DOI：10.7652/xjtuxb201806006

# 基于 Caffe 的嵌入式多核处理器深度学习框架并行实现

高榕，张良，梅魁志
（西安交通大学电子与信息工程学院，710049，西安）

**摘要**：针对开源深度学习快速特征嵌入的卷积框架(Caffe)在 Android 移动端进行前向计算时存在的兼容性和时间性能差的问题，提出了基于 Caffe 的嵌入式同构、异构并行化改进设计方法。该方法将 Caffe 及其第三方库通过交叉编译移植到嵌入式移动平台后，利用同构的多核多线程方法分别对卷积层、输入帧之间的部分前向计算过程进行了并行化；实现了采用开放运算语言(OpenCL)的异构图形处理器(GPU)卷积计算，进一步提升了框架的处理速度。对 3 种经典的深度神经网络模型 MNIST、Cifar-10 和 CaffeNet 进行了测试对比，测试结果表明：在没有任何模型精度损失的条件下，并行后的前向计算耗时明显低于并行前，时间性能提升最高达到 2 倍。所提方法能够将深度学习框架 Caffe 高效地、并行地部署和应用于嵌入式移动多核芯片上。

**关键词**：深度学习；移动端；前向计算；并行；OpenCL
**中图分类号**：TP183 **文献标志码**：A **文章编号**：0253-987X(2018)06-0036-06

# A Deep Learning Frame on Embedded Multicore Processors Based on Caffe and Its Parallel Implementation

GAO Rong, ZHANG Liang, MEI Kuizhi
(School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

**Abstract**: An effective embedded homogeneous and heterogeneous parallel improvement design on the basis of Caffe is proposed to solve the poor compatibility and low efficiency of forward inference in Android mobile terminals by using the open-sourced deep learning frame named Caffe (Convolutional architecture for fast feature embedding). The scheme transplants Caffe and its third-party library to arm architecture using a cross compiler, and then the multi-core and multi-thread technology is used to parallelize partial forward inference between convolution layer and input frame group. An heterogeneous parallel convolutional implementation based on OpenCL is also presented to further improve the time performance of the scheme. Comparison tests with three classic deep learning neural networks MNIST, Cifar-10 and CaffeNet show that in the absence of any model precision loss, the time consuming after parallelization is far less than that before parallel, and time performance increases up to 2 times. It is concluded that the proposal can make the deep learning frame Caffe effectively deploy and work in parallel on portable embedded multicore devices.

**Keywords**: deep learning; mobile terminal; forward inference; parallelism; OpenCL

收稿日期：2018-01-16。 作者简介：高榕(1993—)，男，硕士生；梅魁志(通信作者)，男，教授，博士生导师。 基金项目：国家重点研发计划资助项目(2017YFB1301100)。
网络出版时间：2018-03-27 网络出版地址：http://kns.cnki.net/kcms/detail/61.1069.T.20180327.0847.012.html

深度学习在图像处理、自动控制和智能机器人等方面表现出了良好的应用潜力，已经被广泛应用到物体识别、手势控制等诸多方面。目前，深度学习技术不再局限于各机构的基础理论研究，已经形成了较为成熟的应用系统，例如人脸检票系统、自动阅卷系统以及交通监控系统等。智能移动终端在当今互联网时代背景下应用愈加广泛。随着便携设备的不断发展，人们对其能够提供智能、稳定、高速的应用需求与日俱增。将深度学习技术应用于嵌入式设备中，用以改善用户体验并提高软件智能，具有重要的现实意义。

卷积神经网络是深度学习算法的基础，Caffe (Convolutional architecture for fast feature embedding)[1] 和 TensorFlow[2] 是目前应用范围最广、生命力最活跃的学习框架。Caffe 结构清晰、可读性高，主体以 C++代码编写，硬件兼容性好，同时有着丰富的线上资源和活跃的开发社区，因此被广泛采用。

ARM 嵌入式处理器是当前移动设备主要的处理器，最新的 Samsung Exynos 8895、Qualcomm Snapdragon 835 以及 Apple A10 CPU 主频可达 2 GHz 以上。深度学习算法在嵌入式终端的实现和加速是目前机器学习与人工智能领域的研究热点。当前性能高的移动端深度学习框架有 Facebook 的 Caffe2[3]、腾讯的 ncnn[4] 和百度的 MDL[5]。Caffe2 基于 Caffe 开发，代码结构与 TensorFlow 更为类似，以轻量化、模块化和张量化为特点，强调多平台支持性和移动端高效性，但目前 Caffe2 尚未成熟，仅支持部分嵌入式 GPU；ncnn 支持多种神经网络结构，不依赖第三方数学计算库，具有极为精简的代码体积和高效的汇编级优化，但不支持异构计算；MDL 旨在简化卷积神经网络的部署难度，体积小、速度快，仅支持 iOS GPU。针对现今广泛应用的 Caffe 模型格式，Caffe2、ncnn 和 MDL 均需要额外的模型转换操作。

在移动端加快神经网络前向计算速度主要有修改神经网络的模型和加快框架运行速度两种策略。Han、Lin 等从模型压缩角度出发，采用剪枝、哈夫曼编码和定点小数方式优化训练模型[6-7]；Zhang、Marat 从矩阵乘法角度出发，采用 Neon、快速傅立叶变换方式加速网络计算[8-9]。Oskouei 等使用 RenderScript 异构框架，卷积层计算最高提速达 60 倍[10]。

为探索深度学习框架如何在 Android 手机端高效部署的问题，最大化地精简前向计算流程并保证兼容性，考虑到目前嵌入式 GPU 性能逐渐提升的事实，本文从框架优化角度出发，实现了基于 Caffe 的嵌入式移动端深度学习算法框架前向计算的多核同构并行和 OpenCL[11] 异构并行加速方案。

## 1　基于 Caffe 的嵌入式同构、异构并行化改进方法

Caffe 具有清晰的代码结构，以高度模块化的设计取得了良好的可扩展性。本文提出并实现了一种基于 Caffe 的嵌入式同构、异构深度学习算法框架加速方法，主要包括 3 个部分：基于 CPU 的卷积层 im2col 函数多核多线程同构并行化；基于 CPU 的卷积层、非线性激活、池化层多核多线程帧间同构并行化；基于 GPU 的卷积层 im2col 函数异构并行化，方法的框架如图 1 所示。

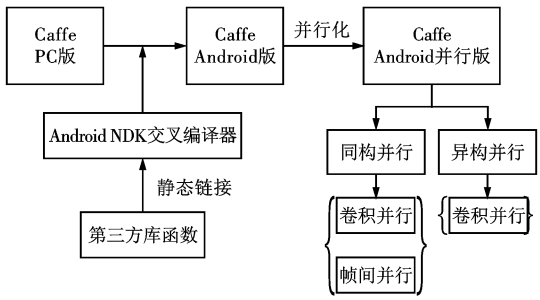


图 1　Caffe 移动端同构、异构并行方法框架

通过对深度神经网络模型前向计算过程各层耗时进行分析表明：卷积层和池化层是深度神经网络前向计算过程中具有并行化潜力的最耗时的结构；卷积层耗时约占整个过程耗时的 60%，池化层耗时约占整个过程耗时的 20%[12]。基于上述事实，利用多核多线程方法，对卷积层中的像素重排操作（即 im2col 函数）进行并行优化，将卷积核在每帧图像的线性扫描操作按照位置对应规则分配到多个线程中做等间隔式处理；采用多核多线程方法对多帧图像的深度神经网络模型前向计算过程进行并行优化，将线性的卷积层、非线性激活和池化层按照数据优先规则分配到多个线程中做流水线式处理；利用异构的 OpenCL 框架，将卷积层中的像素重排操作迁移到 GPU 中进行，充分利用嵌入式移动终端的计算资源，进一步提高算法框架的性能。

## 2　基于 CPU 的同构并行化

卷积层、激活函数层和池化层是深度神经网络模型中不可或缺的组成部分，并起着至关重要的作

用。卷积层用来进行特征提取；激活函数层用来引入非线性因素；池化层主要用来降低数据维度、减少计算量、提高模型的泛化能力。

### 2.1 卷积并行

卷积算法的一般实现主要由 2 部分组成：第 1 部分是输入二维矩阵转一维向量操作，如图 2 所示，对索引的图像块重排为矩阵列（im2col 函数），逐通道、逐行、逐列将较为复杂的滑动卷积分解为像素重排和矩阵乘法；第 2 部分是基本矩阵乘法操作。

卷积核以一定步长沿着输入图像的长、宽和通道遍历滑动，并行度较低。为了加快重排速度，增强函数的非线性性，最大限度地利用多核处理器的计算资源，提出了 im2col 同构并行化方法。卷积核滑动窗口在输入帧的每一个位置处的二维数据相互独立，能够将像素重排操作分配给不同的线程处理；另外，重排结果在数学原理上有严格次序关系，能够将不同线程的结果直接输出到预定位置。采用全局指针共享输入输出的方式，避免了不必要的资源占用和时间损耗[13]。

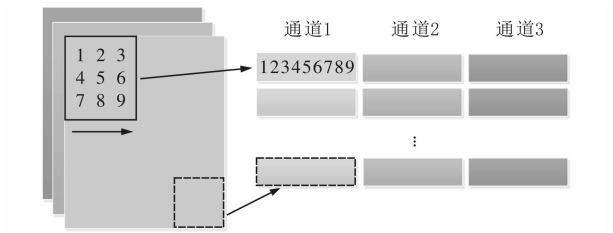


图 2 像素重排函数操作示意图

2.1.1 位置对应的任务分配规则 子线程是进行像素重排的最小操作单元，不同子线程有且仅有自己唯一的从 0 开始的线程 ID；子线程数即为并行度。每一个子线程负责处理当前线程 ID 作为起始位置的滑动扫描过程，即位置对应规则，完成后以并行度作为步长等间隔处理，位置对应的任务分配规则如图 3 所示。

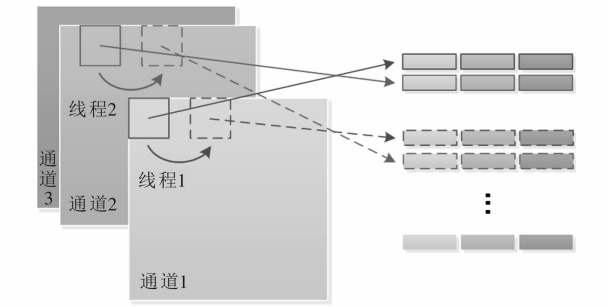


图 3 位置对应的任务分配规则示意图

2.1.2 数据同步与子线程协调 PthreadLib 多线程编程机制中，线程创建时刻与线程执行时刻并不相同，主线程调用线程创建函数创建多个子线程后，系统根据优先级高低和抢占处理器资源的先后，无规律地启动并执行子线程。

本文提出信号量控制线程创建、执行的方法。在每一次子线程创建后、下一次子线程创建前，强制等待信号量激活；在每一次子线程启动执行后、线程 ID 参数正确赋值时，强制释放信号量。通过信号量的等待与释放操作，确保子线程 ID 参数赋值的正确性，即在每一次子线程创建后，主线程函数暂时挂起，避免了循环体连续执行造成的影响；在每个子线程 ID 参数正确赋值后，子线程函数体释放信号量，通知主线程“参数已正确读入，可以进行下次创建”；主线程在收到信号量的释放通知后，解锁等待并进行下一次循环体的执行。信号量同步方法原理如图 4 所示，流程图如图 5 所示。

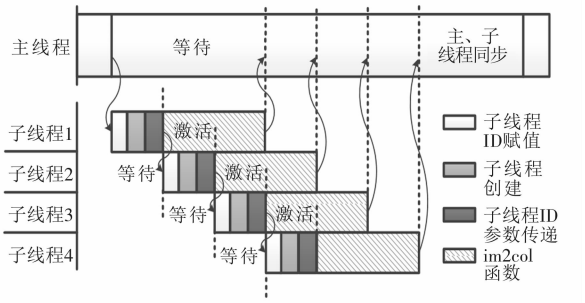


图 4 信号量同步方法原理示意图

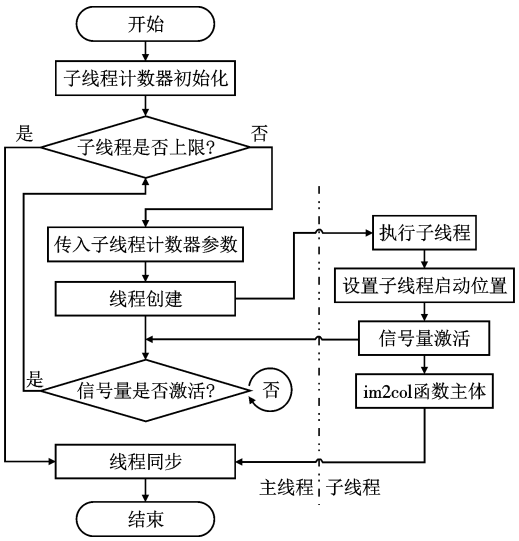


图 5 信号量同步方法流程图

为进一步改善小图像、小模型（MNIST）卷积并行化的加速效果，针对上述方法中存在的多次创建、销毁子线程带来的额外时间开销的问题，提出了改进的信号量同步方法，方法流程如图 6 所示。

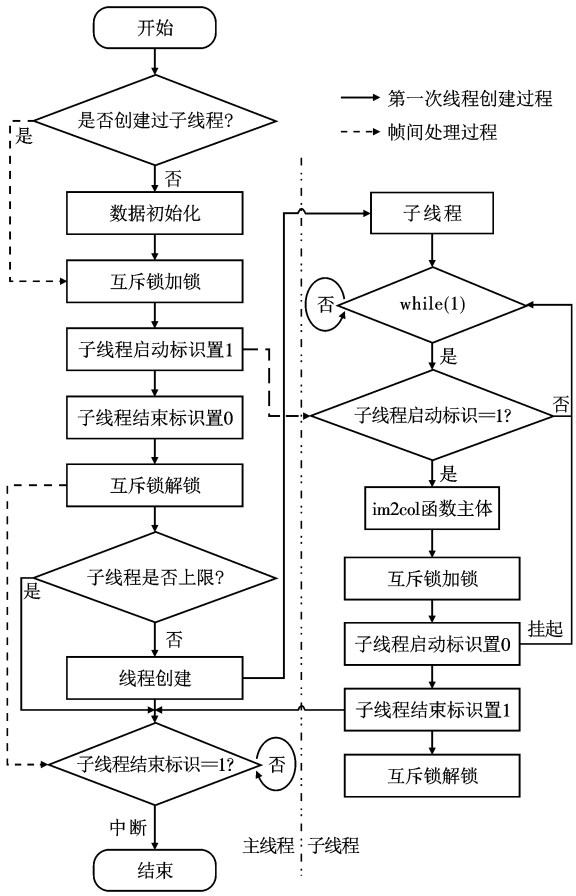


图 6　改进的信号量同步方法流程图

## 2.2　帧间并行

深度学习网络模型的前向计算过程中，在全连接层之前，输入图像的帧与帧之间相互独立。卷积和池化，卷积、非线性激活(Relu)和池化两种结构是深度学习模型最基础的结构。根据上述事实，为了充分利用帧间无关的特性，本文从模型并行的角度出发，对卷积层、非线性激活层和池化层进行了并行，基本原理如图 7 所示，输入组帧数假设为 4。

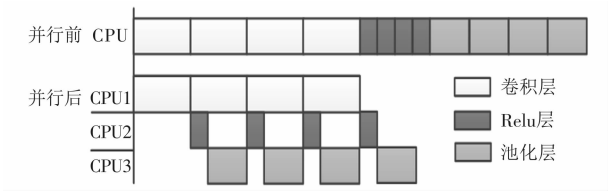


图 7　卷积层、Relu 层及池化层帧间并行原理图

并行前卷积层、非线性激活层和池化层均为串行处理；并行后将每帧的非线性激活或池化分配给不同的核心和线程(以下简称计算单元)，即数据优先规则下的流水线式处理方法，通过信号量作为不同计算单元间的通讯媒介，控制每帧图像不同步骤的启动顺序。

2.2.1　模版匹配、网络复制与 CPU 绑定　帧间并行的处理对象是卷积层、非线性激活层和池化层 3 层，所以首先需要进行模版匹配；深度神经网络各层包含大量的模型信息，为正确的将网络层结构和数据传入子线程中，需要进行网络复制，以保证卷积层、非线性激活层和池化层在不同计算单元的处理独立性；为充分利用多核处理器的资源，合理分配计算权重，将卷积层、非线性激活层和池化层任务绑定到不同的处理器核心上。

2.2.2　层间触发器和接收器的设计　为控制每帧图像不同步骤的启动顺序，需要设置卷积层和非线性激活层的处理触发器，以及非线性激活和池化层的处理接收器。各触发器和接收器相互关系如图 8 所示。

特别地，在上述卷积层、非线性激活函数层和池化层进行实际处理工作前，Caffe 包含一个数据重构操作；非线性激活函数层通常会以“子层”的方式嵌入卷积层中，并与“父层”共享输入、输出数据。为避免卷积层和非线性激活层同时对数据重构产生冲突，需额外对重构操作进行时序控制，如图 8 所示。

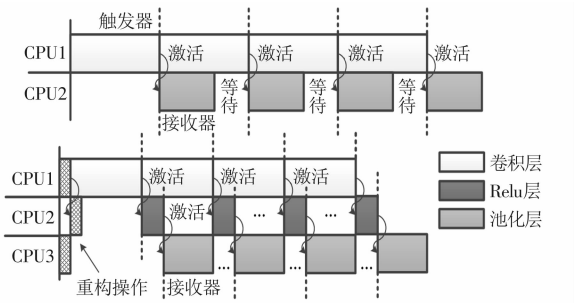


图 8　卷积层、Relu 层和池化层触发器、接收器配置示意图

# 3　基于 GPU 的异构并行化

为利用部分移动设备携带高性能 GPU 的优点，消除 im2col 同构并行时子线程抢占式等待的影响，提高 CPU 的计算效率，本文实现了 im2col 函数的异构并行化。

OpenCL 由多维度的工作组构成，每个工作组承担一次计算任务。im2col 函数一共有 3 层循环，用来定位与卷积核匹配的输出位置。采用三维工作组获取索引的方式将循环展开，每一组索引映射到对应卷积核位置上并分发给各自的工作组并行处理，完成后将结果从 GPU 取出。im2col 异构并行方法原理如图 9 所示。

在对多帧图片进行前向计算时，OpenCL 需要频繁地创建销毁上下文、命令队列和程序，并反复申

请释放固定大小的内存块。为最大程度地提升异构并行效率，本文提出“一次创建，多次使用，一次销毁”的 OpenCL 优化方法，在首帧、末帧图片创建、销毁 GPU 内存空间，所有图片共享计算资源[14]。

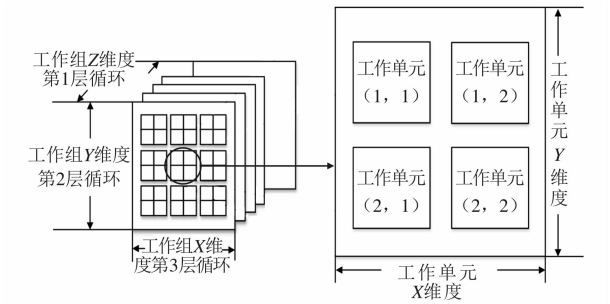


图 9 im2col 异构并行方法原理图

## 4 实 验

本文在 PC 平台和嵌入式 Android 平台，对 MNIST、Cifar-10 和 CaffeNet 3 种经典网络模型进行了测试，验证了同构、异构并行化方案对加速神经网络前向计算过程的可行性、正确性和有效性。

### 4.1 Caffe 的 Android 移植

原嵌入式 Android 系统不支持 Caffe，无法直接在移动端进行编译和运行，为解决上述问题本文采用交叉编译方案，在桌面 Ubuntu 系统使用 Android NDK Build 工具编译生成能够运行于目标平台的 Caffe 及其第三方库，并下载到嵌入式平台[15]。移植框架流程如图 10 所示。

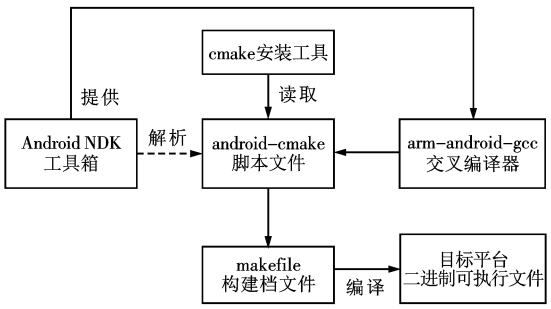


图 10 Caffe 嵌入式移植框架流程图

### 4.2 并行性能测试

本文的 PC 测试平台为 Intel Core i7 3930k、Ubuntu x64 系统、Caffe CPU 版本；嵌入式测试平台为 Cortex-A9 Samsung S5P4418 CPU，动态运行主频 400 MHz～1.4 GHz，4 核心，系统为 Android 5.1 ADB shell；同构、异构测试平台为 Qualcomm SnapDragon 820 CPU，运行主频为 1.8 GHz，4 核心，Adreno 530 GPU，运行主频为 510 MHz，3 GB 运行内存，系统为 Android 7.0，测试环境为 Android 7.0 ADB shell。MNIST、Cifar-10 和 CaffeNet 3 种模型的 Caffe 网络配置见表 1。

表 1 MNIST、Cifar-10、CaffeNet 三种模型网络配置

| 层 | MNIST | Cifar-10 | CaffeNet |
|---|---|---|---|
| 1 | 卷积 | 卷积 | 卷积，Relu |
| 2 | 池化 | 池化，Relu | 池化 |
| 3 | 卷积 | 卷积，Relu | 局部归一化 |
| 4 | 池化 | 池化 | 卷积，Relu |
| 5 | 全连接，Relu | 卷积，Relu | 池化 |
| 6 | 全连接 | 池化 | 局部归一化 |
| 7 | | 全连接 | 卷积，Relu |
| 8 | | 全连接 | 卷积，Relu |
| 9 | | | 卷积，Relu |
| 10 | | | 全连接，Relu |
| 11 | | | 全连接，Relu |
| 12 | | | 全连接，Relu |

MNIST 输入图像为 28 像素×28 像素单通道灰度图，输入帧大小为 100，循环次数为 50 次；Cifar-10输入图像为 32 像素×32 像素三通道彩色图，输入帧大小为 100，循环次数为 50 次；CaffeNet 输入图像为 256 像素×256 像素三通道彩色图，输入帧大小为 5，循环次数为 50 次。

本文选用前向计算的验证过程；验证过程用于模型验证，区别于前向计算的部署过程，验证过程复杂度更高，具有代表性。

PC 平台用来做性能预估测试，测试结果为 MNIST、Cifar-10、CaffeNet 整个网络验证过程耗时，见表 2。本文不对帧间并行进行单独测试，将 CPU 卷积并行与帧间并行的组合优化记作同构并行，将 GPU 卷积并行与帧间并行的组合优化记作异构并行，后文均沿用上述说法。

加速比是同一任务在未并行前和并行后运行消耗时间的比率，由表 2 可知，同构并行算法在 PC 平台下对 3 种模型的最优加速比分别为 1.67、1.60、1.12。

表 2 4 线程 PC 平台同构并行前向计算时间测试结果

| 模型 | 前向计算时间/s | | |
|---|---|---|---|
| | 未并行 | 卷积并行 | 同构并行 |
| MNIST | 6.251 | 5.632 | 3.750 |
| Cifar-10 | 17.135 | 14.932 | 10.758 |
| CaffeNet | 48.188 | 47.262 | 43.082 |

Cortex-A9 Samsung S5P4418 平台同构并行前向计算时间测试结果见表 3。

表 3　2 线程 S5P4418 同构并行前向计算时间测试结果

| 模型 | 前向计算时间/s | | |
|---|---|---|---|
| | 未并行 | 卷积并行 | 同构并行 |
| MNIST | 28.98 | 34.26 | 19.09 |
| Cifar-10 | 148.94 | 120.60 | 103.27 |
| CaffeNet | 245.50 | 205.31 | 170.46 |

由图 3 可见，MNIST 输入图片较小，复杂度较低，卷积并行优化收益小于线程调用开销，所以实际测试耗时不降反升。为避免计算密集型应用阻塞所有核心而影响平台自身性能，S5P4418 同构并行化测试只使用 2 线程。由表 3 可知，同构并行算法在 S5P4418 平台下对 3 种模型的最优加速比分别为 1.52、1.44、1.44。

Qualcomm SnapDragon 820 平台（以下简称 820 平台）同构并行前向计算时间测试结果见表 4。

表 4　4 线程 820 平台同构并行前向计算时间测试结果

| 模型 | 前向计算时间/s | | |
|---|---|---|---|
| | 未并行 | 卷积并行 | 同构并行 |
| MNIST | 27.19 | 25.79 | 13.72 |
| Cifar-10 | 138.52 | 76.06 | 67.81 |
| CaffeNet | 152.04 | 103.21 | 98.22 |

由表 4 可知，同构并行算法在 820 平台下对 3 种模型的最优加速比分别为 1.98、2.04、1.55。

通过 3 种平台同构并行化前后测试结果可以看出，同构并行算法能较大幅度地改善深度神经网络前向计算的时间性能，在没有任何精度损失的情况下最大加速比可达 2 倍。

CaffeNet 模型复杂度高，网络深度高于 20，基本包含所有常用的深度学习结构，具有一般性，因此异构并行仅对 CaffeNet 进行测试。820 平台异构并行单帧处理时间测试结果见表 5。异构并行算法在 820 平台下对 CaffeNet 模型的最优加速比为 1.76。

表 5　820 平台异构并行单帧处理时间测试结果

| 模型 | 单帧处理时间/ms | | |
|---|---|---|---|
| | 未并行 | 同构并行 | 异构并行 |
| CaffeNet | 608 | 392.9 | 344.5 |

针对异构并行提出的“一次创建，多次使用，一次销毁”的 OpenCL 优化方案，本文进行了单独测试，测试结果见表 6。

表 6　异构 OpenCL 优化方案单帧处理时间测试结果

| 模型 | 单帧处理时间/ms | | |
|---|---|---|---|
| | 未并行 | OpenCL 优化前 | OpenCL 优化后 |
| CaffeNet | 608 | 354.3 | 344.5 |

本文对 CaffeNet 前向部署过程也进行了并行化，在 820 平台下最优单帧处理时间可达 124 ms。

## 5　结　论

本文在分析当前深度学习框架对于 Android 移动端前向计算部署局限性的基础上，结合现有嵌入式中央处理器和图形处理器性能飞速发展的特点，提出了一种基于 Caffe 的嵌入式同构、异构并行化设计。本文使用 Android NDK Build 对 Caffe 进行了嵌入式移植，针对卷积层线性度高的事实，结合输入组帧间无关的特点，实现了 Caffe 前向计算过程的并行化。对比优化前后的测试结果，本文所提方法取得了较好的加速效果，CaffeNet 的移动端实现具有可行性，便于从 PC 迁移应用。在未来工作中，将利用 OpenCL 对 Caffe 中矩阵乘法进行异构并行优化。

**参考文献：**


[1] JIA Yangqing, SHELHAMER E, DONAHUE J, et al. Caffe: convolutional architecture for fast feature embedding [C]// Proceedings of the 22nd ACM International Conference on Multimedia. New York, USA: ACM, 2014: 675-678.

[2] Google Inc. Tensorflow [EB/OL]. (2017-09-23) [2017-10-10]. https://github.com/tensorflow/tenso-rflow.

[3] Facebook Inc. caffe2 [EB/OL]. (2017-05-22)[2017-06-24]. https://github.com/caffe2/caffe2.

[4] Tencent Inc. ncnn [EB/OL]. (2017-09-23)[2017-11-04]. https://github.com/Tencent/ncnn.

[5] Baidu Inc. Mobile-deep-learning [EB/OL]. (2017-09-03)[2017-10-02]. https://github.com/baidu/mobile-deep-learning.

[6] HAN Song, MAO Huizi, DALLY W J. Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding [J]. Fiber, 2015, 56(4): 3-7.


（下转第 113 页）