# 面向边缘设备的高能效深度学习任务调度策略

任 杰[1) ] 高 岭[2),3)] 于佳龙[2)] 袁 璐[2)]

[1)](陕西师范大学计算机科学学院 西安 710119)

[2)](西北大学信息科学与技术学院 西安 710127)

[3)](西安工程大学计算机科学学院 西安 710600)

**摘 要** 近年来,深度学习在图像和自然语言处理等诸多领域表现出色,与深度学习相关的各类移动应用发展迅速,但由于移动网络状态的不稳定性及网络带宽的限制,基于云计算的深度模型任务可能出现较大响应延迟,严重影响用户体验.与此同时,深度模型对设备的计算及存储能力有较高的要求,无法直接在资源受限的移动设备中进行部署.因此,亟须设计一种新的计算模式,使得基于深度模型的移动应用能够满足用户对快速响应、低能耗及高准确率的期望.本文提出一种面向边缘设备的深度模型分类任务调度策略,该策略通过协同移动设备与边缘服务器,充分利用智能移动终端的便捷性和边缘服务器强大的计算能力,综合考虑分类任务的复杂度和用户期望,完成深度模型在移动设备和边缘服务器中的动态部署,并对推理任务进行动态调度,从而提升任务执行效率,降低深度学习模型推理开销.本文以基于卷积神经网络的图像识别应用为例,实验结果表明,在移动环境中,相比于准确率最高的深度模型,本文提出的高能效调度策略的推理能耗可降低 93.2%、推理时间降低 91.6%,同时准确率提升 3.88%.

**关键词** 深度学习模型;边缘设备;任务调度策略;能效优化

**中图法分类号** TP18 **DOI 号** 10.11897/SP.J.1016.2020.00440

# Energy-Efficient Deep Learning Task Scheduling Strategy for Edge Device

REN Jie[1)] GAO Ling[2),3)] YU Jia-Long[2)] YUAN Lu[2)]

[1)](*School of Computer Science, Shaanxi Normal University, Xi'an* 710119)

[2)](*School of Information and Technology, Northwest University, Xi'an* 710127)

[3)](*School of Computer Science, Xi'an Polytechnic University, Xi'an* 710600)

**Abstract** The deep neural network has made significant progress in many fields. Its powerful computing ability makes it an efficient tool to solve complex problems, and has been widely used in automatic driving, face recognition, and augmented reality. Due to the outstanding performance of deep learning in the fields of image recognition and natural language processing, applying the deep learning model on mobile application is inevitable. Typically, the deep learning model relies on high-performance servers equipped with strong computing processors and large storage. However, because of the unstable mobile networks and limited bandwidth, running deep learning on the cloud server may cause a response delay, which violates the quality of user experience, and running the inference task on the cloud also has the privacy problem. At the same time, the researcher tries to execute the inference task on the user's own device, mainly focus on the on-device deep learning by using model compression techniques and develop the light-weight deep model, and all of them will sacrifice the model accuracy. Because of the limited resources of the mobile terminal

收稿日期:2019-05-06;在线出版日期:2019-11-01.本课题得到国家自然科学基金(61902229,61872294,61602290)、陕西省自然科学基础研究计划(2019JQ-271)、中央高校基本科研业务专项资金(GK201803063)资助.**任 杰**,博士,讲师,中国计算机学会(CCF)会员,主要研究领域为移动计算、机器学习和性能优化.E-mail:renjie@snnu.edu.cn.**高 岭**(通信作者),博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为网络管理、嵌入式系统.E-mail:gl@nwu.edu.cn.**于佳龙**,硕士研究生,主要研究方向为深度学习模型优化.**袁 璐**,硕士研究生,主要研究方向为移动计算.

(computing power, storage size, and battery capacity), the mobile device cannot satisfy the DNN model. We need to design a new computing paradigm so that the Deep Neural Network (DNN) based model can meet the user's expectations for fast response, low energy consumption, and high accuracy. This paper proposes a novel scheduling strategy, Edge-based strategy, for deep learning inference tasks by using edge devices. The Edge-based strategy combines the mobility of the user's mobile device with the powerful computing processors on edge server. Firstly, the strategy selects and deploys the appropriate DNN models by considering the inference time and accuracy. Specifically, the Edge-based strategy evaluates the candidate deep models on user mobile devices, and record the inference time and failure classification samples, the inference time is the first priority on mobile devices, then the strategy deploy the deep model with the least inference time on mobile devices, and input the failure sample to the other deep models and select the model with highest accuracy and deploy it on the edge device. After deploying the model on both devices, Edge-based strategy focuses on how to schedule the inference task between two devices to achieve the best performance. The core of task scheduling is the pre-trained classification model, it takes account of the input data complexity, and user expectations and schedule the inference task dynamically. This paper compares four typical machine learning techniques to train the classification model, and the random forest gives the highest accuracy. This paper takes the image recognition application as an example, and evaluate 12 popular CNN models on RaspberryPi 3B+, Jetson TX2 respectively, the experimental results show that in the mobile network environment, Edge-based strategy can effectively improve the performance of the deep model and reducing the overhead of inference, our approach outperforms the model with the highest accuracy by 93.2%, 91.6%, and 3.88% for energy consumption, inference time and accuracy.

**Keywords**　deep learning model; edge devices; task scheduling strategy; energy efficiency

## 1　引　言

深度神经网络(Deep Neural Networks,DNN)在抽取变体因子和解构流形数据上的出色表现使其成为一种解决复杂问题的有效工具,并在自动驾驶[1]、人脸识别及生物医学[2]等诸多应用领域取得了巨大的成功.

现如今,研究人员不断改进深度神经网络结构提升任务准确率的同时,网络的深度和参数量也在成倍增长.如图 1 所示,2012 年至 2015 年,深度模型在图像识别领域的错误率由 16%[3] 降到 3.5%[4],但模型规模增长 16 倍.在语音识别领域,Deep Speech[5] 识别精度已经达到 95%以上,而模型训练操作数增长近 10 倍.深度神经网络出色表现的背后是高性能计算设备及大量存储资源的支持,由此,深度学习模型一般部署在云端进行任务推理.但由于网络状态的不稳定性及网络带宽的限制,简单的将深度模型部署在云端的计算模式无法保证移动用户对低延迟响应的期望,特别是面向移动设备的医疗健康和视频监控等对实时性要求较高的关键任务,高延迟不仅严重影响用户体验,甚至可能带来经济和生命财产的损失.

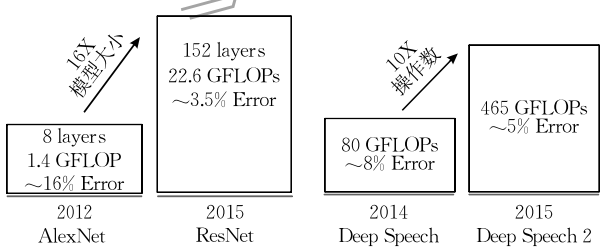


图 1　基于 DNN 的图像识别及语言识别模型

随着嵌入式系统的发展,深度学习在移动设备中的应用研究受到广泛关注,将高性能的深度学习模型部署到移动设备中(如无人机、智能手机等),将会在抢险救灾、快速智能识别等领域取得质的飞跃,从而为军事和民用带来新的发展机遇[6].而受限于智能移动终端的计算资源及存储资源,在移动端执行深度模型推理任务往往伴随着高延迟、高能耗等问题[7].为此,研究人员设计轻量级模型[8-9]或利用裁剪[10]、量化[11]等模型压缩技术,通过减少模型参

数的数量来降低模型对计算及存储资源的需求，这些方法大幅降低了模型规模和推理开销，但也对模型的精度带来了不同程度的损失.除此之外，由于应用程序上下文的动态性和不可预测性，无法通过人工主观判断选择适于当前任务的最优模型.

针对上述问题，本文根据用户对任务精度需求的差异性及当前任务特征，将 DNN 模型动态部署在移动端和边缘端，在保证用户体验的同时，降低推理开销.以图像识别为例，当图像光照条件良好、图片内容结构简单时，用户可选择部署在移动端的轻量级深度模型进行任务推理，而复杂的图片须采用部署在边缘设备中的高性能神经网络模型进行任务推理.此外，根据不同用户对任务精度需求的差异性，选择合适的模型进行推理，可以进一步降低模型推理开销.由此本文考虑，如何根据任务类型及用户需求提供满足用户需求的模型.

本文针对图像识别领域卷积神经网络(Convolutional Neural Networks，CNN)模型在嵌入式系统中的能效问题展开研究，将移动端的便捷性同边缘服务器强大的计算能力相结合，通过对 12 个典型 CNN 模型的准确度、推理时间及能耗进行分析，研究不同模型的适用范围，提出一种面向边缘设备的高能效图像分类任务调度策略.该策略综合考虑不同分类任务的复杂度和用户期望，分别在移动端和边缘服务器选择部署符合用户期望(高准确率、低功耗、快速响应)的 CNN 模型，并构建预分类模型判断当前任务复杂度，根据图像特征，输出当前图像分类任务的调度方案.并将该调度策略在 IMAGENET ILVRC2012[12] 验证集(50 K)中进行实验验证.实验结果表明，在图像分类任务中，相比于在移动端直接部署 CNN 模型进行推理，本文提出的基于边缘设备的图像分类任务调度策略比准确度最高的深度模型 Inception_V4 能耗降低了 93.2%，推理时间加速 10.8 倍，准确率提升至 83.5%.

## 2 相关工作

近年来，工业界和学术界一直探索如何将深度模型同移动设备相结合，以提供低延迟的 DNN 服务.但由于高精度的 DNN 模型对设备的计算及存储资源有较高要求，而移动设备的计算、存储及电量资源有限，导致高性能的 DNN 模型难以直接在移动设备场景中(如智能摄像机、智能手机、可穿戴设备等)进行部署.

目前，研究人员主要通过构造轻量级模型[8-9,13-14]或使用模型压缩方法[10-11,15]来降低 DNN 模型在移动端的计算开销，如 Google 团队使用深度可分离卷积技术开发了适用于移动端的 MobileNet[16]，通过为每一个通道分配卷积核，将结果进行逐点卷积，以生成新的特征图，从而大幅降低了模型参数量及运行开销.在模型压缩方面，主流压缩技术包含模型剪裁、稀疏化和量化等.Han 等人[17-18] 提出 Deep Compression 方法通过对模型压缩技术的综合应用降低执行开销，主要包含权值稀疏、权值共享和权值编码实现进一步的压缩，实验结果显示 Deep Compression 可以在 Alexnet 模型上达到 35 倍的压缩比，但没有优化推理时间，同时 Deep Compression 只对两种较为简单的网络结构进行了实验，对于结构复杂且精度较高的神经网络是否有效没有做出验证.BinaryConnect[11] 通过在神经网络训练过程中使用 1 bit 权值替代网络的浮点数权值，大幅减小模型的存储空间.Boureau 等人[19] 通过将部分网络参数置零即权值稀疏方法对网络参数进行修剪.XNOR-Net[20]，TWN[21] 等网络通过使用较少比特数值来减少网络参数和存储空间，或通过 $K$ 均值法对参数值进行压缩[22-23].此外，Gupta 等人[24] 将全精度浮点数固化到 16 bit 进行存储，同时采用随机约束技术对模型进行训练，实现网络存储空间的缩减.上述研究均通过参数共享方法来降低网络存储空间，但都仅限于单独网络的设计或使用，其实用性和具体性能表现依旧有待探索.在模型裁剪方面，Li 等人[25] 提出了基于量级的裁剪方式，通过权重绝对值的大小评判其重要性，从而裁剪重要性较低的 filter，该方法可有效的降低模型的复杂度，且不会给模型的性能带来明显损失.Yang 等人[26] 提出了基于能耗的剪裁方案，对模型网络结构各层能耗进行测量并排序，对能耗较大的层根据权值大小进行剪裁.然而上述在压缩或训练轻量级神经网络方面的研究，均以牺牲预测精度为代价来降低深度模型运行开销.除此之外，上述方案无法根据当前用户需求及任务场景在移动设备中自动选择部署合适的模型，而由于应用程序上下文(例如模型输入)环境的动态性和不可预测性，无法通过人工主观判断选择最优执行模型，由此需要一种自适应算法，能够根据当前环境、用户需求、任务类型在不同的设备中部署合适的深度模型，以提供高可用低延迟的 DNN 服务.

除了对模型本身进行精简或压缩处理，另一方面，研究人员通过将深度模型计算负载卸载到云端以加速 DNN 模型推理，Teerapittayanon 等人[27] 提

出了分布式深度神经网络，通过考虑用户对能耗和精度的动态需求将深度神经网络拆分为本地部署和云端部署两部分，然而该方法依然无法解决由于不稳定的网络环境状态导致的响应延迟问题，此外，云端部署 DNN 模型会对用户隐私安全带来一定的隐患[28].

本文提出的面向边缘服务器的高能效深度模型任务调度策略，在移动端部署满足用户精度需求的轻量级模型，该模型体积较小并能够提供快速响应服务，同时利用用户近端的边缘服务器，部署高精度的 DNN 模型. 本地部署可以极大降低由于不稳定的网络状态导致的高响应延迟，针对较复杂的推理任务，动态发送到部署在近端边缘服务器上的高性能 DNN 模型，保证分类准确率的同时也更易于保护用户隐私安全.

## 3　深度模型在移动端的能效问题

本文以深度学习模型在图像识别领域的应用为例，将网络结构较复杂的 Inception_V4 和轻量级模型 MobileNet_V1 分别部署在移动平台 NVIDIA Jetson TX2 和 RaspberryPi 3B＋上执行图像识别任务，并记录推理时间、能耗及准确率.

所有模型均基于 IMAGENET ILVRC2012 训练集进行构建，测试集为 IMAGENET ILVRC2012 验证集(50000 张图片). 分类任务分别输入 MobileNet_V1 和 Inception_V4，输出分类标签置信度值列表(降序排列)，该值表示模型将图片物体识别为不同对象的置信度，队列顶端标签表示识别为该对象的置信度值最高. 本文基于 ImageNet 挑战赛评判标准，选用排列第一(Top-1)和前五(Top-5)的置信度进行模型评估. 具体来说，对于 Top-1，检查置信度列表顶端标签是否与对象实际标签匹配，对于 Top-5，检查对象的实际标签是否位于模型输出置信度值前 5 的对象标签中.

图 2～图 4 为 MobileNet_V1 和 Inception_V4 分别在 RaspberryPi 3B＋和 Jetson TX2 平台上的平均推理时间(图 2)、平均推理能耗(图 3)及平均准确率(图 4). 由图可知，在 IMAGENET ILVRC2012 的验证集中，MobileNet_V1 单张图片平均推理时间为 268.8 ms(RaspberryPi：519.63 ms，TX2：18 ms)，平均推理能耗为 0.32 J(RaspberryPi：0.49 J，TX2：0.14 J)，Top-1 的准确率为 70.74%. Inception_V4 平均推理时间 3393.58 ms(RaspberryPi：6680 ms，TX2：107.18 ms)，平均推理能耗为 4.5 J(RaspberryPi：6.66 J，TX2：2.4 J)，Top-1 准确率为 80.18%. 由此可知，对于网络结构简单的轻量级 MobileNet_V1 来说，单张图片平均推理所需时间比 Inception_V4 降低 92.1%，推理能耗节约 92.9%，但 MobileNet_V1 Top-1 准确率比 Inception_V4 低 9.44%. 除此之外，深度模型在高性能开发板 Jetson TX2 上的整体性能表现更优，相比于 RaspberryPi 3B＋，Jetson TX2，平均推理时间节约 98.2%，能耗节约 64.4%.

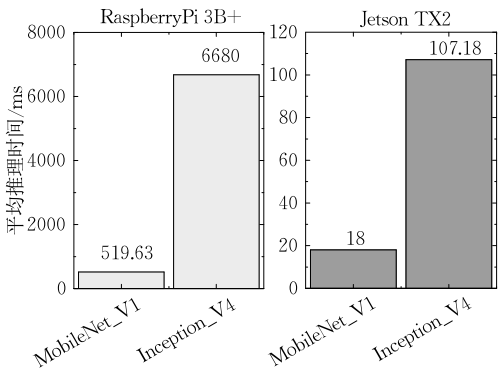


图 2　MobileNet_V1 和 Inception_V4 分别在 RaspberryPi 3B＋，Jetson TX2 的平均推理时间

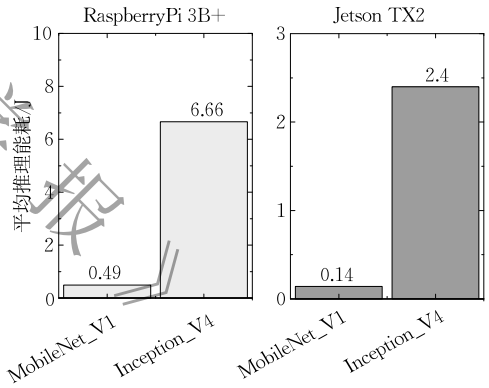


图 3　MobileNet_V1 和 Inception_V4 分别在 RaspberryPi 3B＋，Jetson TX2 的平均推理能耗

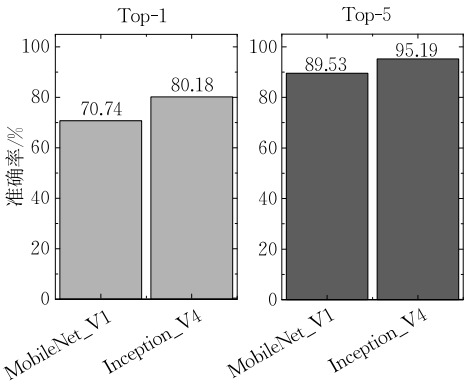


图 4　MobileNet_V1 和 Inception_V4 分别在 RaspberryPi 3B＋，Jetson TX2 的平均准确率

进一步分析实验结果可知，IMAGENET ILVRC-2012 验证集中有 67.2%的图片(场景简单的图片，如图 5(a)所示)在 Inception_V4 和 MobileNet_V1 中均可获得准确的推理结果，另有 12.96%图片(场景复杂的图片，如图 5(b)所示)只在 Inception_V4 中获得了准确的推理结果. 由此，本文考虑是否可以利用任务复杂度的不同，以及用户对性能需求的差异性，在不同性能的设备中部署不同类型的深度学习模型，并动态调度推理任务到合适的模型中进行分类. 例如将 MobileNet_V1 的便捷性同 Inception_V4 的高准确率相结合，基于当前使用场景及图片的复杂度，将高准确率的 Inception_V4 部署在高性能设备 TX2 中，并将复杂图片识别任务调度到 TX2 上执行，此外，当用户所处场景网络状态不佳时，也可利用本地部署的 MobileNet_V1 进行任务推理，最终实现在保证响应速度的同时，尽可能的降低推理能耗、提升准确率的目标.

(a) 麦穗鸟 (b) 巴亚韦弗鸟

图 5 针对不同场景的图像物体分类任务

# 4 基于边缘设备的高能效深度学习任务调度策略

## 4.1 概述

本文结合轻量级模型和高准确率模型各自的优势，提出一种基于边缘设备的高能效深度学习任务调度策略(简称 Edge-based). Edge-based 通过在移动端部署轻量级低开销的深度学习模型，提供低延迟响应，同时在边缘服务器端部署高准确率但开销较大的深度模型，处理复杂场景的图像分类任务，以保证任务的准确率. 该策略旨在保证分类准确率的同时，尽可能的降低推理时间和能耗. 以图像分类任务为例，流程如图 6 所示.

① 移动端应用输入图片分类任务；

② 抽取表示图片复杂度的典型特征，输入到本地预分类模型；

③ 预分类模型根据图片复杂度输出适合当前任务的 CNN 模型标签，并将分类任务调度至边缘服务器或在本地进行推理；

④ 从本地模型/边缘服务器模型返回分类结果.

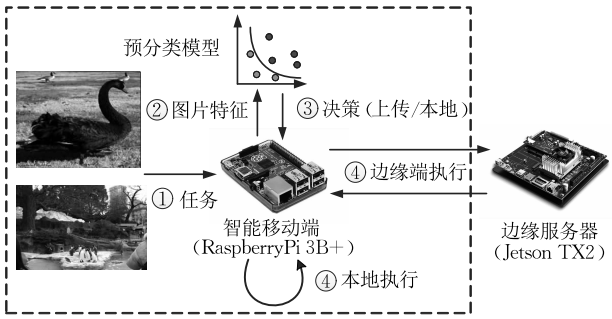


图 6 基于边缘设备的图像分类任务调度策略流程

## 4.2 模型选择及部署方案

本文针对图像分类领域典型的 12 个深度学习模型进行研究分析(如表 1 所示). 其中，Inception Google Net 考虑 3 种结构模型，Inception V1 基于 Inception Module 提高参数利用率. Inception V2 进一步采用 3×3 的小卷积核替代 5×5 的卷积核，降低参数量，此外还通过 Batch Normalization 方法提升网络训练速度，提高收敛后的分类准确率. 随后的 Inception V4 进一步结合 ResNet 以提升准确率. 在残差结构中，本文考虑的 ResNet-50，ResNet-101 以及 ResNet-152 均为三层残差结构，其主要区别是 conv3_x 以及 conv4_x 下的残差结构个数不同，其中，ResNet-50 在 conv3_x 和 conv4_x 下的残差结构个数分别为 4 和 6，ResNet-101 为 4 和 23，ResNet-152 为 8 和 36，该网络结构通过不断加深层数以实现模型精度的提升. 与此同时，本文考虑了轻量级模型 MobileNet 和 ShuffleNet. MobileNet 由 Horward 等人在 2017 年提出，它在计算量和模型尺寸方面具备明显优势，其架构由一个作用于输入图像的标准卷积层、一个深度可分离卷积堆栈以及最后的平均池和全连接层组成. MobileNet 采用深度可分离的卷积来构建轻量级深层神经网络，可将标准卷积分解成一个深度卷积和一个点卷积(1×1 卷积核). 深度卷积将每个卷积核应用到每一个通道，而 1×1 卷积用来组合通道卷积的输出，这种分解可以有效减少计算量，降低模型大小. 2018 年 Google 公开的 MobileNet_V2[29] 通过引入残差结构和 bottleneck 层，使其更加高效. 进一步，Ma 等人[9] 指出当前以 FLOPs 评估模型性能的不合理性，提出以网络实际内存消耗成本为度量标准，设计了更加高效的 ShuffleNet_V2.

**表 1　12 个 CNN 模型的参数量及层数**

| 模型名称 | 参数量/M | 层数 |
|---|---|---|
| Inception_V1 | 7.0 | 22 |
| Inception_V2 | 11.3 | 32 |
| Inception_V4 | 25.6 | 58 |
| ResNet_50(V1,V2) | 25.5 | 50 |
| ResNet_101(V1,V2) | 51.0 | 101 |
| ResNet_152(V1,V2) | 76.5 | 152 |
| MobileNet_V1 | 4.2 | 28 |
| MobileNet_V2 | 3.5 | 55 |
| ShuffleNet_V2 | 3.4 | 50 |

以上模型分别从 TensorFlow-Slim library① 和 TensorPack② 获得，并通过 ImageNet ILSVRC 2012 训练集进行训练. 本文以移动端最低推理开销、边缘服务器端最高精确度为目标，设计模型选择算法，具体模型选择过程如算法 1、2 所示.

**算法 1.**　本地模型选择算法.

输入：训练集(*Data*)，可选模型列表(*ModelList*)
*Min_Inference_Time*=*INT_MAX*;
*Local_Model*=NULL;
FOR *model* IN *ModelList*
DO
　*Time*=*Get_Inference_Time*(*model*,*Data*)
　#获取当前模型推理时间
　IF *Time* <*Min_Inference_Time*
　　*LocalModel*=*model*;
　　*Min_Inference_Time*=*Time*
　EDN IF
DONE
输出：本地部署模型(*LocalModel*)

**算法 2.**　边缘服务器模型选择算法.

输入：训练集(*Data*)，可选模型列表(*ModelList*)，算法 1 选中的模型(*LocalModel*)
*Predict_Results*=*LocalModel*(*Data*);
#本地模型推理结果
*Fail_Cases*=*Data*[*Predict_Results*!=*Real_label*]
#获取本地模型分类失败的数据
*ServerModel*=NULL;
*CurAcc*=0;
*TopAcc*=0;
FOR *model* IN *ModelList*
#其他模型在 *LocalModel* 分类失败数据上的准确率
DO
　*CurAcc*=*Get_Accuarcy*(*FailCases*,*model*)
　IF *CurAcc*>*TopAcc*
　　*ServerModel*=*model*;
　EDN IF
DONE
输出：服务器部署模型(*ServerModel*)

综上，基于深度学习的图像分类模型选择部署过程如算法 1、2 所示，将推理开销最低的 CNN 模型部署在移动端，并将该模型分类失败的任务利用其他候选模型进行推理，选择准确率最高的模型部署在边缘服务器中.

### 4.3　预分类模型

面向边缘设备的高能效深度学习任务调度策略的核心为预分类模型(如图 6 所示)，该模型根据当前分类任务特征(见 4.4 节)，将图像分类任务调度到合适的 CNN 模型(本地/边缘)进行推理. 本节就预分类模型的构建过程进行介绍.

#### 4.3.1　预分类算法

在资源受限的智能移动端选择合适的预分类算法需要考虑两个因素：快速的执行时间和高准确率. 本文分别使用 KNN、随机森林(Random Forest，RF)、Adaboost、OneVsRest 四种经典的分类算法对图片识别任务进行快速分类，并对不同预处理算法的性能进行实验对比(实验见 5.2 节). 其中，KNN 模型中 $K$ 值的选取以及 RF 算法中深度的选择都会对分类过程产生影响，如 RF 算法中过大的深度值将导致模型过拟合，而较小的深度则不能充分发挥模型的分类能力；Adaboost 和 OneVsRest 模型的分类器则是由多个弱分类器构成的，弱分类器的类别选择会影响整体的分类性能. 根据不同算法的特点，本文对以上四种算法进行调参，并选择模型效果最优的参数进行训练. 如 5.2 节实验结果所示，随机森林算法性能最优，由此，本文使用随机森林算法构建预分类模型.

#### 4.3.2　构建预分类模型

预分类模型的构建过程如图 7 所示.

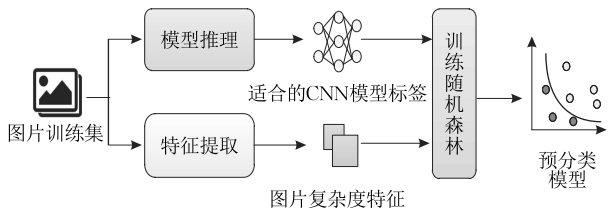


图 7　基于随机森林的预分类模型构建流程

**数据集.** 本文使用 IMAGENET ILVRSC2012 比赛中的验证集作为构建预分类模型的数据集，该数据集包含了 1000 个不同的类别的 50000 张图片，其中训练集为 45 000 张图片，测试集为 5000 张图片. 本文通过十折交叉验证对预分类模型分类效果

① Nathan Silberman. TensorFlow-slim image classification library. https://github.com/tensorflow/models/tree/master/research/slim. 2019

② https://github.com/tensorpack/tensorpack/tree/master/examples/ImageNetModels. 2019

进行评估.具体来说,将 50000 验证图像分成 10 个相等的集合,每个集合包含 5000 个图像.保留一组用于测试预分类模型分类效果,其余 9 组用作训练数据,重复这个过程 10 次,10 组中的每一组都将作为测试数据集,并根据平均准确率对分类算法进行评估,5.2 节将对预分类模型效果进行详细阐述.

**构建训练集**.首先,将上述数据集中的分类图片分别输入 4.2 节算法 1、2 选择的部署在本地和边缘服务器端(LocalModel, ServerModel)的 12 个 CNN 模型(表 1 所示,12 个候选模型均基于 Tensorflow 框架进行构建,并由 ImageNet ILSVRC 2012 训练数据集进行训练构建),将获得预测结果的置信度列表进行降序排列,记录消耗时间.然后根据耗费时间和 Top-1 的结果,确定每个图片的最佳图像分类器(LocalModel/ServerModel)并做标签.同时提取每个图片的 7 种特征值(见 4.4 节表 5).最后将每个图片的特征值与最佳分类器配对,作为预分类模型的训练数据集.

**模型训练**.本文使用随机森林作为预分类模型算法,基于上文生成的预分类模型训练集,通过 Python 机器学习库 Scikit-learn 自动构建模型选择决策树,具体来说,每一棵决策树将对单个图片输出分类结果(LocalModel/ServerModel),并进行投票,统计投票结果后,将得票数多的分类结果作为预处理模型最终输出.

### 4.4 预分类模型特征选择

为了合理表征不同分类任务复杂度的差异性,本节就预分类模型输入的图片分类任务特征进行分析.

#### 4.4.1 特征提取

基于文献[30]对图片特征的描述及研究,首先通过 SimpleCV 的 API(findBlobs、findLines、_getRawKeypoints 等)和 PIL 库函数从图片中提取表征当前图像识别任务复杂度的 29 个初始特征(表 2 所示),如图片的关键点(局部特征突出的点)、亮度、边缘长度(描述图片中包含物体轮廓)、主要目标长宽比等信息.

**表 2 表示图片复杂度的 29 个初始特征**

| 特征 | 描述 |
|---|---|
| n-keypoint | 关键点的个数 |
| Brightness_rms | 亮度均方根 |
| Perceived_brightness_rms | 感知亮度均方根 |
| Edge_length{1~7} | 7 类不同边缘长度的直方图 |
| Area_by_perim | 主要目标的面积除以周长 |
| Hue{1~7} | 7 类不同色调的直方图 |
| Avg_brightness | 平均亮度 |
| Avg_perceived_brightness | 平均感知亮度 |
| Contrast | 对比度 |
| Edge_angle{1~7} | 7 类不同边缘角度直方图 |
| Aspect_radio | 主要目标的长宽比 |

#### 4.4.2 特征选择

为了降低预分类模型时间开销,并防止模型过拟合,本文通过以下三步对 29 个初始特征进行筛选,如图 8 所示.

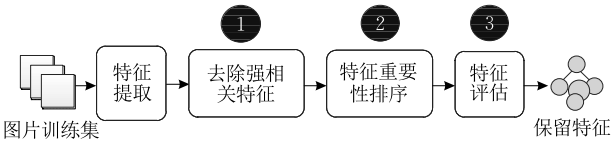


图 8 特征筛选流程

(1)删除强相关性特征.使用皮尔森相关系数构造相关系数矩阵,相关系数值在-1 到 1 之间,若两个特征的相关系数绝对值越接近 1,其相关性越大.在深度模型训练过程中,通常定义特征相关性大于 0.75 时为线性强相关,如图 9 所示,删除相关系数大于 0.75 的特征,只保留一个相关特征,对分类结果性能没有影响,由此为了降低特征处理开销,本文删除特征相关度大于 0.75 的特征,表 3 所示为相关性大于 0.75 的特征,最终保留 16 个特征.

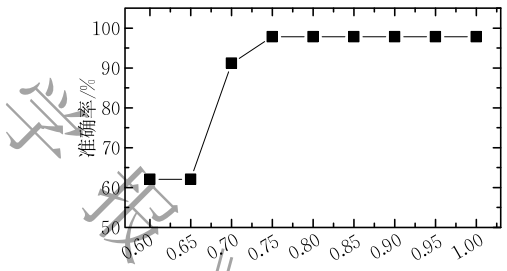


图 9 删除不同相关度特征对预分类模型准确率的影响

**表 3 表示图片复杂度的 7 个特征**

| 保留特征 | 删除特征 | 相关系数 |
|---|---|---|
| Avg_perceived_brightness | Perceived_brightness_rms | 0.98 |
| | Avg_brightness | 0.91 |
| | Brightness_rms | 0.88 |
| Edge_length1 | Edge_length{4~7} | 0.78~0.85 |
| Hue1 | Hue{2~6} | 0.99 |

(2)特征重要性排序.通过 RFE、卡方检测对特征重要性进行排序,并保留评分前 10 的特征(表 4 所示).

**表 4 使用 RFE 和卡方检测选择的 10 种特征**

| 保留特征 |
|---|
| n_keypoints, aspect_radio, area_by_perim, contrast, edge_length1, avg_perceived_brightness, edge_length3, edge_angle5, hue1, hue7 |

(3)对特征的重要性进行验证.针对第 2 步选择的 10 个特征,每次删除一个特征,并观察模型准

确率的变化.图 10 所示,从大到小依次展示单个特征对分类准确率的影响,最终保留对分类准确率有较大影响的前 7 个特征作为预处理模型的输入(表 5 所示).

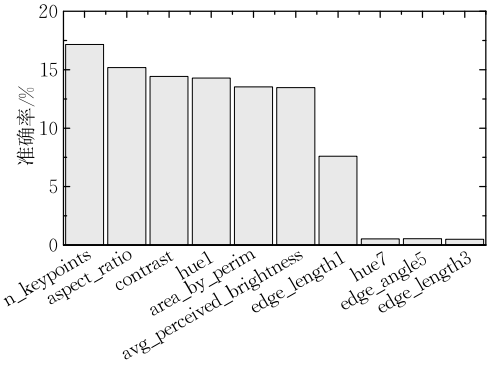


图 10　10 种特征分别对图像分类准确率的影响

**表 5　表示图片复杂度的 7 个特征**

| 保留特征 |
|---|
| n_keypoints, aspect_radio, area_by_perim, hue1, edge_length1, contrast, avg_perceived_brightness |

最后,为了避免由于单一特征值过大,影响预测结果,本文通过离散标准化方法对特征数据进行归一化处理,如式(1)所示.

$$X'=\frac{x-\min(x)}{\max(x)-\min(x)} \tag{1}$$

# 5　实验评估

## 5.1　实验平台

**硬件平台.** 本文使用 NVIDIA Jetson TX2 嵌入式深度学习平台作为边缘服务器,该平台配置了双核 Denver2 和四核 ARM Cortex-A57,GPU 为 256-core NVIDIA Pascal GPU.移动端使用了 Raspberry Pi 3B+嵌入式平台,该平台配置了 64 位四核 A53 处理器,1GB 的 RAM.与此同时,本文实验使用 Monsoon 功耗检测器对 Raspberry Pi 3B+的功耗进行实时测量.

**软件平台.** Jetson TX2 运行 Ubuntu 16.04 系统,搭载 Tensorflow v1.6、CUDNN(v6.0)和 CUDA(v8.0.64).Raspberry Pi 3B+运行 Raspbian (v2018.4),搭载 Tensorflow v1.11 & Tensorflow lite,以及 OpenCV 和 SimpleCV 用来构建特征提取器,Python scikit-learn 构建预分类模型.

**深度学习模型.** 本文考虑了 12 种经典的图像分类 CNN 模型(如表 1 所示),且均基于 Tensorflow 框架进行实现,并在 ImageNet ILSVRC 2012 训练集上完成线下训练.

## 5.2　预分类模型对比实验

预分类模型的主要工作是判定当前用户输入任务复杂度,并输出适合该任务的 CNN 模型标签,移动设备基于输出标签对分类任务进行调度(上传处理/本地处理).本节分别使用 KNN、随机森林、Adaboost 及 OneVsRest 四种分类算法构建预分类模型,并对比不同算法分类的效果.

图 11 为设置不同 $K$ 值时 KNN 算法分类准确率,实验结果显示当 $K$ 近邻个数为 9 时,KNN 准确率最高.图 12 为 RF 不同深度时的准确率,在给定范围内,当深度为 4 时,随机森林准确率不再提升.图 13 和图 14 分别为 OneVsRest 和 Adaboost 采用不同基分类器时的模型准确率,实验结果显示,采用 SVM 作为基分类器时的模型分类效果最佳.进一步,针对不同分类算法各自的最佳参数分别构建预

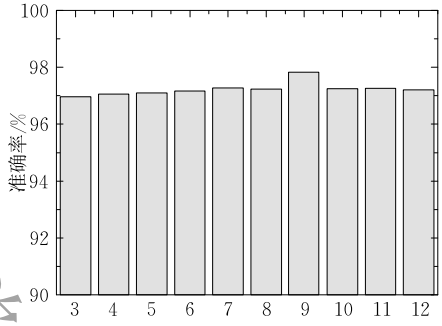


图 11　不同 $K$ 值的 KNN 算法准确率

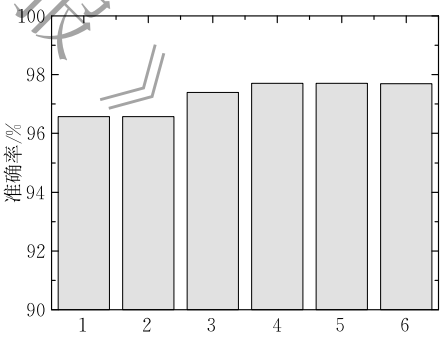


图 12　不同深度随机森林的准确率

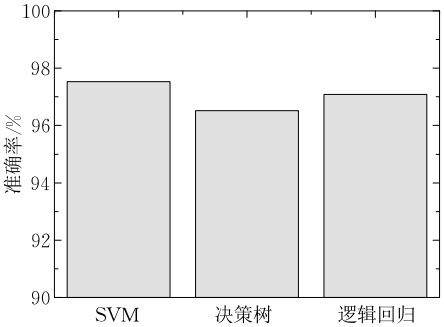


图 13　不同基分类器下 OneVsRest 算法准确率

分类模型，并将图片分类任务分别输入四个模型中，获取查准率、查全率和 $F1$-score 作为模型的性能评估指标，图 15 所示为不同预分类模型的性能指标. 可见 RF 的查准率、查全率和 $F1$-score 均在四种模型中效果最佳. 因此，本文选择 RF 作为预分类模型的核心算法.

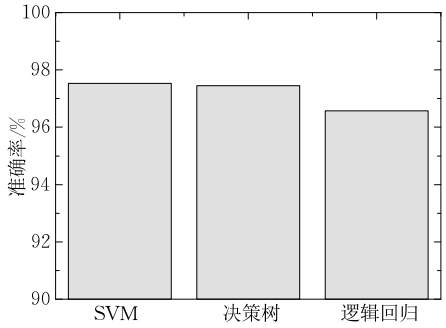


图 14 不同基分类器下 Adaboost 算法准确率

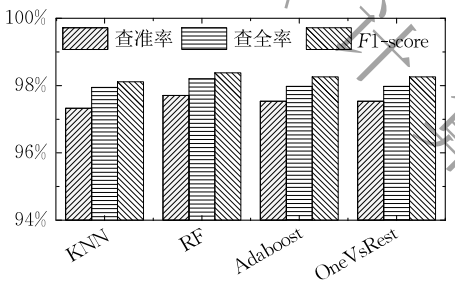


图 15 4 种预分类算法的性能对比

### 5.3 性能分析

本文提出的基于边缘设备的高能效深度模型任务调度策略，通过有效结合移动设备和边缘服务器的各自优势，自动选择及部署深度学习模型，动态调度图片分类任务，从而完成对分类任务的高效处理.

首先，根据算法 1 选择本地部署模型. 图 16、图 17 显示了 12 种典型 CNN 模型在 Jetson TX2 上的推理开销及准确率，由图可知，虽然 MobileNet_V1 准确率只有 70.7%，但推理开销最低（推理时间

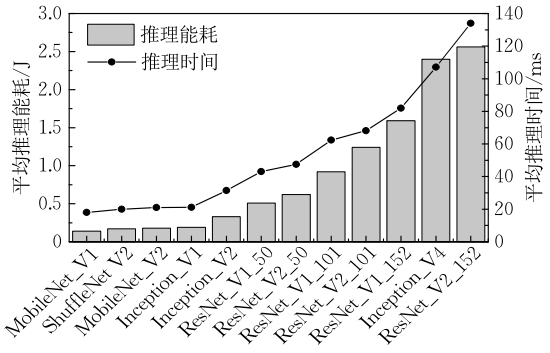


图 16 12 种典型 CNN 模型在 Jetson TX2 上的推理时间及能耗

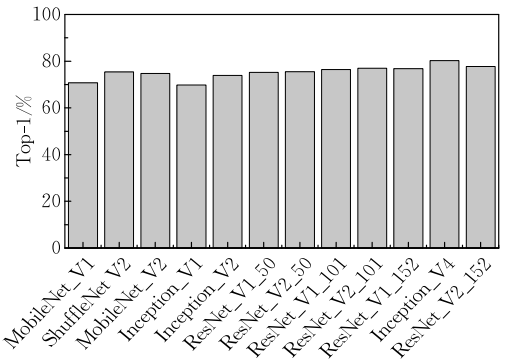


图 17 12 种典型 CNN 模型在 IMAGENET LVRC2012 上的 Top-1

18 ms，推理能耗 0.14 J），相比于 Inception_V4，MobileNet_V1 推理能耗降低了 94.2%，推理时间降低 90.1%. 假设用户期望较低的响应延迟，考虑到移动设备有限的计算、存储及电量资源，由此可以选择将 MobileNet_V1 部署到移动端.

然后根据算法 2 选择部署在边缘服务器端的 CNN 模型. 首先利用另外 11 个深度模型对 MobileNet_V1 识别失败的任务进行推理. 图 18 所示为在 MobileNet_V1 分类失败的 29.3%图片数据集上另外 11 个模型的分类表现. 由图可知，其中 Inception_V4 表现最优，准确率可达 43.91%. 由此，按照算法 2，Edge-based 将自动选择 Inception_V4 部署在边缘服务器上，以处理移动端 MobileNet_V1 无法识别的图像任务. 进一步，本节对 Edge-based、本地部署 MobileNet_V1 和本地部署 Inception_V4 分别进行任务推理的时间、能耗、准确率及相关性能表现进行对比分析. 此外，如 4.3.2 节所示，本文所有实验均通过 10 折交叉验证进行测试，实验结果为交叉验证平均值.

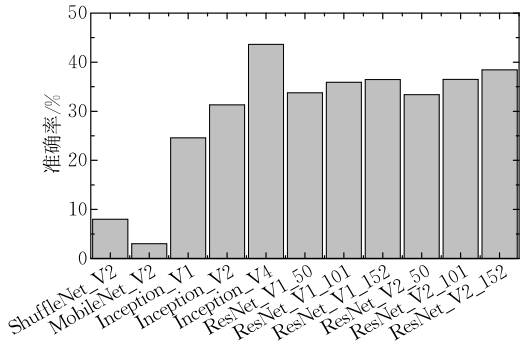


图 18 11 个候选 CNN 模型在 MobileNet_V1 分类失败数据集上的分类表现

**准确率**. 由图 19 可知，MobileNet_V1、Edge-based 和 Inception_V4 的准确率分别为 70.97%、

83.23%、80.12%.本文提出的基于边缘服务器的调度策略准确率相比于 MobileNet_V1 提升了 17.27%,相比于准确率最高的 Inception_V4 提升 3.88%(计算公式[Edge-based(准确率)－其他模型(准确率)]/其他模型(准确率)×100%).具体来说,由于 Edge-based 方法是对 Inception_V4 和 MobileNet_V1 的结合,基于当前图片特征,自动选择分类模型,因而部分在 MobileNet_V1 无法分类准确的图片由预分类模型调度到 Inception_V4 进行分类(12.93%),除此之外,实验显示 Inception_V4 分类错误的 3.5%的任务,在 MobileNet_V1 上得到了准确的分类结果.综上所述,Edge-based 考虑任务特征,通过预分类模型动态将任务调度到合适的深度模型进行分类,由此准确率要高于 Inception_V4.与此同时,MobileNet_V1 和 Inception_V4 相结合可达的理论最优值为 83.71%(理论最优值为预分类模型准确率为 100%时的分类准确率),而 Edge-based 的准确度可达最优值的 99.45%.

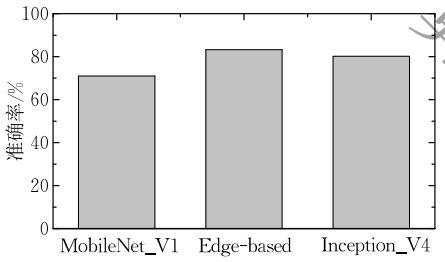


图 19　Edge-based 策略同本地执行 MobileNet_V1 和 Inception_V4 时的 Top-1 准确率对比

**推理时间**.图 20 所示为 Edge-based 的时间开销同在本地执行 MobileNet_V1 和 Inception_V4 的推理时间对比.由图可知,本地部署 MobileNet_V1 平均推理时间最低(519.63 ms),较 Inception_V4(6680 ms)提升 11.85x, Edge-based 的时间开销(561.7 ms)略高于 MobileNet_V1,但比 Inception_V4 提升 10.8x.

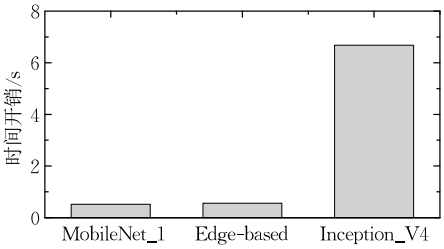


图 20　Edge-based 策略同本地执行 MobileNet_V1 和 Inception_V4 时的推理时间对比

**能量消耗**.图 21 所示为三种策略的能耗开销,由图可知,Edge-based 能耗仅为 0.45J,同本地运行 MobileNet_V1(0.49J)和 Inception_V4(6.66J)相比,Edge-based 能耗分别节约 8.2%和 93.2%.

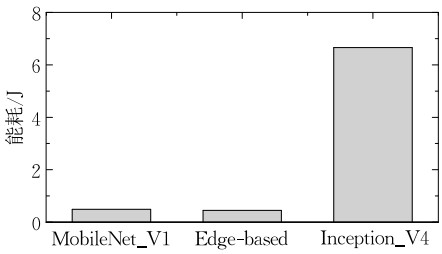


图 21　Edge-based 策略同本地执行 MobileNet_V1 和 Inception_V4 时的推理能耗对比

**查准率&召回率&*F*1**.图 22 给出了 Edge-based 调度策略的其他性能指标,可以看出,相比于 MobileNet_V1 和 Incception_V4,Edge-based 调度策略取得了最高的查准率和 *F*1-score.高查准率在某些领域中尤为重要,如视频监控,因为高查准率可以减少人工对假正例预测的检查.

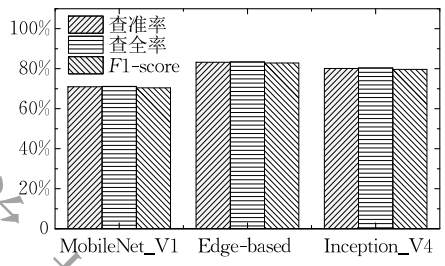


图 22　Edge-based 策略同本地执行 MobileNet_V1 和 Inception_V4 的其他性能指标对比

### 5.4　训练和部署开销

本文提出的预分类模型仅需线下训练一次并部署,不需要重复训练.平均的推理时间低于 0.3 s,同在移动设备中部署高准确率的 Inception_V4 相比,Edge-based 调度策略平均时间开销降低 91.6%,能耗降低了 93.2%.与此同时,本文对移动设备的 RAM 占用进行实时监控分析,模型运行时 RAM 平均使用率仅为 6.1%.

本文实验过程中,生成训练数据时间较长,由于在进行推理模型的选择时,本文使用了 12 个候选深度模型在 Jetson TX2 上进行推理并记录结果,大约花费一周时间.为了降低实验测量过程中的噪声,本文在无负载的机器上进行实验.

本文提出的 Edge-based 调度策略运行开销较低,具体包含图像特征提取、预分类模型处理、基于预分类模型决策的本地推理开销、上传任务至边缘服务器开销及边缘服务器推理时间(不考虑边缘服

务器推理能耗),具体开销如表 6 所示.

**表 6 Edge-based 调度策略的能耗开销分布** (单位:J)

| 特征提取 | 预分类模型 | 本地模型推理 | 任务上传 |
|---|---|---|---|
| 0.047 | <0.001 | 0.345 | 0.058 |

## 6 结 论

现如今,深度学习模型在图像识别、语音识别等领域的识别水平已经达到人类知识水平.然而,深度学习模型优异表现的背后是强大计算能力及海量存储资源的支持,模型在不断刷新任务分类准确率的同时,其深度和规模也在成倍增长.另一方面,基于深度学习模型的移动应用需要为用户提供低时延的响应服务,而移动端资源及能耗限制已成为模型应用的最大瓶颈,单纯的将深度学习模型部署在移动端会产生用户无法忍受的开销.研究人员开发的轻量级模型均以牺牲预测精度为代价来降低深度模型运行开销.本文面向边缘设备,将用户移动端的移动便捷性同边缘服务器的计算及存储能力相结合,提出了一种针对深度模型任务的高能效调度策略,该策略基于用户对准确率、推理时间及能耗的期望,在不同的边缘设备上部署合适的深度学习模型,并利用线下训练的预分类模型对任务复杂度进行分析,自动将任务分配到合适的模型中进行推理.本文将该策略应用在图片识别任务,并在 RaspberryPi 3B+和 Nvidia Jetson TX2 平台进行部署,通过 ImageNet ILSVRC 2012 验证集进行实验验证.实验结果显示,相比于在移动端部署精度最高的深度学习模型,本文提出的高能效深度学习任务调度策略节能 93.2%,推理时间加速 10.8 倍,精度提升 3.88%.

**致 谢** 感谢新型网络智能信息服务国家地方联合工程研究中心的大力支持!

## 参 考 文 献

[1] Chen C, Seff A, Kornhauser A, Xiao J. DeepDriving: Learning affordance for direct perception in autonomous driving//Proceedings of the IEEE International Conference on Computer Vision. Santiago, Chile, 2015: 2722-2730

[2] Esteva A, et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature, 2017, 542(7639): 115-118

[3] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks//Proceedings of the 25th International Conference on Neural Information Processing Systems. Nevada, USA, 2012: 1097-1105

[4] He K, et al. Deep residual learning for image recognition//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA, 2016: 770-778

[5] Amodei C D, Ananthanarayanan S, et al. Deep speech 2: End-to-end speech recognition in English and Mandarin Dario//Proceedings of the 33rd International Conference on Machine Learning. New York, USA, 2016: 173-182

[6] Ji Rong-Rong, et al. Deep neural network compression and acceleration: A review. Journal of Computer Research and Development, 2018, 55(9): 47-64(in Chinese)
(纪荣嵘等. 深度神经网络压缩与加速综述. 计算机研究与发展, 2018, 55(9): 47-64)

[7] Qin Qing, Ren Jie, Yu Jialong, et al. To compress, or not to compress: Characterizing deep learning model compression for embedded inference//Proceedings of the 16th IEEE International Symposium on Parallel and Distributed Processing with Applications. Melbourne, Australia, 2018: 729-736

[8] Iandola F N, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. arXiv preprint arXiv:1602.07360, 2016

[9] Ma Ningning, et al. ShuffleNet V2: Practical guidelines for efficient CNN architecture design//Proceedings of the European Conference on Computer Vision (ECCV). Munich, Germany, 2018: 116-131

[10] Li H, Kadav A, Durdanovic I, et al. Pruning filters for efficient ConvNets//Proceedings of the 5th International Conference on Learning Representations. Toulon, France, 2017: 1-13

[11] Bound L, Bound U. BinaryConnect: Training deep neural networks with binary weights during propagations//Proceedings of the Advances in Neural Information Processing Systems. Montréal Canada, 2015: 3123-3131

[12] Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge. International Journal of Computer Vision, 2014, 115(3): 211-252

[13] Georgiev P, et al. Low-resource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2017, 1(3): 50

[14] Ravi S. Projectionnet: Learning efficient on-device deep networks using neural projections. arXiv preprint arXiv: 1708.00630, 2017

[15] Gong Y, Liu L, Yang M, Bourdev L. Compressing deep convolutional networks using vector quantization. arXiv Prepr. arXiv1412.6115, 2014

[16] Howard A G, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017

[17] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. Fiber, 2015, 56(4): 3-7

[18] Han S, Pool J, Tran J, Dally W. Learning both weights and connections for efficient neural network//Proceedings of the NeurIPS. Montreal, Canada, 2015: 1135-1143

[19] Boureau Y L, Bach F, Lecun Y, et al. Learning mid-level features for recognition//Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Francisco, USA, 2010: 2559-2566

[20] Courbariaux M, Hubara I, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1. arXiv preprint arXiv:1602.02830, 2016

[21] Li Fengfu, Zhang Bo, Liu Bin. Ternary weight networks. arXiv preprint arXiv:1605.04711, 2016

[22] Wu Jun-Ru, Yue Wang, Wu Zhen-Yu, Wang Zhang-Yang. Deep *K*-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. arXiv preprint arXiv: 1806.09228, 2018

[23] Wu Jiaxiang, et al. Quantized convolutional neural networks for mobile devices//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA, 2016: 4820-4828

[24] Gupta S, Agrawal A, et al. Deep learning with limited numerical precision//Proceedings of the International Conference on Machine Learning 2015. New York, USA, 2015: 1737-1746

[25] Li Hao, Kadav A, Durdanovic I, et al. Pruning filters for efficient ConvNets. arXiv preprint arXiv:1608.08710, 2016

[26] Yang Tien-Ju, Chen Yu-Hsin, Sze V. Designing energy-efficient convolutional neural networks using energy-aware pruning//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Hawaii, USA, 2017: 5687-5695

[27] Teerapittayanon S, McDanel B, Kung H T. Distributed deep neural networks over the cloud, the edge and end devices//Proceedings of the IEEE 37th International Conference on Distributed Computing Systems. Atlanta, USA, 2017: 328-339

[28] Ossia S A, Shamsabadi A S, Taheri A, et al. A hybrid deep learning architecture for privacy-preserving mobile analytics. arXiv preprint arXiv:1703.02952, 2017

[29] Sandler, Mark, et al. MobileNetV2: Inverted residuals and linear bottlenecks//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Salt lake, USA, 2018: 4510-4520

[30] Hassaballah M, Abdelmgeid A A, Alshazly H A. Image features detection, description and matching. Image Feature Detectors and Descriptors. Basel, Switzerland: Springer, 2016: 11-45

**REN Jie**, Ph. D., lecturer. His research interests include mobile computing, machine learning, and performance optimization.

**GAO Ling**, Ph. D., professor. His research interests include network management and embedded system.

**YU Jia-Long**, M. S. candidate. His research interest is deep learning model optimization.

**YUAN Lu**, M. S. candidate. Her research interest is mobile computing.



**Background**

Recently, the deep neural network has made significant progress in many fields. Its powerful computing ability makes it a useful tool to solve complex problems, and has been widely used in automatic driving, face recognition, and augmented reality, etc. To reduce time delay and protect user privacy, the model inference process should be migrated to the use's nearby devices (e. g., smartphone or edge devices). However, the limited resources and energy consumption of mobile devices become the biggest bottleneck for deploying deep models. Due to the unstable network state and the limited network bandwidth, deploying the DNN model in the cloud cannot guarantee the mobile user experience expectation. At the same time, due to the limited resources of the mobile terminal (computing power, storage size, and battery capacity), the mobile device cannot satisfy the DNN model. Therefore, it is necessary to design a new computing paradigm, so that the inference task can meet the user's expectation of a fast response, low energy consumption and high accuracy. Based on the energy efficiency and classification characteristics of the deep neural network, this paper studies the deployment of CNN model on the edge devices and the scheduling of image classification tasks.

This paper focuses on how to reduce the delay and energy consumption of deep learning tasks and maintain the inference accuracy on edge devices. We propose an energy-efficient task scheduling strategy by combining the strong computing power of the edge server and the convenient of the smartphone. In this paper, we apply our strategy to the deep learning-based image recognition application.

Specifically, we use IMAGENET ILVRC2012 as the

training and testing data set and choose 12 popular convolutional neural network models to verify the performance. Firstly, we record the inference time and energy consumption of 12 CNN models on two representative embedded systems (RaspberryPi 3B+ and NVIDIA JETSON TX2), and then our approach will select and deploy the lightweight models, which cost least energy consumption with low latency and low accuracy on Raspberry Pi (mobile device), and high accuracy models which need powerful computing processor and big storage on Jetson TX2 (edge device). It is important to note that the selecting and deploying process is performed automatically; after the models have been deployed on the mobile device and edge device respectively. The image recognition tasks are then scheduled to mobile or edge devices by using a pre-trained machine learning model (based on random forest algorithm) on the mobile device. pre-trained machine learning model will extract 7 key features from input tasks that describe their complexity and then output a decision (upload to edge or execute on the local device). Our strategy would like to schedule the simple images running on the local mobile device and the complexity images task on the edge server. The results show that our strategy obtains, on average, 92.6%, 91.8%, and 3.38% improvement for energy consumption, inference time, and accuracy when compared to the most accurate model that runs on mobile devices.

This work was supported in part by National Key R&D Program of China No. 2019YFB2102200, in part by China NSF Grant Nos. 61972252, 61972254, 61672348, and 61672353, and in part by Alibaba Group through Alibaba Innovation Research Program. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government.