能耗优化的神经网络轻量化方法研究进展

郭朝鹏 王馨昕 仲昭晋 宋杰
(东北大学软件学院 沈阳 110819)

摘 要 近年来，神经网络在语音识别、计算机视觉、自然语言处理等领域都取得了良好的进展. 大量的神经网络被部署于诸如手机、摄像头等依赖电池或太阳能供电的小型设备. 但神经网络参数量大计算复杂，需占用大量计算资源并消耗电能，从而限制了其在资源受限平台上的应用. 学术界和工业界逐渐关注于神经网络的高能耗问题. 神经网络轻量化方法可以有效地减少参数数量、降低参数精度或优化计算过程从而降低神经网络能耗. 本文从能耗优化的角度梳理了神经网络能耗估算方法和神经网络轻量化方法的基本思路，综述了近年来该领域主要研究成果，并提出了能耗估算和能耗优化的神经网络轻量化方法存在的挑战及进一步研究的方向. 其中神经网络能耗估算方法包括测量法、分析法和估算法. 能耗优化的神经网络轻量化方法包括剪枝、量化、张量分解和知识蒸馏. 对于进一步研究方向我们认为，首先需要建立可自适应网络类型的能耗模型；然后需要考虑平衡精度和能耗的轻量化方法. 其次需要实现硬件平台可泛化的轻量化方法；最后开发搜索空间可约束的轻量化方法.

关键词 神经网络；能耗估计；能耗优化；神经网络轻量化
中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2023.00085

Research Advance on Neural Network Lightweight for Energy Optimization

GUO Chao-Peng WANG Xin-Xin ZHONG Zhao-Jin SONG Jie
(*Software College, Northeastern University, Shenyang* 110819)

Abstract Recently neural networks have achieved the great progress in speech recognition, computer vision, natural language processing, and other fields. More and more neural networks are deployed in embedded devices such as mobile phones and cameras, which are relying on batteries or solar energy as their power supply. However, neural networks consume a large amount of storage resources and electric energy, limiting their application on resource-limited platforms. Therefore, both of the academia and industry pay attention to the high energy consumption problem of neural networks. Neural network lightweight method can effectively reduce the number and the precision of parameters and simplify the calculation to optimize energy consumption of neural networks. In this paper, we introduce the basic ideas of neural network energy consumption estimation and neural network lightweight methods for energy optimization, and we summarize the primary research achievements of the field in recent years. Moreover, the challenges and future research trends in both aspects are put forward. The neural network energy consumption estimation methods are categorized as measurement methods, analysis methods and estimation methods. The neural network lightweight methods for energy optimization include pruning, quantization, tensor decomposition and knowledge distillation. We summarize four research directions for the further research. First, an energy consumption model needs to be established to adapt to all neural network schemas. Secondly, the balance

收稿日期：2021-08-19；在线发布日期：2022-07-19. 本课题得到辽宁省博士启动基金（2020-BS-054）、中国国家自然科学基金（62162050）的资助. **郭朝鹏**，博士，讲师，中国计算机学会（CCF）会员，主要研究领域为绿色计算、人工智能，E-mail: guochaopeng@swc.neu.edu.cn. **王馨昕**，硕士研究生，主要研究方向是绿色计算. **仲昭晋**，硕士研究生，主要研究方向是高效能计算. **宋 杰**(通信作者)，博士，教授，中国计算机学会（CCF）高级会员，主要研究领域为高效能计算、大数据处理，E-mail: songjie@mail.neu.edu.cn.

between the accuracy and the energy consumption needs to be considered in the neural network lightweight methods. Thirdly, lightweight methods that can be generalized for hardware platforms need to be implemented. Finally, lightweight methods that are able to limit searching space need to be developed.

Keywords neural network; energy consumption estimation; energy consumption optimization; neural network lightweight

1 引 言

作为人工智能实现的范式之一，神经网络采用广泛互联的结构与有效的学习机制来模拟人脑智能信息处理的过程，是人工智能发展历程中的重要方法，也是类脑智能研究中的有效工具[1]. 近年来，人工智能移动化的趋势逐渐显现[2]. 当前，越来越多的神经网络被部署于依靠电池或太阳能供电的小型移动设备中，如智能手机、智能摄像头等. 神经网络的移动端部署助力了许多智能应用的发展，常见的包括语音助手、在线翻译、人脸识别等. 值得注意的是，这些智能移动设备的出货量目前呈几何式增长. 以智能手机为例，到 2022 年，支持 AI 功能的智能手机出货量占比将从 2017 年的不到 10%提升到 80%，年销量将超 13 亿部. AI 手机将是未来行业的产品方向[3]. 到 2023 年，专用的人工智能芯片很可能已经成为智能手机的标准配置[4].

这些移动端设备通常外形尺寸较小，依靠电池或绿色能源供电[5]，因而其可使用的电能有限. 神经网络的执行涉及大量计算，对电能消耗要求较高. 例如，具有 50 个卷积层的 ResNet-50 在推理阶段处理图像时需要占用超过 95MB 的内存，执行超过 38 亿次浮点乘法[6]；图像分类的基础网络 AlexNet[7]在手机端运行不到一个小时就耗光了手机全部电能[8]. 移动端神经网络能耗受限问题日益突出.

能耗受限问题极大地影响了移动端设备的运行时长，缩短了设备的服务时限，阻碍了移动端设备的智能化发展[9]. 为了应对能耗受限问题，学术界和工业界涌现了大量相关研究和解决方案. 一种解决方案是在云边结合的体系结构下将网络部署于云端，该方法较为直接地缓解了能耗受限问题，但云边端通信带来了额外的通信成本，实时性、安全性较差[10]；另一种解决方案是神经网络能耗优化的神经网络轻量化方法. 这种解决方案延迟低、安全性高且具有隐私优势[2].

能耗优化的神经网络轻量化方法是降低神经网络在目标设备上能耗需求的有效方法，该方法关注网络自身的结构特征，压缩成本低、效率高且易于部署. 本文重点关注能耗优化的神经网络轻量化方法，从神经网络能耗估算方法和网络轻量化方法两个方面出发，概述优化思路，叙述具体方法，总结当前能耗优化的神经网络轻量化方法的研究进展并提出进一步研究问题与挑战. 在作者文献查找范围内，本文是绿色计算和高性能计算领域中第一篇针对能耗优化的神经网络轻量化方法的中文综述，系统地总结了当前的研究进展.

本文第 2 节将介绍研究问题和思路；第 3 节介绍能耗估算方法具体细节；第 4 节介绍能耗优化的神经网络轻量化方法的具体细节；第 5 节提出目前能耗估算方法和网络轻量化方法存在的问题与挑战；最后在第 6 节总结全文.

2 研究问题

在能耗优化的相关研究中，往往重点关注两个层面，能耗估算方法和能耗优化方法. 能耗估算方法给出的度量用于评价能耗优化方法的结果或评价能耗优化过程的中间结果，为能耗优化提供依据. 能耗优化方法则给出降低系统能耗或提高系统能效的具体措施. 本节，我们分析能耗估算和能耗优化的神经网络轻量化的具体思路.

2.1 运行时能耗估算

在神经网络能耗优化的相关研究中通常会采用神经网络精度和神经网络能耗作为评价优化效果的度量[11]. 其中神经网络精度可以通过任务相关的精度函数或任务相关的损失函数得到，而神经网络运行时能耗则需要特定的能耗估算方法得到.

本节我们关注能耗估算方法的设计思路. 在当前研究中，能耗估算方法可以大致归结为 3 类：

测量法. 属于硬件层的能耗估算方法，使用功率计等测量设备直接测量能耗.

分析法. 属于系统层面的估算方法，通过分析产生能耗的直接因素，如网络计算次数和数据存取次数等，计算神经网络运行时的能耗.

估计法. 属于应用层的估算方法，通过分析神经网络结构和产生能耗相关的特征，利用机器学习、深度学习等方法，预测网络运行时的能耗.

本文第 3 节将展开叙述上述 3 种能耗估算方法.

2.2　能耗优化的神经网络轻量化

目前神经网络的训练和部署流程如图 1 所示. 神经网络的训练阶段包含神经网络参数训练和网络结构的轻量化. 在训练结束后，生成轻量化的网络模型，该模型被部署于各类移动式或嵌入式设备中，高效且低能耗地执行神经网络预测任务.

根据神经网络的预测过程，神经网络能耗的产生通常来自于两个方面：(1)将参数加载至处理单元(Processing Engine，PE)内的算术逻辑单元(Arithmetic and Logic Unit，ALU)带来的数据移动能耗；(2)参数在 ALU 内计算时产生的乘加操作(Multiply and Accumulate，MAC)能耗. 表 1 将 1 次 MAC 操作能耗标准化为 1，总结了相对于 1 次 MAC 操作，在不同内存层级间移动数据的标准化能耗[12]. 目前神经网络存在过参数化的现象[11]，过度冗余的参数导致了更为频繁的参数存取和计算，使网络在运行时产生了大量的数据存取和计算能耗.

通常意义上讲神经网络轻量化指使网络结构或参数简单化的方法. 所以轻量化方法包含：人工设计的轻量神经网络；神经网络结构搜索技术，即利用计算机搜索的方式自动发现更小结构的方法；神经网络压缩技术，即压缩网络结构或者降低网络精度的一类方法. 然而从优化目标来看，前两者主要用于优化神经网络性能，而神经网络压缩技术既可以用于优化神经网络性能也可以用于优化神经网络能耗. 本文重点关注面向能耗优化的神经网络轻量化技术，所以我们侧重关注于神经网络压缩这类轻量化技术.

目前能耗优化的神经网络轻量化技术从参数存取和计算两个角度入手，产生了一些以优化网络结构、降低参数量为目标的网络结构轻量化方法，以及以简化计算过程为目标的计算轻量化方法.

网络结构轻量化方法主要包括结构压缩和知识迁移. 其中，以剪枝和量化为代表的结构压缩方法的优化对象为网络本身，即通过修剪无用的网络子结构或降低网络参数精度，降低网络的参数量和计算复杂度，从而优化网络的参数存取和计算能耗；以知识蒸馏为代表的迁移方法不直接修改网络结构，而是使用该网络指导训练更简单的学生网络. 学生网络不仅学到了原网络包含的信息，且参数量更小、模型更精炼、运行能耗更低. 同时知识蒸馏也可以被视为一种学习模式上的创新方法，从教师网络出发训练出更简单的学生网络.

计算轻量化方法则考虑到网络中部分网络结构维度高、参数量大、N 计算复杂，因此采用张量分解方法将冗余的高维张量近似为几个低维张量的乘积，从而简化计算过程、加速网络推理、降低推理能耗.

值得注意的是，神经网络能耗优化方法不仅包含能耗优化的神经网络轻量化方法，还有诸如运行时平台优化、硬件加速器等其他方法. 其中运行时平台通过合理的任务调度降低能耗，硬件加速器则设计高性能低功耗的计算单元降低能耗. 相对于其他方法，神经网络轻量化属于软件层面的优化方法.

本文我们重点总结能耗优化的神经网络轻量化方法. 我们将在第 4 节具体阐述面向能耗优化的神经网络轻量化方法.

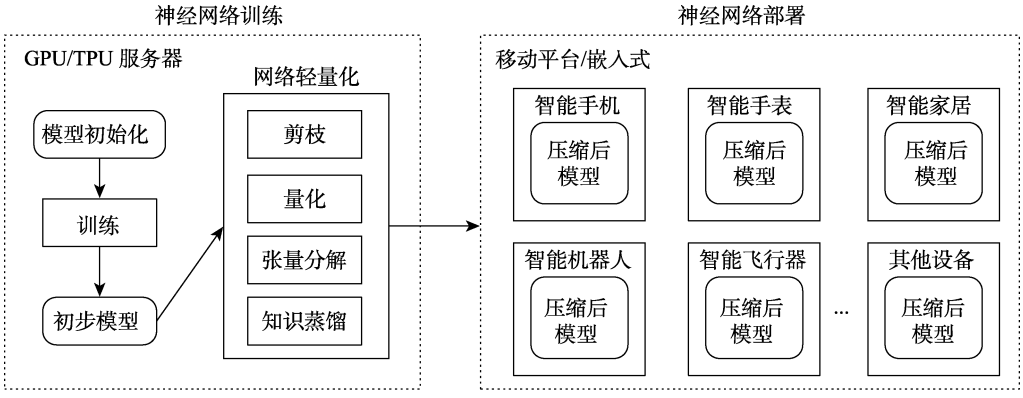

图 1　神经网络轻量化及部署流程图

表 1 数据移动及计算标准化能耗

操作	标准化能耗
1 次 MAC 操作	1
PE 内寄存器文件→PE 内 ALU	1
其他 PE→目标 PE 内 ALU	2
全局缓冲区→ALU	6
内存→ALU	200

3 神经网络运行能耗估算

本节，我们从测量法、分析法、估计法 3 个方面，具体介绍能耗估算方法的相关研究及成果.

3.1 测量法

测量法的一般步骤是：（1）使用功率计测量运行设备的待机功率 P_{standby}；（2）使用功率计测量 T 时间段内 n 次网络运行的实时功率 $P(t)$；（3）由式(1)计算 1 次神经网络运行能耗.

$$E=\frac{\int_0^T P(t)\mathrm{d}_t - P_{\text{standby}}\times T}{n} \tag{1}$$

测量法存在诸多不足：（1）测量法粒度较大，只能测量神经网络整体的能耗，无法测量各个隐层的能耗. 因而无法为能耗优化方法提供细粒度的优化指导，如定位到能耗较高的隐层等，不利于后续的分析和进一步优化；（2）测量法通常需要测量多次才能获得准确的能耗估计，但现存的一些能耗优化方法是迭代的，依赖实时的能耗估算结果，因而测量法的使用存在性能瓶颈；（3）测量法需投入测量设备，无法广泛用于大规模能耗估算，如分布式环境下的能耗估算. 综上所述，测量法不利于高效地评价和优化神经网络运行时能耗.

测量法的优势是方法简单、能耗估值准确且不与特定的神经网络结构以及运行设备相关. 在当前研究中通常用于收集原始的粗粒度能耗数据.

3.2 分析法

在分析法中，通常考虑神经网络运行中产生能耗的直接因素，即计算能耗和数据访问能耗. 两者的定义如下：

计算能耗：由计算产生的能耗. 在分析法中通常根据不同层的结构参数，如神经元个数、权值个数等，统计每秒执行的浮点运算次数(FLOPs, FLoating point Operations)或乘积累加运算数(MAC，Multiply and Accumulate)等硬件参数进行估算.

数据访问能耗：由数据读写产生的能耗. 在分析法中通过统计读取数据的总比特数，结合各存储单元读取单位比特数据的能耗值进行估算.

分析法的一般过程如图 2 所示. 其步骤为（1）以层为单位估算计算能耗和数据访问能耗；（2）汇总所有层的能耗.

Han 等人[11]考虑了神经网络的计算能耗，同时引入通信能耗、反向传播能耗等，提出了一种自适应能耗限制的 DNN 训练框架. 能耗表达式如式(2)所示. 其中：b_s 为推理的数据批量大小，S 为隐藏层数量，n^s 为第 n 个隐层神经元数量，α_{flop} 为 1 次 MAC 操作产生的能耗；α_{act} 为 1 次非线性激活操作产生的能耗. 通过结合 MAC 次数、非线性激活操作次数和相应操作的单位能耗可以算得总推理能耗.

$$E=b_s\left(\alpha_{\text{flop}}\sum_{s=1}^{S-1}n^s n^{s-1}+\alpha_{\text{act}}\sum_{s=1}^{S}n^s\right) \tag{2}$$

Qi 等人[13]同时考虑了计算能耗和数据访问能耗，提出了性能分析框架 Paleo. 不同的是，该文重点分析神经网络运行时间，结合硬件额定功率估算能耗. 作者将神经网络执行时间拆解到每个层分别计算. 每个层的执行时间包括：读取输入时间、计算时间、写入输出时间 3 个部分. 计算时间表示为 FLOPs/S_o，S_o 为硬件处理速度；读写内存的时间表示为 D_b/S_r，D_b 为数据的比特数，S_r 为硬件读取速度. 除此之外，该文考虑到硬件例如 CPU、GPU 不可能以峰值功率恒定运行，提出了峰值百分比 PPP（Platform Percent of Peak），并拟合一个常数估计 PPP. 作者估算了 AlexNet[7]、VGG-16[14]以及 GAN[15]中判别器和生成器的运行时间，估算时间与实际时间的误差分别为 4.54%、8.74%、8.74%、6.28%.

Yang 等人[16] 考虑了计算能耗和数据访问能耗. 同时该文将剪枝后数据的稀疏性、量化后数据的位宽考虑在内. 作者认为稀疏性由 ReLU 等非线性激活函数和剪枝操作产生. 且当乘数因子有 0 值时，该乘法运算不计入能耗. 对于数据位宽，由于内存的层次结构固定、不同层次的容量大小固定，如果数据位宽发生改变，则相应的数据存放、读取策略也会发生变化. 综合以上因素，作者将能耗单位定义为 MAC 次数. 在 AlexNet、GoogLeNet[17]上的实验表明，估算 MAC 次数与实际 MAC 次数的误差分别为 2.5%和 2.6%.

分析法的优势在于其适应性较强，当硬件平台发生改变时，提供当前硬件条件下的单位计算能耗和数据访问能耗即可计算网络运行能耗. 分析法的劣势在于需要了解硬件设备的自身特性，即获取其各个操作的单位能耗，准备工作复杂；另外，分析

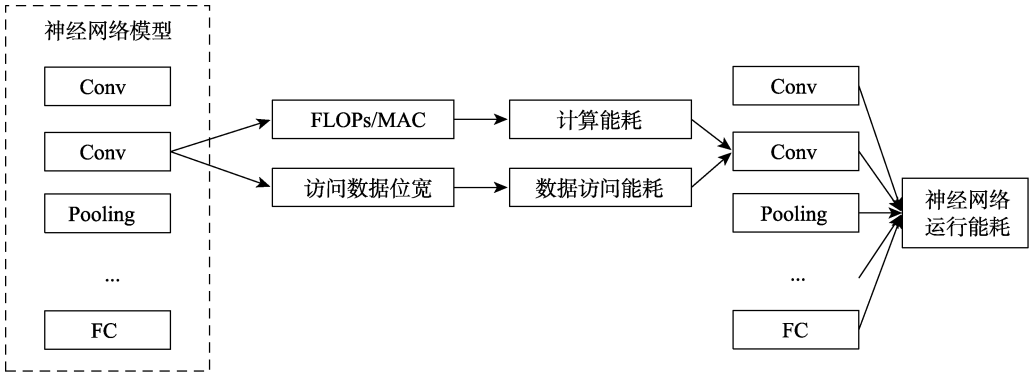

图 2　分析法的一般过程

法的特征由人工抽取并拟合，相对于测量法准确性不高.

3.3　估计法

近年来，很多研究采用机器学习方法，如线性回归、多项式回归、决策树、神经网络等，训练机器学习模型预测能耗. 与分析法相比，估计法不再分析 FLOPs、数据比特数等参数数据，而是在特定的硬件平台上考虑神经网络不同层，如卷积层、池化层、全连接层等的结构参数对能耗的影响.

现有估计法研究的一般目标为卷积神经网络. 将卷积神经网络分解成卷积层、池化层、全连接层后分别提取特征，再训练预测模型. 估计法一般包括以下步骤：（1）抽取层的初始结构参数，如输入通道数、卷积核大小等；（2）根据初始结构参数进一步提取特征；（3）训练能耗预测模型；（4）计算所有层的能耗，并汇总结果得到总的推理能耗.

Cai 等人[18]较早使用估计法的研究，提出了基于功率、能耗、运行时间的预测模型 NeuralPower. NeuralPower 建立了以卷积层、池化层、全连接层为基础的多项式回归模型，模型能够预测每层的推理时间和相应的功率，以计算该层运行能耗. 以推理时间为例，单层网络的回归模型如式(3)所示. 其中 $\boldsymbol{X}_T$ 表示输入的 D_T 维向量，x_i 为 $\boldsymbol{X}_T$ 的第 i 个分量；q_{ij} 为 x_i 在多项式中的系数；c_j、c'_s 为模型系数；$\mathcal{F}$ 表示推理过程中的操作度量，如数据访问次数、浮点运算次数等. 不同层的回归式输入特征略有不同. 该文使用 nvidia-smi 收集功耗数据，分别训练功耗和推理时间预测模型. 并将神经网络的总推理能耗表示为 $\sum_{i=1}^{N} P_n T_n$，其中 N 表示网络总层数. 在 AlexNet 和 VGG-16 等网络上的实验结果表明，平均能耗估算误差约为 2.79%，较分析法[13]在运行时间预测精度上提高了 68.5%.

$$\begin{aligned} &T(\boldsymbol{X}_T)=\sum_j c_j\cdot\prod_{i=1}^{D_r}x_i^{q_{ij}}+\sum_s c'_s\mathcal{F}_s(\boldsymbol{X}_T)\\ &\boldsymbol{X}_T\in\mathbb{R}^{D_r};q_{ij}\in\mathbb{N};\forall j,\sum_{i=1}^{D_r}q_{ij}\leqslant K_T \end{aligned} \tag{3}$$

Rodrigues 等人[19]提出了在嵌入式平台上预测深度神经网络能耗的框架 SyNERGY. 与 NeuralPower[18]不同的是，SyNERGY 没有将不同层的结构特征作为模型输入，而是根据总线访问数量、SIMD 指令数量建立多元回归预测模型. 由于在嵌入式平台分别统计总线访问数量、SIMD 指令数量过于繁琐，作者发现 MAC 数量与 SIMD 指令数量有线性关系，SIMD 数指令量又与总线访问数量有线性关系. 于是建立了两个线性回归模型，通过 MAC 数量预测 SIMD 指令数量，通过 SIMD 指令数量预测总线访问数量. 最后只需要根据各个层的参数计算 MAC 数量即可. 最终实验表明，SyNERGY 的能耗预测误差为 7.08%±5.05%.

Rodrigues 等人[20]权衡预测模型准确性和复杂性为目标，对 SyNERGY[19]做了进一步改进. 同时将使用 MAC 数等计数特征的模型，和使用卷积核大小、步长等结构特征的模型进行了对比. 以计数特征模型为例，作者探究了如下 2 类特征提取方式：（1）以网络整体 MAC 总数 $\mathrm{MAC_{sum}}$ 作为特征，构建回归模型；（2）以每层 MAC 数作为特征构建模型. 在特征选择方面，作者使用贝叶斯信息准则[21]选择最佳特征子集. 在实验中，作者复现了 Neural Power[18]，发现该方法方差较大，很难得到原文所提到的 97.21%准确率. 同时实验表明使用 $\mathrm{MAC_{sum}}$ 预测能耗可以达到最优的平均准确率和最小方差 81.84%±7.8%，而经过特征选择的多项式回归模型得到的准确率 79.58%±13.03%也优于 NeuralPower 的 77.48%±21.21%. 作者将最优的基于 $\mathrm{MAC_{sum}}$ 的模型在不同的软硬件平台上实验，都得到了较好的预测结果.

Dai 等人[22]针对 MobileNet[23]，通过高斯过程回归估计不同超参下的网络能耗. 发现在小数据集上（训练数据仅有 240 条），高斯过程回归的预测结果优于线性回归、多层感知机、决策树、贝叶斯回归、决策树的预测结果. 同时，高斯过程回归输出了预测结果的取值区间，提供了额外的信息. 最后在 MobileNet 上进行实验，预测能耗与实际能耗差距较小.

Bouzidi 等人[24]没有提出新的特征提取方法或者能耗预测模型，而是总结了 5 种机器学习方法——多元线性回归、多层感知机、支持向量机、随机森林、XGBoost[25]的预测结果. 测试了在应对新的输入图像尺寸、新的 CNN 结构参数以及新的 CNN 结构时上述 5 种模型的预测效果. 最终，综合考虑预测模型的训练时间、超参数搜索时间、预测时间等因素，XGBoost 的表现优于其他 4 种模型.

与以上研究采用的传统机器学习方法不同，Justus 等人[26]构建了全连接神经网络预测推理时间. 他们认为深度学习的方法可以拟合更加复杂的函数、适应复杂的场景，比传统机器学习方法更具有优势. 该文将神经网络分为卷积神经网络、循环神经网络两类，针对全连接层、卷积层、池化层和循环神经网络网络的特殊结构分别提取层结构特征、硬件参数特征，将神经网络拆解为不同的层分别预测推理时间.

估计法可以被视为通过学习能耗相关特征预测能耗的方法. 相较于分析法人工抽取特征的方式，估计法准确性更高，相较于测量法则更为简单，无需测量设备投入. 然而其劣势在于估计法往往与网络结构相关，无法适用于所有类型的网络结构；同时估计法一般需要收集某硬件的网络能耗数据，以构建机器学习模型，数据收集和模型构建过程较复杂.

3.4 方法比较

本节总结了测量法、分析法、估计法三种能耗估算方法. 尽管测量法测量范围有限、测量粒度较粗，已不适用于现在的部分能耗优化研究，但这种最直接的测量方法的估算精度还是超过了分析法、估计法的估算结果. 值得注意的是，对于同一硬件，如果多次测量时硬件的温度等物理因素不同，也可能会造成一定的测量误差，而这种测量误差往往是不可控的、难以分析的.

分析法和估计法则更契合现今神经网络能耗优化研究. 分析法的优势是兼容不同的硬件平台，同时其结果更具备可解释性. 模型迁移时只需修改硬件参数即可. 但分析法也存在劣势如下：（1）由于神经网络中非线性计算的存在，以及计算资源利用不充分等，网络计算操作数量与 FLOPs 不成线性比例[26]. （2）在计算能耗时，FLOPs 也不一定与能耗成线性关系. Bouzidi 等人[24]通过实验佐证了这一观点，该文发现相同的 FLOPs 数量下，神经网络运行时间差距高达 3 倍. Rouhani 等人[27]也通过实验发现，在某些情况下，增大卷积层输入尺寸大小反而会减少推理时间，这是由于在 cuDNN 中所用卷积算法不同导致的. 因此，往往需要深入了解不同网络的结构特征和计算过程，才能解决上述问题、得到更优的结果.

不同于分析法在构建模型时需要具备大量的专业知识，估计法只需找到与神经网络能耗相关的网络结构特征、硬件特征，即可使用机器学习方法学习特征与能耗的关系. 与分析法相比，估计法更加易用，但也存在如下缺点：（1）估计法模型往往不具备很好的泛化能力，为了跨硬件平台使用估计法，需要在足够多不同硬件参数的硬件平台上生成数据集进行训练；（2）估计法需要针对不同种类的神经网络层分别训练模型. 若出现了超出训练范围的神经网络结构时，估计法无能为力；（3）为了适配更多的神经网络层，研究人员需要花更多的精力设计模型、调整超参数、构造数据集等.

最后，我们从估算方法、适用网络、平均估算误差、设备平台 4 个角度总结上文提到的分析法、估计法的相关研究，如表 2 所示. 由于每个研究使用的数据集、数据收集方法、硬件平台都不尽相同，因此估算精度不能作为评价估算方法好坏的唯一评价指标. 例如在上文提到，Rodrigues 等人[20]在自己的数据集和硬件平台上复现 NeuralPower[18]得到的预测精度不及原文的实验结果. 表中 T 表示推理时间预测误差，MAC 表示 MAC 预测误差，E 表示能耗估计误差.

表 2 神经网络能耗估算方法

文献	估算方法	适用网络	估计误差	硬件平台
[28]	分析法	FCNN	–	移动型 GPU
[13]	分析法	CNN	$7.08\%^{T}$	桌面型 GPU
[16]	分析法	CNN	$2.55\%^{MAC}$	–
[18]	估计法	CNN	$2.79\%^{E}$	桌面型 GPU
[19]	估计法	CNN	$8.04\%^{E}$	移动型 GPU
[22]	估计法	CNN	–	移动型 GPU
[20]	估计法	CNN	$21.18\%^{E}$	移动型 GPU
[24]	估计法	CNN	$10.66\%^{T}$	移动型 GPU
[26]	估计法	CNN、RNN	–	桌面型 GPU

4　能耗优化神经网络轻量化方法

针对神经网络的过参数化(Over-Parameterization)[11]导致的网络参数量大、计算复杂等问题，能耗优化的网络轻量化方法从结构轻量化和计算轻量化两个方向入手，精炼网络结构、简化计算过程. 常见的轻量化方法包括剪枝、量化、张量分解、知识蒸馏等. 神经网络轻量化方法的优势是显而易见的：(1)可以减小模型计算量，提高网络性能；(2)减少模型内存占用；(3)有利于降低模型能耗；(4)有利于模型的更新和发布. 本小节将依次介绍利用剪枝、量化、张量分解和知识蒸馏四种模型轻量化手段的能耗优化方法.

4.1　剪　枝

剪枝方法也称为稀疏化方法，按照一定的规则删除权重或神经元避免不必要的计算. 不同的剪枝算法在剪枝对象、剪枝粒度、剪枝标准以及剪枝后神经网络精度恢复方法等方面有所不同. (1)根据剪枝标准的不同，剪枝方法可以分为权重剪枝和神经元剪枝，前者减少了网络的连接数量，后者减少了网络的节点数量；(2)根据剪枝粒度的不同，剪枝方法还可以分为结构化剪枝和非结构化剪枝，前者以块为单位进行剪枝，后者以元素为单位进行剪枝. 例如，既可以对权重进行结构化的修剪，去掉整个卷积核或整个通道；也可以对其进行非结构化的修剪，去掉单个权重值. 可以对神经元进行结构化的修剪，去掉整个特征图；也可以进行非结构化的修剪，去掉单个激活值. 下面将介绍由不同剪枝对象和剪枝粒度组合所产生的一些剪枝方法.

Han 等人[29]提出了一种以权重为修剪对象的非结构化剪枝方法，根据预定义的阈值，修剪低于阈值的小权重连接. 此外，对于剪枝带来的网络精度损失的问题，作者重新训练网络微调(fine-tune)剩余连接的权重恢复网络精度. 文献[11]在文献[29]剪枝方法的基础上进一步使用了量化和霍夫曼编码. 实验对大型网络 AlexNet 和小型网络 LeNet-5 等均取得了很好的轻量化效果，分别将二者的参数量从 240MB 和 1720KB 压缩到了 6.9MB 和 44KB. 另外，修剪后的 AlexNet 和 VGG-16，在 CPU、GPU 和移动 GPU TK1 上的平均能耗分别降低了 7 倍、3.3 倍和 4.2 倍. 证明了剪枝方法可以大幅度压缩网络、优化网络能耗. Zhu 等人[30]为每层的权重配置了一个与权重维度大小相同的二值掩码变量表示是否剪裁某个权重. 二值权重在固定步数后更新，在达到最后的稀疏度目标时停止更新.

与前两项以权重值为标准的剪枝方法不同，Yang 等人[8]以能耗为标准指导剪枝过程. 作者首先提出了一种能耗估算方法，用以估计各隐层的能耗. 然后根据能耗估算结果修剪能耗最高的层. 修剪方法与前述工作类似，通过设定阈值，非结构化地修剪低于阈值的权重. 最后，在精度恢复的过程中，通过最小二乘法局部恢复每一层的权重，再通过继续训练的方式恢复全局权重. 将该剪枝方法应用到 AlexNet、GoogLeNet 和 SqueezeNet 上的实验表明，通过该方法将网络能耗分别降低了 3.7 倍、1.6 倍和 1.3 倍.

除了修剪权重，还有一些剪枝操作对神经元进行了非结构化的修剪. Hu 等人[31]对零激活值进行了修剪，并使用修剪前的权重做初始值重新训练网络. 在 LeNet 和 VGG-16 上的实验表明，该方法能够实现 2 到 3 倍的压缩率，并且不影响模型精度.

以上方法为非结构化的修剪方法，这些方法虽然有利于模型的轻量化，但是在网络参数中引入了大量的空值，在没有下层硬件优化的情况下无法减少神经网络计算量. 近年来，研究中出现了一些结构化剪枝的方法，对特定的层进行修剪，如减少卷积神经网络中的通道数(Channel-Wise Pruning)、减少 CNN 中卷积核数(Filter-Wise Pruning)等. 结构化的剪枝方法不需要结合特定的硬件即可实现推理加速，但由于其修剪粒度较大，修剪更粗糙，因此相对于非结构化剪枝，其对模型的精度影响也更大.

Singh 等人[32]提出以整个卷积核为剪枝单元的结构化剪枝方法. 该方法由自适应滤波器剪枝模块和剪枝率控制模块组成. 其中自适应滤波器剪枝模块的目标是实现滤波器数目最小化，剪枝率控制器模块则动态控制剪枝率，以实现网络精度最大化. 这种方法将 VGG-16 的参数数量减少了 17.5 倍，FLOPs 数量减少了 6.43 倍，并且没有产生的精度的损失. 王国栋等人[33]以权重的梯度绝对值大小衡量卷积核的重要性，修剪低重要性的卷积核. 该结构化剪枝算法将 VGG-16 的参数数量减少了 13.6 倍，精度损失为 0.45%.

Molchanov 等人[34]的剪枝对象是整个特征图，修剪过程中以泰勒展开为标准，评估去掉每个特征图后损失函数的变化，修剪掉影响最小的特征图. 在不断迭代修剪达到准确度和修剪目标(如内存利用率)之间的平衡后停止剪枝. 对 VGG-16 剪枝实验中，将特征图大小修剪到了原模型的 66%，GFLOPs 数缩小为原来的 37%，在不同的 GPU 上实现了最高 2.5 倍的加速. 但由于修剪粒度大，造成了 6.3%的

TOP-5 精度损失. 经过微调(fine-tune)后精度恢复到87%，仍存在 2.3%的精度损失. 相对于修剪整个特征图，徐晓等人[35]以特征图中的通道为剪枝目标，修剪粒度更小，最终将 MobileNet-v3 压缩了 3.6 倍，精度损失为 4.2%.

与以上人工设计算法剪枝的方法不同，部分研究使用深度学习方法指导剪枝过程. Hey 等人[36]提出使用强化学习来修剪卷积核，实现自动轻量化模型. 作者使用强化学习中的深度确定性策略梯度法产生每一层压缩比率，通过约束 FLOPs 和准确率设定奖励值. 实验表明，这种基于强化学习的方法将 MobileNet-v1 了压缩 1.4 倍，只造成 0.2%的精度损失. 虽然没有评估能耗，但由更小的模型参数量和更低的计算量可以推断出，该方法在降低网络能耗上有较大的潜力. 文献[37]借助了迁移学习的思路，加入了正则项惩罚低贡献特征，结构化地修剪特征图通道. 在 YOLOv3 上的实验表明，该方法能将参数量压缩 9.54 倍，平均精度均值(mAP，mean Average Precision)损失约为 0.063.

虽然神经网络剪枝的相关研究主要集中在卷积神经网络、循环神经网络模型，但最近也有学者将剪枝技术引入 Transformer 模型[38]和基于 Transformer 的 BERT 模型.

Gordon 等人[39]提出了一种针对 BERT 网络的非结构化剪枝方法，该研究认为在预训练阶段修剪 30%到 40%的权重并不会影响网络处理下游任务的性能. Sanh 等人[40]认为以权重大小为标准剪枝网络的效果有待提升，因此提出了一种以权重重要性为剪枝标准的非结构化剪枝方法. 该方法在句子相似性检测任务集中，将权值数量修剪到原模型的 10%，只造成了 1.5%的精度损失.

Yu 等人[41]提出一种以多头注意力层的注意力头为剪枝对象的结构化剪枝方法，根据 KL 散度(Kullback-Leibler 散度)，修剪重要性最低的注意力头，直到达到预设的剪枝比例. 在修剪 DeiT-B 模型的实验中，剪枝后的模型实现约 3 倍加速，只造成 0.25%的精度损失. Fan 等人[42]提出的 LayerDrop 方法同样对层进行了结构化的修剪，并在未进行微调的情况下保证了网络修剪后的性能. 证明了从大型网络中提取高效的小型子网络是可能的.

4.2 量 化

量化指将网络模型中的高精度参数映射到低精度参数，从减小参数位宽的角度轻量化模型. 量化的对象一般为权值或激活值，量化的位宽一般包括 2 位、8 位或 16 位等. 量化过程如式(4)所示. 其中 x_{pre} 为量化前的值，$x_{quantized}$ 为量化后的值. x_{scale} 为缩放因子，由参数量化前后的值域确定. X_{zero_point} 为量化前的 0 值对应的量化后的值. 量化方法降低了数据存储要求和计算的复杂度，有利于加速计算，降低神经网络运行能耗.

$$\begin{aligned} x_{pre} &= x_{scale} \times (x_{quantized} - x_{zero\_point}) \\ x_{scale} &= \frac{\max(x_{pre}) - \min(x_{pre})}{\max(x_{quantized}) - \min(x_{quantized})} \end{aligned} \tag{4}$$

一些文献将权值和激活值量化到同一位宽，Courbariaux 等人[43]提出了一个二值化的神经网络，将权值和激活值二值化(+1 或–1)，二值化神经网络比 32 位精度的网络降低了 32 倍的存储大小和访问需求，并且通过二值化神经网络将大多数算术运算替换为按位计算，带来能耗的大幅度降低. Guo 等人[44]设计了支持低精度数据的加速器，支持 8 位权值和激活值的运算，将能效提高了 16 倍.

另外一些文献对权值和激活值进行了不同程度的量化. Cai 等人[45]认为，量化后的权值可以加载到设备上直接运算，而激活值的量化还需要在设备中随神经网络运行同步进行. 因此 Cai 等人[45]将权值量化到了 4 位、不量化激活值，消除了激活值本地量化的计算过程、加速了运算，只产生了 0.78%的精度损失. 此外，Mishra 等人[46]的实验表明，降低激活值精度比降低权值精度对模型准确度的影响更大. 因此激活值量化后的位宽一般略高于权值，以保证模型压缩率和准确度的平衡. Baskin 等人[47]将权值和激活值分别量化到 4 位和 8 位，将 MobileNet 压缩了 8.05 倍，top-1 精度损失为 2.2%. Elhoushi 等人[48]通过使用卷积移位和全连接移位方法，在训练和推理过程中使用按位移位和符号翻转代替乘法，将权重值压缩至 5 位以下. 在 GoogLeNet 上压缩后精度基本没有损失，并降低了约 9.7 倍能耗. Cao 等人[49]提出了一种新颖的量化方法，作者认为在经过激活和池化层后，卷积层的大部分输出被置为零或丢弃. 作者通过在极低比特网络上预测输出特征图的稀疏性，利用上述稀疏性信息引导原始精度模型的推理，省去了非零输出所对应的计算，最终降低推理能耗.

神经网络每层的冗余度和重要性都有所不同，混合精度(Mixed Precision)量化进一步评估每一层的重要性，并对每一层进行不同程度的量化. Wang 等人[50]提出了一个基于强化学习的自动量化框架 HAQ，以每一层的位宽作为行为，模型精确度作为

奖励来确定最优量化策略. 与固定位宽(8 位)量化相比，该框架降低了 1.9 倍能耗和 1.95 倍的延迟，精度损失可以忽略不计. 蔡瑞初等人[51]分析了网络中权值和激活函数所需的数据位宽，动态量化网络将 VGG-19 网络的卷积层和全连接层分别量化到 9 位和 8 位时未损失精度.

针对激活值精度下降带来的网络准确性下降的问题，Mishra 等人[46]提出了宽减精度网络(Wide Reduced-Precision Networks，WRPN). 该方案降低了激活值和权值的精度，但进一步通过增加层中卷积核的个数，弥补了精度降低对模型准确度造成的损失. 例如,应用了 4 位激活值和 2 位权值的 ResNet 和 GoogLeNet 的简单变体，在卷积核数量变为原网络的两倍之后，网络精度保持不变，但计算成本降低为原来的 39%和 38%. 相对于全精度网络，宽减精度网络在 ASIC 上的能效可以提高 2-3 倍.

除了仅降低参数的精度，权重共享也是量化方法中的一种. 权重共享指将多个数值大小相近的权重值用一个权重值表示，建立共享权重表压缩模型大小. Han 等人[11]量化剪枝后的网络. 通过 k 均值聚类和线性初始化对每一层的权重聚类，将质心作为每一层的共享权重建立共享权重表. 减少了需要存储的有效权重数量，并将表示每个权重所需要的位数从 32 位降低到 5 位. 实验证明，该方法将存储 LeNet-5 所需的空间压缩了 33 倍，且未影响网络准确性. 通过压缩，网络可以完全存储在片上 SRAM 中，而无需访问片外 DRAM，使得网络在移动设备中运行时更加节能.

此外，最近也有相关研究将量化技术引入 Transformer 和 BERT 模型. 这类模型中的参数量和计算量绝大多数来自于多头自注意力模块. Zafrir 等人[52]提出了 BERT 网络量化方法——Q8BERT. Q8BERT 使用对称线性量化方法将 BERT 中所有的全连接层和 Embedding 层权值和激活值量化为 8 位. 实验结果表明，Q8BERT 将模型压缩了 4 倍，量化后准确率降低约 1%. Shen 等人[53] 认为具有较高 Hessian 矩阵谱的网络层参数对量化更敏感，需要更高的量化精度. 于是基于 Hessian 矩阵提出了混合精度量化方法 Q-BERT. 实验表明，该方法将 BERT 压缩了约 13 倍，准确率损失在 2.3%以内. 然而，为了确保 Softmax、Layer Normalization、GELU 的操作精度，上述量化方法在运算时反量化整数值为浮点数后再做运算. Kim 等人[54]首先提出全整数值量化，使得 Transformer 模型可以部署在针对整数运算优化的处理器上，得到更优运算速率. 作者通过设计上述三种操作的整数运算近似方法加速计算，最终实现了 4 倍的推理加速，未造成精度损失. 虽然未见面向能耗的 Transformer 模型优化方法，但更小的参数量和简化的计算过程有利于推理能耗的优化.

4.3　张量分解

在实现层面，神经网络的全连接层和卷积层通常是张量形式. 卷积层中的卷积核可以被视为一个关于卷积核宽高、输入通道数和输出通道数的四维张量. 随着张量维度的升高，内存和计算量都随着维度呈指数增长[55]. 张量分解通过将高维张量分解为几个低维张量的乘积以降低计算量，加速神经网络推理，降低推理能耗.

Kim 等人[56]提出了全网络压缩方法 one shot. 将网络压缩划分为秩选择、Tucker 分解、微调三个部分. Tucker 分解过程如式(5)所示，将大小为 $D\times D\times S\times T$ 的原张量 $\boldsymbol{K}$ 分解为大小为 $D\times D\times R_3\times R_4$、$S\times R_3$ 和 $T\times R_4$ 的三个张量的乘积. 其中 R_3、R_4 的选择由贝叶斯矩阵分解确定. 结果表明，该方案在 AlexNet 和 GoogLeNet 上分别实现了 2.72 倍和 1.42 倍的加速，以及 3.41 倍和 1.60 倍的能耗节省. 宋冰冰等人[57]基于参数估计和遗传算法，提出了自动化张量分解算法. 其中，基于参数估计的自动化 Tucker 分解算法在 LeNet5 上实现了约 4 倍的加速，精度损失为 0.44%.

$$\boldsymbol{K}_{i,j,s,t}=\sum_{r_3=1}^{R_3}\sum_{r_4=1}^{R_4}\boldsymbol{C}_{i,j,r_3,r_4}\boldsymbol{U}^{(3)}_{s,r_3}\boldsymbol{U}^{(4)}_{t,r_4} \tag{5}$$

Tensor-train 分解将一个高维张量分解为多个三维张量的积. Tensor-train 分解过程如式(6)所示，其中 $\boldsymbol{A}\in\mathbb{R}^{n_1\times n_2\times\ldots\times n_d}$，因子张量 $\boldsymbol{G}_k\in\mathbb{R}^{n_k\times r_{k-1}\times r_k}$ $r_0=r_d=1$，因子张量之间执行缩并计算. Huang 等人[58]和 Deng 等人[59]分别基于 tensor-train 分解设计加速器. Huang 等人[58]通过张量分解压缩权重矩阵并结合并行计算矩阵乘法设计加速器，和 3D CMOS-ASIC 相比，实现了 1.283 倍的性能提升和 4.276 倍的能耗节省. Deng 等人[59]则进一步分析并优化了 tensor-train 格式的推理方案. 优化的主要目标是消除 tensor-train 格式在推理过程中固有的冗余计算. 例如，对于共享部分索引的任何输出特征图中的元素对，在计算这些元素对的过程中均存在相同的矩阵乘法阶段. 因此，Deng 等人[59]提出了一种高效的 tensor-train 格式推理方案，实现了乘法次数的理论限制，消除了冗余计算. 实验表明，由于张量分解带来的高压缩比，与 Eyeriss[12]相比，该加速器在吞吐量和能效方面分

别改进了约 3.61 倍和 5.01 倍. Tjandra 等人[60]将低维的权重矩阵 $\boldsymbol{W}\in\mathbb{R}^{M\times N}$ 张量化，即用更高维的张量 $\boldsymbol{W}\in\mathbb{R}^{m_1\times m_2\times..\times m_d\times n_1\times n_2\times..\times n_d}$ 表示原矩阵，其中 $M=\prod_{k=1}^{d}m_k$，$N=\prod_{k=1}^{d}n_k$. 之后使用 tensor-train 方法分解该张量. 在 GRU[61]上的实验表明，该方法将网络压缩了 139.82 倍，只造成了 0.28%的精度损失，有利于降低神经网络能耗.

$$\boldsymbol{A}=\boldsymbol{G}_1\times^1\boldsymbol{G}_2\times^1...\times^1\boldsymbol{G}_d \tag{6}$$

Ma 等人[62]对 Transformer 模型应用了块项分解（Block-Term Tensor Decomposition，BTD）. 块项分解可以看作是 CP 分解和 Tucker 分解的结合. 首先根据 CP 分解张量分解为 P 个块项的和. 每个块项内近似执行 Tucker 分解. 在多项翻译数据集上的实验表明，该方法可以将 Transformer 参数量压缩约 1 倍且未造成性能损失，低秩近似算法也很好地降低了 Transformer 的计算复杂度，有利于优化神经网络能耗.

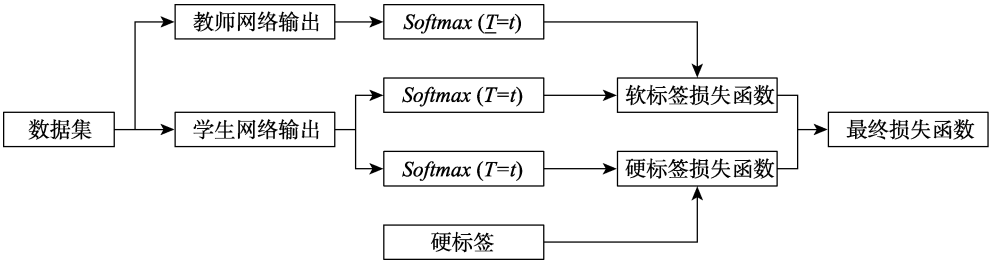

图 3 知识蒸馏过程示意图

4.4 知识蒸馏

语音识别、对象识别等任务常常需要使用冗余的数据集大量训练，这导致了其网络模型通常为单个非常复杂的模型，或者是许多单独训练的模型的集成. 这些模型结构复杂、计算量大，对部署时的存储空间、计算资源和能耗都有比较严格的要求. 知识蒸馏为这种问题提供了解决方法，它通过使用结构复杂的教师网络诱导训练结构简单、性能相当的学生网络来实现知识迁移，最终降低网络参数量，实现模型的轻量化和加速. 尽管现有蒸馏方法的主要设计目标不是降低能耗，但参数量更少的学生网络推理速度更快、内存存取量和计算量更少，有利于降低推理能耗.

Bucila 等人[63]最先提出了基于知识迁移的模型压缩思路—使用较小的学生模型近似较大、性能更好的教师模型. 作者使用训练好的教师模型标注不含标签的数据集，之后使用该数据集训练学生模型. 在 ADULT、COVTYPE 等数据集上的实验评估表明，用这种方式训练得到的学生模型比教师模型小 1000 倍、速度快 1000 倍，且几乎没有造成性能损失. Ba 等人[64]认为 one-hot 编码无法体现标签之间的关联性，例如数字 1 与数字 7 形状相似. 因此作者选择使用学生网络和教师网络的末端输出层结果(计算 *Softmax* 前的结果)作为损失函数的输入，以充分挖掘标签类别信息以及教师网络输出的概率分布. Hinton 等人[65]将这类模型压缩方法统一命名为“蒸馏(Distillation）”，并在上述使用末端输出层结果训练的基础上进一步改进，提出使用带有温度系数 T 的 *Softmax* 函数以提供更多的类别间和类内信息，如式(7)所示，其中 z_i 表示末端输出层结果中第 i 个值. 作者将通过 *Softmax* 函数计算的分类概率称为“软标签”(Soft Target)，将数据集的 0/1 编码标签称为“硬标签”(Hard Targets). 学生网络输出的软标签分别与硬标签、教师网络输出的软标签计算损失函数，加权得到最终的损失函数，训练过程如图 3 所示. 实验表明，蒸馏得到的学生网络大小比教师网络小了 10 倍，虽然造成了 0.3%的精度损失，但其精度仍然比只使用硬标签训练的基线模型高 1.9%. Luo 等人[66]则针对人脸识别进一步优化，筛选出与人脸识别最为相关的神经元并保留其输出，去除不相关的神经元.

$$q\left(z_i,T\right)=\frac{exp\left(z_i/T\right)}{\sum_j exp\left(z_i/T\right)} \tag{7}$$

这种基于模型末端输出蒸馏的方法忽视了教师网络中间层的知识，在构建浅而宽的学生网络时性能较好，但在构建深层学生网络时表现较差. Romero 等人[67]拓展 Hinton 等人[65]所提出的蒸馏方法，将神经网络中间层、最后一层的输出共同作为监督学生模型训练的知识. 作者首先训练教师网络，选取若干教师网络的中间层作为指导层、学生网络的中间层作为学习层，计算指导层和学习层输出的差

异作为损失函数，训练学生网络的中间层. 最后将上述学习结果作为学生网络的初始权重，按照 Hinton 等人[65]的蒸馏方法再次训练整个学生网络. Zagoruyko 等人[68]将注意力引入到上述蒸馏过程中，作者认为一个隐藏神经元通过激活函数后的绝对值越大，代表这个神经元越重要. 因此作者通过在通道维度对特征图进行统计计算，即 $\mathbb{R}^{(C\times H\times W)}\to\mathbb{R}^{(H\times W)}$，得到注意力图作为知识. 而 Heo 等人[69]认为不仅神经网络中间层的输出值可以作为知识，在 *ReLU* 激活函数下神经元的激活状态也可以作为知识传递到学生网络. 如果教师网络神经元被激活，则学生网络的神经元激活值应大于 0，反之应小于 0. 作者参照铰链损失函数(Hinge Loss)的思路设计了针对神经元激活状态的损失函数，实验表明这种方法在大数据集、小数据集以及各类型的网络上的精度损失都小于前有的研究.

Tang 等人、Sanh 等人、Jiao 等人[70−72]分别蒸馏复杂 BERT 网络. Tang 等人[70]第一个从事蒸馏 BERT 网络的研究，提出了以教师网络的末端输出层作为知识，将 BERT 网络蒸馏到浅层双向长短期记忆网络的方法. 实现了 100 倍的参数压缩率，将神经网络运行延迟降低了 5 倍. Sanh 等人[71]对 BERT 网络在预训练阶段和面向特定任务训练阶段的特点进行了研究，认为在预训练阶段蒸馏网络不会影响到后续训练特定任务时精确度. 于是在预训练阶段选取了教师网络中约一半隐藏层的状态训练学生网络. 在保留网络 97%的理解能力的同时，神经网络运行速度提升了 60%. Jiao 等人[72]进一步提出了 TinyBERT 压缩框架，框架定义了三种分别基于嵌入层输出、隐层状态和末端输出层的损失函数，分别在预训练阶段和面向特定任务阶段蒸馏网络. TinyBERT 压缩方法保证了学生网络可以同时学到一般领域和特定领域任务的知识. 实验表明，该框架将模型大小和神经网络运行时间分别降低了 7.5 倍和 9.4 倍，有利于降低神经网络能耗.

4.5　方法比较

能耗优化的神经网络轻量化方法从网络结构和计算过程角度有效降低了网络的参数量和精度，简化了计算过程，优化了网络能耗，有利于模型在移动端的部署. 虽然部分文章没有直接测量能耗优化结果，但根据第 3 节能耗估算方法的研究结果，更小的参数量和简化的计算过程分别有利于降低神经网络运行时参数的存取和计算能耗. 本节所介绍的剪枝、量化、张量分解、知识蒸馏分别总结于表 3、表 4、表 5、表 6. 表中压缩倍数、能耗节约倍数表示在该研究的对比实验中，轻量化后的网络对比原网络，网络参数文件减少的倍数和能耗节约的倍数.

表 3　神经网络剪枝方法小结

文献	轻量化规则	精度恢复	实验网络	压缩倍数	Top5 精度损失	能耗节约倍数	适用类型
[29]	修剪权重	结束后全局	AlexNet	9	≈0	–	CNN
[8]	修剪权重	修剪时局部 结束后全局	GoogLeNet	2.94	0.98%	1.6	CNN
[8]	修剪权重	修剪时局部	SqueezeNet	3.57	0.14%	1.3	CNN
[30]	修剪权重	修剪时局部	MobileNet-v1	1.98	≈0	–	CNN RNN
[30]	修剪权重	修剪时局部	NMT	4.8	≈0	–	CNN RNN
[32]	修剪卷积核	结束后全局	VGG-16	17.5	≈0	–	CNN
[33]	修剪卷积核	结束后全局	VGG-16	13.6	0.45%	–	CNN
[36]	修剪卷积核	修剪时局部	MobileNet-v1	1.4	0.2%	–	CNN
[31]	修剪激活	修剪时局部	LeNet	3.85	0.01%	–	CNN
[34]	修剪特征图	修剪时局部	VGG-16	1.515	2.3%	–	CNN
[35]	修剪特征图	修剪时全局	MobileNet-v3	3.6	4.2%	–	CNN
[37]	修剪特征图	修剪时全局	YOLOv3	9.64	0.063(mAP)	–	CNN
[40]	修剪权重	修剪时局部	BERT	10	1.5%	–	Transformer
[41]	修剪注意力	结束后全局	DeiT-B	3	0.25%	–	Transformer
[42]	修剪层	无精度恢复	RoBERTa	4	≈0	–	Transformer
[11]	修剪并共享权重(+量化)	修剪和量化后全局	LeNet-300-100	32	≈0	–	CNN
[11]	修剪并共享权重(+量化)	修剪和量化后全局	AlexNet	33	≈0	7	CNN

表 4 神经网络量化方法小结

文献	轻量化规则	精度恢复	实验网络	压缩倍数	Top-5 精度损失	能耗节约倍数	适用类型
[43]	1 位权重和激活	结束后全局	–	32	<4%	–	CNN
[47]	4 位权重 8 位激活	量化时局部	MobileNet-v1	8.05	2.2%(top-1)	–	CNN
[46]	2 位权重 4 位激活	结束后全局	ResNet-34	–	≈0	2.56	CNN
[50]	2-8 位权重和激活	量化时局部	MobileNet-v2	7.47	0.09%	1.9	CNN
[45]	4 位权重	无精度恢复	GoogLeNet	–	0.78%	–	CNN
[44]	8 位权重和激活	量化后全局	GoogLeNet	4	3.20%	16	CNN
[45]	5 位权重	量化后全局	GoogLeNet	–	≈0	9.7	CNN
[46]	4 位权重 1 位激活	无精度恢复	ResNet-18	–	0.18%	–	CNN
[51]	动态量化	量化后全局	SqueezeNet	4	0.5%	–	CNN
[52]	8 位权重和激活	量化后全局	BERT	4	1%	–	Transformer
[53]	2-8 位权重和激活	量化后全局	BERT	13	2.3%	–	Transformer
[54]	8 位权重和激活	量化后全局	BERT	4	≈0	–	Transformer

表 5 神经网络张量分解方法小结

文献	轻量化规则	精度恢复	实验网络	压缩倍数	Top-5 精度损失	能耗节约倍数	适用类型
[56]	Tucker 分解	结束后全局	AlexNet	5.46	1.70%	3.41	CNN
[56]	Tucker 分解	结束后全局	GoogLeNet	1.28	0.24%	1.60	CNN
[57]	CP 分解、Tucker 分解	分解时全局	LeNet5	–	0.44%	–	CNN
[58]	TT 分解	结束后全局	自定义网络	14.85	1.50%	4.28	CNN
[60]	TT 分解	结束后全局	GRU	139.82	0.28%	–	RNN
[59]	TT 分解	结束后全局	GRU	195	≈0	–	CNN RNN
[62]	BTD 分解	结束后全局	Transformer	2.45	≈0	–	Transformer

表 6 神经网络知识蒸馏方法小结

文献	轻量化规则	网络	压缩倍数	Top-5 精度损失	能耗节约倍数	适用类型
[66]	基于末端层蒸馏	DeepID2+	10.42	3.10%	–	CNN
[64]	基于末端层蒸馏	自定义网络	8.6	≈0	–	CNN
[65]	基于末端层蒸馏	AlexNet	–	–	–	CNN
[67]	基于中间层蒸馏	Maxout Network	36	1.17%	–	CNN
[68]	基于中间层蒸馏	ResNet-34	1.9	1.70%	–	CNN
[69]	基于中间层蒸馏	自定义网络	6.21	1.07%	–	CNN
[70]	基于末端层蒸馏	BERT	349	>1.2%	–	Transformer
[71]	基于部分隐层蒸馏	BERT	1.67	1.1%	–	Transformer
[72]	基于全部层蒸馏	BERT	7.5	≈0	–	Transformer

对于轻量化过程导致的模型精度损失，还需要考虑对模型进行精度恢复. 例如，Han 等人[11]在剪枝和量化实现高压缩比的同时，将网络模型精度恢复到压缩前，并且节省了 7 倍的能耗. 精度恢复主要通过压缩后重新训练模型的方法来实现，不同研究的精度恢复方法有所不同. 我们在表 3、表 4、表 5 中将其总结并简称为四类方法——“无精度恢复”代表不对轻量化后的模型进行精度恢复；“结束后全局”代表在模型轻量化完成后对模型整体进行重新训练；“修剪/量化时局部，结束后全局”代表在轻量化过程中伴随着局部模型的重新训练，压缩结束后对模型整体进行重新训练；“修剪/量化时局部”代表只在轻量化过程中对局部模型重新训练，轻量化结束后不再重新训练模型；“修剪/量化/分解时全局”代表在轻量化的几次迭代结束后都要重新训练模型. 另外，未造成精度损失和精度上升的情况，在精度损失一栏中用“≈0”表示. 最后，由于知识蒸馏过程中同时对学生网络训练，因此蒸馏方法不包含精度恢复过程.

5 问题与挑战

能耗优化的神经网络轻量化方法正在受到越来

越多的关注，各种软硬件能耗估算和神经网络轻量化方法也都取得了不错的效果. 但从研究现状来看，能耗估算和网络轻量化方法还存在一些问题与挑战如下.

5.1　可自适应网络类型的能耗模型

目前能耗估算和能耗优化的轻量化方法主要针对具体问题和场景，如特定类型的网络和设备提出一种能耗估算或轻量化的方法，缺乏统一且系统的能耗模型，并在能耗模型的基础上系统地研究神经网络能耗优化问题. 神经网络目前大致可以分为全连接网络、卷积神经网络、循环神经网络、Transformer网络、图神经网络等. 然而随着技术的发展，神经网络结构越发复杂，甚至出现了多种类型网络的复合体. 所以考虑到目前神经网络结构和应用上的复杂性，建立可自适应网络类型的能耗模型是极具挑战性. 但构建可自适应网络类型的能耗模型是极具必要性的. 可自适应网络类型的能耗模型可以屏蔽网络结构的复杂性，有利于得到更为平凡化的能耗估算和能耗优化的轻量化方法. 然而考虑到网络类型的多种多样，建立可自适应网络类型的能耗模型中关键问题是解决海量网络模型归纳难问题.

考虑到神经网络本身以及其计算过程是一个图结构，可以利用图的性质，建立图结构模型作为能耗模型. 其结构信息可以利用图论的相关方法抽取. 若结合深度学习方法，可以选用目前较为流行的Transformer[38]结构或图神经网络(Graph Neural Network)[73]作为能耗模型的基础. 其中 Transformer 能够通过注意力机制(Attention Mechanism)获取图中节点间的相互作用关系，抽取节点特征. 图神经网络则可利用图的邻接矩阵，基于邻居节点更新节点间的信息. 在近期研究中，出现了结合图神经网络和注意力机制的一些新模型，其中比较有代表性的是 GAT(Graph Attention Network)[74]网络. 与传统图神经网络不同的是，GAT 利用注意力机制学习所有邻居节点的“贡献度”，即权重. 在更新节点时，同时考虑其邻居节点的信息及权重. 实验表明，GAT 可以有效地抽取有向图的结构信息，有利于抽取神经网络的结构特征.

5.2　平衡精度和能耗的轻量化方法

目前能耗优化的神经网络轻量化方法的评价指标侧重于准确率、吞吐量、模型压缩比、能耗降低比等方面，许多优化方法以上述指标为优化目标优化网络模型，无法综合考虑能耗降低比和精度. 考虑一种极端情况，若轻量化方法压缩率极高、单次运行能耗极低，但轻量化模型精度损失极大，几乎无法预测出正确结果. 这种低能耗低精度模型显然不是我们所需要的. 通常我们认为网络结构越复杂其精度越高能耗越高. 但该关系并不是线性的. 甚至在一些情况中并不是相关的. 例如当网络结构发生巨大变革时，往往结构变简单精度变高同时能耗变低. 因此，在神经网络轻量化方法中如何平衡精度和能耗是极具挑战的问题. 其关键问题是提出一种综合且易于计算的度量以涵盖精度和能耗.

$$EE = \frac{1 \times \boldsymbol{A}}{E} \tag{8}$$

在绿色计算和高性能计算领域，能效通常是“有用功”(即计算量或任务量)和能(即用电量)的比值. 最大化能效指在单位能耗下，完成尽可能多的计算或任务[75]. 我们可以在神经网络能效优化的背景下借鉴该能效的定义作为平衡精度和能耗的度量. 较为简洁的定义如式(8)所示，其中 EE 指网络能效，E 指单次运行神经网络的能耗，A 指的网络的精确度. 由该式可知，仅当神经网络轻量化方法最小化神经网络能耗且最大化模型精度时，才能获得最优网络能效.

能效的度量还必须是易于得到或计算的. 在式(8)中，E 可由能耗估算方法给出，但精确度 A 仍需要一种精度度量方法给出. 较为简单的方式是直接使用损失函数评估. 然而在轻量化过程中，通常在轻量化结束后进行精度恢复，导致中间网络并未达到其最佳精确度. 同时，精度恢复方法往往是重训练网络，从而导致轻量化模型精度无法在短时间内收敛. 所以还需建立一种神经网络精度的预测模型. 目前常用方法是使用精度恢复前几轮的精度增长率构建预测模型如线性模型进行预测，或是通过神经网络结构建模的方式，直接从结构特征出发构建精度的预测模型.

5.3　硬件平台可泛化的轻量化方法

在能耗估算和能耗优化相关研究中，硬件是能耗产生和优化的载体. 在神经网络应用过程中，存在大量不同结构设计和优化的硬件设备，所以硬件差异是客观存在且必须纳入考量的，轻量化方法必须是硬件平台可泛化的. 如何考虑到嵌入式设备和 IoT 设备的差异，使轻量化方法具备硬件平台泛化性是一个巨大的挑战. 其中如何综合地考虑硬件特性去除干扰性和无关性因素是一个关键性问题.

目前能耗估算方法和能耗优化的轻量化研究均与硬件平台存在巨大的耦合. 在能耗估计法中，训练数据从某种硬件平台上收集得到，当硬件平台变

更时，需要针对新的硬件平台重新调整或设计估算模型，导致难以大规模推广使用. 在能耗优化的轻量化方面，许多神经网络压缩方法需结合特定的硬件才能更好地实现优化效果. 以剪枝方法为例，一些非结构化剪枝方法产生大量值为 0 的神经元，需要结合特定的硬件设计，跳过数值为 0 的神经元的计算，才能最大化能耗的优化效果. 轻量化方法硬件依赖性会造成对不同硬件需要设计不同的轻量化策略，造成模型轻量化和部署的困难. 因此未来需要更多地考虑高效的、无硬件依赖的能耗估算和能耗优化的轻量化方法.

在能耗估算中若要考虑硬件可泛化性，势必需抽取硬件的能耗特征，建立统一的硬件能耗模型，该模型可以是白盒的也可以是黑盒的. 假设模型是白盒的，那么该模型适用于不同种类的硬件，区别在于模型参数. 目前使用 Roofline 模型可以较好的分析神经网络的运行效率[76]. 可以借用相似的思想，建立硬件能耗估算方法. 如果认为模型是黑盒的，那么硬件信息则体现于能耗数据收集过程中. 可以借助迁移学习的理念，在已训练的能耗预测模型基础上，收集新硬件平台下神经网络运行的能耗数据迁移评估模型.

在能耗优化的轻量化中若要考虑硬件可泛化性，轻量化方法需要在轻量化过程中考虑不同硬件的特点. 如果把硬件视作黑盒，那么轻量化方法需依赖能耗估算方法的硬件可迁移性. 如果把硬件视作白盒，那么轻量化方法需要考虑不同硬件的特点，做有针对性的优化. 例如，当硬件支持更低维度整型计算时，量化方法可以取得更好的压缩效果；当硬件计算器支持跳过因子为 0 的乘法运算时，非结构化的剪枝可以取得更好的效果. 所以可以针对具体的硬件特征，综合选择多种轻量化方法.

5.4 搜索空间可约束的轻量化方法

能耗优化的神经网络轻量化可以看作以轻量化为目标的搜索问题，其结果是一个高能效的轻量化策略. 然而，结合多个优化目标、硬件差异、网络结构差异等因素时，搜索算法的搜索空间是异常巨大的. 同时考虑到网络结构的复杂性和网络参数数量的指数级增长，整体的搜索空间也在爆发性增长. 如何制定搜索空间可约束的轻量化方法，在轻量化过程中缩小搜索空间具有巨大的挑战. 在约束搜索空间中的关键性问题，是如何找到一种可采纳式启发精准地衡量当前网络状态和目标网络状态间的距离，从而快速地在所有可选轻量化方向中找到最有潜力的方向.

可采纳式启发常见于人工智能搜索问题. 可采纳式启发作为一种度量评估，势必依赖能耗模型表征网络结构特征. 同时利用结构特征推理较优的轻量化策略. 目前推理系统可以采用逻辑推理、贝叶斯网络以及认知图(Cognitive Graph)技术. 以认知图网络技术为例，其包含 3 个主体结构：表示学习系统、认知系统和推理系统. 表示学习系统用于向量化网络结构的特征，服务于认知和推理系统，类似于能耗模型. 认知图网络总是尝试着模拟人类的学习和过程. 对于不熟悉或者未知的事务，认知系统凭借一般经验做出最合理的猜想和尝试. 而推理系统则依照长期的经验构建基于规则的推理系统. 认知图技术可以被视为结合了符号主义和连接主义的方法之一. 认知系统是连接主义的代表类似于传统的监督学习器，通过拟合数据的分布获得一个分类或回归函数. 推理系统则是符号主义的代表，类似于传统的贝叶斯网络或知识图谱，通过自动地发觉数据中相应的规则自动构建推理系统. 通过认知图技术可以构建一个合理的可采纳式启发解决轻量化中搜索空间过大问题.

6 结 论

作为人工智能实现的典范，近年来，神经网络在语音识别、计算机视觉、自然语言处理等许多领域都取得了良好的表现. 但神经网络参数量大、结构复杂，不仅占用了大量的存储空间，也在运行时消耗了大量电能，阻碍了移动端设备的智能化发展. 为了解决神经网络在移动设备上能耗受限问题，能耗优化的神经网络轻量化方法从神经网络结构入手，降低网络运行时能耗.

本文介绍了神经网络能耗估算和能耗优化的神经网络轻量化方法的基本思路，随后介绍了测量法、分析法、估计法三类能耗估算方法，以及剪枝、量化、张量分解、知识蒸馏四类神经网络轻量化方法，最后提出了能耗估算和能耗优化的轻量化方法存在的问题和挑战及进一步研究的方向. 总而言之，能耗优化的神经网络能耗轻量化领域仍然有许多问题亟待解决，希望本文能够为相关领域的研究者提供参考和借鉴.

参 考 文 献

[1] Jiao Li-Cheng, Yang Shu-yuan, Liu Fang, et al. Seventy years beyond neural networks: Retrospect and prospect. Chinese Journal of Computers, 2016, 39(8): 1697-1716 (in Chinese)

(焦李成, 杨淑媛, 刘芳, 等. 神经网络七十年：回顾与展望. 计算机学报, 2016, 39(8): 1697-1716)

[2] Technology, Media and Telecommunications Predictions, https://www2.deloitte.com/cn/en/pages/technology-media-and-telecommunications/articles/tmt-predictions-2018.html 2018

[3] AI 手机出货量将占比 80%, https://baijiahao.baidu.com/s?id=1623998367334176942&wfr=spider&for=pc 2019,01,29

[4] Pal S, Beaumont J, Park D-H, et al. OuterSPACE: An outer product based sparse matrix multiplication accelerator// Proceedings of the IEEE International Symposium on High Performance Computer Architecture. Vienna, Austria, 2018: 724–736

[5] Takhirov Z, Wang J, Saligrama V, et al. Energy-efficient adaptive classifier design for mobile systems//Proceedings of the International Symposium on Low Power Electronics and Design. San Francisco, USA, 2016: 52-57

[6] Cheng Y, Wang D, Zhou P, et al. A Survey of model compression and acceleration for deep neural networks. arXiv, 2020, 1710. 09282 [cs], 1-10

[7] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 2017, 60(6): 84-90

[8] Yang T-J, Chen Y-H, Sze V. Designing energy-efficient convolutional neural networks using energy-aware pruning// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Honolulu, USA, 2017: 6071-6079

[9] Ren Jie, Gao Ling, Yu Jia-Long, Yuan Lu. Energy-efficient deep learning task scheduling strategy for edge device. Chinese Journal of Computers, 2020, 43(3): 440–452 (in Chinese)
(任杰, 高岭, 于佳龙， 袁璐. 面向边缘设备的高能效深度学习任务调度策略, 面向边缘设备的高能效深度学习任务调度策略. 计算机学报, 2020, 43(3): 440-452)

[10] Zhang Yi-Lin, Liang Yu-Zhu, Yin Mu-Jun, et al. Survey on the methods of computation offloading in mobile edge computing. Chinese Journal of Computers, 2021, 44(12): 2406-2430 (in Chinese)
(张依琳, 梁玉珠, 尹沐君, 等. 移动边缘计算中计算卸载方案研究综述. 计算机学报, 2021, 44(12): 2406-2430)

[11] Han S, Mao H, Dally W J. Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding//Proceedings of the International Conference on Learning Representations. San Juan, Puerto Rico, 2016: 1-14

[12] Chen Y-H, Emer J, Sze V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks// Proceedings of the ACM/IEEE 43rd Annual International Symposium on Computer Architecture. Seoul, South Korea: 2016: 367-379

[13] Qi H, Sparks E R, Talwalkar A. Paleo: A performance model for deep neural networks//Proceedings of the International Conference on Learning Representations. Toulon, France: 2017: 1-10

[14] Simonyan K, Zisserman A, et al. Very deep convolutional networks for large-scale image recognition//Proceedings of the International Conference on Learning Representations. San Diego, USA, 2015: 1-14

[15] Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial network//Proceedings of the International Conference on Learning Representations. San Juan, Puerto Rico, 2016: 1-16

[16] Yang T-J, Chen Y-H, Emer J, et al. A method to estimate the energy consumption of deep neural networks//Proceedings of the 51st Asilomar Conference on Signals, Systems, and Computers. Pacific Grove, USA: 2017: 1916-1920

[17] Szegedy C, Wei Liu, Yangqing Jia, et al. Going deeper with convolutions//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, USA, 2015: 1-9

[18] Cai E, Juan D-C, Stamoulis D, et al. Neuralpower: predict and deploy energy-efficient convolutional neural network//Proceedings of the Ninth Asian Conference on Machine Learning. Seoul, Kore, 2017: 622-637

[19] Rodrigues C F, Riley G, Luján M. SyNERGY: An energy measurement and prediction framework for Convolutional Neural Networks on Jetson TX//Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. Las Vegas, USA, 2018: 375-382

[20] Rodrigues C F, Riley G, Lujan M. Energy predictive models for convolutional neural networks on mobile platforms. arXiv, 2020, abs/2004.05137, 1-9

[21] James G, Witten D, Hastie T, et al. An introduction to statistical learning. USA, Springer, 2013

[22] Dai X, Jia Y, Vajda P, et al. ChamNet: Towards efficient network design through platform-aware model adaptation//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Long Beach, USA, 2019: 11390-11399

[23] Howard A G, Zhu M, Chen B, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv, 2017, abs/1704.04861: 1-9

[24] Bouzidi H, Ouarnoughi H, Niar S, et al. Performance prediction for convolutional neural networks on edge GPUs//Proceedings of the 18th ACM International Conference on Computing Frontiers. New York, USA, 2021: 54-62

[25] Chen T, Guestrin C. XGBoost: A scalable tree boosting system// Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, USA: 2016: 785-794

[26] Justus D, Brennan J, Bonner S, et al. Predicting the computational cost of deep learning models//Proceedings of the IEEE International Conference on Big Data. Seattle, USA: 2018: 3873- 3882

[27] Eshratifar A E, Pedram M. Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment//Proceedings of the Great Lakes Symposium on VLSI. Chicago, USA, 2018: 111-116

[28] Rouhani B D, Mirhoseini A, Koushanfar F. DeLight: Adding Energy dimension to deep neural networks//Proceedings of the International Symposium on Low Power Electronics and Design. San Francisco, USA, 2016: 112-117

[29] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network//Advances in Neural Information Processing Systems. Montreal, Canada, 2015: 1135-1143

[30] Zhu M, Gupta S. To prune, or not to prune: exploring the efficacy of pruning for model compression//Proceedings of the 6th International Conference On Learning Representations. Vancouver, Canada, 2018: 1-10

[31] Hu H, Peng R, Tai Y-W, et al. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv, 2016, abs/1607.03250: 1-9

[32] Singh P, Verma V K, Rai P, et al. Play and Prune: Adaptive Filter

pruning for deep model compression//Proceedings of the 28th International Joint Conference on Artificial Intelligence. Macao, China, 2019: 3460–3466

[33] Wang Guo-Dong, Ye Jian, Xie Ying, Qian Yue-Liang. An adaptive threshold structured pruning algorithm based on gradient. Computer Engineering, 2021: 1-12 (in Chinese)
(王国栋, 叶剑, 谢萦, 钱跃良. 基于梯度的自适应阈值结构化剪枝算法. 计算机工程, 2021: 1-12)

[34] Molchanov P, Tyree S, Karras T, et al. Pruning convolutional neural networks for resource efficient inference//Proceedings of the 5th International Conference On Learning Representations. Toulon, France, 2017: 1-17

[35] Xu Xiao, Jiang Zhi-Xiang, Zhang Yang. Convolution neural network comprehension method based on channel pruning and quantization. Computer Engineering and Design, 2021, 42(10): 2860-2866 (in Chinese)
(徐晓, 蒋志翔, 张扬. 基于通道剪枝和量化的卷积神经网络压缩方法. 计算机工程与设计, 2021, 42(10): 2860-2866)

[36] He Y, Lin J, Liu Z, et al. AMC: automl for model compression and acceleration on mobile devices//Proceedings of the 15th European Conference. Munich, Germany, 2018: 815-832

[37] Feng Jing-Xiang. Channel pruning of convolutional neural network based on transfer learning. Computer and Modernization, 2021, 0(12): 13-18 (in Chinese)
(冯敬翔. 基于迁移学习的卷积神经网络通道剪枝. 计算机与现代化, 2021, 0(12): 13-18)

[38] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Proceedings of the Annual Conference on Neural Information Processing Systems. Long Beach, USA: 5998-6008

[39] Gordon M A, Duh K, Andrews N. Compressing BERT: Studying the Effects of weight pruning on transfer learning//Proceedings of the 5th Workshop on Representation Learning for NLP. Online, 2020: 143-155

[40] Sanh V, Wolf T, Rush A M. Movement Pruning: Adaptive Sparsity by fine-tuning//Proceedings of the Advances in Neural Information Processing Systems. Online, 2020: 20378-20389

[41] Yu H, Wu J. A unified pruning framework for vision transformers. arXiv, 2021, abs/2111.15127: 1-11

[42] Fan A, Grave E, Joulin A. reducing transformer depth on demand with structured dropout//Proceedings of the International Conference on Learning Representations. Addis Ababa, Ethiopia, 2020: 1-15

[43] Courbariaux M, Hubara I, Soudry D, et al. Binarized Neural networks: training deep neural networks with weights and activations constrained to +1 or -1. arXiv, 2016, abs/1602.02830: 1-12

[44] Guo K, Sui L, Qiu J, et al. Angel-eye: A complete design flow for mapping CNN onto embedded FPGA. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(1): 35-47

[45] Cai J, Takemoto M, Nakajo H. A deep look into logarithmic quantization of model parameters in neural network//Proceedings of the 10th International Conference on Advances In Information Technology, IAIT. Bangkok, Thailand, 2018: 1-8

[46] Mishra A, Nurvitadhi E, Cook J J, et al. WRPN: Wide Reduced-precision networks//Proceedings of the International Conference on Learning Representations. Vancouver, Canada, 2018: 1-11

[47] Baskin C, Liss N, Schwartz E, et al. UNIQ: Uniform noise injection for non-uniform quantization of neural networks. ACM Transactions on Computer Systems, 2021, 37(4): 1-15

[48] Elhoushi M, Chen Z, Shafiq F, et al. DeepShift: Towards Multiplication-less neural networks//Proceedings of the IEEE/CVF Conference On Computer Vision And Pattern Recognition. Online, 2021: 2359-2368

[49] Cao S, Ma L, Xiao W, et al. SeerNet: Predicting Convolutional neural network feature-map sparsity through low-bit quantization//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Long Beach, USA: 2019: 11208-11217

[50] Wang K, Liu Z, Lin Y, et al. HAQ: Hardware-Aware automated quantization with mixed precision//Proceedings of the Computer Vision and Pattern Recognition. Long Beach, USA, 2019: 8604-8612

[51] Cai Rui-Chu, Zhong Chun-Rong, Yu Yang, et al. Cnn quantization and compression strategy for edge computing applications. Journal of Computer Applications, 2018, 38(9): 2459-2454 (in Chinese)
(蔡瑞初, 钟椿荣, 余洋等. 面向边缘应用的卷积神经网络量化与压缩方法. 计算机应用, 2018, 38(9): 2459-2454)

[52] Zafrir O, Boudoukh G, Izsak P, et al. Q8BERT: Quantized 8Bit BERT//Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing. Vancouver, Canada, 2019: 36-39

[53] SHEN S, DONG Z, YE J, et al. Q-BERT: Hessian Based ultra low precision quantization of BERT//Proceedings of the AAAI Conference on Artificial Intelligence. New York, USA, 2020: 8815- 8821

[54] Kim S, Gholami A, Yao Z, et al. I-BERT: Integer-only BERT quantization. arXiv, 2021, 2101.01321: 1-13

[55] Oseledets I V. Tensor-train decomposition. SIAM Journal on Scientific Computing, 2011, 33(5): 2295-2317

[56] Kim Y-D, Park E, Yoo S, et al. Compression of deep convolutional neural networks for fast and low power mobile applications//Proceedings of the International Conference on Learning Representations. San Juan, Puerto Rico, 2016: 1-16

[57] Song Bign-Bing, Zhang Hao, Wu Zi-Feng et al. Automated tensor decomposition to accelerate convolutional neural network. Journal of Software, 2021, 32(11): 3468-3481 (in Chinese)
(宋冰冰, 张浩, 吴子锋, 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)

[58] Huang H, Yu H. A Highly-Parallel and Energy-Efficient 3D Multi-Layer CMOS-RRAM Accelerator for Tensorized Neural Network. IEEE Transactions on Nanotechnology, 2017, 17(4): 645-656

[59] Deng C, Sun F, Qian X, et al. TIE: Energy-efficient tensor train-based inference engine for deep neural network//Proceedings of the 46th International Symposium on Computer Architecture. Phoenix Arizona: 2019: 264-278

[60] Tjandra A, Sakti S, Nakamura S. Tensor decomposition for compressing recurrent neural network//Proceedings of the International Joint Conference on Neural Networks, Rio de Janeiro, Brazil, 2018: 1-8

[61] Chung J, Gülçehre Ç, Cho K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv, 2014, abs/1412.3555: 1-9

[62] Ma X, Zhang P, Zhang S, et al. A tensorized transformer for language modeling//Proceedings of the Advances in Neural Information Processing Systems. Vancouver, Canada, 2019: 2229-2239
[63] Bucila C, Caruana R, Niculescu-Mizil A, et al. Model compression// Proceedings of the twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, USA, 2006: 535-541
[64] Ba J, Caruana R, Ghahramani Z, et al. Do deep nets really need to be deep?//Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conference On Neural Information Processing Systems, Montreal, Canada, 2014: 2654-2662
[65] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv, 2015. abs/1503.02531:1-9
[66] Luo P, Zhu Z, Liu Z, et al. Face model compression by distilling knowledge from neurons//Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, USA. 2016: 3560–3566
[67] Romero A, Ballas N, Kahou S E, et al. FitNets: Hints for thin deep nets//Proceedings of the 3rd International Conference On Learning Representations, San Diego, USA, 2015: 1-13
[68] Zagoruyko S, Komodakis N. Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer//Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 2017: 1-13
[69] Heo B, Lee M, Yun S, et al. Knowledge transfer via distillation of activation boundaries formed by hidden neurons// Proceedings of the 3rd AAAI Conference on Artificial Intelligence, Honolulu, USA, 2017: 3779-3787
[70] Tang R, Lu Y, Liu L, et al. Distilling task-specific knowledge from bert into simple neural networks. arXiv, 2019, abs/1903.12136
[71] Sanh V, Debut L, Chaumond J, et al. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv, 2020, abs/1910.01108v4: 1-5
[72] Jiao X, Yin Y, Shang L, et al. TinyBERT: Distilling BERT for natural language understanding//Proceedings of the Findings of the Association for Computational Linguistics. Online, 2020: 4163-4174
[73] Xu K, Hu W, Leskovec J, et al. How powerful are graph neural networks?//Proceedings of the International Conference on Learning Representations. New Orleans, USA, 2019: 1-19
[74] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks//Proceedings of the International Conference on Learning Representations. Vancouver, Canada, 2018:1-12
[75] Ye Ke-Jiang, Wu Zhao-Hui, Jiang Xiao-Hone, et al. Power management of virtualized cloud computing platform. Chinese Journal of Computers, 2012, 35(6): 1262-1285 (in Chinese)
(叶可江, 吴朝晖, 姜晓红, 等. 虚拟化云计算平台的能耗管理. 计算机学报, 2012, 35(6): 1262-1285)
[76] Wang Y, Yang C, Farrell S, et al. Time-based roofline for deep learning performance analysis//Proceedings of the IEEE/ACM Fourth Workshop on Deep Learning on Supercomputers (DLS). Atlanta, USA: 2020: 10-19

GUO Chao-Peng, Ph.D., lecturer. His main research interests include green computing and high-performance computing.

WANG Xin-Xin, master student. Her research interest is green computing.

ZHONG Zhao-Jin, master student. His research interest is high performance computing.

SONG Jie, Ph.D., professor. His research interests include big data management, green computing, and blockchain.

Background

In recent years, neural networks have developed rapidly in speech recognition, computer vision, natural language processing and other fields. The deployment of neural networks on the mobile devices has also greatly facilitated people's lives. Due to its complex structure and parameters, neural networks occupy a lot of storage space and consume lots of electric energy during running. However, embedded devices are usually small in size, limited in storage space and electric energy. Therefore, the deployment of neural network in embedded devices is restricted. In recent years, many neural network lightweight methods for energy optimization have emerged in the field of neural network optimization. The goal of these lightweight methods is to reduce the amount of network parameters and simplify the calculation process while running, so as to shorten network’s inference time and lower its energy consumption.

Firstly, we introduce three types of energy consumption estimation methods. With the help of energy consumption estimation methods, we can better analyze the causes of energy consumption. Then, this paper summarizes four kinds of network lightweight methods, including pruning, quantization, tensor decomposition and knowledge distillation. We compare the experimental ideas, experimental results, advantages and disadvantages of each method in detail. Finally, this article summarizes the challenges of energy consumption estimation and compression methods, and proposes possible solutions.

In the field of green computing and high-performance computing, the previous review of neural network optimiza-

tion methods only focused on the performance optimization of neural networks, that is, how to optimize the amount of network parameters and its running time, and ensure that the accuracy of the network is not affected. This article summarizes neural network energy consumption estimation methods and neural network lightweight methods for energy optimization. It is the first Chinese review for energy consumption optimization in the field of green computing and high-performance computing. This paper is helpful for researchers to understand the development status and research direction of neural network lightweight methods for energy optimization.

This project was funded by the Liaoning Provincial Natural Science Foundation of China (2020-BS-054) and by the National Natural Science Foundation of China under Grant No. 62162050.