嵌入式智能计算机计算能力评测方法

马春燕 陈 晶 姚 鼎 张 涛
(西北工业大学软件学院 西安 710072)

摘 要 计算能力评测方法是嵌入式智能计算机领域的研究热点之一. 嵌入式智能计算机存在多种神经形态计算的优化方案和机器学习加速器，导致其评测难度大于通用计算机. 基准测试是目前普遍采用的评测方法，但是在资源受限的嵌入式设备上，基准测试集和评价指标的复用能力有限，难以适应配置多样化的嵌入式智能系统；测试集中神经网络模型的计算强度存在一定的随机性，无法充分挖掘待测设备的计算潜力；评价指标不统一，难以对不同嵌入式智能计算机的计算能力进行对比分析. 本文提出了一种基于神经进化算法的嵌入式智能计算机计算能力评测方法. 首先，基于 Roofline 理论模型，融合计算潜力挖掘、资源适配和评价指标统一等优势，提出了一种适配各种嵌入式智能计算机的计算能力评测框架，并对其合理性进行分析；其次，提出了一种评测计算能力的神经网络模型生成算法，利用神经进化算法，使生成模型的计算强度逼近嵌入式智能计算机的计算强度上限，充分挖掘待测设备的计算潜力，使评测结果更客观；然后，采用环境固定的上位机作为对照，分别在待测设备和上位机交叉运行生成的神经网络模型，并以两次执行推断任务时的每秒浮点运算次数作为计算因子，给出计算能力评测的通用公式，可以实现不同嵌入式智能计算机计算能力的对比分析；最后，在 Mindspore-cpu、Tensorflow-cpu 和 Mindspore-ascend310 框架下评测华为 Atlas200，相比基准测试中常用的 5 种神经网络模型，采用本文生成的神经网络模型的测评结果更合理，证实两个 DaVinci 核心的智能计算能力是八个 Cortex-A55 核心的 42.37 倍.

关键词 神经进化算法；嵌入式智能计算机；计算能力评测；神经网络模型；Atlas200
中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2023.02279

Computing Capability Evaluation Method of Embedded Intelligent Computer

MA Chun-Yan CHEN Jing YAO Ding ZHANG Tao
(*School of Software, Northwestern Polytechnical University, Xi'an* 710072)

Abstract Computing capability evaluation is one of the research hotspots in the field of embedded intelligent computing. It is an important means to test the execution speed and performance of intelligent computers when performing inference tasks. The rationality of its evaluation results directly affects the optimization and improvement direction of embedded intelligent computers. Due to the various optimization schemes for neural form computing and machine learning accelerators, the evaluation of embedded intelligent computers is more difficult than that of general-purpose computers. Benchmark testing is currently a commonly used evaluation method, but on resource-limited embedded devices, the reuse ability of benchmark test sets and evaluation indicators is limited, making it difficult to adapt to the diversified configuration of embedded intelligent systems. The computational intensity of the neural network models in the test set has a certain randomness, which cannot fully explore the computing potential of the device under test, and the evaluation indicators are not unified, making it difficult to compare and analyze the computing capabilities of different embedded intelligent computers. To solve the problem that embedded intelligent de-

收稿日期：2022-08-23；在线发布日期：2023-04-04. **马春燕**（通信作者），博士，副教授，主要研究领域为嵌入式软件系统验证、软件自动化测试与故障定位. E-mail: machunyan@nwpu.edu.cn. **陈 晶**，硕士，主要研究方向为嵌入式系统测试、故障定位. **姚 鼎**，硕士，主要研究方向为嵌入式系统测试、智能化软件开发. **张 涛**，博士，教授，主要研究领域为软件工程、故障定位、智能化软件开发.

vices have different configurations and cannot be performance tested through fixed models, this paper proposes a neural network model generation algorithm based on neural evolution algorithm to generate neural network models that can characterize the intelligent computing capabilities of embedded intelligent devices, and inversely infer the intelligent computing capabilities of embedded intelligent computers based on the complexity of the model. Firstly, based on the Roofline theory model, the advantages of integrating computing potential mining, resource adaptation, and unified evaluation indicators are used to propose a computing capability evaluation framework that can adapt to various embedded intelligent computers, and its rationality is analyzed. Secondly, a neural network model generation algorithm with the goal of generating the maximum complexity model is proposed. By using the neural evolution algorithm, the computational intensity of the generated model approaches the upper limit of the computational intensity of the embedded intelligent computer, fully exploring the computing potential of the device under test, and making the evaluation results more objective. Then, using a fixed upper machine as a reference, the generated neural network model is cross-operated between the device under test and the upper machine. The floating-point operation per second during the two inference task executions is used as the calculation factor to provide a general formula for computing capability evaluation, which can realize the comparative analysis of computing capabilities of different embedded intelligent computers. Finally, in the Mindspore-cpu, Tensorflow-cpu, and Mindspore-ascend310 frameworks, Huawei Atlas200 is evaluated. Compared with the five neural network models commonly used in benchmark tests, the evaluation results of the neural network models generated in this paper are more reasonable, proving that the intelligent computing capability of two DaVinci cores is 42.37 times that of eight Cortex-A55 cores. The experimental results are within the theoretical expected range, indicating that the model generation method based on neural evolution algorithm proposed in this paper exhibits stable evaluation effects under different computing frameworks. In summary, the computing capability evaluation method for embedded intelligent computers proposed in this paper mainly addresses issues such as resource adaptation, computing potential mining, neural model structural changes, and unified performance evaluation indicators during the evaluation process. It can improve the accuracy of computing capability evaluation for embedded intelligent computers and shorten the product evaluation cycle.

Keywords neural evolutionary algorithm; embedded intelligent computer; computing capability evaluation; neural network models; Atlas200

1 引 言

嵌入式智能计算机的评测是前沿研究热点之一. 随着嵌入式智能计算机在航空航天、零售、交通运输、制造业及农业等领域的推广应用，嵌入式智能系统软硬件结构也越来越复杂化、智能化. 嵌入式智能计算机根据不同的应用场景形成了不同的分类. 在设计各种嵌入智能计算机时，其智能计算能力评测是测试智能计算机执行推断任务时的执行速度和运行性能的重要手段，评测结果的合理性直接影响嵌入式智能计算机的优化改进方向.

嵌入式智能计算机的评测复杂度和难度都大于通用计算机. 和通用计算机相比，嵌入式智能计算机存在多种神经形态计算的优化方案和机器学习加速器，例如，利用卷积神经网络中权值共享的特点进行设计和优化的 ShiDianNao 架构[1]；关注 k-nn、k-means 等机器学习算法的 PuDianNao 架构[2]；利用硬件解决稀疏神经网络非规整性的 Cambricon-X 架构[3]. 同时，嵌入式智能计算机相较于通用计算机，对基准测试集的约束更多，要求更高. 一方面，嵌入式环境下存储和计算资源相对紧缺，已有的基准测试集模型规模大，需要人工进行裁剪和适配；另一方面，嵌入式设备之间的计算环境各异，已有模型的平台环境（如：库文件、操作系统等）不支持适用于某些嵌入式设备的测试集可能无法在其他设备上使用. 因此，嵌入式基准测试集需要进行针对

性设计，人工构建测试集非常费时，且难以迁移和复用，导致测试成本增加，难度提升. 目前，对嵌入式智能计算机的性能评测，往往是结合不同应用场景的特点，建立不同的性能评价指标，通过评测框架对其进行定量分析.

业内普遍采用基准测试[4]对嵌入式智能计算机进行评测. 基准测试由一组基准测试程序集和评价指标构成. 基准测试程序集在被测平台上的执行速度可以体现被测平台的计算能力. 因此，基准测试常用于评估各种软件和硬件平台之间的相对计算能力[5]. 采用基准测试的智能计算能力评估方法需要提供统一且可信的评价体系[6]，以评估并对比当前存在的各种软硬件结构所实现的神经网络运算加速方法[2, 7-8].

采用基准测试评测嵌入式智能系统的方法存在严峻的挑战. 目前，嵌入式智能系统从 CMOS 工艺[9]、集成电路到架构的各个层面均可以定制，用于加速神经网络的数学运算，所以其在运算和能耗效率方面都显著超越桌面处理器. 针对这些工艺、材料、器件和体系结构，给出合适的性能指标和量化方法，对嵌入式智能计算评测领域的研究起到至关重要的作用. 但根据在通用计算芯片设计中得到的经验，难以找到普适的最优器件、架构或算法[10-11]，目前已经出现了多种架构和技术来实现各种形式的神经形态计算和机器学习加速，所以，需要根据不同的嵌入式智能计算机分类，分别设计基准测试集和评价指标，这为基准测试带来了大量工作. 目前一组基准测试程序集和评价指标难以适应配置多样化的嵌入式智能系统，其复用能力有限. 不同的基准测试方法拥有不同的评价指标体系，难以给出统一的性能指标对嵌入式智能计算能力进行量化，使不同嵌入式智能计算机的评测结果对比分析与论证存在困难. 此外，不论是理论研究还是应用需求，神经网络结构和计算模型不断地引入新的算法，这为采用基准测试评测嵌入式智能计算机的方法在实时应对新神经网络结构和计算模型方面带来了极大挑战. 而且基准测试集中神经网络模型的计算强度是否最大限度接近被测嵌入式智能计算机的计算强度，在目前的研究中尚未提及，也未对其进行分析或证明，所以，测试集中神经网络模型的计算强度存在一定的随机性，导致评价结果的参考价值打折扣.

本文提出一种动态生成高复杂度神经网络模型作为测试程序进行智能计算机计算能力评测的新方法. 首次借助神经进化算法动态生成计算强度逼近嵌入式智能计算机的神经网络模型作为测试程序. 论文方法可以解决评测过程中嵌入式资源适配、计算潜力挖掘、神经模型结构变化以及统一性能评价指标等问题. 主要解决以下技术挑战：

（1）如何保证算法生成的神经网络模型计算强度接近于被测设备？计算强度匹配是评测结果可靠的必要条件. 神经网络模型的计算强度取决于其时间复杂度和空间复杂度的比值，然而通过改变网络节点的参数和数量，单一调整模型的时间复杂度或空间复杂度几乎不可能实现，两者往往同增或同减. 因此，本文采用统一量化指标公式，通过适配嵌入式资源，给定神经网络模型在嵌入式智能设备上的最慢执行速度，设计神经进化算法并不断迭代，选取收敛后的最大复杂度对应的网络模型，将其作为对应嵌入式智能计算机的匹配模型.

（2）如何设计高效的基于神经进化的计算能力评测算法？神经进化算法收敛所需的时间是整个评测算法的瓶颈，减少神经进化算法的执行次数可以很大程度上提高算法效率. 考虑到在大量嵌入式设备的计算能力评测过程中，需要选择配置相对不变的上位机作为对照，往往采用虚拟机或云服务器. 因此，本文针对相对固定配置的上位机，在执行速度的预测范围内，选取合适的步长，预先生成上位机模型集. 算法执行过程中仅需根据模型的执行速度进行搜索，在大量设备评测场景下，仍能保持较高的评测效率.

本文工作的主要贡献包括：

（1）提出了基于神经进化算法的智能计算机能力评测框架. 基于 Roofline 理论模型，从计算潜力挖掘、资源适配、评价指标统一等 3 方面提出了嵌入式智能计算机计算能力评测框架，并对框架的合理性进行分析. 该评测框架适配不同的嵌入式智能计算机分类和配置，采用统一测评指标公式对嵌入式智能计算机进行评测.

（2）提出了基于神经进化算法的神经网络模型生成方法. 与目前采用固定神经网络模型作为测试用例对嵌入式智能计算机的计算能力进行评测相比，本文首次利用神经进化算法动态生成神经网络模型作为测试用例对其进行评测. 方法选取深度学习中具有代表性的网络层作为基因节点进行编码，并构建节点模型以描述不同种类神经网络层中所包含的各种参数，在限制模型推断任务执行速度的情况下，不断加强种群中单个神经网络个体在空间复杂度、时间复杂度和嵌入式智能计算机计算能力等 3 个目标的适应度，最终生成的神经网络模型其计算强度逼近嵌入式智能计算机的计算强度. 该方

法充分挖掘嵌入式设备的计算潜力，使评测结果更客观.

（3）给出嵌入式智能计算机计算能力评测的通用评测指标公式. 采用环境固定的上位机作为对照，分别在待测设备和上位机上交叉运行生成的神经网络模型，并以两次执行推断任务时的每秒浮点运算次数（Floating-point Operations Per Second，缩写为 FLOPS）作为计算因子，给出嵌入式智能计算机计算能力评测的通用公式，以实现对不同嵌入式智能计算机的评测结果进行对比分析与论证.

（4）在 Mindspore-cpu、Tensorflow-cpu 和 Mindspore-ascend310 框架下对 Atlas200 设备进行计算能力评测，相比基准测试中常用的 5 种神经网络模型，采用本文生成的神经网络模型的测评结果更趋于合理，证实两个 DaVinci 核心的智能计算能力是八个 Cortex-A55 核心的 42.37 倍，处于 Atlas200 设备参数理论计算范围之内，说明方法的有效性.

接下来，本文第 2 节介绍计算能力评测方面的相关工作. 第 3 节介绍了嵌入式智能计算机计算能力的评测框架及合理性分析. 第 4 节详细阐释基于神经进化算法的嵌入式智能计算机计算能力评测方法，并对方法进行讨论. 第 5 节对实验对象、实验方法和实验结果进行阐释. 第 6 节对本文工作进行总结.

2 相关工作

在计算能力理论研究方面，Williams 等人在 2009 年提出了 Roofline 模型，该模型提出了计算强度的概念，通过计算能力和内存带宽两个指标对计算能力进行抽象，有效地衡量了应用程序性能与峰值性能之间存在差距[12]. Marques 等人在 2021 年提出了 Mansard Roofline 模型，该模型针对处理器核心数增加、处理器类型增加的情况，改善了 Roofline 方法过度抽象微观体系结构复杂性的问题，提供了对计算系统更加有效的建模[13].

在通用计算能力评测方面，已经有较为成熟的方案. 1988 年，由工作站制造商 HP、DEC、MIPS、SUN 等联合推出的 SPEC 标准性能评测机构，为软硬件制造商、学术研究组织等提供了基准评测平台，是目前 CPU 性能评价较为客观而可信的性能评测标准之一[14]. Guthaus 等人在 2001 年针对 ARM 架构的嵌入式系统提出了 Mibench 基准测试程序，相较于 SPEC 测试集，Mibench 基准测试程序能够更好地反映嵌入式系统在指令分配、内存行为以及并行性方面的性能[15]. Poovey 等人在 2009 年提出了 EEMBC 基准测试集，通过分析基准评测指标的内在程序行为，以抽象属性量化嵌入式平台的性能指标[16]. 瑞典布京理工学院在 2010 年提出了面向多处理器嵌入式系统的 ParMiBench 基准测试集，该测试集在 Mibench 的基础进行了并行化实现[17].

在智能计算评测领域，学术界和产业界同样存在大量的基准测试方法，旨在评价和指导各种类型的智能计算芯片. 由于各类嵌入式智能计算芯片的目标功能不同，其设计的思路、基准测试方法的具体内容以及性能评估指标也不尽相同. Ignatov 等人在 2018 年通过分析移动端可用硬件加速资源，并结合了移动端可用智能计算框架、编程模型以及模型运行限制，提出了关注于 Android 端 AI 性能评测的 AIBenchmark 基准测试集[18]. 斯坦福大学 Coleman 等人在 2019 年提出 DAWNBench 基准测试集，该测试集将计算系统在深度学习时所花费的训练时间和训练精度作为系统性能的主要评价指标[19]. 谷歌、百度、英特尔、AMD、哈佛大学与斯坦福大学等在 2020 年提出了 MLPerf 基准测试集[20]，MLPerf 关注于计算系统在模型训练时的计算能力，该测试集包含 ResNet、SSD、Transformer 等常见的网络模型. Brewer 等人在 2020 年提出了 iBench 基准测试集，并为高性能边缘计算系统的测试提供了性能指标[21].

国内在智能技术评测方面也取得了不错的成果. 中国人工智能产业发展联盟在 2017 年针对深度学习性能提出了 BenchIP 基准测试集，该测试集包含宏观和微观两部分基准测试集，用于分析系统存在的瓶颈以及对系统进行优化[22]. 徐青青等人在 2019 年针对卷积神经网络提出了 DLBenchmark 评测方法[23]. 清华大学与鹏城实验室在 2021 年针对大规模计算集群联合提出了 AIPerf 测试集[24]，该测试集基于 NNI 开源框架，以自动化机器学习[25]为负载，能够自适应地扩展到各种规模的机器，同时提供了评价分数.

本文工作和已有工作的对比如表 1 所示，文献[19-22]采用固定测试集从不同方面评测设备的性能，虽然未给出各维度指标的归一化计算方法，但有利于从不同性能角度对比设备. 文献[24]和本文均从计算能力角度对设备进行评测，均以自适应的方法产生测试集，但前者以自动化机器学习为工作负载，侧重于考虑分布式集群环境的计算能力评测，而本文侧重于嵌入式环境的自适应和计算潜力充分挖掘. 已有的基于基准测试集的评测方法，采用固定的神经网络模型集，在多设备评测过程中面临的局限性有：无法充分挖掘设备潜力、部分测试程序

表 1　本文工作与已有方法的对比

方法	关注点	测试集自适应	评测指标	核心策略	可扩展性	归一化计算
DAWNBench[19]	端到端深度学习训练和推理	×	训练时间、训练成本、推理时间、推理成本	限制模型训练时间	受限于模型的规模	×
MLPerf [20]	机器学习性能评测	×	吞吐量、每秒处理的数据量、训练时间	超参数限制和借用	受限于工作负载的大小	×
iBench [21]	分布式边缘设备性能评测	×	吞吐量、延迟、带宽、预处理时间、效能	数据模拟、负载均衡	受限于网络延迟和预训练模型的规模	×
BenchIP [22]	智能处理器设计	×	性能、能耗、面积、准确性	微基准和宏基准	嵌入式处理器、桌面处理器、服务器处理器	×
AIPerf [24]	高性能计算评测	√	每秒操作数	贝叶斯优化	智能计算机&其他设备	√
本文方法	嵌入式智能计算评测	√	每秒操作数	进化算法	嵌入式智能计算机&其他设备	√

无法适配设备资源、测评指标差异大导致难以对比多台设备的计算能力. 其中，无法充分挖掘设备潜力的主要原因是部分模型的计算强度与待测设备的计算强度不匹配；无法适配设备资源的主要原因是部分模型过大，待测嵌入式设备的资源无法满足模型运行的最低要求；多台计算机的计算能力难以对比的主要原因是运行多个模型会得到多个评测结果，如何对其统一量化，并进行多台设备间计算能力的对比，尚未明确. 已有工作提供多维的评测结果主要是针对实际应用中的神经网络模型，选择其更适合的设备，但选择过程中往往需要在多个维度之间进行权衡甚至取舍. 如何对不同架构的嵌入式智能计算机的智能计算能力采用一致标准进行排序的问题，以及评测过程中灵活适配不同类型的嵌入式资源、充分挖掘嵌入式计算机的计算潜力、动态调整测试集中神经模型结构等问题，都面临新的挑战. 而本文主要关注嵌入式智能计算机的计算能力，通过生成尽可能逼近待测计算机计算强度上限的网络模型，以配置相对固定的上位机为参照，提供统一的量化指标，在计算能力比较、设备采购场景等有一定优势.

本文给出的评测方法同样使用了神经网络模型的执行速度作为智能计算能力的衡量的参与要素. 不同之处在于，上述评测方法都是使用已有模型进行智能计算能力评测，难以考虑到模型和被测计算机之间计算强度的匹配程度. 而本文方法使用的测试模型是通过神经进化算法自动生成的，可以让模型的计算强度尽可能接近被测计算机的计算强度. 同时，本文将神经网络模型的执行速度作为智能计算能力的要素，推导出测试框架下统一评测指标公式. 本文方法拥有以下优势：

（1）用于评测的神经网络模型可以充分挖掘嵌入式设备的计算潜力，使评测结果更客观；

（2）用于评测的神经网络模型可以适配各种嵌入式智能计算机，使评测方法更灵活；

（3）采用统一评测标准对嵌入式智能计算机进行评测，使不同嵌入式智能计算机的评测结果可以被对比分析与论证.

3　嵌入式智能计算机计算能力评测框架及合理性分析

3.1　基于Roofline模型的嵌入式智能计算机计算能力评测分析

Roofline 理论模型描述了在计算平台的算力和带宽限制下，计算任务能够达到的理论性能上限[26]. 嵌入式智能计算机主要通过运行工作负载（即神经网络模型），以工作负载的运算速度（在本文的描述中，运算速度均指执行推断任务时的每秒浮点运算次数 FLOPS）反映其计算能力. 因此，基于 Roofline 理论模型，在嵌入式智能计算机计算能力评测中，计算任务和计算平台的相关含义如下：

A. 计算任务的计算量、访存量和计算强度

计算任务指神经网络模型的推断任务. 计算任务的计算量（C，单位为 Floating-point Operations，缩写为 FLOPs）指模型实现一次前向传播需要的浮点运算次数，即模型的时间复杂度；计算任务的访存量（V，单位为 Byte）指模型实现一次前向传播过程中，内存数据交换的总量，即模型的空间复杂度. 计算任务的计算强度（I，单位为 FLOPs/Byte）为计算量与访存量的比值.

B. 计算平台的算力、带宽和计算强度上限

嵌入式智能计算机的算力（π，单位为 FLOPS）和带宽（β，单位为 Byte/s）与其结构以及神经网络框架有关. 例如，在仅有 CPU 作为计算核心或安装 CPU 计算的神经网络框架的嵌入式智能系统中，

算力指 CPU 的理论计算能力峰值，带宽指系统主存的理论性能峰值；在以 GPU 计算为主且安装相应神经网络框架的嵌入式智能系统中，算力指 GPU 的理论计算能力峰值，带宽指显存的理论性能峰值. 计算机的计算强度上限（$I_{\max}$，单位为 FLOPs/Byte）为算力与带宽的比值.

根据 Roofline 理论模型，计算任务的实际可达性能（P，单位为 FLOPS）与其计算强度的关系如图 1 所示，图 1 描述的是计算任务的实际可达性能 P 及其计算强度 I 之间的关系，其中计算强度 I 为计算任务的计算量 C 和访存量 V 的比值. π 和 β 分别表示计算平台计算性能和存储性能的理论峰值. 计算任务的实际可达性能受限于 π 和 β，当计算任务的计算强度接近 $I_{\max}$ 时，才能充分利用嵌入式智能计算机的计算和存储性能. 计算任务的实际可达性能同时受限于计算平台的算力和带宽，且理论性能峰值为计算平台的算力. 当计算任务的计算强度大于计算平台的计算强度上限时，计算任务在当前计算平台处于计算瓶颈区域，其单次运行时间由计算平台的算力决定($t=C/\pi$)；反之，则处于带宽瓶颈区域，其单次运行时间由计算平台的带宽决定($t=V/\beta$). 因此，为了精确地评测嵌入式智能计算机的计算能力，需要调整优化神经网络模型的架构和配置，使模型的计算强度接近待测计算机的计算强度上限，以充分挖掘待测计算机的计算潜力.

本文所做的工作是评测嵌入式智能系统的最大计算能力，在这个过程中，需要结合模型的计算强度和智能计算设备的计算强度. 在给定模型最小运行速度的前提下，当模型的计算强度和设备的计算强度上限相等时，算法生成的神经网络模型最复杂.

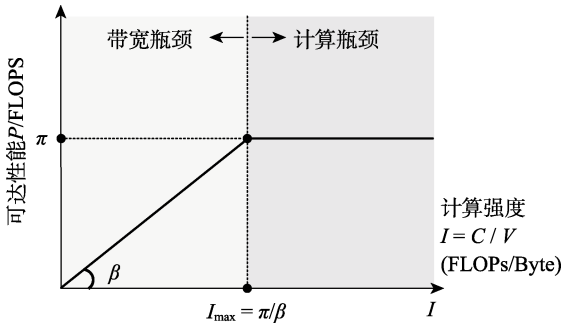


图 1 计算任务瓶颈分析

最大计算强度，即计算强度上限，是指嵌入式智能计算机的算力和带宽的比值，该值往往难以精确计算. 本文根据 Roofline 模型，通过神经进化算法不断调整神经网络模型的模型的结构和参数，以寻找适配待测设备计算强度上限的最佳模型，从而更加精准地测试嵌入式计算机的最大计算能力，方便对不同设备的计算能力进行对比.

3.2 嵌入式智能计算机计算能力评测框架

本文根据 Roofline 理论模型，借鉴 Linpack 和 Top500 的成功经验[24]，以每秒操作数（Operations Per Second，OPS）作为基准分数来量化衡量 AI 计算机能力，从计算潜力挖掘、资源适配和指标统一 3 个方面提出了嵌入式智能计算机计算能力评测框架. 该框架以神经网络模型在待测计算机上的运算速度作为评测指标，论文采用固定配置的上位机作为对照，综合模型在两个平台上运算速度参数，对计算能力进行评测. 提出的评测框架自上而下分为模型生成层、模型评估层和能力评测层，如图 2 所示，三个层次分别解决潜力挖掘、资源适配和指标统一三个问题. 模型生成层基于神经进化算法，不

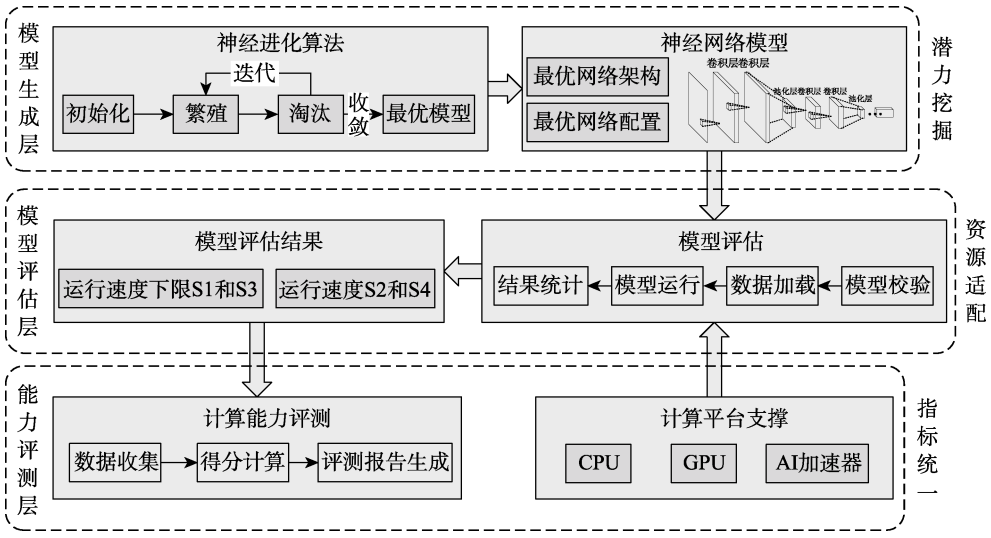


图 2 计算能力评测框架

断优化神经网络模型的架构和配置，以充分挖掘被测平台的计算潜力. 模型评估层对模型进行校验，加载相应的数据，并在不同的计算平台上适配运行. 能力评测层对不同平台上模型的运行结果进行收集，计算评测得分并生成评测报告.

为了对不同设备进行相对公平的评测，本文评测框架采用环境固定的上位机作为对照，分别在待测设备和上位机上执行神经进化算法，生成两个神经网络模型并交叉运行. 图 2 中，模型评估模块对两次执行神经进化算法生成的神经网络模型的运行结果进行统计. 第一次执行神经进化算法时，设定模型在下位机的 OPS 下限 S_1（单位为 FLOPS）作为约束条件，并将得到的最大复杂度神经网络模型放到上位机运行，得到 OPS 为 S_2（单位为 FLOPS）. 第二次执行神经进化算法时，设定模型在上位机的 OPS 上限 S_3（单位为 FLOPS）作为约束条件，并将得到的最大复杂度神经网络模型放在下位机运行，得到 OPS 为 S_4（单位为 FLOPS）. 运算速度下限 S_1 和 S_3 取值与平台的配置大小正相关. 在运算速度下限不变的情况下，则神经网络模型的复杂度越高，运算速度 S_2 和 S_4 越大，即对应计算平台的计算能力越强. 论文提出了以 S_1、S_2、S_3 和 S_4 作为计算因子的计算能力评测的公式（9），该公式及其推理过程将在 4.2 节进行详细介绍.

3.3　嵌入式智能计算机计算能力评测框架的合理性分析

本文评测框架主要解决目前基于基准测试的嵌入式智能计算机计算能力评测过程中出现的计算潜力未充分挖掘、嵌入式资源适配性差以及性能评价指标未统一问题.

（1）现有基准测试方法分析

基准测试方法主要面向固定领域进行计算能力评测，评测过程中的工作负载相对固定，具有评测周期短的优势. 但在采用固定负载的情况下，由于不同嵌入式智能计算机的计算强度不同，测试过程中会遇到不同的性能瓶颈，导致测试的侧重点不同. 通过扩充测试集，可以在一定程度上平衡测试集对不同嵌入式智能计算机的适配情况，但该方法仍然存在以下问题：

①在嵌入式智能计算机领域，实验和生产环境中使用到的神经网络推断模型的结构变化非常快. 基准测试程序集和评价指标复用能力有限，难以适应配置多样化的嵌入式智能系统，需要根据不同的嵌入式智能计算机的硬件和资源限制，分别设计基准测试集和评价指标，或者对已有模型进行裁剪，导致人力成本增加、评测周期变长.

②基准测试集中神经网络模型的计算强度是否最大限度接近被测嵌入式智能计算机的计算强度，在目前的研究和实践中尚未进行分析和论证. 因此，测试集中的神经网络模型的计算强度存在一定的随机性，可能无法充分挖掘待测设备的计算潜力，导致评测结果的参考价值打折扣.

③使用多个测试集会得到多个不同评分，对嵌入式智能计算机的计算能力排序带来不便，即便通过一些公式将多个维度的得分综合起来得到一个分数，公式的有效性、准确性也需要进一步的考证. 难以给出统一的性能指标和量化参数，对嵌入式智能计算能力进行量化，使不同嵌入式智能计算机的评测结果对比分析存在困难.

（2）本文评测框架的合理性分析

为解决以上问题，本文没有采用固定神经网络模型对嵌入式智能计算机的计算能力进行评测，而是利用神经进化算法生成神经网络模型进行评测，并让生成模型的计算强度尽可能接近于嵌入式智能计算机的计算强度. 同时采用执行推断任务时的每秒操作数 OPS 作为嵌入式智能计算机计算能力的衡量因素. 本文的评测框架的合理性体现在以下三个方面：

①将神经网络模型编码为基因序列，通过随机交叉和变异扩充神经网络模型集，依据模型的复杂度设计适应度函数，在限制运算速度的前提下，筛选复杂度更高的模型，多次进行算法迭代，将收敛后的最高复杂度模型作为适配当前待测设备的模型，解决了模型与嵌入式设备间的适配问题，无需人工对模型进行裁剪.

②基于 Roofline 模型，选择神经网络模型的复杂度（时间复杂度和空间复杂度）作为衡量其计算强度的重要因素，即在给定模型最慢运算速度的条件下，一个嵌入式智能计算机能够正常运行的模型越复杂，则该计算机的计算能力也就越高. 该方法解决了基准测试集模型计算强度随机，无法充分挖掘待测设备计算潜力的问题.

③评测过程中采用固定配置的上位机作为参照（S_3），不同嵌入式设备的计算能力均以上位机作为基准进行归化，在下位机上模型的运算速度 OPS 下限 S_1 不变的前提下，统计算法生成的模型在上位机和待测设备上的运算速度（S_2 和 S_4）. 基于理论推理得到的标准化评分公式计算不同设备的最终得分，解决多种基准测试集导致的计算能力难以统一量

化，以及不同设备间的计算能力排序问题.

4 嵌入式智能计算机计算能力评测方法

4.1 基于神经进化算法的神经网络模型生成算法

神经网络模型生成算法是为了在限制模型推断任务执行速度的情况下，使生成模型的复杂度最大化，为之后通过最大复杂度模型的运行结果评估嵌入式智能计算机的计算能力奠定基础. 在模型生成的过程中，会涉及模型的时间复杂度、模型的空间复杂度以及模型在下位机上运行的 OPS 三个参数.

神经网络模型生成算法需要实现 3 个目标：

（1）让模型运行的 OPS 尽可能地靠近给定的限制；

（2）当存在计算瓶颈时，需要增加模型的空间复杂度，降低模型的时间复杂度；

（3）当存在内存瓶颈时，需要降低模型的空间复杂度，增加模型的时间复杂度.

但在实际操作中，因为模型的时间复杂度和空间复杂度一般是一起增大或者减小的，这使得直接输出想要的模型存在很多困难，所以本文采用了神经进化算法来生成符合条件的模型.

神经进化使用基于群体的优化方法不断提高群体中每个神经网络的质量，从而根据其要解决的问题生成越来越好的神经网络[27]. 神经进化算法得到问题全局最优解的过程是在种群一次次迭代繁衍过程中保持最优解并用它们改进其他解的过程，该过程是收敛且可靠的，并且整个种群会在一次次的迭代中不断优化. 在种群迭代过程中，交叉算子可以探索给定的两个父代个体之间的解空间. 变异算子通过随机的方式改变染色体中的某个基因节点，保证种群中个体的多样性，增加神经进化算法的搜索空间，引导算法在迭代中找到全局最优解.

4.1.1 算法框架

神经网络模型生成算法的流程如图 3 所示，包括种群初始化、种群繁殖、基因序列更新、种群淘汰以及最终结果模型的生成.

（1）种群初始化. 需要构造初始神经网络模型集合，并进行初始化，包括设置集合大小以及每个神经网络模型的初始化方法.

（2）种群繁殖. 种群繁殖主要通过交叉算子和变异算子，在已有的神经网络模型基础上产生新的神经网络模型，增加种群中神经网络模型的多样性.

（3）基因序列更新. 利用变异或交叉算子产生新的基因序列时，需要同步更新该基因序列对应的神经网络模型在上位机和下位机上执行速度，并重新计算神经网络模型在上位机和下位机上运行的 OPS，用于之后的种群淘汰操作.

（4）种群淘汰. 种群淘汰的主要任务是淘汰种群中较差的基因序列，使整个神经网络模型集合朝着复杂度更高的方向迭代. 为了防止结果过拟合，淘汰过程需引入一定程度的随机计算，总体上，复杂度越高的模型对应的基因序列，越不容易被淘汰.

（5）结果模型生成. 使用神经进化算法多次迭代后，种群中的神经网络模型在下位机上的运算速度收敛于给定的运算速度限制. 此时，找出种群中复杂度最高的模型对应的基因序列，并根据该基因序列生成对应的神经网络模型，最终生成的神经网络模型即为神经网络模型生成算法的最优解.

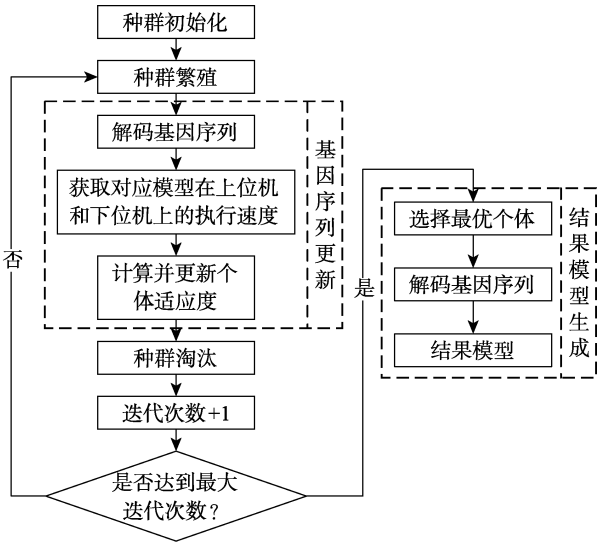


图 3 神经网络模型生成算法流程

神经网络模型生成算法的伪代码如算法 1 所示（第 1 行~第 12 行）. 关于算法的收敛性和迭代次数限制，具体描述如下：

算法 1. 模型生成算法(*GeneratingModel*).

输入: 最大迭代次数 *count*,初始种群的大小 *size*,变异操作的执行次数 *k*，下位机上模型的运算速度下限 *limit*.

输出: 最优基因序列对应的模型 *m*.

1. *srand*()
2. *E*←*initPopulation*(*size*, *k*) //构造初始模型集合
3. WHILE (*count* > 0)
4. *breedingPopulation*(*E*, *size*) //增加模型多样性
5. *M*←*decodedGeneSequences*(*E*)
6. R_{up}←*getModelRunResultsByUp*(*M*) //获取模型集在上位机的运行结果
7. R_{down}←*getModelRunResultsByDown*(*M*) //获取模

型集在下位机的运行结果

8. g←*selectingPopulation*(E, R_{up}, R_{down}, *limit*, *size*)　// 淘汰无法运行或复杂度较低的神经网络模型

9. *count*=*count*−1

10. END WHILE

11. m←*decodedGeneSequence*(g)

12. RETURN m

本文神经进化算法以给定条件下的神经网络模型复杂度反映待测计算平台的计算能力，根据 Roofline 理论模型，当且仅当模型的计算强度（时间复杂度与空间复杂度的比值）等于计算平台的计算强度上限时，神经网络模型与嵌入式智能计算平台之间最适配，此时算法计算的神经网络模型得分最高. 当嵌入式智能模型的计算强度大于或小于嵌入式智能计算平台的计算强度时，存在计算瓶颈或带宽瓶颈，此时模型的得分随着两者计算强度的差异增加而递减. 综上，以模型计算强度为自变量，以模型得分为因变量，该函数存在唯一极大点. 因此，经过多次迭代，模型的计算强度将趋于计算平台的计算强度上限，即算法收敛于唯一极大点.

神经进化算法初始化时，给定迭代次数的一个经验值，之后在多次实验过程中，通过下述方法判断算法是否收敛，确定算法的迭代次数：当连续 5 代种群中最优个体的适应度变化不超过 2%，则认为算法已经收敛. 以算法收敛时的最优个体作为适配待测设备的最佳神经网络模型，并确定迭代次数，形成算法执行的时间约束.

4.1.2　基因序列子算法设计

神经进化算法是一种基于种群的算法，每个解对应一条基因序列. 基因序列的编码方式以及基因序列的交叉变异算子是神经进化算法的基础，以下将从基因序列编码、基因序列交叉算子以及基因序列变异算子 3 个方面对基因序列子算法进行设计.

（1）基因序列编码

本文采用链式架构空间作为搜索空间. 为了描述模型的拓扑结构，将网络中每一层结构都作为一个基因节点，每一个网络拓扑结构都可以映射为一条基因序列. 同时，为了能够将基因序列映射到网络拓扑结构，每个基因节点需要包含一系列的参数，用于描述模型中各层的详细结构. 基因节点类型及包含的相关参数如表 2 所示.

表 2　基因节点类型及其参数

序号	基因节点类型	参数
1	全连接层	激活函数、神经元个数
2	卷积层	激活函数、过滤器个数、卷积核大小
3	Flatten 层	无
4	池化层	池化类型、池化核大小

由于神经网络模型的构造存在约束，层与层之间的顺序不能随意修改. 为了保证生成的网络模型的有效性，将基因序列的链式结构分为卷积节点列表和全连接节点列表. 卷积节点列表包含卷积层网络和池化层网络；全连接层列表包含全连接层网络. 卷积节点列表和全连接节点列表间通过一个 Flatten 层相互连接. 为了保证模型能够正常运行，在构建模型时，默认为模型添加了一个输入层和一个输出层. 输入层是一个输入形状为（1，3，32，32）的卷积层；输出层是一个有 10 个神经元的全连接层. 基因序列的链式结构如图 4 所示.

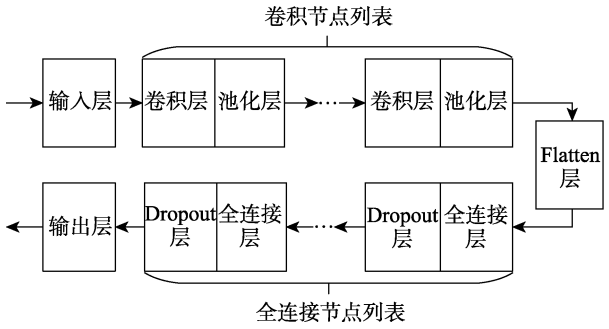


图 4　基因序列链式结构

（2）基因序列交叉算子

在神经进化算法中，通过随机挑选出两个个体当作父代个体，然后通过交叉算子生成两个子代个体，并加入到种群中. 常用的交叉算子有单点交叉、多点交叉、均匀交叉、半均匀交叉、三亲交叉、循环交叉等. 本文通过随机方法选出用于交叉的两个父代个体，通过单点交叉的方法实现交叉算子，如图 5 所示.

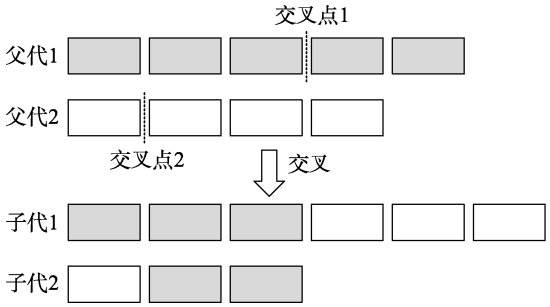


图 5　基因序列的交叉过程

在本文中，为了保证交叉算子产生的基因序列能够生成可正常运行的模型，交叉算子仅作用于卷积节点列表和全连接节点列表之间. 两条基因序列

交叉时，首先选择在卷积节点序列或全连接节点序列交叉，随后再执行交叉算子. 基因序列交叉算法的伪代码如算法 2 所示（第 1 行~第 13 行）.

算法 2. 基因序列交叉算法(*CrossingGeneSequence*).

输入: 待交叉的两条基因序列 g_1, g_2.

输出: 交叉后的两条基因序列 g_3, g_4.

1. g_3=g_1
2. g_4=g_2
3. *choice*←*random*(0, 1)
4. $spot_1$=*random*(0, *len*(g_1) –1) //随机产生交叉点1
5. $spot_2$=*random*(0, *len*(g_2) –1) //随机产生交叉点2
6. IF (*choice* == 0) //卷积节点序列交叉
7. *copyConvolutionSequence*(g_3, g_2, $spot_2$)
8. *copyConvolutionSequence*(g_4, g_1, $spot_1$)
9. ELSE //全连接节点序列交叉
10. *copyFullConnectionSequence*(g_3, g_2, $spot_2$)
11. *copyFullConnectionSequence*(g_4, g_1, $spot_1$)
12. END IF
13. RETURN g_3, g_4

（3）基因序列变异算子

神经进化中常见的变异算子有功率变异、均匀变异、非均匀变异、高斯变异等，这些变异算子对种群进行改进，直至得到最优解. 本文中采用均匀变异，通过增加或删除基因节点以及修改基因节点的参数，针对性地调整模型的时间复杂度和空间复杂度.

在实际的变异过程中，不同的变异操作对模型的时间复杂度和空间复杂度的调整程度不一样，为了能够产生出满足条件的模型，本文实现的变异操作均是随机产生，以尽可能增加种群多样性.

在对基因序列进行变异操作时，需要对卷积层的过滤器个数、全连接层的神经元个数等参数设定一个最小值，本文中将最小值设置为 4，并确保随机修改后的数据不小于该值. 该方案作为一种兜底策略，有效保证了基因序列生成的模型的正常执行. 同时，在随机修改时的修改差值需要以 4 为倍数，保证变异能够产生明显的效果，提高算法收敛的速度. 基因序列变异算法的伪代码如算法 3 所示(第 1 行~第 13 行）.

算法 3. 基因序列变异算法(*VariatingGeneSequence*).

输入: 原始基因序列 g_1.

输出: 变异后的基因序列 g_2.

1. *choice*←*random*(0, 2)
2. *index*←*random*(0, *len*(g_1)–1)
3. g_2=g_1
4. IF (*choice* == 0) //添加节点
5. *type*←*random*(0, 1)
6. *node*=*generateGeneNode*(*type*)
7. *insertGeneNode* (g_2, *node*, *index*)
8. ELSE IF (*choice* == 1) //删除节点
9. *removeGeneNode*(g_2, *index*)
10. ELSE
11. *modifyGeneNode*(g_2, *index*) //随机修改节点的参数，确保为4的倍数
12. END IF
13. RETURN g_2

4.1.3 种群子算法设计

种群算法由三个子算法组成，分别为种群初始化算法、种群繁殖算法和种群淘汰算法，下面依次对三个算法进行详细介绍.

（1）种群初始化算法

神经进化算法始于一个随机的种群.初始的种群需要包含各种各样的个体，在本文中，每个个体为一条基因序列，对应一个神经网络模型. 初始化种群的一个重要的目标是使得种群中的个体尽可能均匀分布于解空间，合适的种群初始化方式可以提升解的多样性，提高找到全局最优解的概率.

每条基因在初始化时，基因序列中的卷积节点列表和全连接节点列表是空的. 所以在初始化种群时，需要对每条基因节点进行一定次数的变异，以保证种群的多样性. 种群初始化算法的伪代码如算法 4 所示（第 1 行~第 11 行）.

算法 4. 种群初始化算法(*InitPopulation*).

输入：初始种群的大小 *size*,变异操作的执行次数 *k*.

输出：初始化后的种群 *E*.

1. *E*←∅
2. WHILE (*size* > 0)
3. *g*←*generateInitialGeneSequence*() //构造初始基因序列
4. WHILE (*k* > 0) //利用变异算子将初始序列变异 *k* 次
5. g_2←*variateGeneSequence*(*g*)
6. *k*=*k*–1
7. END WHILE
8. *addGeneSequenceIntoPopulation*(*E*, g_2)
9. *size*=*size*–1
10. END WHILE
11. RETURN *E*

（2）种群繁殖算法

种群的繁殖通过基因序列交叉算子和变异算子实现. 在设定一个繁殖上限后，通过随机变异和随机交叉的方式扩充种群中的基因序列个数，直到种群中基因序列的个数达到设定的上限.

随机变异从已有的种群集合中随机选择一条基因序列并将其复制,之后按照 4.1.2 节中介绍的基因变异操作对复制得到的基因序列进行变异，并将变异后的基因序列加入到种群集合中.

随机交叉操作在选择基因时不同于完全随机. 为了保证算法能更快地得到有效结果，选择基因时会优先选择更加优秀的基因序列. 第 1 条父基因的选择是完全随机的，从全部的基因序列中随机选择出一条基因序列；第 2 条父基因的选择会根据基因序列数据按照一定的概率选择其作为交叉操作的另一基因序列，基因序列数据包含基因序列对应模型在上位机上运行的 OPS 以及在下位机上运行的 OPS.

若一个基因序列在上位机上运行的 OPS 为 OPS_{up}，在下位机上运行的 OPS 为 OPS_{down}. 同时，在整个种群中，上位机运行的 OPS 最小值为 OPS_{min}，下位机运算速度最小限制值为 OPS_{limit}. 第 2 次选择的概率计算如公式（1）所示.

$$p=\frac{OPS_{min}}{OPS_{up}}\cdot e^{-(OPS_{down}-OPS_{limit})^2} \tag{1}$$

其中，概率公式的前半部分表示基因序列对应的模型在上位机上运算速度越慢，越容易被选择. 后半部分表示基因序列对应的模型在下位机上运算速度越接近于下位机运算速度最小限制值为 OPS_{limit}，基因序列越容易被选择. 种群繁殖算法的伪代码如算法 5 所示（第 1 行~第 13 行）.

算法 5. 种群繁殖算法(*BreedingPopulation*).

输入：种群 *E*,初始种群的大小 *size*.

1. WHILE (*len*(*E*) < *size*× 1.2) //扩充初始种群20%的模型序列
2. *choice*←*random*(0, 1)
3. IF (*choice* == 0) //随机交叉
4. g_1, g_2←*randomChooseGeneSequences*(*E*)
5. g_3, g_4←crossGeneSequence(g_1, g_2)
6. *addGeneSequenceIntoPopulation*(*E*, g_1)
7. *addGeneSequenceIntoPopulation*(*E*, g_2)
8. ELSE //随机变异
9. *g*←*randomChooseGeneSequence*(*E*)
10. g_1←*variateGeneSequence*(*g*)
11. *addGeneSequenceIntoPopulation*(*E*, g_1)
12. END IF
13. END WHILE

（3）种群淘汰算法

现有神经进化算法往往采用“优胜劣汰”的淘汰策略，存在算法难以收敛或容易陷入局部最优解的难题. 此外，在嵌入式智能系统计算能力评测过程中，由于交叉和变异操作的随机性，可能出现部分基因序列对应的神经网络模型无法在设备上正常运行. 因此本文针对上述问题，结合嵌入式智能系统计算评测的特点，从以下三点进行优化：

①首先淘汰在嵌入式智能设备上无法正常运行的个体（见算法 6 的第 2~3 行）；

②由于适应度越高的个体越有可能被选择为最佳解，本文从父代种群中适应度在前 1/4 的优秀个体中随机保留一部分（见算法 6 的第 14、17~19 行），以加速算法的收敛；

③为了解决陷入局部最优的难题，算法需要有一定的机率挑选适应度相对不够优秀的个体. 即使部分基因序列陷入了局部最优，依然还存在一些基因序列可以跳出当前的局部最优情况. 剩余个体中随机选择一条基因序列，根据该基因序列对应神经网络模型的时间和空间复杂度（见算法 6 的第 10~11 行），以及该模型在上位机和待测设备上运行的 OPS（见算法 6 的第 7 行），按公式（2）所描述的概率确定其是否被淘汰.

算法 6. 种群淘汰算法(*SelectingPopulation*).

输入：种群 *E*,模型在上位机和下位机运行结果 R_{up} 和 R_{down},下位机模型运算速度下限 *limit*, 初始种群的大小 *size*.

输出：最优基因序列 g_{opt}.

1. FOR (each *g* in *E*)
2. IF (*isValid*(*g*)) //淘汰无法运行的模型序列
3. *removeGeneSequence*(*E*, *g*)
4. END IF
5. END FOR
6. FOR (each *g* in *E*)
7. *OPS*←*getOPS*(R_{down}, *g*)
8. IF (*OPS* >= *limit*)
9. $OPS_{\min}$←*searchMinimumOPS*(R_{up})
10. $C_{\max}$←*calculateMaximumTimeComplexity*(*g*)
11. $V_{\max}$←*calculateMaximumSpaceComplexity*(*g*)
12. END IF
13. END FOR
14. *greatSeqs*←*getGreatSeqs*(0.25) //随机选择适应度在前 1/4 的一部分

15. WHILE (*len*(*E*) > *size*×0.8) //淘汰初始种群 20%的模型序列
16. *g*←*randomChooseGeneSequence*(*E*)
17. IF (*g* in *greatSeqs*)
18. CONTINUE
19. END IF
20. p_1←*calculateProbabilityByLowerOPS*(*g*, R_{down}, *limit*)
21. p_2←*calculateProbabilityByUpperOPS*(*g*, R_{up}, OPS_{min})
22. p_3←*calculateProbabilityByComplexity* (*g*, C_{max}, V_{max})
23. $p=p_1+p_2+p_3$
24. IF (*random*(0, 300) > p×100)
25. *removeGeneSequence*(*E*, *g*)
26. END IF
27. END WHILE
28. g_{opt}←*getOptimalGeneSequence*(*E*)
29. RETURN g_{opt}

$$\begin{cases} p = p_1 + p_2 + p_3 \\ p_1 = e^{-(OPS_{down}-OPS_{limit})^2} \\ p_2 = \delta(OPS_{min} / OPS_{up}) \\ p_3 = \delta(C_m / C_{max}) \cdot \delta(V_m / V_{max}) \end{cases} \tag{2}$$

本文的淘汰策略采用两次随机选择的方式进行. 第一次随机选择是完全随机，在整个种群样本中随机选择出一条基因序列；第二次选择会根据基因序列数据按照一定的概率选择将其保留，保留概率由三部分的乘积构成，概率计算如公式（2）所示. 其中：

（1）OPS_{down} 为基因序列对应模型在下位机上运行的 OPS，OPS_{limit} 为下位机运算速度最小限制值，p_1 可以理解为模型在下位机上运行的 OPS 越接近于下位机运算速度最小限制值，对应基因序列的保留概率越大.

（2）OPS_{min} 为当前种群中所有满足速度最小限制条件下的基因序列对应模型在上位机运行的 OPS 最小值，OPS_{up} 为基因序列对应模型在上位机上运行的 OPS，式中 $\delta(x)$ 函数当 $x \geqslant 1$ 时，取值为 1，当 $x<1$ 时，取值为 x. p_2 可以理解为模型在上位机上运行的 OPS 越小，对应基因序列的保留概率越大.

（3）C_m 为基因序列对应模型的时间复杂度，C_{max} 为当前种群中所有满足速度最小限制条件下的基因序列对应模型的时间复杂度最大值，V_m 为基因序列对应模型的空间复杂度，V_{max} 为当前种群中所有满足速度最小限制条件下的基因序列对应模型的空间复杂度最大值. p_3 可以理解为模型的时间复杂度和空间复杂度越大，即模型越复杂，对应基因序列的保留概率越大.

种群淘汰算法的伪代码如算法 6 所示（第 1 行~第 29 行）.

4.1.4 模型复杂度计算方法

在执行进化算法的过程中，如何比较模型的复杂度至关重要. 以模型的时间复杂度和计算平台的算力为横坐标轴，以模型的空间复杂度和计算平台的带宽为纵坐标轴，将带宽、算力、空间复杂度和时间复杂度放在同一坐标系中进行分析，如图 6 所示. 根据 Roofline 理论模型，在模型的推断阶段，完成一次前向传播的时间 $t=\max(C_m/\pi, V_m/\beta)$，单位为 s，其中 C_m 和 V_m 分别表示模型的时间复杂度和空间复杂度，π 和 β 分别表示计算平台的算力和带宽. 图 6 中，点 P 表示一个算力和带宽分别为 x 和 y 的嵌入式智能计算机. 根据前文，在执行神经进化算法时，需要将最终筛选出来的模型的执行时间控制在一个特定的值. 假定将生成模型的 OPS 控制为 1，当得到一个在 P 平台上推断任务执行速度为每秒一次的模型时，该模型的时间复杂度和空间复杂度一定处于 L_1 和 L_2 两条虚线段上，需要在这些模型中筛选出复杂度最高的模型.

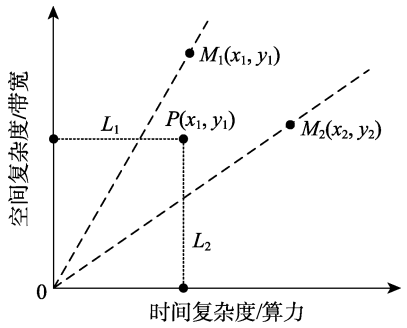


图 6 模型复杂度比较

根据上位机的计算强度的大小可能会出现三种情况. 当上位机的计算强度大于嵌入式智能计算机的计算强度时，上位机可以表示为图中的 M_2 点，L_1 虚线段上的模型在上位机上的执行时间大于 L_2 虚线段上的模型，且 L_1 虚线段上的模型之间执行时间近似；当上位机的计算强度小于嵌入式智能计算机的计算强度时，上位机可以表示为图中的 M_1 点，L_2 虚线段上的模型在上位机上的执行时间大于 L_1 虚线段上的模型，且 L_2 虚线段上的模型之间执行时间近似；当上位机的计算强度等于嵌入式智能计算机的计算强度时，在上位机运行时间最长的模型即复杂度最高的模型.

由于上位机的计算强度一般不会和嵌入式智能计算机的计算强度相等，因此第三种情况一般不会出现. 对于剩下的两种情况，在上位机上运行并取执行时间更长的模型仅能筛选出一部分. 但剩余的模型中，所有模型的时间复杂度相近或者空间复杂度相近，这有利于之后进一步的筛选.

在上位机上运行后，只需要单独比较模型的时间复杂度或空间复杂度即可. 本文中所使用的各个网络层的时间和空间复杂度的计算方式如表 3 所示. M 表示卷积核输出特征图的边长，K 表示每个卷积核的边长，C_{in} 表示本层的输入通道数，C_{out} 表示本层的输出通道数.

表 3　各网络层的复杂度计算方式[24]

网络层类型	时间复杂度	空间复杂度
卷积层	$O(M^2\cdot K^2\cdot C_{in}\cdot C_{out})$	$O(K^2\cdot C_{in}\cdot C_{out}+M^2\cdot C_{out})$
池化层	$O(M^2\cdot K^2\cdot C_{in}\cdot C_{out})$	$O(1)$
全连接层	$O(C_{in}\cdot C_{out})$	$O(C_{out})$
Flatten 层	$O(1)$	$O(1)$

4.2　嵌入式智能计算机计算能力评测标准

本文将根据所得模型在上位机上运行的 OPS 来对嵌入式智能计算机的计算能力进行评分. 模型在上位机上运行的 OPS 越小，则代表模型的复杂度高，相应的嵌入式智能计算机的计算能力也越高，因此使用上位机上模型运行的 OPS 的倒数作为模型的评分.

但是，根据 Roofline 理论模型，实际的模型复杂度与该评分之间存在一定偏差. 如图 7 所示，P_1 点表示下位机，即嵌入式智能计算机，P_2 点表示上位机，即测试平台.

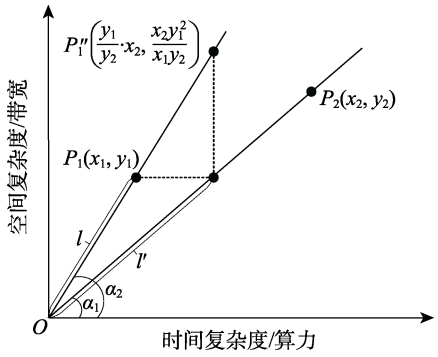


图 7　模型复杂度计算

假设在限制下位机运行 OPS 为 S_1 下，复杂度最大的模型为 M_1；若 M_1 模型在上位机上运行的 OPS 为 S_2，则在限制上位机运行 OPS 为 S_2 (S_3)下，复杂度最大的模型为 M_2；S_4 是将 M_2 模型放在下位机上运行的 OPS.

图 7 中 OP_1 和 OP_1'' 的比值如公式（3）所示.

$$\frac{OP_1}{OP_1''}=\frac{S_4}{S_1}=\frac{\tan\alpha_1}{\tan\alpha_2} \tag{3}$$

同时可以使用 $1/S_2$ 作为模型 M_2 的复杂度，即 OP_1' 的长度为如公式（4）所示.

$$l'=\frac{1}{S_2} \tag{4}$$

l 和 l' 的关系可以表示如公式（5）所示.

$$\frac{l}{l'}=\frac{\sin\alpha_1}{\sin\alpha_2}. \tag{5}$$

模型复杂度由时间和空间复杂度两部分组成，如果把时间复杂度记为 C，空间复杂度记为 V，那么模型复杂度可以表示如公式（6）所示.

$$W=k_1\sqrt{C^2+k_2V^2}. \tag{6}$$

式中 k_1 和 k_2 均是常数，k_2 的值决定了时间和空间复杂度的权重. 如果将上位机的算力和带宽看为一个较为均衡的状态，即 α_1 的度数为 45°，汇总得到公式（7）.

$$\begin{cases}\dfrac{S_4}{S_1}=\dfrac{\tan\alpha_1}{\tan\alpha_2}\\ \dfrac{l}{l'}=\dfrac{\sin\alpha_1}{\sin\alpha_2}\\ l'=\dfrac{1}{S_2}\\ \alpha_1=\dfrac{\pi}{4}\\ 0<\alpha_2<\dfrac{\pi}{2}\end{cases} \tag{7}$$

化简计算后，模型的评分计算如公式（8）所示.

$$l=\frac{\sqrt{S_1^2+S_4^2}}{\sqrt{2}S_2S_1} \tag{8}$$

在上面的计算过程中，模型 M_2 可以提前生成. 因为上位机平台已经确定，可以针对上位机预先生成一个模型集，覆盖到所有可能需要的模型. 如果预测 OPS 出现的范围是 1~100（FLOPS），那么可以预先按 0.1 的步长使用本文提到的模型生成算法生成 1000 个模型，构成上位机模型集. 在得到模型 M_1 后，可以直接在上位机模型集中搜索到对应的模型 M_2 即可，这样可以减少一次模型生成，提高对模型评分计算的效率. 由于以上模型评分的计算方法是基于特定上位机的，即不同配置的上位机会测试出不同的数据. 为了保证测试结果的一致性，可以将

在上位机运行测试的相关操作迁移到特定配置的虚拟机或者云服务器上.

在实际评测过程中，可能存在 OPS 限制无法满足的情况. 例如当上位机性能远大于下位机时，上位机得到的 S_2 的值可能会远小于 1 导致计算误差较大. 这时需要对 OPS 的值进行缩放，然后再进行计算. 根据 Roofline 理论模型，若将 S_1 的值缩放为 kS_1，则模型的复杂度会变为缩放前的 $1/k$ 倍，S_2 的值会缩放为 kS_2. 同时，由于设备性能之间可能存在较大差异，模型在待测设备上的运算速度下限 S_1 可能不同. 为了在同一尺度下比较不同的嵌入式智能计算机，为了比较不同的嵌入式智能计算机，需要将 S_1 的值缩放到同一个值. 若将 S_2 的值缩放为 S_{limit}，那么计算能力评分的计算如公式（9）所示.

$$L=\frac{\sqrt{S_1^2S_3^2+S_2^2S_4^2}}{\sqrt{2}S_{limit}S_2S_3} \tag{9}$$

公式（9）描述算法生成的神经网络模型在待测平台和上位机（基础对照）的运算速度综合指标，单位为 1/FLOPS，其取值表示为运算速度的倒数. L 的取值越大，则表明算法生成的模型在被测平台上的运算速度越慢，同时反映了该神经网络模型更加复杂，可以更加充分地挖掘被测平台的计算潜力.

4.3 方法的讨论

4.3.1 方法的实用性

现有的基准测试方法往往采用一个相对固定的程序集，在测试不同设备时可能出现设备的计算资源或带宽资源无法满足程序的运行要求，因此需要人工对已有测试集进行裁剪以适配具体测试设备的不同配置情况，例如：CPU、内存等配置信息. 本文提出的基于神经进化算法生成的网络模型则自动适配测试设备的配置情况，无需人工调整.

基于神经进化算法生成可以表征嵌入式智能计算机计算能力的模型，从模型的复杂度逆推嵌入式智能计算机的智能计算能力. 为了保证生成模型的有效性，本文选取了深度学习中具有代表性的网络层作为基因节点进行编码. 同时，本文构建节点模型来描述不同种类网络层中所包含的各种参数，以保证基因序列能正确完整地映射到网络拓扑结构上.

在以往的嵌入式智能计算机计算能力评测过程中，需要根据设备的配置情况，对已有基准测试集进行扩充或裁剪. 由于不同设备之间的差异，导致相互之间进行计算能力对比时，难以构造合适的测试集. 本文提出的方法给定执行速度限制的条件下最大化生成模型的复杂度，尽可能挖掘设备的潜力，可以适应不同设备的差异，实现设备计算能力的定量评估.

4.3.2 方法的合理性

目前不同评测方法之间在测试集选择（面向具体领域或设备类型）和测评指标定义方面有较大差异. 不同的评测指标下，由于不清楚各种神经网络模型推断任务的计算量，无法评估嵌入式智能计算机对应的理论性能真值. 所以，单独看一个嵌入式智能设备的评测得分无法衡量其是否接近真值，虽然可以通过对比不同嵌入式智能计算机的评测得分，衡量哪个计算机的智能计算能力更优，但是该方法计算的评测得分是否合理仍需考究.

本文旨在解决当前计算能力评测过程中使用固定测试集无法充分挖掘设备潜力、部分测试程序无法适配设备资源、测评指标差异大导致难以对比多台设备的计算能力三个问题. 根据 Roofline 理论模型，固定测试集在待测设备上运行时可能存在计算瓶颈或带宽瓶颈，而基于神经进化算法生成的网络模型，经过多次迭代，尽可能优化模型的结构和参数，使得模型的计算强度尽可能接近待测设备的计算强度上限，以充分挖掘设备的计算潜力.

本文采用链式架构空间作为神经网络模型的搜索空间，虽然神经进化算法生成的网络模型结构和实际应用中的模型之间的确存在差异，但卷积层和全连接层仍是所有神经网络模型的主流结构，基本涵盖了网络模型的所有运算. 因此，本文算法生成的模型能代表现有网络模型中运算相关的特征，评测结果可以作为多个嵌入式智能计算机计算能力比较的依据. 未来工作中可以考虑增大搜索空间，或者基于特定应用领域对模型结构做一定的限制，使算法生成的神经网络模型更加贴近实际.

4.3.3 方法成本的评估

相较于使用基准测试集的方法，使用神经进化算法的评测方法需要更长的评测时间. 实验采用的 Atlas200 嵌入式智能设备支持运行的模型为 om 格式（华为的推理架构模型格式），需要使用 ATC（Ascend Tensor Compiler）工具将 onnx 格式（深度学习模型的开放格式）的模型转换为 om 格式，而此过程耗时较长，是总测试时长过长的主要原因. 如果上位机和下位机分别根据神经进化算法生成神经网络模型，需要分别将生成的模型转换为 om 格式，耗时较长. 由于上位机配置信息是确定的，本文为优化总评时长，通过预估上位机生成的神经网络模型运行的 OPS 范围，在神经进化算法运行之前

按 0.1 的步长预先生成上位机的神经网络模型集，神经进化算法运行过程中可以在上位机生成的神经网络模型集中,根据 OPS 挑选合适的神经网络模型，节省上位机生成和转换模型的时间，仅生成下位机神经网络模型并对其进行转换，以大幅优化总测评时长. 目前取得的平均测试时长为 2 小时 50 分钟. 实际生产环境中，基准测试集包含多种网络模型（iBench 测试集包含 10 种网络模型，BenchIP 测试集包含 11 种网络模型），总测试时长为 1 小时左右. 为尽量降低设备间歇运行引入的误差，本文在利用算法生成模型后，多次运行模型，以运行速度的平均值作为最终结果. 相较于产品长达数周的评测周期而言，2 小时 50 分钟的测试时长是可接受的.

此外，评测不同的设备时，上位机的配置和环境应保持不变，且上位机的计算能力和内存性能应为一个较为均衡的状态，将测试基准从模型的计算强度转移到上位机的计算强度上. 然而，相对于模型计算强度不匹配的问题，保证上位机之间计算强度接近更加容易. 为了保证测试结果的一致性，可以将在上位机运行测试的相关操作迁移到特定配置的虚拟机或者云服务器上.

4.3.4　影响评测结果的因素分析

本文方法评测嵌入式智能计算机的计算能力，仅需要考虑神经网络模型的计算强度，与模型是用来处理图像、视频还是音频没有关联关系. 同时不同模态的输入和模型推断任务的准确度等指标依赖于具体神经网络模型、数据集、硬件可靠性等，对神经网络模型的强度没有影响. 因此，评测过程不同神经网络模型执行任务的指标不属于本文计算能力评测的范围，不会对评测结果产生影响. 实验过程中神经网络模型的输入均为随机生成的(3，32，32)矩阵向量，计算能力评测结果与数据集的质量无依赖关系，由于图像相较于视频、音频等模态更易于随机构造，所以，本文在测试过程中选用了分类和目标检测任务，都是针对图像的.

影响评测结果的主要因素是在嵌入式智能设备和上位机上分别执行神经网络的推断任务时的每秒操作数 OPS. 由于设备性能的差异，神经网络模型单次运行的 OPS 统计信息可能存在波动，因此本文在实验过程中以一百次运行结果的平均值作为最终的 OPS，降低波动对评测结果合理性的干扰.

5　实验验证

5.1　实验设置

本文实验使用的上位机 CPU 为 AMD2600，主频锁定为 3.6GHz，内存为 16GDDR4 以及频率为 3200MHz. 被测嵌入式智能计算机使用 Atlas200，集成了昇腾 310 AI 处理器，可以实现图像识别、图像分类等，广泛用于智能摄像机、机器人、无人机等物联网端侧智能场景. 被测设备提供两种架构的处理器，一种是 DaVinci AI Core，该处理器是 AI 处理器，为神经网络模型的训练和运行提供了优化，另一种是 A55 Arm Core，该处理器是 ARM 处理器，未做 AI 相关的优化. 下位机部署 Mindspore-cpu、Mindspore-ascend310 以及 Tensorflow-cpu 框架，其中 Mindspore-cpu 框架和 Tensorflow-cpu 框架均使用 Atlas200 的 Cortex-A55 核心进行计算，Mindspore-ascend310 框架使用 Atlas200 的 DaVinci AI 核心进行计算. 该嵌入式智能计算设备的硬件参数如表 4 所示.

表 4　Atlas200 平台硬件参数

产品名	Atlas200
处理器	昇腾 310 AI 处理器 • 2 个 DaVinci AI Core • 8 个 A55 Arm Core (最大主频 1.6GHz)
AI 算力	• 半精度(FP16): 4/8/11 TFLOPS • 整数精度(INT8): 8/16/22 TOPS
内存	• 类型：LPDDR4X • 位宽：128bit/64bit • 容量：8GB/4GB • 速率：3200Mbps • 支持 ECC

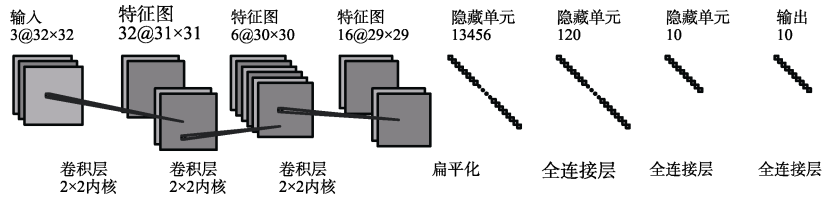
目前不同评测方法之间在测试集选择（面向具体领域或设备类型）和测评指标定义方面有较大差异. 不同的评测指标下，由于不清楚各种神经网络模型推断任务的计算量，无法评估嵌入式智能计算机对应的理论性能真值，所以，单独看一个嵌入式智能设备的评测得分无法衡量其是否接近真值. 虽然可以通过对比不同嵌入式智能计算机的评测得分，衡量哪个计算机的智能计算能力更优，但是无法衡量该方法计算的评测得分是否合理. 为此，本文基于华为 Atlas200 表 4 中的处理器和 AI 算力的硬件参数,预估 Mindspore-Ascend310 和 Mindspore-cpu 的计算能力比值. 理论浮点峰值=CPU 主频×CPU 每个时钟周期执行浮点运算次数×CPU 数量，Atlas200 的处理器的通用计算核心 Cortex-A55 采用 ARMv8.2 架构,支持 4 个单精度浮点的 SIMD 指令，所以,Cortex-A55 核心计算的最高理论算力=处理器核心数×主频×单个指令处理浮点数个数，8×1.6GHz×4FLOPS/Hz=51.2GFLOPS，同时还需要考虑到并行过程中会出现的性能损失. 昇腾 310 处理器的

DaVinci AI 计算核心的单精度计算理论算力在 2×10^3GFLOPS~5.5×10^3GFLOPS 之间，即(4TFLOPS~11TFLOPS)/2，所以，根据华为官方提供的处理器、主频等信息计算得到：昇腾 310 处理器的 DaVinci AI 计算核心的计算能力约是 Cortex-A55 核心的 40~100 倍左右，由于 DaVinci AI 计算核心和 Cortex-A55 核心使用同一套内存系统，即带宽相等，而前者的算力显然大于后者，导致算法生成模型时 DaVinci AI 计算核心的带宽瓶颈更加突出. 此外，实际生成神经网络模型的计算强度和设备的计算强度上限之间存在一定偏差，导致对该设备的评测得分变低，最终得分的比值会略小于两者计算能力的比值，因此将比值预期设定为 30~100.

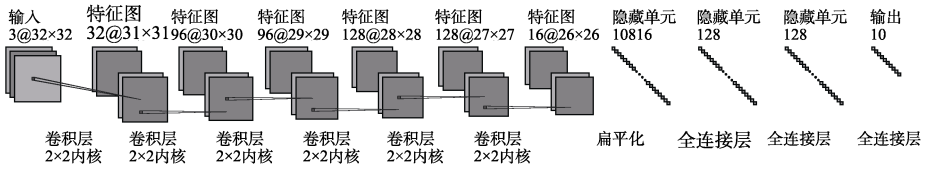
本文的计算能力评测得分根据公式(9)计算得到，计算结果是精确值，但由于 Atlas200 平台的 AI 算力参数是范围值，因此 DaVinci AI 计算核心和 Cortex-A55 核心的评测得分的比值同样为范围值. 平台提供 4/8/11TFLOPS 三级 AI 算力支持，运行过程中根据神经网络模型规模自动调整算力级别，导致比值预期范围较大.

目前大多采用基准测试集的方法进行计算能力评测，本文以现有基准测试[20,22]中 5 种常用的神经网络模型作为基准评判程序，对不同的神经网络模型在同一华为 Atlas200 平台上进行测试，得到 Mindspore-Ascend310 和 Mindspore-cpu 的计算能力性能评分的比值. 因此，本实验设置两组对比试验，分别是 5 种神经网络模型之间及其与本文方法的对比实验和不同计算框架下本文方法的对比实验.

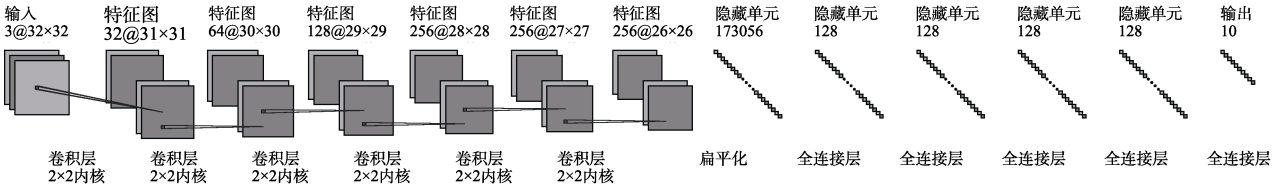
不同神经网络模型之间的对比实验中，以表 5 所示的 5 种神经网络模型作为对照，与采用神经进化算法生成的网络模型进行对比，涉及的 5 种神经模型结构实例如图 8 所示，不同神经网络模型的差



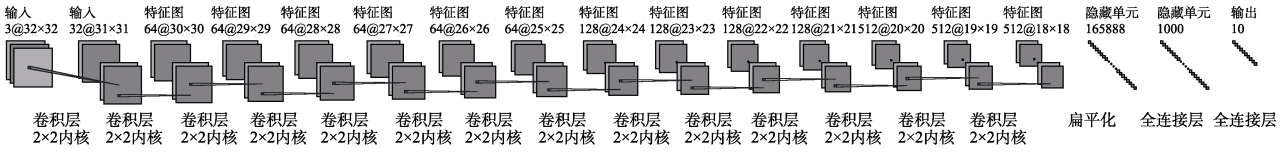
(a) LeNet



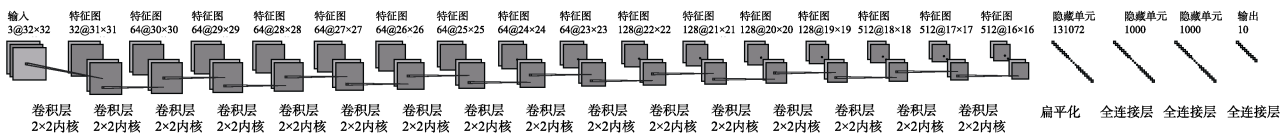
(b) AlexNet



(c) VGGNet



(d) ResNet



(e) Faster R-CNN

图 8 神经网络模型结构实例

表 5　神经网络模型对比实验的对象

网络模型	数据集	应用场景
LeNet	MNIST	手写数字识别
AlexNet	ImageNet	图像分类
VGGNet	ImageNet	图像分类
ResNet	ImageNet	图像分类
Faster R-CNN	PASCAL VOC 2012	目标检测

异主要体现为卷积层和全连接层个数和比例不同，尽可能体现不同的计算强度. 神经网络模型结构的差异主要影响不同应用场景下具体任务的求解精度，而计算能力评测过程中则更多关心的是模型的时间复杂度和空间复杂度，因此本文忽略神经网络结构的区别对计算能力评测过程中的影响.

不同计算框架下的对比实验中，涉及三种神经网络框架（Mindspore-cpu、Mindspore-ascend310 和 Tensorflow-cpu），以及 DaVinci AI 计算核心与 Cortex-A55 核心的两种处理器框架，DaVinci AI Core 处理器为 AI 处理器，为神经网络模型的训练和运行提供了优化，A55 Arm Core 处理器是 ARM 处理器，未做 AI 相关的优化，所以，两种架构处理器下的对比实验可以增强实验结果的普适性和可靠性. 实验过程中神经网络模型的输入均为随机构造的(3,32,32)向量，与模型的计算强度无关.

现有工作主要是评测计算机的性能，而本文主要从计算能力角度对嵌入式智能计算机进行评测. 由于关注点不同，且大多数已有工作受限于设备环境而难以复现，导致无法从定量角度进行对比. 因此，在第 2 节相关工作部分的表 1 中介绍了本文工作和已有工作的定性比较，从不同维度体现本工作的优势和局限.

5.2　实验方法

图 9 是本文采用 python 语言研制的实验工具原理图. 任务配置模块负责解析外部任务参数，并将评测任务发送到计算能力评估模块. 计算能力评估模块根据任务配置检查上位机的环境，利用算法生成模型并统计在上位机运行的结果. 通信模块负责向下位机发送评测任务和指令，并接收和解析指令处理模块返回的结果. 指令处理模块用于接收针对典型智能应用的评测指令，并对指令进行解析，将解析后的结果发送到计算任务执行模块. 计算任务执行模块根据上位机发送的评测指令，检查下位机的环境，运行生成的模型，并返回统计结果. 评测管理模块负责管理评测过程中的任务数据和结果数据，并对其进行本地化存储. 日志模块负责对管理软件运行过程中的操作和异常进行记录，方便后续进行问题定位. 采用该实验工具的实验步骤如下：

（1）搭建实验环境. 在上位机上部署 Mindspore-cpu 以及 Tensorflow-cpu 框架，在被测下位机上部署 Mindspore-cpu、Mindspore-ascend310 以及 Tensorflow-cpu 框架.

（2）安装实验工具. 在上位机上安装代理软件，在被测下位机上安装管理软件.

（3）配置评测任务. 包括网络模型（包括激活函数、神经元个数、过滤器个数、池化类型等）、模型生成算法的初始化参数（包括迭代次数、初始种群大小、变异次数等）等配置. 提供对上位机和下位机中使用到的神经网络框架、计算核心等软硬件环境的设置，以便于管理软件能够发送合适的神经网络模型.

（4）运行 5 种神经网络模型. 采用目前常用的神经网络模型 LeNet，AlexNet，AlexNet-S，VGGNet，ResNet 和 Faster R-CNN，在 Mindspore-ascend310 和 Mindspore-cpu 计算框架下运行，其中，AlexNet-S 表示对 AlexNet 模型的神经元个数进行一定比例缩小后得到的模型. 通过构造相应的网络结构，并将其转换为 Mindspore 框架支持的 om 模型，最后在 Atlas200 平台上运行.

（5）收集 5 种神经网络模型运行数据. 由于上述神经网络模型是确定的，无法计算 S_1、S_2、S_3 和 S_4 的取值并采用公式（9）对 DaVinci AI 计算核心

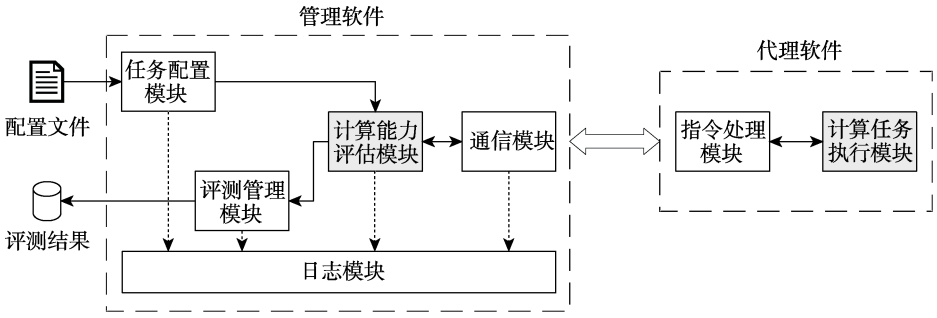


图 9　实验工具原理图

和 Cortex-A55 核心计算能力进行评分. Linpack 是国际上使用最广泛的测试高性能计算机系统浮点性能的基准测试,所以,本文借鉴借鉴 Linpack 和 Top500 的成功经验[24]，以每秒操作数作为基准分数来量化上述 5 种神经网络模型运行时嵌入式计算机的计算能力. 分别统计上述网络模型的运行时间，计算每秒运行的浮点操作数（FLOPS），并计算不同的神经网络模型在两种框架下每秒运行的浮点操作数比值，以衡量昇腾 310 处理器的 DaVinci AI 计算核心和 Cortex-A55 核心的计算能力的比值.

（6）运行本文评测框架. 执行基于神经进化算法的神经网络模型生成程序.

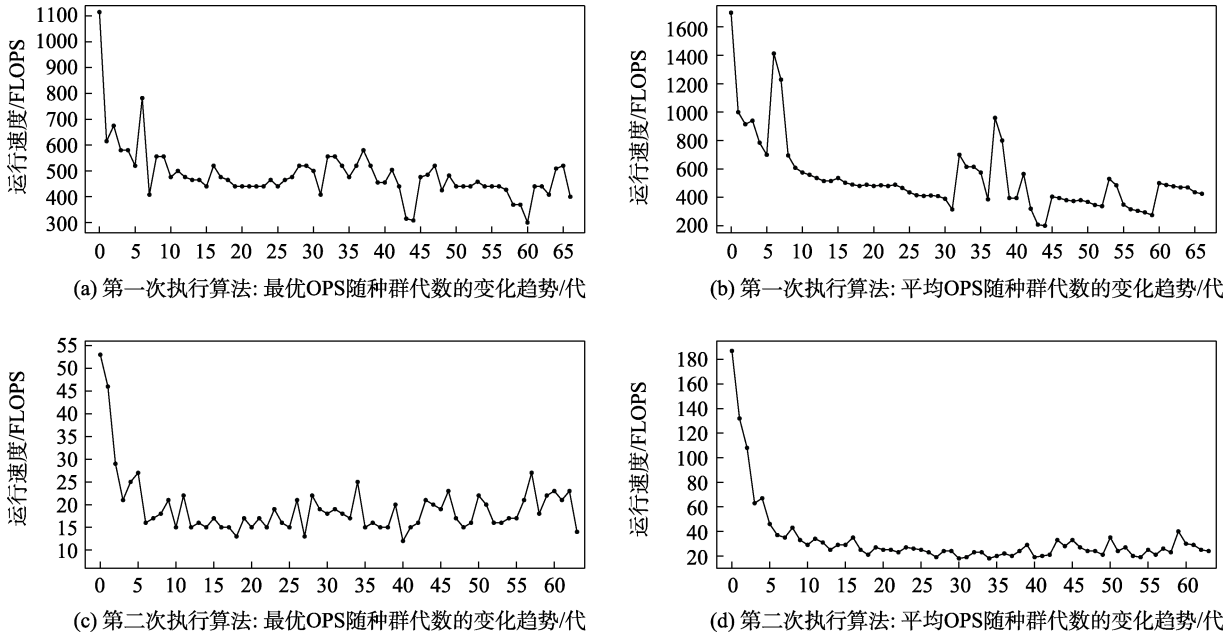
（7）采用本文算法生成的神经网络模型，分别在 Mindspore-ascend310、Mindspore-cpu 和 Tensorflow-cpu 计算框架下运行，首先统计运行时间，其次按照公式（9）计算生成的神经网络模型在 3 种框架下计算能力评分，最后，计算 DaVinci AI 核心和 Cortex-A55 核心计算能力评分的比值.

5.3 实验结果分析

针对 Mindspore-cpu、Mindspore-ascend310 以及 Tensorflow-cpu 三个框架进行性能评估测试,且统一使用 Tensorflow-cpu 作为上位机的模型运行框架. 因为图像处理常用任务推断处理帧数常见的为 24、30、60、120 和 240 等，帧数如果设置太小，导致计算误差较大，如果设置太大，模型的复杂度没有上升空间，所以，根据多次实验的评估，本次实验中将 S_{limit} 的值设为 60 较为合适，对部分代数没有满足条件解的情况，使用上一次迭代的最优值作为当前代数的最优值. 每次测试需要进行两次算法迭代，最后根据公式（9）计算出每次测试的最终评分.

图 10 展示了 Mindspore-cpu 框架的测试结果. 第一次执行神经进化算法时，将 OPS 下限设为 60，图 10(a)表示第一次使用神经进化算法迭代时，种群中最优解的 OPS 值为 400. 图 10(b)表示第一次使用神经进化算法迭代时种群平均 OPS 已经收敛. 第二次执行神经进化算法时，将 OPS 下限设为 400，图 10(c)表示第二次使用神经进化算法迭代时，种群中最优解的 OPS 值为 15. 图 10(d)表示第二次使用神经进化算法迭代时种群平均 OPS 已经收敛. 对应于公式（9），S_1 的值为 60，S_2 的值为 400，S_3 的值为 400，S_4 的值为 15.

图 11 展示了 Mindspore-ascend310 框架的测试结果. 第一次执行神经进化算法时，将 OPS 下限设为 600. 图 11(a)表示第一次使用神经进化算法迭代时，种群中最优解的 OPS 值为 94. 图 11(b)表示第一次使用神经进化算法迭代时种群平均 OPS 已经收敛，此时算法生成的最优神经网络模型实例如图 13 所示，模型由输入、7 个卷积层、扁平化、3 个全连接层和输出组成，其中，输入为(3,32,32)的向量，卷积层的卷积核大小为 2*2，步长为 1，输入为长度为 10 的一维向量. 第二次执行神经进化算法时，将 OPS 下限设为 600，图 11(c)表示第二次使用神经进化算法迭代时，种群中最优解的 OPS 值为 275. 图 11(d)表示第二次使用神经进化算法迭代时种群平均



(a) 第一次执行算法: 最优OPS随种群代数的变化趋势/代

(b) 第一次执行算法: 平均OPS随种群代数的变化趋势/代

(c) 第二次执行算法: 最优OPS随种群代数的变化趋势/代

(d) 第二次执行算法: 平均OPS随种群代数的变化趋势/代

图 10 Mindspore-cpu 框架测试结果

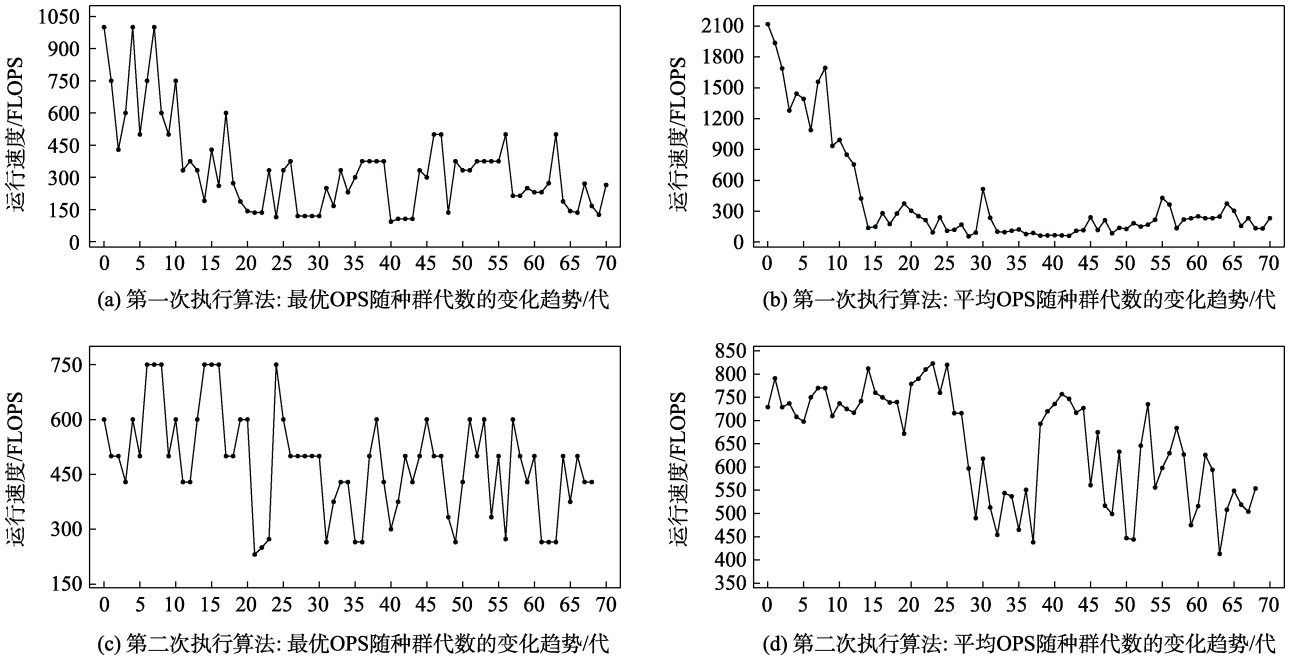


图 11　Mindspore-ascend310 框架测试结果

OPS 已经收敛. 对应于公式（9），S_1 的值为 600，S_1 的值为 94，S_3的值为 600，S_4 的值为 275.

图 12 展示了 Tensorflow-cpu 框架的测试结果. 第一次执行神经进化算法时，将 OPS 下限设为 60，图 12(a)表示第一次使用神经进化算法迭代时，种群中最优解的 OPS 值为 400. 图 12(b)表示第一次使用神经进化算法迭代时种群平均 OPS 已经收敛. 第二次执行神经进化算法时，将 OPS 下限设为 400，图 12(c)表示第二次使用神经进化算法迭代时，种群中最优解的 OPS 值为 8. 图 12(d)表示第二次使用神经进化算法迭代时种群平均 OPS 已经收敛. 对应于公式（9），S_1 的值为 60，S_2 的值为 400，S_3的值为 400，S_4 的值为 8.

5 种常用神经网络模型和本文生成的神经网络模型对比实验结果如表 6 所示. 表 6 中第 1 列为使用的神经网络模型；第 2、3 列分别为不同神经网络模型在 Mindspore-ascend310 和 Mindspore-cpu 框架下的计算能力评测得分；第 4 列为得分的计算方法，

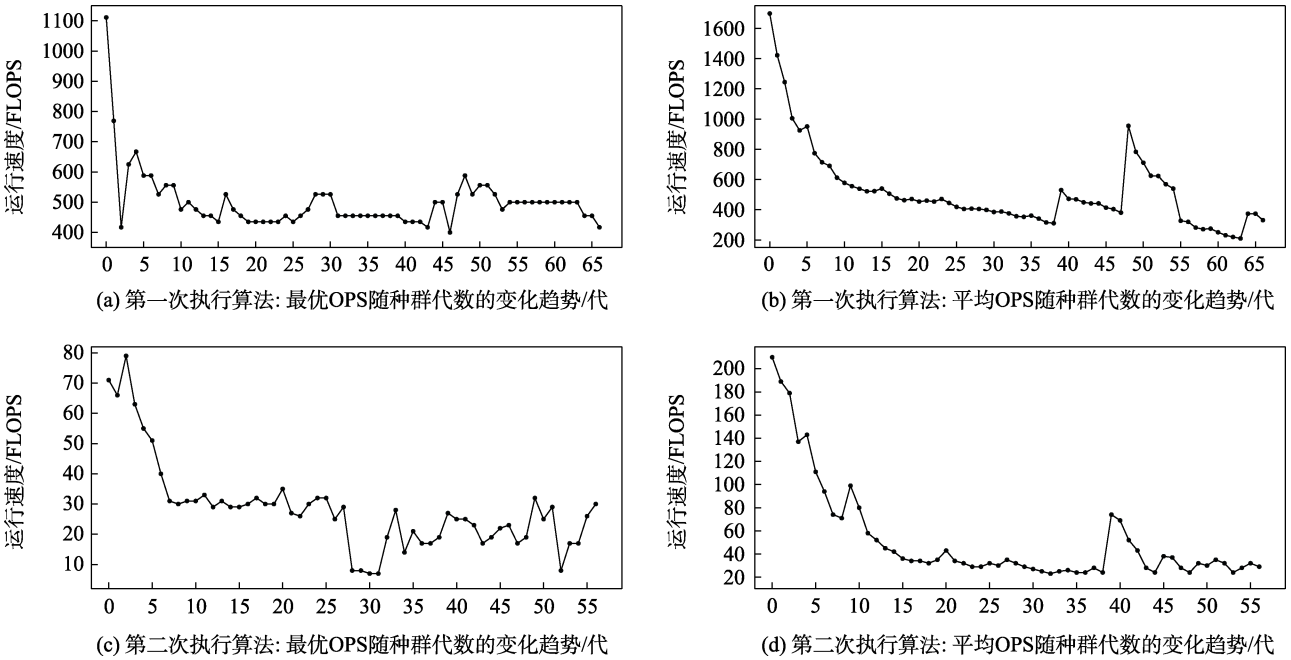


图 12　Tensorflow-cpu 框架测试结果

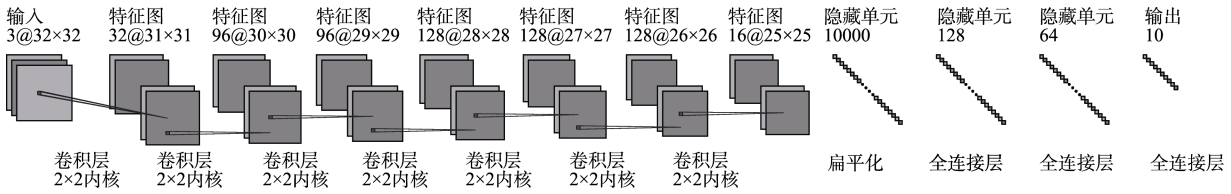


图 13 Mindspore-ascend310 框架下神经进化算法生成的网络模型实例

表 6 常用神经网络模型和本文生成的神经网络模型对比实验结果

网络模型	Mindspore-ascend310 计算能力评测得分	Mindspore-cpu 计算能力评测得分	得分计算方法	总测试时长	Mindspore-ascend310 和 Mindspore-cpu 框架下评测得分的比值
LeNet	625.01	90.90	OPS 统计	1 分钟	6.88
AlexNet	11.21	0.31	OPS 统计	5 分钟	36.16
AlexNet-S	714.51	14.97	OPS 统计	2 分钟	47.73
VGGNet	232.16	1.88	OPS 统计	1 分钟	123.49
ResNet	47.38	0.50	OPS 统计	5 分钟	94.76
Faster R-CNN	56.02	0.63	OPS 统计	5 分钟	88.92
算法生成的神经网络模型	754.2×10^{-4}	17.8×10^{-4}	公式(9)计算	2 小时 50 分钟	42.37

对于确定的神经网络模型则直接统计其在待测设备的 OPS，对于算法生成的神经网络模型，利用公式（9）计算得分；第 5 列为总测试时长，对于算法生成的模型，该时长包括算法收敛、模型转换和模型运行所需的时间；第 6 列为不同模型在 Mindspore-ascend310 和 Mindspore-cpu 框架下评测得分的比值，用于验证是否符合实验预期，即昇腾 310 处理器的 DaVinci AI 计算核心的计算能力大约是 Cortex-A55 核心的 30~100 倍.

根据实验结果，采用已有的神经网络模型进行智能计算能力评测时，不同神经网络模型测试出的评分比值存在较大差异的主要原因是网络模型的卷积层和全连接层的数量以及规模不同，同时 ascend310 和 cpu 两种芯片对卷积和全连接操作的优化程度不同. 此外，Atlas200 硬件平台提供 4/8/11TFLOPS 三级算力支持，运行过程中根据模型规模自动调整算力级别，增加了不同模型评分比值间的差异. AlexNet、AlexNet-S、Faster R-CNN 和 ResNet 四个模型计算的昇腾 310 处理器的 DaVinci AI 和 Cortex-A55 的计算能力的比值在预期范围内，LeNet 和 VGGNet 模型计算的比值结果与预期范围相差较大. LeNet 模型的评测结果超出预期范围的主要原因为模型的规模较小且卷积层和全连接层的占比相同，导致该模型在 Mindspore-cpu 框架下仍有较好的表现. VGGNet 模型的评测结果超出预期范围的主要原因是全连接层较多，产生了大量矩阵向量乘积操作，然而 Mindspore-ascend310 框架对向量乘积有更好的加速效果，因此在该框架下运算速度更快，导致比值大于预期值. 这反映了论文 3.3 节分析的问题，即基准测试集中的神经网络模型的计算强度存在一定的随机性，可能无法充分挖掘待测设备的计算潜力，导致评测结果的参考价值打折扣.

在采用神经进化算法自动生成用于计算能力评测的神经网络模型时，Mindspore-ascend310 框架评分是 Mindspore-cpu 框架评分的 42.37 倍，符合理论预期. 由于算法收敛和模型转换需要大量时间，本文方法与单个网络模型相比，测试时长的差异较大. 但在实际生产环境中，由于基准测试集包含多种网络模型（iBench 测试集[21]包含 10 种网络模型，BenchIP 测试集[22]包含 11 种网络模型），因此总测试时长为 1 小时左右，且尚未考虑模型裁剪和设备适配所需的时间. 因此，相较于产品长达数周的评测周期而言，2 小时 50 分钟的测试时长是可以接受的. 此外，为尽量降低设备间歇运行引入的误差，本文在利用算法生成神经网络模型后，多次运行神经网络模型，计算能力评测公式中计算因子 S_1、S_2、S_3 和 S_4 的取值均为多次运行的平均值，以可接受的时间换取评测结果的精度.

综上，通过上述比较发现，本文方法更接近真值，更客观. 不同神经网络模型之间的对比实验结果表明本文介绍的智能计算能力评测方法有如下创新优势：

（1）无需考虑模型的计算强度是否与被测嵌入式智能计算机的计算强度相匹配. LeNet 和 VGGNet 模型因其计算强度和被测设备不匹配，导致测试的结果偏离预期范围，而本文的评测方法可以使生成

模型的计算强度尽可能的接近于被测设备.

（2）无需考虑模型在被测嵌入式智能计算机上是否能正常. 由于 AlexNet 模型太大，无法直接放在 Atlas200 设备上进行测试，需要进行一定的压缩处理. 在使用固定测试集的测试过程中，因为被测设备的内存资源是不确定的，可能会出现测试项过大导致模型测试异常终止，而本文的评测方法使用的模型是自动生成的，在评测过程中会自动舍弃无法生成可执行模型的基因序列，能有效地适用于各种嵌入式智能计算机的评测任务.

不同计算框架下本文方法的对比实验结果如表 7 所示. 表 7 中第 1 列为 3 种计算框架，第 2 列为 2 类计算核心，第 3 列至第 6 列为 S_1 至 S_4 的取值，第 7 列为采用公式(9)计算的评分结果，Tensorflow-cpu 和 Mindspore-cpu 框架都用 Atlas200 的 Cortex-A55 核心进行计算，且两个框架使用了同一套内存. 根据 Roofline 理论模型，预计两者理论得分相差不大. 本文的方法测试中 Tensorflow-cpu 框架的评分为 18.2×10^{-4}，Mindspore-cpu 框架的评分为 17.8×10^{-4}，两者相差不大，符合实验预期. 实际测试中 Mindspore-ascend310 框架得分是 Mindspore-cpu 框架得分的 42.37 倍，处于理论预期范围 30~100 之间. 该实验结果表明本文提出的基于神经进化算法的模型生成方法在不同的计算框架下均表现出稳定的评测效果.

表 7　不同计算框架下本文方法的对比实验结果

计算框架	计算核心	S_1	S_2	S_3	S_4	计算能力评分/10^{-4}(公式(9)计算)
Mindspore-cpu	Cortex-A55 Arm Core	60	400	400	8	17.8
Tensorflow-cpu	Cortex-A55 Arm Core	60	400	400	15	18.2
Mindspore-ascend310	DaVinci AI Core	600	94	600	275	754.2

6　总　结

为了解决了嵌入式智能设备配置各异，难以通过固定模型进行性能测试的问题，本文基于神经进化算法，生成可以表征嵌入式智能设备智能计算能力的神经网络模型，从模型的复杂度逆推嵌入式智能计算机的智能计算能力. 为了保证生成模型的有效性，本文选取了深度学习中具有代表性的网络层作为基因节点进行编码. 同时，本文构建节点模型来描述不同种类网络层中所包含的各种参数，以保证基因序列能正确完整的映射到网络拓扑结构上. 在节点模型基础上，本文基于神经进化算法框架，在限制模型推断任务执行速度的情况下，提出了一种以生成最大复杂度模型为目标的神经网络模型生成算法. 随后本文给出了嵌入式智能计算机的智能计算能力得分计算评测公式，并将计算结果作为嵌入式智能计算机智能计算能力的指标. 未来继续探索一下工作：

（1）持续将新兴网络结构纳入到神经网络模型生成算法的搜索空间中. 随着深度学习技术的发展，新兴的网络结构被不断地提出. 本文采用链式架构空间作为搜索空间，目前无法充分覆盖到所有可能的神经网络结构. 因此需要伴随技术的发展将新兴的网络结构纳入到算法的搜索空间中，以保证智能计算能力评测的充分性和可靠性.

（2）评估模型适配更多嵌入式智能计算机. 本文在框架的适配方面的实验验证仅考虑到了 Tensorflow 和 Mindspore 框架，后续可以将更多的智能计算框架添加到智能计算能力的评估模块中.

致　谢　感谢教育部-华为“智能基座”产教融合协同育人基地赠送的嵌入式智能设备 Atlas200.

参　考　文　献

[1] Du ZD, Fasthuber R, Chen TS, et al. ShiDianNao: shifting vision processing closer to the sensor. Computer Architecture News, 2015, 43(3): 92-104

[2] Chen YJ, Lan HY, Du ZD, et al. An instruction set architecture for machine learning. ACM transactions on Computer Systems, 2018, 36(3): 1-35

[3] Wu XX, Ou Y, Li WM, et al. Acceleration of sparse convolutional neural network based on coarse-grained dataflow architecture. Journal of Computer Research and Development, 2021, 58(7): 1504-1517 (in Chinese)
(吴欣欣, 欧焱, 李文明等. 基于粗粒度数据流架构的稀疏卷积神经网络加速. 计算机研究与发展, 2021, 58(7): 1504-1517)

[4] Price WJ. A benchmark tutorial. IEEE Micro, 1989, 9(5): 28-43

[5] Luther K. Embedded tutorial: IC test cost benchmarking// Proceedings of the 12th IEEE European Test Symposium. Freiburg, Germany, 2007: 200-200

[6] Tang F, Gao W, Zhan JF, et al. AIBench training: Balanced industry-standard AI training benchmarking//Proceedings of the 2021 IEEE International Symposium on Performance Analysis of Systems and Software. Stony Brook, USA, 2021: 24-35

[7] Song BB, Zhang H, Wu ZF, et al. Automated tensor decomposition to accelerate convolutional neural networks. Journal of

Software, 2021, 32(11): 3468-3481 (in Chinese)
(宋冰冰, 张浩, 吴子峰等. 自动化张量分解加速卷积神经网络. 软件学报, 2021, 32(11): 3468-3481)

[8] Wang HL, Guo Y, Qu WX. Deep learning hardware acceleration based on general vector DSP. Science in China (Information Sciences), 2019, 49(3): 256-276 (in Chinese)
(王慧丽, 郭阳, 屈晓霞. 基于通用向量 DSP 的深度学习硬件加速技术. 中国科学(信息科学), 2019, 49(3): 256-276)

[9] Wang F, Zhang XL, Pei WH, et al. Resistant light noise neural microelectrode based on CMOS process. Semiconductor Optoelectronics, 2018, 39(5): 671-674 (in Chinese)
(王飞, 张雪莲, 裴为华等. 基于 CMOS 工艺的抗光噪声神经微电极. 半导体光电, 2018, 39(5): 671-674)

[10] Sze V, Chen YH, Emer J, et al, Zhang ZD. Hardware for machine learning: challenges and opportunities//Proceedings of the 2017 IEEE Custom Integrated Circuits Conference. Austin, USA, 2017: 1-8

[11] Liu ZC, Zhu YX, Wang H, et al. Parallel acceleration design of convolutional neural network based on FPGA. Microelectronics & Computer, 2018, 35(10): 80-84 (in Chinese)
(刘志成, 祝永新, 汪辉等. 基于 FPGA 的卷积神经网络并行加速结构设计. 微电子学与计算机, 2018, 35(10): 80-84)

[12] Williams S, Waterman A, Patterson D. Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM, 2009, 52(4): 65-76

[13] Marques D, Ilic A, Sousa L. Mansard roofline model: reinforcing the accuracy of the roofs. ACM Transactions on Modeling and Performance Evaluation of Computing Systems, 2021, 6(2): 1-23

[14] Ould-Ahmed-Vall EM, Woodlee J, Doshi KA, et al. Characterization of SPEC CPU2006 and SPEC OMP2001: regression models and their transferability//Proceedings of the 2008 IEEE International Symposium on Performance Analysis of Systems and software. Austin, USA, 2008: 179-190

[15] Guthaus MR, Ringenberg JS, Ernst D, et al. MiBench: a free, commercially representative embedded benchmark suite//Proceedings of the 4th Annual IEEE International Workshop on Workload Characterization. Austin, USA, 2001: 3-14

[16] Poovey JA, Conte TM, Levy M, et al. A benchmark characterization of the EEMBC benchmark suite. IEEE Micro, 2009, 29(5): 18-29

[17] Iqbal SMZ, Liang YC, Grahn H. Parmibench-an open-source benchmark for embedded multiprocessor systems. IEEE Computer Architecture Letters, 2010, 9(2): 45-48

[18] Ignatov A, Timofte R, Chou W, et al. AI benchmark: Running deep neural networks on android smartphones//Proceedings of the European Conference on Computer Vision Workshops. Munich, Germany, 2018: 288-314

[19] Coleman C, Kang D, Narayanan D, et al. Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark. Operating Systems Review, 2019, 53(1): 14-25

[20] Mattson P, Tang HL, Wei GY, et al. MLPerf: an industry standard benchmark suite for machine learning performance. IEEE Micro, 2020, 40(2): 8-16

[21] Brewer W, Behm G, Scheinine A, et al. iBench: a distributed inference simulation and benchmark Suite//Proceedings of the 2020 IEEE High Performance Extreme Computing Conference. Waltham, USA, 2020: 1-6

[22] Tao JH, Du ZD, Qi G, et al. BENCHIP: benchmarking intelligence processors. Journal of Computer Science & Technology, 2018, 33(1): 1-23

[23] Xu QQ. Design and implementation of benchmarks for deep learning processors [M.S. Thesis]. University of Science and Technology of China, HeFei, 2019 (in Chinese)
(徐青青. 深度学习处理器基准测试程序的设计与实现[硕士学位论文]. 中国科学技术大学, 合肥, 2019)

[24] Ren ZX, Liu YH, Shi TH, et al. AIPerf: automated machine learning as an AI-HPC benchmark. Big Data Mining and Analytics, 2021, 4(3): 208-220

[25] He X, Zhao KY, Chu XW. AutoML: a survey of the state-of-the-art. Knowledge-Based Systems, 2021, 212(5): 1-27

[26] Leinhauser M, Widera R, Bastrakov S, et al. Metrics and design of an instruction roofline model for AMD GPUs. ACM Transactions on Parallel Computing, 2022, 9(1): 1-14

[27] Liu W, Fu J, Zhou DY, et al. Research on shallow neural network evolution method based on improved coyote optimization algorithm. Chinese Journal of Computers, 2021, 44(6): 1200-1213 (in Chinese)
(刘威, 付杰, 周定宇等. 基于改进郊狼优化算法的浅层神经进化方法研究. 计算机学报, 2021, 44(6): 1200-1213)

MA Chun-Yan, Ph. D., associate professor. Her research interests include embedded software system verification, software automated testing and fault localization.

CHEN Jing, M.S. candidate. His research interests include embedded system testing and fault localization.

YAO Ding, M.S. His research interests include embedded system testing and intelligent software development.

ZHANG Tao, Ph. D., professor. His research interests include software engineering, fault localization and intelligent software development.

Background

The evaluation of intelligent computing capability is an important means to test the execution speed and performance of intelligent computers when performing inference tasks. The rationality of the evaluation results directly affects the optimization and improvement direction of embedded intelligent computers.

Compared with general-purpose computers, embedded intelligent computers have a variety of neuromorphic computing optimization schemes and machine learning accelerators, which lead to new challenges in the process of embedded computing capability evaluation with the benchmark testing methods used in the past. First, it is difficult to give unified performance indicators and quantitative parameters to quantify the embedded intelligent computing capabilities. Secondly, a set of benchmark test programs and evaluation indicators are difficult to adapt to the embedded intelligent system with diverse configurations, and its reuse ability is limited. Furthermore, the benchmark test set has not been analyzed or proved to ensure that the computational intensity of the neural network model is as close as possible to the computational intensity of the embedded intelligent computer under test.

This paper proposes an embedded intelligent computer computing capability evaluation method based on a neural evolution algorithm. Firstly, based on the Roofline model, a computing capability evaluation framework for various embedded intelligent computers is proposed, which integrates computing potential mining, resource adaptation, and evaluation index unification, and its rationality is analyzed. Secondly, a neural network model generation algorithm is proposed to evaluate the computing capability. The neural evolution algorithm is used to make the computing strength of the generated model as close as possible to the upper limit of that of the embedded intelligent computer, fully tap the computing potential of the computer to be tested, and make the evaluation more objective. Then, using the upper computer with a fixed environment as a comparison, the neural network model generated by the cross operation of the computer to be tested and the upper computer, and taking the number of floating-point operations per second when the inference task is executed twice as the calculation factor, the general formula for computing capacity evaluation is given, which can realize the comparative analysis of the computing capacity of different embedded intelligent computers. Finally, Huawei atlas 200 is evaluated under the framework of Mindspore-cpu, Tensorflow-cpu, and Mindspore-ascend310. Compared with the five neural network models commonly used in the benchmark test, the results using the neural network model generated in this paper are more reasonable, which confirms that the intelligent computing capability of the two DaVinci cores is 42.37 times that of the eight Cortex-A55 cores. This achievement can simplify the evaluation cycle of embedded intelligent computers and enable designers to perform rapid iterative optimization of the system.