2020 年 10 月 第 10 期
Vol. 41 No. 10 2020

切比雪夫逼近的深度神经网络并行加速

李方舒,钱 慧,陈晓旭

(福州大学 物理与信息工程学院,福州 350116)
E-mail: qianhui@ fzu. edu. cn

摘 要: 深度神经网络(Deep Neural Network,DNN)中数据量巨大,且卷积层计算复杂度高,使得其难以在资源有限的嵌入式 GPU 上进行部署,因此需要对其进行并行加速设计. 本文提出采用切比雪夫多项式对卷积核进行逼近,并将该优化方案应用在面向图像重构的 DNN 中以实现卷积操作的并行化处理,降低计算复杂度. 然后为优化后的网络卷积层进行基于 GPU 的并行加速设计,最后将网络整体移植到 NVIDIA AGX Xavier 嵌入式开发板上来实现图像的重构推理过程. 实验结果表明,并行加速后的网络重构推理的速度是原始网络的 2. 2 倍.

关 键 词: 深度神经网络; 图像重构; 并行计算; 嵌入式 GPU; 切比雪夫逼近
中图分类号: TP311 **文献标识码:** A **文 章 编 号:** 1000-1220(2020)10-2206-06

Parallel Acceleration for Deep Neural Network of Chebyshev Approximation

LI Fang-shu,QIAN Hui,CHEN Xiao-xu
(School of Physics and Information Engineering,Fuzhou University,Fuzhou 350116,China)

Abstract: Due to the large amount of data of Deep Neural Network(DNN) and the high computational complexity of convolutional layer,it is difficult for DNN to be deployed on the embedded GPU with limited resources,therefore,parallel acceleration design is needed. This paper proposes to use Chebyshev polynomial to approximate the convolution kernel,and applies this optimization scheme to image reconstruction DNN to realize parallel processing of convolution operation and reduce computational complexity. Then the GPU-based parallel acceleration designs are carried out for the convolutional layers of the optimized network. Finally,the network is transplanted to NVIDIA AGX Xavier embedded development board to realize the reasoning process of image reconstruction. The experimental results show that the reconstruction reasoning speed of the parallel accelerated network is 2. 2 times faster than that of the original network.

Key words: deep neural networks; image reconstruction; parallel computing; embedded GPU; Chebyshev approximation

1 引 言

近年来深度神经网络(Deep Neural Network,DNN)在图像分类,识别,检测等领域取得了巨大的成功. 然而 DNN 普遍存在数据量大,计算复杂度高的问题,在卷积层中尤为突出[1],这使得网络难以移植到资源有限的嵌入式开发板上. 然而随着高科技设备渗透到生活的方方面面,在诸多设备上实现计算密集型的应用变得越来越普遍也越来越重要,因此在资源有限的平台上实现 DNN 的推理成为了研究者们关注的重点.

为了能在嵌入式终端快速部署 DNN,研究者们提出了多种方案. 2015 年韩松等人[2] 提出了去除网络的冗余连接并调整权重以实现网络的压缩,使得网络能够在嵌入式移动应用终端上进行部署. 文献[3] 通过对网络模型进行压缩与高度优化,将网络从重量级模型变换为能够部署在嵌入式开发板上的轻量级模型,实现了一种基于 GPU 嵌入式系统的实时驾驶员睡意检测技术. 文献[4] 通过采用传感器融合超声波来更好的过滤噪声的方法,结合 GPU 硬件特性,提出了一种基于嵌入式 GPU 的实时立体视觉碰撞检测自适应系统. 文献[5] 采用异步多线程并行计算,在 GPU 嵌入式平台上实现微笑检测器过程中,采用模块化的方案对线程进行优先级排序,并通过增加关键任务的线程数量来实现任务之间的负载平衡. Mohammad 的团队针对移动终端上 DNN 的推理进行了诸多研究[6-8],发现根据设备的资源量合理的采用不同的线程粒度进行并行化设计能够显著提高网络的推理速度,并提出了数据重新排列,非精确计算等数据处理方案来实现网络的加速设计. 面向检测,识别等任务的 DNN 开始逐渐应用到资源有限的移动设备上.

为了满足人们对高质量,沉浸式影像日益增长的需求,基于深度神经网络的图像重构引起了研究学者的广泛关注[9-12]. 然而在面向图像重构的 DNN 中,为了从退化的高维图像中不断提取代表图像细节的关键特征,导致网络各层的输出数据量与输入数据量相同甚至更大,网络卷积层负载情况更为严峻. 相较于面向检测识别等问题的 DNN,面向图像

收稿日期: 2019-12-05 收修改稿日期: 2020-01-19 基金项目: 数字福建物联网工程应用实验室建设项目(82917002)资助. 作者简介: 李方舒,女,1995 年生,硕士研究生,研究方向为 GPU 并行加速设计; 钱 慧,女,1977 年生,博士,副教授,研究方向为压缩采样、压缩感知; 陈晓旭,男,1994 年生,硕士,研究方向为 GPU 并行加速设计.

重构的 DNN 进行硬件移植时面临着更严峻的挑战.

切比雪夫多项式能够实现最佳的函数逼近[13],因此本文采用切比雪夫多项式对网络卷积核进行并行化处理以简化卷积层的计算,将该优化方案应用在面向重构的深度神经网络中,随后对优化处理后的网络卷积层进行基于 GPU 的并行化设计. 为了保证方案的普适性,采用 zhu 等人[14]在 2016 年提出的基于流形学习的图像重构网络,该网络能够适用于多种不同的图像重构任务. 最后将经过并行化设计后的网络移植到 NVIDIA AGX Xavier 嵌入式开发板上,并对其进行性能分析.

2　面向图像重构的流形学习深度神经网络

基于流形学习的深度神经网络[14]结构如图 1 所示,传感器数据作为全连接层 1 的输入,由于传感器数据一般为复数,存在实部与虚部,对于 $n \times n$ 的复杂传感器数据,需将其转换为 $2n^2$ 个实数值输入全连接层 1,因此全连接层 1 也表示网络的输入层,神经元数量为 $2n^2$. 全连接层 2 和 3 的神经元个数均为 n^2,采用 *tanh* 激活函数. 为了匹配卷积操作,将全连接层 3 输出的数据进行维度转换处理,由 $n^2 \times 1$ 变换为 $n \times n$. 卷积层 1 采用 64 个 5×5 大小的卷积核以 1 为步长进行卷积操作,卷积层 2 采用 64 个 $5 \times 5 \times 64$ 大小的卷积核以 1 为步长进行卷积操作. 卷积层均采用 *Relu* 作为激活函数,最后对卷积层 2 采用了 1 个 $7 \times 7 \times 64$ 大小的卷积核以 1 为步长进行反卷积得到重构图像.

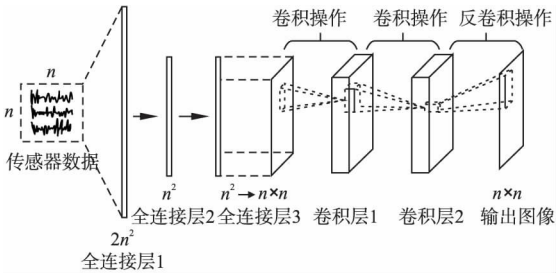

图 1　网络结构

Fig. 1　Structure of the network

DNN 中卷积层主要用于实现输入信号高维特征的提取. 卷积过程中通过卷积核 h 对输入数据 f 进行局部感知,在更高层将局部信息进行综合以提取输入的特征,经典的卷积层计算如公式(1)所示:

$$y = \xi(h * f) \tag{1}$$

y 表示当前卷积层的输出,"*"表示卷积操作. ξ 表示激活函数,基于流形学习的网络模型中采用 *Relu* 函数:

$$Relu(x) = \begin{cases} x & x \geqslant 0 \\ 0 & x \leqslant 0 \end{cases}$$

在卷积计算过程中,常常需要四个嵌套的循环,内部在卷积核上迭代,外部在输入图像的行列方向上迭代,循环效率很低. 其次,二维图像和卷积核通常采用连续的内存块以行串行化的顺序进行存储,按列访问数据可能会造成内存子系统中较高的高速缓存丢失率,因此卷积操作不容易实现快速计算. 考虑采用多项式来对卷积层计算进行简化处理.

3　基于切比雪夫多项式的卷积层优化

3.1　切比雪夫多项式逼近原理

近似计算的核心在于能够在多项式迭代时间内给出问题的近似最优解. 而多项式能够实现函数逼近,通过多项式的逼近转换可以将问题转化为对多项式类的研究. 且由于多项式的和,差,积仍然是多项式,因此可以通过迭代规律来完成复杂计算的快速实现.

切比雪夫多项式能够实现最佳一致逼近. 考虑到实际应用中往往需要对一个已知的复杂函数 $f(x)$ 进行求解,为了对计算进行化简,通常需要寻找一个函数 $Q_n(x)$,使得两者之间的误差能够在某种度量意义下达到最小. 切比雪夫最佳一致逼近理论中,函数 $Q_n(x)$ 为切比雪夫多项式,其满足在某区间 $[a,b]$ 内与 $f(x)$ 之间的差值是区间内所有多项式 $Q(x)$ 与 $f(x)$ 之间插值中最小的,如公式(2)所示:

$$\max_{a \leqslant x \leqslant b} |Q_n(x) - f(x)| = \min |\max |Q(x)_{a \leqslant x \leqslant b} - f(x)|| \tag{2}$$

切比雪夫多项式[13]的函数逼近理论表明,这样的多项式 $Q_n(x)$ 存在且唯一. 令 $E_x = \max_{a \leqslant x \leqslant b} |Q_n(x) - f(x)|$,当 $E(x)$ 在 $[a,b]$ 上至少存在 $n+2$ 个交错点 $[x_1 \cdots x_{n+2}]$ $(a \leqslant x_1 < \cdots < x_{n+2} \leqslant b)$,使得 $E(x_i) = \pm E_n$,其中 $i \in [1, n+2]$,此时 $Q_n(x)$ 是 $f(x)$ 的最佳一致逼近.

3.2　卷积核的切比雪夫多项式逼近

由于切比雪夫多项式能够实现最佳一致逼近,因此考虑采用切比雪夫多项式来进行卷积核的优化. 假设输入图像大小为 3×3,卷积核大小为 2×2(此处为了进行简单的展示,实际上较少出现卷积核大小为 2×2 的情况). 如图2(a)所

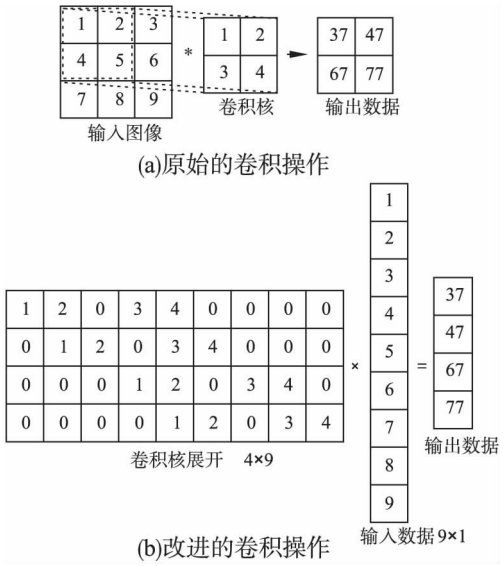

图 2　卷积操作

Fig. 2　Convolutional operations

示,原始的卷积操作中,卷积核大小普遍较小,每次仅能够提取到部分特征,常需要在每个输入特征图上都进行遍历,最后在高维度上进行信息的整合,计算过程复杂.

本文提出将输入数据表示成向量形式,如图 2(b)所示,卷积核中仍然保留对应的权重信息,并采用补 0 的方式对其

进行扩充展开，其中补 0 的地方用灰色方块表示. 此时卷积核矩阵用 H 表示，H 的维度 $p \times q$ 计算方式如公式(3)、公式(4)所示:

$$p = (I_{row} - d_{row})/stride + 1) \times (I_{col} - d_{col})/stride + 1) \quad (3)$$

$$q = I_{row} \times I_{col} \quad (4)$$

其中 I_{row} 和 I_{col} 分别表示输入图像的长和宽，d_{row} 和 d_{col} 分别表示卷积核的长和宽，$stride$ 为步长.

此时的卷积计算被展开成为矩阵向量乘法，可以通过一次计算得到结果而无需进行卷积核对输入图像的遍历，且此时输出同样为一个向量，其能够直接作为下一层的输入，无需进行维度转换.

由于卷积操作被展开成了矩阵向量乘法，此时可以对卷积核采用切比雪夫多项式逼近来进行计算化简，以使得其可以进行并行化处理. K 阶切比雪夫多项式用 $T_k(H) \in \mathbb{R}^{n \times n}$ 表示，其递归公式如式(5)所示:

$$T_k(H) = 2HT_{k-1}(H) - T_{k-2}(H) \quad (5)$$

其中 $T_0 = 1$，$T_1 = H$. 因此卷积核的近似可表示为公式(6):

$$g_\theta(H) = \sum_{k=0}^{K-1} \theta_k T_k(H) \quad (6)$$

$\theta = (\theta_0, \theta_1, \cdots, \theta_{K-1}) \in \mathbb{R}^K$ 表示切比雪夫多项式的系数向量，也表示网络中需要训练的卷积核参数. 卷积层的输入采用全连接层 3 直接输出的 $n^2 \times 1$ 维的数据，不进行维度转换，用 f_{in} 表示，此时网络卷积层计算可用公式(7)表示:

$$y_{cheby} = \xi_{Relu}\left(\sum_{k=0}^{K-1} \theta_k T_k(H) f_{in}\right) \quad (7)$$

y_{cheby} 表示卷积层的输出.

由于切比雪夫的 K 次多项式 $T_k(H)$ 可由公式(5)所示的稳定递归关系来计算，此时假设 $\bar{f}_k = T_k(H) f_{in}$，由切比雪夫迭代可以得出递归方程，如公式(8)所示:

$$\bar{f}_k = 2H\bar{f}_{k-1} - \bar{f}_{k-2} \quad (8)$$

其中 $\bar{f}_0 = f_{in}$，$\bar{f}_1 = Hf_{in}$. 任意 $\bar{f}_k$ 均可写成 $\bar{f}_0$ 和 $\bar{f}_1$ 为基向量的多项式形式. 此时卷积层计算可表示为公式(9):

$$y_{cheby} = \xi_{Relu}([\bar{f}_0, \bar{f}_1, \cdots, \bar{f}_{k-1}]\theta) \quad (9)$$

3.3 卷积层的并行加速设计

CUDA 架构[15] 是 Nvidia 公司推出的 GPU 异构编程架构，本文结合 CUDA 架构对网络卷积层进行并行设计，关键在于切比雪夫多项式迭代的并行设计.

假设以计算 $m(m \geqslant 3)$ 个数据为一轮迭代，图 3 展示了迭代计算的过程，其中每一轮迭代过程中的基向量采用 A 和 B 表示. 在第一轮迭代中基向量为 $\bar{f}_0$ 和 $\bar{f}_1$. 从第二轮迭代开始，每一轮迭代的基向量为前一轮迭代计算出来的最后两个数据值. 其中 $C_{(s,0)}$ 和 $C_{(s,1)}$ 分别表示每轮迭代过程中计算第 $s(s \in [1,m])$ 个值时两个基向量对应的系数值，即多项式系数值. 第 r 轮迭代中计算得到的第 s 个数据值采用 $F_{(r,s)}$ ($r \in [1, K/m]$，$s \in [1,m]$) 表示，其计算公式如式(10)所示:

$$F_{(r,s)} = \begin{cases} C_{(s,0)}\bar{f}_0 - C_{(s,1)}\bar{f}_1 & (r = 1) \\ C_{(s,0)}F_{(r-1,m-1)} - C_{(s,1)}F_{(r-1,m)} & (r \in [2, K/m]) \end{cases} \quad (10)$$

结合公式(10)和图 3 可以得出，在第一轮迭代中即可计算得出全部多项式系数，后面每次迭代中对应顺位的多项式系数相等，如图中 $C_{(s,0)}$ 和 $C_{(s,1)}$ 所示，将它们按照顺序存储在 GPU 全局内存(Global Memory)中.

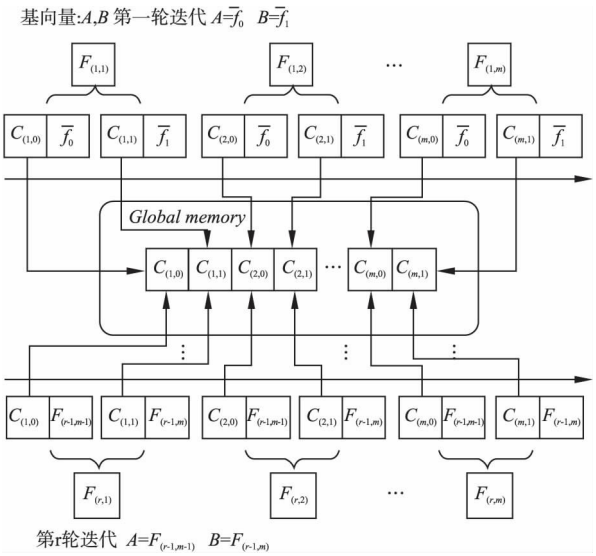

图 3 切比雪夫迭代计算流程

Fig. 3 Process of Chebyshev iterative

在 CUDA 加速设计时，采用线程并行度为 m，在后续迭代过程中直接从 GPU 全局内存调用提前计算好的对应多项式系数来计算对应多项式的值，从而可以避免每次迭代中多项式系数值的重复计算，也减少了 GPU 端与 CPU 端的数据传输次数. 图 4 展示了 CUDA 进行并行处理的过程，全局内存中开辟出两块内存，一块用于存放计算好的多项式系数，一块存放计算好的多项式的值. 迭代计算开始，线程块从全局内存读取多项式系数，为块内线程分配相应的值，每轮迭代结束，采用线程广播机制，将最后两个线程计算得到的多项式值在块内进行广播，作为下一轮迭代的基向量.

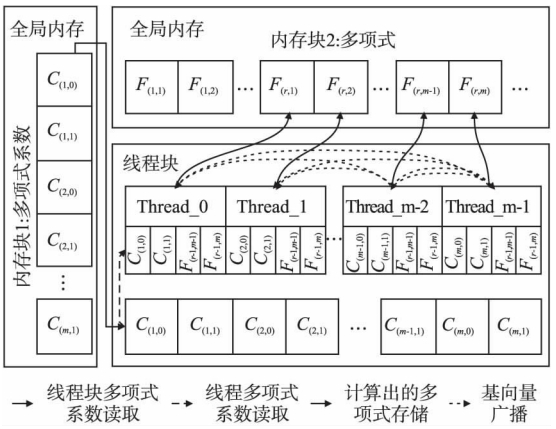

图 4 基于 CUDA 的切比雪夫迭代并行设计

Fig. 4 Cuda-based chebyshev iterative parallel design

采用上述并行方案，计算 k 个数值仅需要迭代 k/m 次，大大缩短了计算时间. 除了第一轮迭代中需要进行复杂的多项式系数计算外，后续的迭代中对每个 $\bar{f}_k$ 的计算均只需要进行两次乘法一次减法.

并行度 m 可以根据硬件平台资源情况进行灵活的设置.

由于每轮迭代中所有多项式的计算都在同一时刻进行,每一轮迭代过程中计算时间由两次乘法和一次减法运算得到,因此每次迭代计算所需的时间固定,记为 $Time1$. 假设将并行度增大到 $m+e$,所需的迭代次数变为 $k/(m+e)$,增大并行度之后迭代过程中减少的时间为 $(k/m-k/(m+e))\times Time1$. 与此同时,迭代开始前所需计算的多项式系数个数增加,由 $2m$ 增加至 $2(m+e)$. 为了保证增大并行度能够带来计算时间的增益,此时需要关注两点: 1) 所需计算的多项式系数增加,硬件平台是否有足够的线程资源来支撑它们的计算,而无需等待前面的计算结束并完成资源释放; 2) 增加的多项式系数计算时间记为 $Time2$,需要保证 $Time2\leqslant(k/m-k/(m+e))\times Time1$. 通过上述两点的考量对并行度进行设置,可以在硬件上实现资源的充分利用. 本文中切比雪夫多项式迭代过程的线程并行度 m 设置为 3. 相对于串行的迭代过程,采用并行计算的方式灵活度更高,计算效率能够获得显著提升.

4　实验分析

采用来自 MGH-USCHCP 公共数据集库[16] 医学脑图数据集来进行模型的训练,数据集图片大小处理成 64×64. 优化前后的网络采用相同的训练环境: CPU 为 Intel(R) Xeon(R) E5-4610 V4,总内存为 15G; GPU 为 Tesla P100,显存为 256G; Tensorflow 版本为 1.8; CUDA 版本为 9.0. 优化后的网络进行重构验证采用的推理环境: CPU 为 Core i7-7700,内存为 32G; GPU 为 Nvidia GeForce GT 730,显存为 2G; CUDA 版本为 9.0. 将优化后的模型与原模型进行对比,验证优化后的模型重构性能,采用峰值信噪比(Peak Signal to Noise Ratio, PSNR)来评估重构图像的质量. 然后将优化后的模型搭载在 Jetson AGX Xavier 嵌入式平台上并进行加速效果分析. Jetson AGX Xavier 是 NVIDIA 推出的一款高性能计算开发板,其中支持 CUDA 7.0,GPU 为 NVIDIA Volta 架构,CPU 为 8 核 Carmel ARM v8.2,提供了两个深度学习加速器引擎,可提供 30TOPs 的深度学习运算能力,是目前深度神经网络实现最佳的 GPU 嵌入式平台.

4.1　网络优化前后计算复杂度分析

DNN 的时间复杂度和空间复杂度是评价网络模型架构的重要指标. 时间复杂度决定了其进行训练与推理的用时,时间复杂度高时模型训练时间长使得研究者们的想法无法得到快速验证,模型的改善时间也会相应拉长. 因此网络的时间复杂度是研究者们关注的评价指标,同时在近似计算中时间复杂度也是一个重要指标. 空间复杂度决定了网络模型的参数数量,网络进行硬件移植时需要考虑到硬件平台有限的资源是否能够满足网络部署的要求.

由于 DNN 中时间复杂度主要来源于网络卷积层,因此主要进行卷积层的时间复杂度分析,公式(11)表示单个卷积层的时间复杂度,其中 M 表示输出特征图的大小,S 表示卷积核的大小,C_{in} 和 C_{out} 分别表示卷积层输入和输出的通道数:

$$Complex_{Time}=O(M^2\times S^2\times C_{in}\times C_{out}) \tag{11}$$

由公式(9)可以得出优化后网络的卷积层时间复杂度仅与该层卷积核参数数量有关,如公式(12):

$$Complex_{Time}=O((S^2\times C_{out})^2) \tag{12}$$

卷积层的空间复杂度计算公式如式(13)所示:

$$Complex_{Space}=O(S^2\times C_{in}\times C_{out}) \tag{13}$$

由公式(11)和公式(12)可以计算得出优化前后网络各层的时间复杂度如表 1 所示. 卷积层 1 和反卷积层为卷积部分的输入和输出层,由于在这两层网络中存在输入或者输出层通道数为 1 的情况,因此表 1 中优化后的模型卷积层 1 和反卷积层的时间复杂度下降不多,但是卷积层 2 的时间复杂度下降了 125.44 倍. 可以得出通过优化后网络中间隐藏的卷积层的复杂度能够获得明显的时间复杂度下降.

表 1　网络优化前后各卷积层时间复杂度

Table 1　Time complexity of convolutional layers of the network before and after network optimization

网络模型	卷积层 1	卷积层 2	反卷积层
原始模型	O(5760000)	O(321126400)	O(12845056)
优化后模型	O(2560000)	O(2560000)	O(9834496)

由公式(9)可知,卷积操作被转换为切比雪夫多项式迭代,网络中的参数被转换为多项式迭代中的多项式系数,而此时需要预先计算的多项式系数个数由并行度 m 来决定,迭代开始前需要计算 $2m$ 个多项式系数值,因此并行处理后的网络卷积层空间复杂度如公式(14)所示:

$$Complex_{Space}=O((2m)^2\times C_{in}\times C_{out}) \tag{14}$$

由于 m 远小于原始卷积核的大小 S,因此经过切比雪夫迭代处理后,各卷积核的空间复杂度明显下降.

4.2　优化后的网络模型重构效果验证

优化前后的模型训练方式一致,批次数据量大小设置为 100,学习率为 0.00002,采用 RMSProp 算法实现梯度下降,以加快收敛速度. 采用均方误差作为损失函数.

图 5 为训练过程中两个模型损失-迭代曲线图,可以看出优化后的模型起始损失值要大于原始模型,随着迭代次数增加,两条曲线逐渐贴近,最终能够达到相近的重构性能.

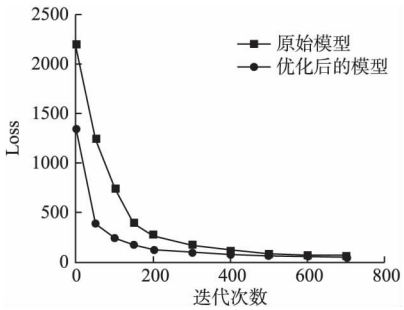

图 5　两种模型损失-迭代曲线

Fig. 5　Loss-iteration curves of the two models

图 6 中,左边两张图为网络输入图像,右边两张图为优化网络的重构结果,重构出的图像 PSNR 能达到 30dB 左右,可以证明优化后的模型可以成功的完成图像重构任务.

4.3　优化后模型的硬件加速实现

实验通过远程交叉编译将模型导入到嵌入式平台上,以文本读取的方式加载权重参数来实现网络的推理过程. 采用 CUDA 中 nvprof 工具对并行加速设计后的网络模型推理一张图片的过程进行整体分析. 网络硬件实现结果如表 2 所示,

显示了 warp 占用率(*Achieved Occupancy*,*AO*) ,全局内存负载吞吐量(*Global Memory Load Throughput*,*GMLT*) ,全局内存负载效率(*Global Memory Load Efficiency*,*GMLE*) ,全局内存存储效率(*GlobalMemoryStoreEfficiency*,*GMSE*) ,所有参

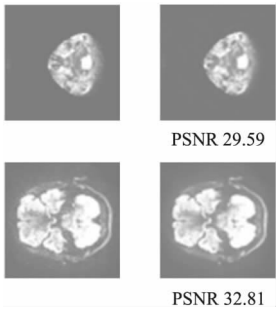

图 6 重构结果

Fig. 6 Reconstruction results

数都取平均值,如公式(15) -公式(17) 所示. 其中活跃线程数量为 n,最大线程数量为 $Max\_threads$,请求的全局内存负载吞吐量为 $GMLT\_Q$,所需的全局内存负载吞吐量为 $GMLT\_N$,请求的全局内存存储吞吐量为 $GMST\_Q$,所需的全局内存存储吞吐量为 $GMST\_N$.

$$AO = \frac{n}{Max\_threads} \tag{15}$$

$$GMLE = \frac{GMLT\_Q}{GMLT\_N} \tag{16}$$

$$GMSE = \frac{GMST\_Q}{GMST\_N} \tag{17}$$

本文基于重构网络进行并行化处理,实验中数据集大小均为 64 × 64,以固定的分辨率图像进行重构. 由于完成对网络的并行化设计后,硬件平台上线程调度的方式被确定下来,相应的资源使用方案和计算中的操作数均固定下来,因此对同样分辨率的图像进行重构时,各参数指标理论上保持一致,此处以一张图像的重构为例,对网络的硬件加速情况进行分析.

表 2 网络硬件实现分析

Table 2 Hardware implementation analysis of network

核函数	AO(%)	GMLT(GB/S)	GMLE(%)	GMSE(%)
激活函数	83.5119	24.94	100	100
全连接层计算	17.8268	452.60	100	100
卷积层计算	18.1159	767.59	25.17	34.38

表 2 中可以看到激活函数(此处的激活函数为卷积层的激活函数,*Relu*) 的平均占用率最高,达到了 83.5119%,表明该核函数中 warp 的利用最充分. 卷积层计算的核函数的全局内存负载吞吐量最高,达到了 767.59GB/S,同时全连接层核函数和卷积层核函数的全局内存负载效率、全局内存存储效率都达到了 100%,充分利用了设备内存带宽.

由于全连接层 1 为输入层,因此计算从全连接层 2 开始. 由表 3 可得,除去网络中各层进行权重矩阵读取的时间,整个网络模型运行时间为 0.35s.

表 4 展示了原始网络在 Tensorflow 环境下进行推理的时间和并行加速设计后的网络在嵌入式开发板上进行重构推理的时间,可以看出并行加速设计后的网络在嵌入式开发板上进行推理的速度比原始网络在 PC 端推理的速度快 2.2 倍.

表 3 网络运行时间

Table 3 Running time of the network

核函数	核函数计算时间(s)	各层运行总时间(s)
全连接层 2	0.100392	0.198768
全连接层 3	0.044379	0.053295
卷积层 1	0.002751	0.006069
卷积层 2	0.037086	0.041219
反卷积层	0.043530	0.054631

表 4 推理时间对比

Table 4 Comparison of the inference time

模型	原始网络模型	并行处理后模型	加速倍数
推理时间(s)	0.77	0.35	2.2

5 总 结

本文针对深度神经网络中数据负荷大,网络卷积层计算复杂度很高的问题,提出采用切比雪夫多项式对网络卷积核进行逼近. 通过将卷积核进行展开,使得其与输入数据的卷积操作被转换成矩阵向量乘法的形式,能够无需遍历操作直接计算出卷积结果,然后采用切比雪夫多项式对卷积核中参数进行迭代拟合,使得卷积操作能够实现并行化处理. 在网络优化后的基础上,基于 CUDA 编程架构为卷积层切比雪夫多项式迭代设计了对应的并行加速方案. 最后将网络移植到 NVIDIA Jetson AGX Xavier 嵌入式开发板上,其在开发板上进行推理的时间仅为 0.35s. 相较于原始网络的推理可以获得 2.2 倍的加速.

References:

[1] Chellapilla K,Puri S,Simard P. High performance convolutional neural networks for document processing [C] //10th International Workshop on Frontiers in Handwriting Recognition,2006.

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

[3] Reddy B,Kim Y H,Yun S,et al. Real-time driver drowsiness detection for embedded system using model compression of deep neural networks [C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops,2017: 121-128.

[4] Stanoev A,Audinet N,Tancock S,et al. Real-time stereo vision for collision detection on autonomous UAVs [C] //2017 IEEE International Conference on Imaging Systems and Techniques(IST) ,IEEE,2017: 1-6.

[5] Ghazi P,Happonen A P,Boutellier J,et al. Embedded implementation of a deep learning smile detector [C] //2018 7th European Workshop on Visual Information Processing (EUVIP) , IEEE, 2018: 1-6.

[6] Motamedi M,Fong D,Ghiasi S. Fast and energy-efficient cnn inference on iot devices [J] . arXiv preprint arXiv: 1611.07151,2016.

[7] Motamedi M,Fong D,Ghiasi S. Machine intelligence on resource-constrained IoT devices: the case of thread granularity optimization

for CNN inference [J]. ACM Transactions on Embedded Computing Systems(TECS),2017,16(5s):1-19.

[8] Motamedi M,Fong D,Ghiasi S. Cappuccino: efficient cnn inference software synthesis for mobile system-on-chips [J]. IEEE Embedded Systems Letters,2018,11(1):9-12.

[9] Jin K H,McCann M T,Froustey E,et al. Deep convolutional neural network for inverse problems in imaging [J]. IEEE Transactions on Image Processing,2017,26(9):4509-4522.

[10] Schlemper J,Caballero J,Hajnal J V,et al. A deep cascade of convolutional neural networks for dynamic MR image reconstruction [J]. IEEE Transactions on Medical Imaging,2017,37(2):491-503.

[11] Rivenson Y,Zhang Y,Günaydın H,et al. Phase recovery and holographic image reconstruction using deep learning in neural networks [J]. Light: Science & Applications,2018,7(2):17141,doi: 10.1038/lsa.2017.141.

[12] Lu H,Li Y,Uemura T,et al. Low illumination underwater light field images reconstruction using deep convolutional neural networks [J]. Future Generation Computer Systems,2018,82:142-148.

[13] Mason J C,Handscomb D C. Chebyshev polynomials [M]. Chapman and Hall/CRC,2002.

[14] Zhu B,Liu J Z,Cauley S F,et al. Image reconstruction by domain-transform manifold learning [J]. Nature,2018,555(7697):487-492.

[15] Sanders J,Kandrot E. CUDA by example: an introduction to general-purpose GPU programming [M]. Addison-Wesley Professional,2010.

[16] Fan Q,Witzel T,Nummenmaa A,et al. MGH-USC human connectome project datasets with ultra-high b-value diffusion MRI [J]. Neuroimage,2016,124:1108-1114.

征 稿 简 则

一、征稿范围 《小型微型计算机系统》杂志刊登文章的内容涵盖计算技术的各个领域(计算数学除外). 包括计算机科学理论、体系结构、计算机软件、数据库、网络与通讯、人工智能、信息安全、多媒体、计算机图形与图像、算法理论研究等各方面的学术论文.

二、来稿要求: 本刊主要刊登下述各类原始文稿:

1. 学术论文: 科研成果的有创新、有见解的完整论述. 对该领域的研究与发展有促进意义. 论文字数最好在 10000 字左右.

2. 综述: 对新兴的或活跃的学术领域或技术开发的现状及发展趋势的全面、客观的综合评述(各类综述稿件一经录用三个月见刊).

3. 技术报告: 在国内具有影响的重大科研项目的完整的技术总结.

三、注意事项

1. 来稿务求做到论点明确、条理清晰、数据可靠、叙述简练,词义通达.

2. 来稿必须是作者自己的科研成果,无署名和版权争议. 引用他人成果必须注明出处.

3. 本刊采用在线投稿方式,可登陆 http://xwxt.sict.ac.cn/进行在线投稿.

4. 格式要求: 题目(中、英文)、摘要(中、英文)、作者的真实姓名(中、英文)、作者的单位、城市(中、英文)、邮政编码、E-mail(便于联系的)、关键词(中、英文 4~7 个),中图分类号、作者简介、基金项目.

(1) 英文部分的作者姓名使用汉语拼音,单位英文名称须给出英文全称,不要使用缩略语;

(2) 作者简介包含作者姓名、性别、出生年、最高学历、技术职称、研究方向(若作者中有中国计算机学会(CCF)会员,请注明,并给出会员号). 凡第一作者为 CCF 会员/高级会员/学生会员者,并在初次上传稿件中注明的,将享受八五折的版面费优惠;

(3) 基金项目的类别与项目编号.

5. 中、英摘要: 文章摘要具有独立性和自明性,含正文等量的主要信息,一般为 250~300 字,采用第三人称表述.

6. 参考文献: 未公开发表的文献不得列入. 文后所列参考文献统一排序,且必须在正文中引用. 中文参考文献应给出对应的英文译文. 其具体书写格式为:

(1) 图书 [编号] 作者姓名(姓在前,名在后),书名,出版社地址,出版社,出版年.

(2) 期刊 [编号] 作者姓名、文章题目、刊物名称,出版年,卷号(期号):起止页码.

(3) 会议论文 [编号] 作者姓名. 论文题目. 见: 编者、论文集全名、出版地: 出版者,出版年,起止页码.

(4) 网络文献: 请给出文献作者或单位名,文章题目、网址、发布日期.

7. 插图和表: 插图必须精绘并用计算机激光打印,一般不超过 7 幅. 图应结构紧凑,不加底纹,不要做成彩色的,图宽最好不超过 8 厘米,图内字号统一使用 6 号宋体,字迹、曲线清晰,必要时给出坐标名称和单位. 每个图、表均给出中英文图注(如 "图 1: ＊＊＊图;" "Fig. 1: ＊＊＊") 和表注(如 "表 1: ＊＊＊表", "Table 1: ＊＊＊").

8. 计量单位: 稿件中一律使用《中华人民共和国法定计量单位》. 外文和公式中应分清大、小写和正、斜体,上、下角的字母、数码位置准确. 易混淆的字母或符号,请在第一次出现时标注清楚.

9. 本刊在收到作者稿件经初审后立即给作者电子邮箱发"稿件收到通知". 除作者另有明确要求外,本刊原则上只与第一作者联系,作者投稿后若 4 个月无消息,可自行改投它刊. 通过初审的稿件将收到本刊给予的编号,并需邮寄审稿费.

10. 本刊对不拟录用的稿件只发给"退稿通知",恕不退回原稿,请自留底稿.

11. 稿件一经发表,将酌致稿酬,并寄送样刊.

本刊文章现被国内外多家数据库收录,作者著作权使用费与本刊稿酬一并给付,作者若不同意将文章收录,请在投稿时说明.

编辑部地址: 沈阳市浑南区南屏东路 16 号《小型微型计算机系统》编辑部　邮政编码: 110168

电　话: 024-24696120　　E-mail: xwjxt@sict.ac.cn　　网　址: http://xwxt.sict.ac.cn