行.

在软件实现上,采用 PyQt 设计 UI 主界面,其优点在于统一使用 python 接口,可与轻量化算法的输入输出无缝对接. 采用多线程、多进程机制充分利用计算资源,实现流畅的人机交互体验. 在可视化效果上,引入 ECharts[25] 进行表格和曲线的可视化,并提供模型和特征图的可视化,对轻量化算法的运行过程进行回放和展示.

2.2 输入界面

参数配置:包括数据集、待压缩神经网络、剪枝方法和量化方法的选择,在线/离线模式的选择,神经网络超参数配置,如学习率、训练轮数和批处理大小等,剪枝算法超参数配置,如剪枝的优化学习率、微调轮数和稀疏度等;此外还包含数据回放的路径配置. 神经网络模型和剪枝算法均对超参数的设置敏感,在软件中,对每种可能的轻量化算法组合均提供初始的超参数. 在 UI 实现上采用 Qt 的输入、按钮等组件,结合 Qt 的信号槽机制,实现对应的回调函数.

指标设置:对轻量化算法的指标设置,有 2 个功能,第一是作为剪枝过程中的约束条件,比如 NetAdapt[22] 方法;而对于 FPGM[12] 和 AGP[21] 等这类无法添加优化约束的方法,指标设置仅用于可视化展示. 实际使用中的指标可根据具体需求设置,常用的包括参数量、计算量、存储和推理时间等. 与参数配置模块一致,在 UI 实现上采用了 Qt 内置的组件.

2.3 输出界面

训练可视化:本文轻量化的输入是已完成训练的模型,StarLight 首先判断该模型是否存在,若不存在则重新训练,再进行轻量化处理,所以此处沿用训练可视化表示该过程. 模块主要实现训练过程中间结果的可视化. 包括损失值、精度和时间分布等指标. 实现上采用开源 JavaScript 图表库 Echarts[25],如图 2 所示,可视化的元素包括柱状图、饼状图、折线图和曲线图等. 基于 Qt 的 QWebEngineView 类进行 Echarts 图库的实时渲染. 模型训练过程时间较长,StarLight 具备后端和前端分离的功能,关闭主界面后再次打开,训练可视化自动加载历史数据,并动态更新当前数据.

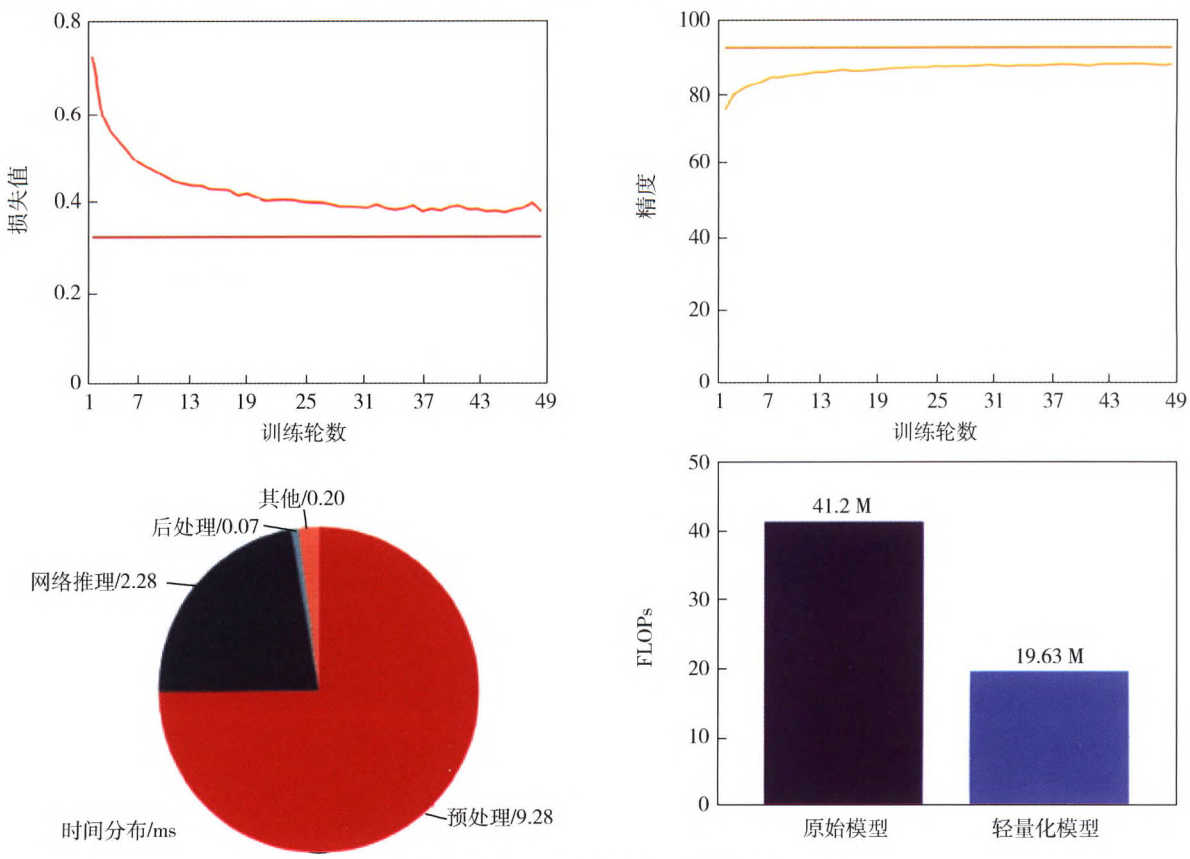

图 2 模型训练和推理过程的可视化组件

Fig. 2 The visualization elements used in model training and testing

推理可视化:该模块的功能相对独立,加载轻量化后的模型和数据集,实时动态显示推理过程中各项指标的变化,比如参数量、计算量和存储等,同时提供原始模型的指标,以方便结果对比. 在可视化实现上,采用和训练可视化类似的组件,此处不再详述.

模型可视化:该模块实现了模型拓扑图的可视化,以及特征图的可视化. 如图 3 所示,左边显示的是模型节点,点击后会显示节点的名称和特征图形状等属性;右边显示的特征图,点击不同的节点,则实时可视化对应的特征图. 该模块有助于模型的调试,以及剪枝前后结果的对比. 本模块基于开源工具 QuiverPytorch[26] 实现.

图 3　模型可视化效果图

Fig. 3　Model visualization

2.4　轻量化算法的集成

StarLight 支持的轻量化算法和神经网络模型如表 4 所示,后续随着新的算法和网络模型提出,我们也将同步更新软件. 对 StarLight 在分类、检测、分割和跟踪等多种任务上进行了验证,轻量化效果在后续实验中进行详细说明.

基于松耦合的设计思路,StarLight 具有良好的可扩展性,如图 1(b)所示,轻量化算法模块的输入是参数配置,输出日志信息以及模型,仅需修改算法的输入输出接口;这种方式降低了算法集成过程中的代码改动量,另外一个优点是算法可以通过命令行启动,方便对算法进行单独调试. 在 StarLight 运行过程中,启动轻量化算法会创建一个新的独立线程,实现与主 UI 线程的分离,如果用户不主动从 UI 界面结束算法线程,UI 的意外退出或关闭不会对算法的运行产生影响,轻量化算法运行结束后自动退出线程.

表 4　StarLight 集成的轻量化算法和神经网络模型

Tab. 4　The neural network models and lightweight algorithms in StarLight

轻量化算法	剪枝量化[19]	AGP[21], FPGM[12], Taylor[20], L1, L2
		FP16, INT8, Mix
神经网络模型	分类任务	ResNet50
	目标检测	Cascade RCNN[27]
	语义分割	DeepLabv3 +[28]
	目标跟踪	Siamese RPN[29]

3　实验验证

如图 1(b)所示,轻量化计算软件 StarLight 主要包含后端轻量化算法和前端可视化 2 部分,其中核心是轻量化算法,可视化主要提供用户友好的接口,降低软件的调试和使用难度. StarLight 软件仍处于更新中,本文主要对后端的轻量化算法进行实验验证.

3.1　实验平台

实验中使用高算力 V100 GPU 服务器和低算力嵌入式设备 Xavier,这 2 种设备的软件环境统一为 pytorch1.5、TensorRT 7.1.3.4. StarLight 运行在 V100 GPU 服务器上,得到轻量化模型后,在 Xavier 上部署测试,实验中将 Xavier 的功耗模式设置为 15 W.

表 5　数据集统计信息

Tab. 5　The distribution of dataset

	训练	验证	测试	合计
图片数量	765	232	216	1 213
比例	63%	19%	18%	100%

3.2　网络模型

目标检测、语义分割和目标跟踪算法可有效提升火星车实验系统的智能化水平. 为此,我们分别选取 3 种主流的深度神经网络模型,测试 StarLight 的轻量化能力. 针对目标检测,选取 Cascade RCNN[27],评价指标为正确率,即正确检测的物体数量和总物体数量的比值. 针对语义分割,选取 DeepLabv3 +[28],评价指标为平均交并比(mIoU). 针对目标跟踪,选取 Siamese RPN[29],评价指标为跟踪的成功率和 FPS.

3.3　数据集

如图 4 所示,采用模拟的火星场景数据集,主要材质有石头、沙粒等. 算法对场景中的高风险物体,比如石头,进行识别、分割和跟踪. 如表 5 所示,数据集包含 1213 张 RGB 图像,将其分为训练集、验证集

和测试集. 针对目标跟踪的 Siamese RPN 模型,还使用公开数据集 VOT2015 进行训练[29].

图 4 火星模拟场样例数据

Fig. 4 Samples of a simulated Mars scene

3.4 实验分析

如表 6 所示,以目标跟踪算法为例,使用 V100 GPU 服务器进行实验分析. baseline 表示轻量化前的模型. FPS1 表示对 1 个目标跟踪的跟踪速度,FPS2 表示 2 个目标,其他以此类推. 剪枝方法选取 FPGM,测试不同稀疏度对效果的影响,如 FPGM0.2 代表稀疏度 0.2,稀疏度越大表示网络层剪去的通道数越多. 随着稀疏度增加,模型计算量和存储量随之降低,且运行速度得到提升. 稀疏度为 0.2 和 0.5 时,成功率没有降低,但性能得到了提升. 当稀疏度到达 0.8 时,成功率降幅较大. 基于以上观察,选择稀疏度 0.5 能够较好的兼顾各项指标. 而对于量化算法,3 种方法均能做到不降低精度的条件下,大幅降低存储并提升 FPS,需要注意的是,量化并不会改变计算量和参数量. 最后,结合 FPGM0.5 和 3 种量化方法进行组合测试,即先剪枝后量化的操作;从结果来看,相比于 baseline,在成功率不降或小幅降低的条件下,实现了神经网络模型的轻量化且大幅提高计算速度;针对目标检测任务,考虑到成功率指标,我们推荐 FPGM0.5 + FP16 的组合方式. 针对目标检测和语义分割任务,我们进行类似的组合实验测试,得到最优轻量化途径.

火星车实验系统计算资源受限,无法使用高性能的 GPU 服务器,在获得轻量化模型后,我们将其部署到低算力嵌入式平台 Nvidia Xavier 上,结果如表 7 所示. 在任务指标基本不变的情况下,对检测、分割和跟踪网络实现了约 30% ~75% 的计算量压缩、20% ~65% 的参数量压缩和 20% ~80% 的存储压缩;同时功耗均低于 15 W,主频低于 1.2 GHz,存储低于 1TB,实现了算法在低算力嵌入式平台运行的功能.

表 6 基于 V100 GPU 的目标跟踪算法 Siamese RPN 测试结果

Tab. 6 The comparison of object tracking algorithm on V100 GPU

方法	计算量/G	参数量/M	存储/MB	FPS1	FPS2	FPS3	FPS4	成功率/%
baseline	6.48	6.25	24	229	124	81	64	99.58
FPGM0.2	4.39	4.42	17	242	130	83	64	99.58
FPGM0.5	1.98	2.20	8.5	274	146	80	66	99.58
FPGM0.8	0.51	0.67	2.6	272	138	94	68	68.82
FP16	6.48	6.25	13	585	294	189	143	99.58
INT8	6.48	6.25	7	550	284	185	138	99.58
Mix	6.48	6.25	7	571	294	197	148	99.58
FPGM0.5 + FP16	1.98	2.20	5.7	668	330	222	167	99.58
FPGM0.5 + INT8	1.98	2.20	2.4	642	325	217	164	98.95
FPGM0.5 + Mix	1.98	2.20	2.4	650	328	219	164	98.95

表 7 基于 Xavier 平台的轻量化模型部署结果

Tab. 7 The deployment of lightweight model on Xavier

神经网络模型(指标)	轻量化方法	性能	计算量/G	参数量/M	存储/MB	功耗/W	主频/GHz
Cascade RCNN (正确率/%)	baseline	99.09	382	71.7	275	6 ~ 8	< 1.2
	FPGM0.5	99.07	268	55.4	212	6 ~ 7	< 1.2
DeepLabv3 + (平均交并比/%)	baseline	90.6	52.9	13.9	56	8 ~ 9	< 1.2
	FPGM0.5	89.6	13.5	4.8	19	6 ~ 7	< 1.2
Siamese RPN (成功率/%, FPS1)	baseline	成功率:99.6 FPS1:26	6.5	6.3	24	6 ~ 7	< 1.2
	FPGM0.5 + FP16	成功率:99.6 FPS1:73	2	2.2	4.5	1 ~ 2	< 1.2

4 结论

本文首先分析基于剪枝和量化的轻量化方法,之后在公开数据集上对多种剪枝、量化方法以及它们的组合进行定量分析. 以轻量化算法为核心,实现了一种基于模块化配置的轻量化计算软件 StarLight,包含剪枝模型、量化模型及可视化方法,对深度神经网络模型进行快速轻量化和性能评估. 基于 StarLight,对应用于火星车这种典型无人系统的多种任务模型实现轻量化,并完成嵌入式平台部署,在保证任务指标的前提下,解决了深度神经网络模型难以直接应用到计算资源受限系统的问题.

参 考 文 献

[1] CHEN N. Asteroids, ganymede and uranus: China's deep space exploration plan to 2030 and beyond[EB/OL]. [2021-10-17] http://english.cas.cn/newsroom/archive/news_archive/nu2017/201709/t20170908_182912.shtml.

[2] 胡航天, 刘凯, 马士超, 等. 专用指令集在基于 FPGA 的神经网络加速器中的应用[J]. 空间控制技术与应用, 2020, 46(3): 36-41.
HU H T,LIU K,MA S C,et al. Application of special command set in neural network accelerator based on FPGA[J]. Aerospace Control and Application,2020,46(3):36-41.

[3] HOWARD A G, ZHU M L, CHEN B, et al. Mobilenets: efficient convolutional neural networks for mobile vision applications [EB/OL]. [2021-10-17]. https://arxiv.org/pdf/1704.04861.pdf.

[4] TANG Q L, LI J, SHI Z P, et al. Lightdet: a lightweight and accurate object detection network[C]// The 45th International Conference on Acoustics, Speech, and Signal Processing. New York: IEEE, 2020.

[5] ELSKEN T, METZEN J H, HUTTER F. Neural architecture search: a survey[J]. Journal of Machine Learning Research, 2019, 20(55): 1-21.

[6] LU S, HU Y, YANG L X, et al. DU-DARTS: decreasing the uncertainty of differentiable architecture search [C]// The 32nd British Machine Vision Conference. Virtual: British Machine Vision Association, 2021.

[7] YANG L X, HU Y, LU S, et al. DDSAS:dynamic and differentiable space-architecture search[C]// The 13rd Asian Conference on Machine Learning. Virtual:ACML,2021.

[8] ZHU Z Y, LI Y Z, LIANG Y Y. Learning and generalization in overparameterized neural networks, going beyond two layers[C]// The 43rd Conference on Neural Information Processing Systems. Vancouver: Neural Information Processing Systems Foundation, 2019.

[9] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural network[C]// The 29th Conference on Neural Information Processing Systems. Montréal: Neural Information Processing Systems Foundation, 2015.

[10] HE Y H, ZHANG X Y, SUN J. Channel pruning for accelerating very deep neural networks[C]// International Conference on Computer Vision. New York: IEEE, 2017.

[11] LIU Z, LI J G, SHEN Z Q, et al. Learning efficient convolutional networks through network slimming[C]// International Conference on Computer Vision. New York: IEEE, 2017.

[12] HE Y, LIU P, WANG Z W, et al. Filter pruning via geometric median for deep convolutional neural networks acceleration[C]// Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2019.

[13] LOUIZOS C, ULLRICH K, WELLING M. Bayesian compression for deep learning[C]// The 31st Conference on Neural Information Processing Systems. Long Beach: Neural Information Processing Systems Foundation, 2017.

[14] DONG X, YANG Y. Network pruning via transformable architecture search[EB/OL]. [2021-10-16]. https://arxiv.org/pdf/1905.09717.pdf.

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

[16] COURBARIAUX M, BENGIO Y, DAVID J P. Binaryconnect: training deep neural networks with binary weights during propagations[C]// The 29th Conference on Neural Information Processing Systems. Montréal: Neural Information Processing Systems Foundation, 2015.

[17] GUPTA S,AGRAWAL A,GOPAKAJRISHNAN K,et al. Deep learning with limited numerical precision[C]// International Conference on Machine Learning. Lille: ICML, 2015.

[18] HAN S, MAO H, DALLY W J. Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding[C]// The 4th International Conference on Learning Representations. San Juan: ICLR, 2015.

[19] MIGACZ S. 8-bit inference with TensorRT [EB/OL]. [2021-10-18]. http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf.

[20] MOLCHANOV P, MALLYA A, TYREE S, et al. Importance estimation for neural network pruning[C]// Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2019.

[21] ZHU M, GUPTA S. To prune, or not to prune: exploring the efficacy of pruning for model compression[C]// The 6th International Conference on Learning Representations. Vancouver: ICLR, 2018.

[22] YANG T J, HOWARD A G, CHEN B, et al. NetAdapt: platform-aware neural network adaptation for mobile

applications[C] // European Conference on Computer Vision. Munich: Springer, 2018.
[23] LIU N, MA X L, XU Z Y, et al. AutoCompress: an automatic DNN structured pruning framework for ultra-high compression rates[C] // The 34th AAAI Conference on Artificial Intelligence. New York: Association for the Advancement of Artificial Intelligence, 2020.
[24] A quick view of high-performance CNNs inference engines on mobile devices [EB/OL]. [2021-10-18]. https: // github. com/holmesshuan/cnn-inference-engine-quick-view.
[25] Apache ECharts 一个基于 JavaScript 的开源可视化图表库 [EB/OL]. [2021-10-18]. https: // echarts. apache. org/zh/index. html.
[26] Quiver Pytorch: interactive CNN visualization tool for pytorch [EB/OL]. [2021-10-18]. https: // github. com/mjlsuccess/quiverpytorch.
[27] Detectron2[EB/OL]. [2021-10-18]. https: // github. com/facebookresearch/detectron2.
[28] CHEN L L, ZHU Y K, PAPANDREOU G, et al. Encoder-decoder with atrous separable convolution for semantic image segmentation[C] // European Conference on Computer Vision. Munich: Springer, 2018.
[29] LI B, YAN J J, WU W, et al. High performance visual tracking with siamese region proposal network[C] // Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2018.

作者简介：梅继林(1990—),男,助理研究员,研究方向为自动驾驶环境感知、轻量化计算;**杨隆兴**(1996—),男,博士研究生,研究方向为神经网络架构搜索、神经网络压缩;**孙自浩**(1996—),男,博士研究生,研究方向为深度学习、神经网络架构搜索;**陆顺**(1997—),男,博士研究生,研究方向为神经网络架构搜索、神经网络压缩;**邢琰**(1974—),女,研究员,研究方向为故障诊断与重构,航天器导航、制导与控制;**姜甜甜**(1987—),女,高级工程师,研究方向为航天器智能自主控制;**胡瑜**(1975—),女,研究员,研究方向为自动驾驶感知与决策、神经网络架构搜索.

Design and Application of a Lightweight Deep Neural Network Software for Resource Constrained Unmanned Systems

MEI Jilin[1,2], YANG Longxing[1,2], SUN Zihao[1,2], LU Shun[1,2], XING Yan[3,4], JIANG Tiantian[3,4], HU Yu[1,2*]

1. Institute of Computing Technology, CAS, Beijing 100090, China
2. University of Chinese Academy of Sciences, Beijing 100049, China
3. Beijing Institute of Control Engineering, Beijing 100094, China
4. Science and Technology on Space Intelligent Control Laboratory, Beijing 100094, China

Abstract: The unmanned system in interplanetary exploration has the characteristics of limited storage, computing power, energy and so on. The perception, localization and decision-making algorithms based on deep neural network can effectively improve the intelligence level, but these algorithms generally require huge computing power, which is difficult to be directly applied to unmanned systems. Therefore, this paper reviews the existing lightweight methods including pruning and quantization, and makes a quantitative analysis on public dataset. Furthermore, this paper proposes pruning and quantization solutions, establishes a lightweight computing software StarLight, realizes rapid lightweight and evaluation of deep neural network, and solves the problem that the deep model is difficult to be directly applied to resource constrained systems. Finally, based on StarLight, various models used in the Mars rover are compressed, and deployed in the embedded platform; under the premise of ensuring performance, the power≤15 W, CPU frequency≤1.2 GHz and storage≤1 TB. Experiments show that the software can meet the lightweight requirements of resource constrained systems, and builds a foundation for further improving the intelligent level of unmanned systems for interplanetary exploration.

Keywords: unmanned system; deep neural network; lightweight computing

Received:2021-11-01;**Accepted**:2021-12-06
Foundation item: Supported by the National Key R&D Program of China(2018AAA0102700)
* **Corresponding author.** E-mail: huyu@ict.ac.cn