文章编号: 1000-5641(2021)06-0112-12

基于网络压缩与切割的深度模型边云协同加速机制研究

王 诺, 李丽颖, 钱栋炜, 魏同权

(华东师范大学 计算机科学与技术学院, 上海 200062)

摘要: 人工智能 (Artificial Intelligence, AI) 的先进技术已被广泛应用于实时地处理大量数据, 以期实现快速响应. 但是, 部署基于 AI 的各种应用程序的常规方法带来了巨大的计算和通信开销. 为了解决这一问题, 提出了一种基于网络压缩与切割技术的深度模型边云协同加速机制, 该技术可以压缩和划分深度神经网络 (Deep Neural Networks, DNN) 模型, 以边云协同的形式在实际应用中实现人工智能模型的快速响应. 首先压缩神经网络, 以降低神经网络所需要的运行时延, 并生成可用作候选分割点的新层, 然后训练预测模型以找到最佳分割点, 并将压缩的神经网络模型分为两部分. 将所获得的两部分分别部署在设备和云端服务器中, 这两个部分可以协同地将总延迟降至最低. 实验结果表明, 与 4 种基准测试方法相比, 本文所提出的方案可以将深度模型的总延迟至少降低 70%.

关键词: 边云协同; 深度神经网络压缩; 深度神经网络切割

中图分类号: TP393 **文献标志码**: A **DOI**: 10.3969/j.issn.1000-5641.2021.06.012

Research on an Edge-Cloud collaborative acceleration mechanism of deep model based on network compression and partitioning

WANG Nuo, LI Liying, QIAN Dongwei, WEI Tongquan

(*School of Computer Science and Technologye, East China Normal University, Shanghai* 200062, *China*)

Abstract: The advanced capabilities of artificial intelligence (AI) have been widely used to process large volumes of data in real-time for achieving rapid response. In contrast, conventional methods for deploying various AI-based applications can result in substantial computational and communication overhead. To solve this problem, a deep model Edge-Cloud collaborative acceleration mechanism based on network compression and partitioning technology is proposed. This technology can compress and partition deep neural networks (DNN), and deploy artificial intelligence models in practical applications in the form of an Edge-Cloud collaboration for rapid response. As a first step, the proposed method compresses the neural network to reduce the execution latency required and generates a new layer that can be used as a candidate partition point. It then trains a series of prediction models to find the best partitioning point and partitions the compressed neural network model into two parts. The two parts obtained are deployed in the edge device and the cloud server, respectively, and these two parts can collaborate to minimize the overall latency. Experimental results show that, compared with four benchmarking methods, the proposed scheme can reduce the total delay of the depth model by more than 70%.

Keywords: Edge-Cloud collaboration; DNN compression; DNN partitioning

收稿日期: 2020-09-10

通信作者: 魏同权, 男, 副教授, 博士生导师, 研究方向为边缘计算、深度学习、数据分析与服务计算.
E-mail: tqwei@cs.ecnu.edu

0　引　　言

随着云端服务器计算能力的不断增强以及无线网络的不断发展, 越来越多的基于人工智能 (Artificial Intelligence, AI)的应用在汹涌的工业 4.0 浪潮中得到了广泛的部署[1]. 在这一浪潮中面临的两个关键挑战是如何在资源受限的设备上部署深度神经网络, 以及如何在实际应用程序中实时地应用这些深度模型进行推理. 诸如卷积神经网络 (Convolutional Neural Networks, CNN) 之类的深度 AI 模型虽然在许多任务中实现了显著的准确性提高, 但它们始终是计算密集和存储密集型的[2]. 例如, 流行的 CNN 模型 VGG-19[3] 具有近 1.4 亿个参数, 这会占用约 500 MB 的存储空间, 并且需要近 310 亿个浮点运算 (Floating-Point Operation per Second, FLOP) 来对单个简单的图像进行分类[2]. 这样复杂的模型可能会轻易超过设备的计算和存储限制, 从而导致较长的响应延迟[4]. 因此, 针对深度模型压缩和推理延迟的降低已经引起了学术界和工业界的极大关注.

神经网络模型压缩是一种有效的方法, 压缩神经网络可以降低在设备上部署的深度模型的计算强度和存储需求[5]. 深度模型压缩方法可分为三类: 1) 参数调整和共享; 2) 紧凑结构设计; 3) 低秩分解[3]. 基于参数调整和共享的压缩方法主要通过删除深度模型中的冗余和非关键参数, 以减少运行深度模型的计算. 紧凑结构设计的方法将其注意力集中在参数有效的神经网络架构的构建上. 基于低秩分解的方法使用张量分解来估计深度模型的信息参数. 上述压缩方法可以降低计算复杂性, 从而可以在资源受限的设备上部署深度神经网络模型.

深度 AI 模型在设备上的部署策略显著影响这些模型的响应延迟. 在设备上部署神经网络模型的常规方法可以分为两类: 云端部署方法和设备端部署方法[6]. 对于云端部署方法, 设备将原始数据发送到云, 云端服务器执行 AI 模型并将推理结果发送回设备端. 这种方法需要将大量数据上传到云中, 因此通信开销较大[7]. 仅在设备上本地执行 AI 模型的设备端部署方法则可以大大减少通信延迟问题的出现. 但是, 由于有限的计算资源, 大多数设备仍无法执行连续且准确的推理[8]. 例如, 在移动设备上连续运行人脸检测会在 45 min 内耗尽设备电池的电量[9]. 此外, 绝大多数深度模型(如 CNN)的存储要求过高, 这极大地限制了深度模型在移动设备上的部署[6].

缩短设备上深层模型推理时间的有效解决方案是协同智能, 它可以通过使用终端设备与云之间的协作来最大限度地降低 AI 模型的总体响应延迟[10-11]. 在协同智能中, 用于推理的深度 AI 模型通常在选定的中间层被分为两部分. 所选层之前的神经网络层在设备上执行, 而其余层在云端服务器中执行. 本质上, 与传统的仅使用云的方法相比, 计算和原始数据生成源之间的物理接近性提供了多种好处, 例如低通信延迟、低能效、高隐私保护和低带宽消耗等. 与传统的设备端部署的方法相比, 在云中执行一些计算带来了更高的准确性和更短的计算等待时间[12].

研究表明, 目前轻量级压缩模型无法满足相关应用的严格精度和时序要求. 此外, 据我们所知, 目前的协同智能方法都没有考虑神经网络的压缩, 这仍然对在设备上部署 AI 模型提出了持续的挑战.

在本文中, 我们同时对神经网络压缩和协同智能进行了研究, 以降低在应用中部署深度神经网络所需的计算延迟和通信延迟. 所提出的方法克服了现有技术方法的缺点, 已有的这些方法要么忽略了云的强大计算能力, 要么由于将大量数据传输到云而导致过长的传输延迟. 本文的主要贡献概述如下:

(1) 针对应用提出了神经网络加速技术, 该技术使用低秩分解技术压缩深度神经网络模型, 从而可以生成更少输出的层; 基于知识蒸馏 (Knowledge Distillation, KD) 的参数调整方案用于恢复压缩所引起的精度损失并克服消失的梯度.

(2) 提出了一种基于协同智能的神经网络切割方式, 即将压缩和调整后的 CNN 模型分为两部分, 分别部署在终端设备和云中, 以实现整体最小延迟; 通过训练二次回归模型以找到压缩模型的最佳划分点.

(3) 在 VGG-19 模型上的实验结果表明, 与基准模型相比, 所提出的二次回归模型可以将实际测

量的均方根误差 (Root Mean Squard Error, RMSE) 至少降低 90%, 并且与 4 种基准测试方法相比, 所提出的方法可以降低总体延迟至少 70%.

本文的其余部分: 第 1 章简要回顾有关神经网络压缩和协同智能的相关工作; 第 2 章介绍用于快速协同智能的压缩和切割方法; 第 3 章介绍实验结果; 第 4 章总结论文.

1 相关研究

深度神经网络压缩已经投入了大量的研究工作. 该领域的主要工作可以分为以下不同的类别: 参数调整和共享, 紧凑的结构设计和低秩分解. 参数调整和共享压缩方法去除了 CNN 模型中多余和不重要的参数, 以减少所需的计算和存储量[5]. Luo 等[4] 设计了一个框架, 通过过滤级修剪在训练和推理过程中同时加速和压缩深层 CNN 模型. 所提出的框架从理论上将过滤器的修剪确定为一个优化问题, 然后识别并丢弃对于压缩 CNN 模型不是至关重要的过滤器. Ullrich 等[12] 采用经验贝叶斯来学习先验算法, 该先验算法用于对深度模型中的权重参数进行编码, 该方法可以在一个简单的再训练过程中实现量化和修剪. Lin 等[13] 提出了一种无标签的生成对抗性学习方法, 以修剪 CNN 模型中的诸如滤波器之类的异类冗余结构, 从而将特定结构的输出缩放为零. 但是, 以上参数调整和共享压缩方法需要专用的硬件设计来支持内存与或操作组件中的稀疏性设置和位级操作, 以最大限度地提高模型压缩的效率[14].

第二种加速方案是稀疏推理结构设计, 通过该设计方案可以构造参数量小的深度神经网络架构. 在文献 [15] 中, 作者设计了一个额外的循环稀疏连接层, 以通过自下而上的方法训练模型来减少全连接层中的参数数量. Wang 等[15] 设计了一种无冗余的体系结构, 该体系结构可检测并消除 CNN 中的重复计算和存储模式, 所提出的架构利用了一个常规的参数分析器和一个数据流加速器来减少重复的操作和存储空间. Iandola 等[16] 提出了 SqueezeNet, 这是一种神经体系结构类, 其中包含参数有效的全卷积模块, 以减少深度模型中的参数数量. 尽管这些紧凑的结构似乎是有效的, 但它们无法充分利用过滤器间的依赖性, 从而为进一步压缩留有空间[3].

第三种流行的神经网络压缩方法采用基于低秩分解的权重近似和信息理论原理来压缩神经网络参数. Sotoudeh 等[17] 在低秩逼近中增加了非均匀性, 并设计了一个框架来加速 AI 模型中的推理. Lin 等[18] 提出了一种基于分解的压缩方法, 以消除卷积核和 CNN 模型的全连接权重参数矩阵之间的冗余, 在不修改 CNN 结构的情况下, 以低秩近似的形式将噪声注入权重. 这使得注入的噪声能够进行局部最小值搜索.

使用以上 3 种方法压缩的 CNN 模型通常被部署在终端设备上或云中, 这样仍然会导致大量的计算或通信延迟. 为了缓解这种情况, 研究者提出了一种协同智能的概念来部署深度模型, 从而实现低推理延迟. Kang 等[10] 研究了仅云处理的现状方法, 并设计了一个调度程序以神经网络层的粒度去划分深度 AI 模型. 提出的策略利用云和终端设备联合处理来实现低延迟、低能耗和高数据中心吞吐量. 在文献 [19] 中, 作者提出了一种切割划分的设计准则, 该准则在云端分配并执行神经网络中的卷积层, 而在最终设备处分配了其余的全连接层. 他们提出了一种具有特征编码的切割推理方法, 其中设备将推理处理到网络的中间层, 并将输出特征传输到云中以进行网络其余部分的推理. Eshratifar 等[1] 通过引入一个负责减少需要上传到云的特征数据大小的单元, 提出了一种协同智能架构, 将该设计的单元放置在深度 CNN 模型的选定直接层之后, 提出了一种基于容器化切割的运行时自适应 CNN 加速框架, 该框架根据计算资源的可用性和网络条件动态选择最佳切割点进行切割.

上述利用本地终端设备的邻近性和强大的云计算能力的协作方法可以显著降低深度模型的推理延迟. 但是, 这些研究没有考虑深度模型压缩对推理延迟的影响. 在本文中, 我们对应用程序中的响应延迟进行了深度模型压缩和协同智能的联合研究.

2　压缩划分神经网络实现快速协同智能

2.1　介绍

如图 1 所示, 所提出的深度神经网络边云协同加速机制由两个关键组件组成: 模型压缩和模型切割. 在模型压缩过程中, 计算密集型卷积层通过使用基于过滤分解的技术进行压缩, 而通过使用基于奇异值分解 (Singular Value Decomposition, SVD) 的方法来压缩存储密集的全连接层. 然后, 使用基于知识蒸馏 (KD) 的方案对压缩的神经网络模型的参数进行微调, 以恢复压缩造成的精度损失并克服梯度消失问题.

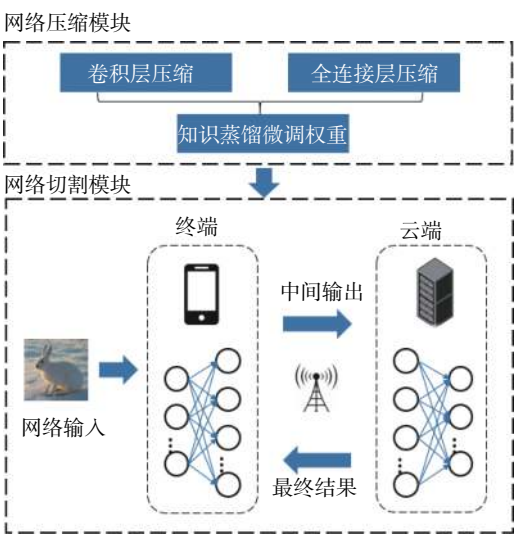

图 1　所提出的深度神经网络边云协同加速机制概览

Fig. 1　Overview of the proposed edge-cloud collaborative acceleration mechanism

在对神经网络模型进行压缩之后, 神经网络切割将压缩的模型分为两部分, 分别部署在终端设备和云中. 该切割方案通过训练的二次回归模型找到使总延迟最小的最佳划分点. 分割点实际上是模型压缩过程中的一个新生成的层.

总体而言, 所提出的方案首先在设备上执行压缩的模型, 直到选定的特定层, 然后将该层的输出传输到云中, 执行云中的其余层. 通过这种方式, 使得包括设备处理延迟、数据通信延迟和云处理延迟在内的总体延迟得以最小化.

2.2 节首先介绍了用于卷积层和全连接层的模型压缩方法, 接着是基于 KD 的参数调整方法. 2.3 节介绍了一种基于预测的模型划分方法, 该方法通过使用训练的预测模型来分割压缩后的神经网络模型.

2.2　基于低秩分解的模型压缩

所提出的基于低秩分解的模型压缩方案首先压缩卷积层和完全连接层, 然后使用基于 KD 的参数技术来调整压缩后的模型参数.

卷积层压缩: 一个典型的卷积层通过使用 N 个滤波器将输入张量 $\boldsymbol{I}$ 转换为输出张量 $\boldsymbol{O}$, 其中输入张量大小, 滤波器大小和输出大小分别为 $H\times W\times C$, $d\times d\times C\times N$ 和 $H'\times W'\times N$. 如图 2 所示, 该变换可描述为[20]

$$\boldsymbol{O}_n=\boldsymbol{F}_n*\boldsymbol{I}=\sum_{c=1}^{C}\boldsymbol{F}_n^c*\boldsymbol{I}_c. \tag{1}$$

其中,$*$代表卷积操作, $\boldsymbol{F}_n$ $(1 \leqslant n \leqslant N)$ 表示第 n 个 3-D 滤波器, 包含 C 个 2-D 滤波器. 第 c 个二维滤波器由 $\boldsymbol{F}_n^c \in \mathbb{R}$ $(1 \leqslant c \leqslant C)$ 表示. $\boldsymbol{I}_c$ 是第 c 个二维输入张量, $\boldsymbol{O}_n$ 则表示第 n 个输出张量.

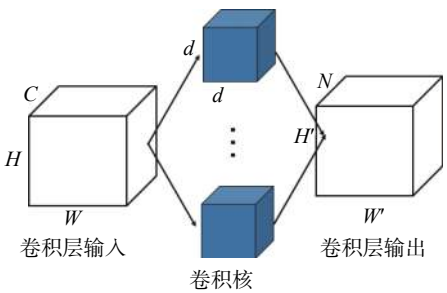

图 2 典型的卷积过程

Fig. 2 A typical convolution process

通过文献 [20] 中的基于滤波分解的方法来压缩卷积层. 首先式 (1) 中给出的卷积过程分为两个卷积, 以使中间特征图的大小显著减小. 第一卷积层具有大小为 $d \times 1 \times C$ 的 R 滤波器, 用 $\boldsymbol{\gamma}_r \in \mathbb{R}^{d \times 1 \times C}(1 \leqslant r \leqslant R)$ 表示, 如图 3 所示. 该卷积层产生中间特征图 $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{H' \times W \times R}$. 第二卷积层包含大小为 $1 \times d \times R$ 的 N 个滤波器, 用 $\boldsymbol{\Gamma}_n \in \mathbb{R}^{1 \times d \times R}(1 \leqslant n \leqslant N)$ 表示, 并生成输出特征图 $\boldsymbol{O} \in \mathbb{R}^{H' \times W' \times N}$.

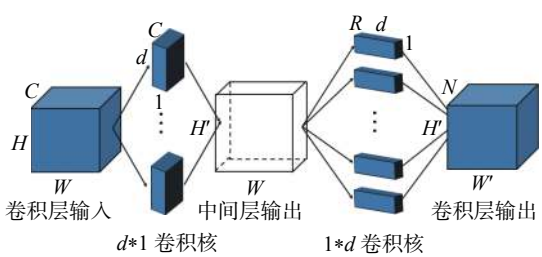

图 3 基于低秩分解的卷积过程

Fig. 3 Convolution process based on low-rank decomposition

然后, 压缩方案使用两个新的卷积层 $\boldsymbol{\gamma}_r$ 和 $\boldsymbol{\Gamma}_n$ 逼近式 (1) 中给出的卷积过程. 将 $\boldsymbol{\gamma}_r$ 和 $\boldsymbol{\Gamma}_n$ 分别代入式 (1), 得到的近似值为 $\boldsymbol{O}_n$:

$$\boldsymbol{O}_n \approx \boldsymbol{\Gamma}_n * \boldsymbol{\mathcal{S}} = \sum_{r=1}^{R} \boldsymbol{\Gamma}_n^r * \left(\sum_{c=1}^{C} \boldsymbol{\gamma}_r^c * \boldsymbol{I}_c \right) = \sum_{c=1}^{C} \left(\sum_{r=1}^{R} \boldsymbol{\Gamma}_n^r * \boldsymbol{\gamma}_r^c \right) * \boldsymbol{I}_c. \tag{2}$$

其中, $*$代表卷积操作, R 是给定的超参数, 用于控制新过滤器的等级. 通过解决以下优化问题, 可以获得两个新滤波器 $\boldsymbol{\Gamma}$ 和 $\boldsymbol{\gamma}$ 的参数:

$$\min_{\boldsymbol{\Gamma},\boldsymbol{\gamma}} L(\boldsymbol{\Gamma},\boldsymbol{\gamma}) = \sum_{n=1}^{N} \sum_{c=1}^{C} \left\| \boldsymbol{F}_n^c - \sum_{r=1}^{R} \boldsymbol{\Gamma}_n^r * \boldsymbol{\gamma}_r^c \right\|_F^2. \tag{3}$$

其中, $||\bullet||_F$ 表示 Frobenius 范数, 它定义为矩阵中元素的绝对平方和的平方根.

在不失一般性的前提下, 假设输入张量宽度 $W \gg d$ ($\gg$代表远大于) 且 $R = N = C$. 压缩方案的加速比 S_R 由式 (4) 计算:

$$S_R = \frac{d^2 CNH'W'}{R(CW + NW')dH} = \frac{dCNH'W'}{R(CW + NW')H}. \tag{4}$$

全连接层压缩: CNN 模型中的全连接层执行逐矩阵乘法[21], 即输入矩阵 $\boldsymbol{X} \in \mathbb{R}^{d\times b}$ 乘以权重矩阵 $\boldsymbol{W} \in \mathbb{R}^{h\times d}$ 来产生输出矩阵 $\boldsymbol{Z} \in \mathbb{R}^{h\times b}$:

$$\boldsymbol{Z} = \boldsymbol{W}\boldsymbol{X}. \tag{5}$$

所提出的基于奇异值分解 (SVD) 的方法使用此特性来减少全连接层所需要的存储空间. 该方法实质上通过使用低秩矩阵 $\widehat{\boldsymbol{W}} \in \mathbb{R}^{h\times d}$ 来近似权重矩阵 $\boldsymbol{W}$, 以减少每个完全连接层中的参数数量. 以这种方式, 全连接层被压缩, 公式为 (式 (6) 中 r 是一个给定的整数)

$$\min_{\widehat{\boldsymbol{W}}} \left\| \boldsymbol{W} - \widehat{\boldsymbol{W}} \right\|_F^2 \text{ , 其中 } \operatorname{rank}(\widehat{\boldsymbol{W}}) < r. \tag{6}$$

通过使用奇异值分解[22] 分两步来解决此优化问题. 在第一步中, 权重矩阵 $\boldsymbol{W}$ 根据以下公式得出:

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}. \tag{7}$$

其中, $\boldsymbol{U} \in \mathbb{R}^{h\times h}$ 是单一矩阵, $\boldsymbol{V}^{\mathrm{T}}$ 是矩阵 $\boldsymbol{V} \in \mathbb{R}^{d\times d}$ 的转置, $\boldsymbol{\Sigma}$ 是 $h \times d$ 矩阵, 对角元素均为正. 这些对角元素定义为矩阵 $\boldsymbol{W}$ 的奇异值. 请注意, 矩阵的共轭转置实际上是其逆. 即对单位矩阵 $\boldsymbol{E}$, 有 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} = \boldsymbol{E}$ 且 $\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{E}$. 在第二步中, 低阶矩阵 $\widehat{\boldsymbol{W}}$ 通过下式计算:

$$\widehat{\boldsymbol{W}} = \widehat{\boldsymbol{U}}_r \widehat{\boldsymbol{\Sigma}}_r \widehat{\boldsymbol{V}}_r^{\mathrm{T}}. \tag{8}$$

其中, $\widehat{\boldsymbol{U}}_r \in \mathbb{R}^{h\times r}$ 和 $\widehat{\boldsymbol{V}}_r \in \mathbb{R}^{h\times r}$ 是两个子矩阵, 分别包含 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 中的 r 个奇异向量. $\widehat{\boldsymbol{\Sigma}}_r$ 中的对角元素实际上是 $\boldsymbol{\Sigma}$ 中 $r(1 \leqslant r \leqslant R)$ 的最大奇异值.

可以看出, 基于 SVD 的压缩方法将完全连接层中的参数数量从 hd 减少到 $r(h + d)$. 相应的压缩比 C_r 可通过以下公式计算:

$$C_r = \frac{r(h+d)}{hd}. \tag{9}$$

基于知识蒸馏 (KD) 的参数调整: 将以上压缩方法直接应用于深度模型会导致每一层的近似误差增大, 这可能会导致压缩后的损失进一步累积和传播[23-24]. 在这项工作中, 采用 KD 来缓解这一问题. KD 旨在训练一个小型网络 (也称为学生网络), 以模仿一个预先训练的较大网络 (也称为教师网络). 该学生网络倾向于捕获全局和本地知识. 全局知识包含来自训练数据的真实标签所提供的信息, 以及教师网络最终输出的信息. 从教师网络的每一层的输出中学习本地知识, 以克服梯度消失问题. 我们重建了一个整体损失函数, 该函数考虑了全局和局部重建误差, 以降低整体近似误差.

首先, 基于 KD 的方法将学生网络 s 和教师网络 t 的 softmax 输出分别定义为 $\boldsymbol{q}_s = f(h(\boldsymbol{I}; \boldsymbol{W}_s))$ 和 $\boldsymbol{q}_t = f(g(\boldsymbol{I}; \boldsymbol{W}_t))$, f 代表 softmax 函数. h 和 g 是分别将输入图像 $\boldsymbol{I}$ 映射到学生和教师网络的输出的两个功能. $\boldsymbol{W}_s$ 和 $\boldsymbol{W}_t$ 分别代表学生和教师网络中的超参数. 期望学生网络输出 $\boldsymbol{q}_s$ 的概率分布不仅接近教师网络 ($\boldsymbol{q}_t$) 的概率分布, 而且接近真实标签 $\boldsymbol{l}$. 为此, 参数 τ 用于软化来自教师网络输出的信息, 因为软化的信息在训练中提供了更多信息. 相同的温度参数也被应用于学生网络的输出, 以生成软化的目标概率分布 $\boldsymbol{q}_s^{\tau}$. 两个网络的软化目标概率分布可通过式 (10) 得出:

$$\boldsymbol{q}_s^{\tau} = f\left(\frac{h(\boldsymbol{I}; \boldsymbol{W}_s)}{\tau}\right), \ \boldsymbol{q}_t^{\tau} = f\left(\frac{g(\boldsymbol{I}; \boldsymbol{W}_t)}{\tau}\right). \tag{10}$$

因此, 考虑了教师网络的真实标签和输出的全局损失函数被定义为

$$L_{\text{global}} = \lambda H(\boldsymbol{q}_s^{\tau}, \boldsymbol{q}_t^{\tau}) + H(\boldsymbol{l}, \boldsymbol{q}_s). \tag{11}$$

式 (11) 中的第一项是从教师网络的软化输出中学习学生网络, 第二项是指导学生网络从真实标签中学习. H 是指基于交叉熵的损失函数, λ 是用于平衡 $H(\boldsymbol{q}_s^\tau, \boldsymbol{q}_t^\tau)$ 和 $H(\boldsymbol{l}, \boldsymbol{q}_s)$ 的交叉熵的参数.

为了进一步提高压缩模型的精度并克服消失梯度, 还考虑了学生网络每一层与教师网络每一层之间的损失. 对于学生网络中的第 i 层, 本地损失函数定义为

$$L_{\text{local}}^i = \frac{1}{m_i} \left\| \boldsymbol{O}_s^i - \boldsymbol{O}_t^i \right\|_F^2. \tag{12}$$

其中, $\boldsymbol{O}_s^i = h(\boldsymbol{I}; \boldsymbol{W}_s^i) \in \mathbb{R}^{\boldsymbol{H}_i \times \boldsymbol{W}_i \times \boldsymbol{C}_i}$ 和 $\boldsymbol{O}_t^i = h(\boldsymbol{I}; \boldsymbol{W}_t^i) \in \mathbb{R}^{\boldsymbol{H}_i \times \boldsymbol{W}_i \times \boldsymbol{C}_i}$ 分别是学生和教师网络的第 i 层的输出张量. $\boldsymbol{W}_s^i$ 和 $\boldsymbol{W}_t^i$ 分别代表学生和教师网络中第 i 层的超参数.

考虑到上述本地和全局损失, 我们通过将总体损失函数最小化来训练学生网络:

$$L(\boldsymbol{W}_s) = L_{\text{global}} + \sum_{i=1}^{L} \lambda_i L_{\text{local}}^i = \lambda H(\boldsymbol{q}_s^\tau, \boldsymbol{q}_t^\tau) + H(\boldsymbol{l}, \boldsymbol{q}_s) + \sum_{i=1}^{M} \frac{\lambda_i}{m_i} \left\| \boldsymbol{O}_s^i - \boldsymbol{O}_t^i \right\|_F^2. \tag{13}$$

其中, M 表示学生或者教师网络中的层数. λ_i $(1 \leqslant i \leqslant L)$ 是一组惩罚参数, 用于平衡第 i 层的全局损失和局部损失.

2.3 基于预测的 CNN 划分

对 CNN 模型进行切割是一种有效的解决方案, 可进一步降低执行压缩深度模型的总体延迟. 不同的神经网络具有不同的最佳切割方案[11]. 如果切割模型在深层模型的前端执行 (即在云中部署了更多层), 则需要将大量数据上传到云中. 在这种情况下, 数据通信等待时间主导了总体等待时间[11]. 如果在模型的后面执行切割, 则可以减少数据通信开销. 但是, 由于在设备上执行了更多的层, 因此增大了处理延迟. 因此, 找到一个最佳的分割点以最小化总延迟至关重要. 本文所提出的训练回归模型以划分使用 2.2 节中的压缩方法获得的深度模型. 压缩将生成输出较小的新层, 非常适合用作协同智能的切割点. 下面, 我们首先介绍基于二次对数的预测模型, 然后介绍基于该预测的切割.

基于二次对数的预测模型: 为压缩的模型中的每种类型的层建立基于二次对数的预测模型, 以基于其可配置参数估计层的处理延迟. 本文所提出的方案从两个方面改进了文献 [11] 中的预测模型: 1) 本文所提出的方案中使用更细粒度的可配置参数来描述深度神经网络模型中的层, 以提高相应预测模型的准确性; 2) 本文所提出的方案中采用新的回归函数提升模型对非线性关系的拟合能力.

对于第一个方面, 文献 [11] 中的预测模型采用可配置的参数 (通道数) × (特征图数) × (特征图大小) 和 (滤波器大小/步幅)^2 × (滤波器数) 来描述卷积层, 其中 (滤波器大小/步幅)^2 × (滤波器数量) 表示输入特征图每个像素的计算量. 但是, 使用上述粗粒度可配置参数来描述卷积层配置与其处理延迟之间的关系还不够准确. 为了提高预测模型的准确性, 我们使用特征图的数量, 输入特征图的大小以及参数 (滤镜宽度/宽度跨度) × (滤镜高度/高度跨度) × (滤镜数量) 来模拟模型的卷积层. 对于池化层, 将使用输入特征图的大小以及输出特征图的数量来建立相应的预测模型. 选择输入神经元的数量和输出神经元的数量来对完全连接的层, softmax 层和 argmax 层进行建模. 对于激活层和归一化层, 选择神经元数量作为可配置参数. 本文所建议的预测模型的有效性在 3.3 节中进行了说明.

关于第二方面, 文献 [11] 中的预测模型采用对数和的线性函数作为其回归函数. 但是, 使用此回归函数的预测模型拟合非线性关系的效果不佳. 本文所提出的方案使用二次对数之和作为回归函数, 从而使预测模型拟合线性和非线性关系.

基于预测的划分: 算法 1 中显示了用于协同智能的所提出的总体算法流程. 该算法采用预训练的深度神经网络模型, 主成分分析 (Principal Component Analysis, PCA) 比率, 温度参数 τ, 当前无线网

络带宽 B 作为输入, 并输出加速后的深度模型. 其中 PCA 比率表示压缩水平, 较高的 PCA 比率表示较低的压缩水平. 较低的 PCA 比率表示较高的压缩水平, 这可能导致更大的精度损失.

本文所提出的方案的工作方式如下, 它首先使用基于过滤分解的压缩方法 (第 1 行) 在预训练的神经网络模型中压缩卷积层. 然后, 该算法使用基于 SVD 的压缩方法 (第 2 行) 压缩神经网络模型中的全连接层. 为了减少由于压缩的神经网络模型而导致的累积和传播错误, 同时克服消失的梯度, 通过基于 KD 的方法 (第 3 行) 调整压缩的神经网络模型中的参数. 之后, 使用细粒度的可配置参数和改进的回归函数 (第 4 行) 训练了一组预测模型. 令 M 为压缩的神经网络模型中的层数. 对于深度模型中的每个 $L_i\,(1 \leqslant i \leqslant M)$ 层, 通过使用第 5 到第 7 行中的相应预测模型来估计 L_i 的处理延迟. 在第 8 行中, 该算法通过以下方式获得了最优的分割点 P:

$$P = \underset{j=1,2,\cdots,M}{\arg\min} \left(\sum_{i=1}^{j} T^i_{\text{device}} + \sum_{k=j+1}^{M} T^k_{\text{cloud}} + T^j_{\text{trans}} \right). \tag{14}$$

其中, T^i_{device} 是在设备上执行压缩的 CNN 模型直到 L_i 层的等待时间. T^k_{cloud} 表示执行云中其余层 (从 L_i 后面的层到最终层) 的延迟. T^j_{trans} 表示无线数据通信延迟, 它取决于上载数据的大小和无线网络带宽. 注意, T^i_{device} 和 T^k_{cloud} 是通过使用相应的预测模型获得的, T^j_{trans} 通过式 (15) 计算获得:

$$T^j_{\text{trans}} = \frac{D_j}{B}. \tag{15}$$

其中, D_j 是 L_j 层输出的大小, 即要上传的数据的大小. B 表示无线网络带宽. 然后, 在第 9 行中根据所获得的最佳划分点 P 对压缩的模型进行划分. 最后, 返回已划分的神经网络模型以进行协作推理 (第 10 行).

算法 1　一种基于网络压缩与切割的深度神经网络边云协同加速机制

输入: 预训练的神经网络模型、PCA 比值、知识蒸馏温度参数 τ 、当前无线网络带宽 B

输出: 加速后的神经网络模型

1: 利用基于低秩分解的压缩方案压缩神经网络中的卷积层
2: 利用基于奇异值分解的压缩方案压缩神经网络中的全连接层
3: 利用基于知识蒸馏的方案微调压缩后的神经网络参数
4: 对压缩后的神经网络结构训练预测模型
5: **For** 在压缩模型中的每一层 L_i **do**
6: 使用相应的预测模型来估计 L_i 的处理延迟
7: **End for**
8: 用式 (14) 计算出分割点 P
9: 根据分割点 P 对压缩后的神经网络模型进行切割
10: **Return** 加速后的神经网络模型

3　评　　价

在本章中, 我们进行了一系列实验, 以验证所提出方案的有效性. 首先使用在 2.2 节中介绍的基于低秩分解的方法对预训练的 VGG-19 模型进行压缩和调整[4]. 然后, 通过将它们与文献 [11] 中介绍的

预测模型进行比较, 来验证 2.3 节中所给出的预测模型的性能. 最后, 根据所提出的预测模型对压缩的 CNN 模型进行切割, 以最大限度降低总延迟. 需要注意的是, 不失一般性地, 本文所提出的基于网络压缩与切割的深度模型边云协同加速机制可以被应用于很多常见的深度神经网络模型, 例如 AlexNet、LeNet、VGG-16 等.

3.1 实验设置

采用了在 ImageNet 数据库上预训练的 VGG-19 模型[4]. 预先训练的 VGG-19 模型取自 Caffe 模型, 并使用 Caffe 在 VGG-19 模型上进行了压缩. 实验是在 NVIDIA Jetson TK1 (作为设备端) 和 NVIDIA Tesla (作为云端) 上进行的. NVIDIA Jetson TK1 配备了 NVIDIA Kepler GPU, NVIDIA 4-Plus1 ARM Cortex-A15 CPU, 192 CUDA 内核, 2 GB × 16 DDR4 内存和 64 位 16 GB 4.51 eMMC 内存. NVIDIA Tesla 配备了 P100 GPU 和 16GB DDR4 内存.

为了调整压缩的 VGG-19 模型, 本文所提出的方案使用随机梯度下降 (Stochastic Gradient Desent,SGD) 求解器来降低式 (13) 中的总损耗函数计算值. 求解器的学习率被设置为 0.001. 温度参数 τ, 超参数 λ 和 λ_i $(1 \leqslant i \leqslant M)$ 分别设置为 1、0.003 和 0.000 5[20].

3.2 CNN 压缩结果

本文所提出的方案的压缩性能是通过使用加速比和压缩比来衡量的, 这可以分别由式 (4) 和式 (9) 得出. 表 1 显示了不同 PCA 比的压缩结果. 请注意, 在 VGG-19 中, 当 PCA 比设置为低于 0.7 时, 模型的精度非常低. 因此, 表 1 中不包括将 PCA 比设置为 0.7 以下时的压缩结果. 从表 1 可以看出, 当 PCA 比率设置为 0.7 时, 所提出的方案可以获得最佳的压缩结果. 因此, 在以下实验中, 我们将 PCA 比设置为 0.7 以压缩 VGG-19.

表 1 不同 PCA 比率的压缩结果

Tab. 1 Compression results with different PCA ratios

PCA比	加速比	压缩比/%
0.70	3.20	48.98
0.80	2.81	38.84
0.80	2.29	18.43

3.3 预测模型结果

图 4 在卷积层和全连接层的预测模型的准确性方面比较了本文所提出的方案和基准测试方法[11]. 请注意, 该图的纵轴为对数刻度, 横轴为真实延迟. 图 4 中的红线表示估计的延迟等于真实延迟. 即预测模型准确地估计了实际的延迟. 蓝点离红线越近, 预测模型的估计性能越好. 可以看出, 与基准预测模型相比, 我们的预测模型可以更准确地估计卷积层和完全连接层的延迟. 原因是本文所提出的方案采用了细粒度特征来对延迟进行建模. 另外, 我们将二次对数之和作为回归函数. 这些改进提高了本文所提出的预测模型的准确性, 以及预测模型拟合非线性关系的能力.

我们还为压缩的 VGG-19 模型中的其他层 (例如 argmax 和 softmax 层) 建立了预测模型. 由于页数的限制, 图 4 仅显示了卷积层和完全连接层的预测结果. 实验结果表明, 与文献 [11] 中的基准预测模型相比, 本文所提出的预测模型可以将延迟的均方根误差 (RMSE) 至少降低 90%.

本文所提出的方案通过使用我们的预测模型准确估计压缩的 VGG-19 中每一层的处理延迟. 然后, 在 2.3 节中采用基于预测的 CNN 切割, 以探索最佳切割点, 该切割将包括设备处理延迟, 数据通信延迟和云处理延迟在内的总体延迟降至最低.

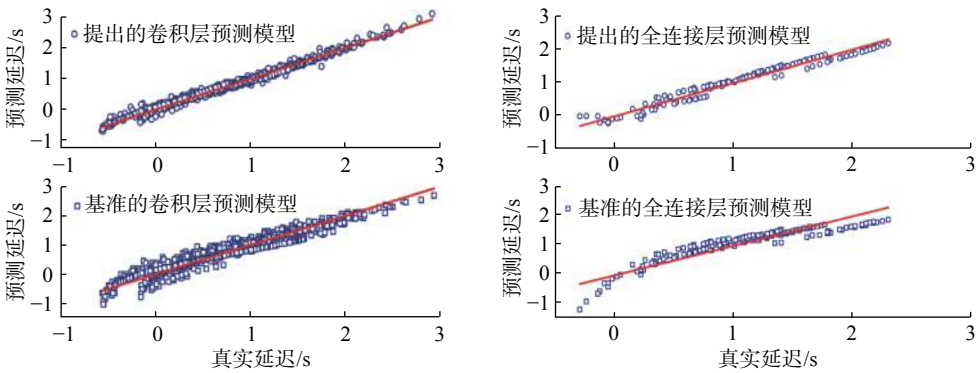

图 4　比较本文提出的预测模型和基准的预测模型

Fig. 4　Comparison of the proposed model against benchmarking prediction models

图 5 显示了压缩的 VGG-19 模型的预测总体延迟的半对数图. 每个条形图表示在每层之后划分 VGG-19 时的总体延迟. 图 5 中的第 1 个数据条表示在云中部署压缩的 VGG-19 模型的延迟. 在这种情况下, 设备会将原始数据上传到云, 然后云执行推理. 第 65 个数据条是在设备上处理整个压缩的 VGG-19 模型的延迟. 黑色五角星表示切割压缩后的 VGG-19 的最佳分割点. 所选最佳切割点之前的层部署在设备上, 而其余层部署在云中. 可以看出, 可以在黑色五角星所对应的深度模型的层次处对 VGG-19 模型进行划分, 实现最低的延迟.

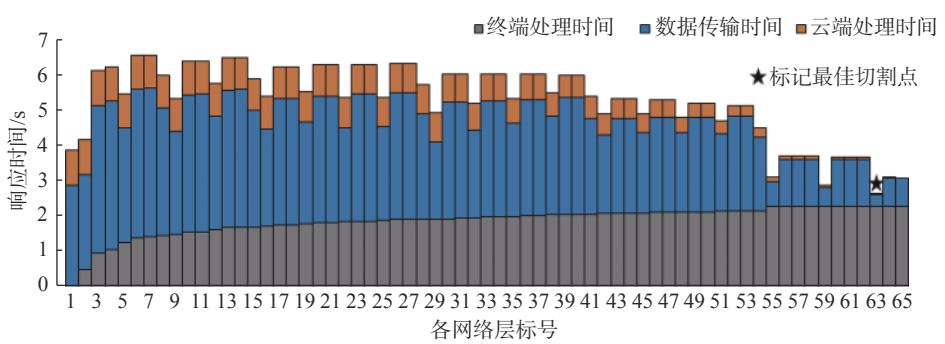

图 5　在不同切割点切割时压缩 VGG-19 的预测总延迟

Fig. 5　Predicted overall latency of the compressed VGG-19 model when the model is partitioned at different points

表 2 列出了 5 种方法的总体延迟, 根据 VGG-19 是否被压缩分为两种模式. 在仅云 (Cloud Only, CO) 方法中, 设备将原始数据发送到云, 然后云运行 VGG-19 进行推理并将结果发送回去. 在仅设备 (Device Only, DO) 方法中, 设备执行整个推断. 从表 2 中可以看出, 与未压缩模式下的 DO 方法相比, 压缩模式下的 DO 方法可以实现更低的延迟. 原因是压缩方法降低了神经网络模型的计算复杂度, 从而降低了 DO 方法的总体延迟. 由于云的强计算能力, 通信时延主导了 CO 方法的总延迟, 并且 2 种 CO 方法的总等待时间很接近. 实验结果还表明, 与 4 种基准测试方法相比, 本文所提出的方案可以将总体延迟至少降低 70%. 表 2 为具体的实验结果.

4　贡　　献

本文提出了一种基于网络压缩与切割的深度神经网络模型边云协同加速机制, 该机制通过压缩和切割深层神经网络模型以在应用中实现快速协同智能. 具体来说, 我们所提出的深度神经网络边云协同加速机制首先利用基于低秩分解和 SVD 的技术来分别压缩卷积层和完全连接层, 从而显著降低深度模型的复杂性和对存储空间的要求. 为了降低压缩引起的累积误差和消失梯度, 采用基于知识蒸

馏 (KD) 的方案对压缩后的模型参数进行微调. 然后将压缩的神经网络模型分为两部分, 分别部署在终端设备和云中. 采用二次回归函数对压缩后的神经网络层的延迟进行建模, 在此基础上找到最佳分割点. 在 VGG-19 深层模型上的实验结果表明, 与基准测试方法相比, 本文所提出的方案可以将总延迟至少降低 70%.

表 2 总延迟与 4 种基准测试方法比较

Tab. 2 Comparison of the proposed method against four benchmarking methods in terms of overall latency

模式	方法	总时延/ms
压缩后的	**本文所提出的方案**	**200.74**
	CO	742.80
	DO	223.30
未压缩的	CO	747.32
	DO	739.18

[参 考 文 献]

[1] ESHRATIFAR A E, ESMAILI A, PEDRAM M. Towards collaborative intelligence friendly architectures for deep learning [C]// The 20th International Symposium on Quality Electronic Design (ISQED). 2019: 14-19.

[2] DUBEY A, CHATTERJEE M, AHUJA N. Coreset-based neural network compression [C]// European Conference on Computer Vision. 2018: 454-470.

[3] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition [EB/OL]. (2015-04-10)[2021-01-04].https://arxiv.org/pdf/1409.1556.pdf.

[4] LUO J H, WU J, LIN W. ThiNet: A filter level pruning method for deep neural network compression [C]// 2017 IEEE International Conference on Computer Vision. IEEE, 2017: 5068-5076.

[5] FU S, LI Z, LIU K, et al. Model compression for IoT applications in industry 4.0 via multi-scale knowledge transfer [J]. IEEE Transactions on Industrial Informatics, 2020(9): 6013-6022.

[6] SODHRO A H, PIRBHULAL S, ALBUQUERQUE V. Artificial intelligence-driven mechanism for edge computing-based industrial applications [J]. IEEE Transactions on Industrial Informatics, 2019(7): 4235-4243.

[7] LÜ L, BEZDEK J C, HE X L, et al. Fog-embedded deep learning for the internet of things [J]. IEEE Transactions on Industrial Informatics, 2019(7): 4206-4215.

[8] WANG T, LUO H, JIA W, et al. MTES: An intelligent trust evaluation scheme in sensor-cloud-enabled industrial internet of things [J]. IEEE Transactions on Industrial Informatics, 2020, 16(3): 2054-2062.

[9] LIKAMWA R, WANG Z, CARROLL A, et al. Draining our glass: An energy and heat characterization of google glass [C]// Proceedings of 5th Asia-Pacific Workshop on Systems. 2014. DOI: 10.1145/2637166.2637230.

[10] KANG Y, HAUSWALD J, CAO G, et al. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge [J]. ACM SIGPLAN Notices, 2017, 52(1): 615-629.

[11] ZHOU Z, CHEN X, LI E, et al. Edge intelligence: Paving the last mile of artificial intelligence with edge computing [J]. Proceedings of the IEEE, 2019, 107(8): 1738-1762.

[12] ULLRICH K, MEEDS E, WELLING M. Soft weight-sharing for neural network compression [EB/OL]. (2017–05–09) [2020–03–01]. https://arxiv.org/pdf/1702.04008.pdf.

[13] LIN S, JI R, YAN C, et al. Towards optimal structured CNN pruning via generative adversarial learning [C]// Conference on Computer Vision and Pattern Recognition. 2019: 2790-2799.

[14] HOSSEINI M. On the complexity reduction of dense layers from $O(N^2)$ to $O(N\log N)$ with cyclic sparsely connected layers [D]. Baltimore County, Maryland: University of Maryland, 2019.

[15] WANG Y, LIANG S W, LI H W , et al. A none-sparse inference accelerator that distills and reuses the computation redundancy in CNNs [C]// Proceedings of the 56th Annual Design Automation Conference. 2019. DOI: 10.1145/3316781.3317749.

[16] IANDOLA F N, HAN S, MOSKEWICZ M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size [EB/OL]. (2016–11–04) [2020–03–15]. https://arxiv.org/abs/1602.07360.

[17] SOTOUDEH M, BAGHSORKHI S S. C3-Flow: Compute compression co-design flow for deep neural networks [C]// Proceedings of the 56th Annual Design Automation Conference. 2019: Article No.86.

[18] LIN S, JI R, CHEN C, et al. Holistic CNN compression via low-rank decomposition with knowledge transfer [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, 41(12): 2889-2905.

[19] KO J H, NA T, AMIR M F, et al. Edge-Host partitioning of deep neural networks with feature space encoding for resource-constrained Internet-of-Things platforms [C]// 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance. IEEE, 2018. DOI: 10.1109/AVSS.2018.8639121.

[20] LI P, CHEN Z, YANG L, et al, Deep convolutional computation model for feature learning on big data in internet of things [J]. IEEE Transactions on Industrial Informatics, 2018, 14(2): 790-798.

[21] CHEN Z, GRYLLIAS K, LI W. Intelligent fault diagnosis for rotary machinery using transferable convolutional neural network [J]. IEEE Transactions on Industrial Informatics, 2020, 16(1): 339-349.

[22] ZHOU J H, PANG C K, LEWIS F L, et al. Intelligent diagnosis and prognosis of tool wear using dominant feature identification [J]. IEEE Transactions on Industrial Informatics, 2009, 5(4): 454-464.

[23] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network [EB/OL]. (2015-03-09) [2020-01-10]. https://arxiv.org/abs/1503.02531.

[24] ZHANG Y, HONG G S, YE D, et al. Powder-bed fusion process monitoring by machine vision with hybrid convolutional neural networks [J]. IEEE Transactions on Industrial Informatics, 2020, 16(9): 5769-5779.

(责任编辑: 陈丽贞)