

面向内存受限设备的新型卷积计算方法

孙雁飞^{1,2}, 王子牛³, 孙莹^{2,4}, 齐晋^{1,2}, 董振江^{2,4}

- 1.南京邮电大学 物联网学院,江苏 南京 210003
- 2.南京邮电大学 江苏省高性能计算与智能处理工程研究中心,江苏 南京 210023
- 3.南京邮电大学 自动化学院、人工智能学院,江苏 南京 210023
- 4.南京邮电大学 计算机学院,江苏 南京 210023

摘要:针对卷积神经网络预测过程中内存使用量大,难以部署在内存受限设备上的问题,提出一种面向内存受限设备的新型卷积计算方法。该方法对输入矩阵中部分数据进行卷积计算,并将计算结果存储在临时内存;然后,将临时内存中的计算结果复制到输入矩阵不再使用的内存并重复上述步骤,从而实现对输入矩阵的卷积计算;最后,对单个卷积计算和 LeNet 进行验证。实验结果表明,该方法计算速度较直接卷积方法更快,且相比 im2col、MEC 和直接卷积方法,单个卷积计算内存平均使用量分别下降 89.29%、82.60% 和 57.15%,LeNet 内存使用量分别下降 89.90%、82.21% 和 28.07%,有效降低了卷积神经网络的内存使用量,有助于在内存受限设备上部署使用。

关键词:深度学习;卷积计算;内存优化;数据复用;边缘设备

中图分类号:TP391 **文献标志码:**A **文章编号:**1673-5439(2022)05-0054-08

A novel convolution calculation algorithm on memory-limited devices

SUN Yanfei^{1,2}, WANG Ziniu³, SUN Ying^{2,4}, QI Jin^{1,2}, DONG Zhenjiang^{2,4}

- 1.School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
- 2.Jiangsu HPC and Intelligent Processing Engineer Research Center, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
- 3.College of Automation & College of Artificial Intelligence, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
- 4.School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

Abstract: In the prediction process of convolutional neural network, the memory consumption is large and it is difficult to deploy on memory-limited devices. This paper presents a novel convolution calculation algorithm for memory-limited devices. In this method, part of data in the input matrix is convolved and the result is stored in the temporary memory. Then, the calculation result in the temporary memory is copied to the memory no longer used by the input matrix and the above steps are repeated, so as to realize the convolution calculation of the input matrix. Finally, the single convolution calculation and LeNet are verified. The experimental results show that the average memory usage of single convolution calculation is reduced by 89.29%, 82.60% and 57.15%, and the memory usage of LeNet is reduced by 89.90%, 82.21% and 28.07% compared with im2col, MEC and direct convolution methods, respectively, when the calculation speed is faster than that of direct convolution method. It effectively reduces the memory usage of convolutional neural networks, which is helpful for the deployment on memory-limited devices.

Keywords: deep learning; convolution calculation; memory optimization; data reuse; edge device

卷积神经网络因其出色的特征提取能力,在图像分类、目标检测、语义分割等计算机视觉领域被广泛应用并发挥重要作用^[1-2]。但卷积神经网络需要较高的计算复杂度和内存消耗量,通常部署在云端进行计算,这在一定程度上限制了卷积神经网络的应用场景。近年来,随着物联网设备的普及,这些设备产生的数据量呈现爆炸式增长,给云端计算带来极大的负担。此外,将数据传输到云端进行计算也带来了数据安全和响应延迟等问题^[3]。为解决上述问题,边缘智能的概念被提出^[4-5],并广泛应用于预测性维护、精准农业、智慧城市等领域。

边缘智能是边缘计算和人工智能的结合,当前研究主要集中在轻量级推理框架^[6-7]和轻量级卷积神经网络^[8-9]。轻量级推理框架提供了在边缘设备上部署卷积神经网络的方法;轻量级卷积神经网络研究高效的网络结构,降低卷积神经网络对算力和内存的需求。因此,如何将卷积神经网络部署在边缘设备并进行低复杂度、小内存计算,已成为当前的研究热点。

尽管在边缘设备上部署卷积神经网络成为可能,但对于内存受限的边缘设备如微控制器、数字信号处理器等,其内存小于 1 MB,难以满足卷积神经网络的内存需求。此外,内存使用量的增加使得 SRAM (Static Random Access Memory) 的漏电流增大^[10],从而缩减这些设备的续航时间。因此,有必要进一步研究降低卷积神经网络内存使用量的方法,从而满足边缘设备的内存约束条件。因此,本文旨在面向内存受限设备,提出一种低内存使用的新型卷积计算方法。

1 相关工作

目前降低卷积神经网络内存使用量的研究主要集中在轻量化卷积神经网络,通过设计特殊的卷积操作,降低网络结构的冗余,从而减少内存使用量。然而,卷积神经网络的内存使用量不仅取决于卷积神经网络结构,还取决于组成卷积神经网络的卷积算子计算方法,不同计算方法的内存使用量可以相差数十倍,有必要研究降低卷积计算内存使用量的方法。现有卷积计算方法主要包括直接卷积、im2col+GEMM 卷积和快速卷积等。

(1) 直接卷积:直接卷积根据卷积算子的定义,通过几层嵌套循环进行计算,输入矩阵中灰色窗口与卷积核的内积即为输出的一个元素,存放在输出矩阵中对应位置。灰色窗口按照步长参数从左到

右,从上到下滑动,依次计算出对应位置输出结果,直至计算全部数据。使用该方法卷积时只需分配存放输出矩阵的内存空间,如图 1 所示。

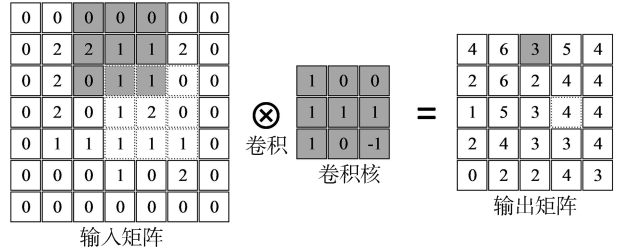


图 1 直接卷积

文献[11]通过动态编译方法在 x86 架构上优化直接卷积实现,其性能接近于理论水平;文献[12]通过优化直接卷积数据布局和计算中的循环顺序,提高了直接卷积的计算速度。然而上述方法并未降低直接卷积的内存使用,难以部署在内存受限设备上。

(2) im2col+GEMM 卷积:该方法依靠 im2col 转换将卷积问题转化为通用矩阵乘 (General Matrix Multiplication, GEMM) 问题,利用高度优化后的线性代数库,如 MKL、BLAS 等加速卷积计算^[13]。该方法被主流深度学习框架如 TensorFlow、Pytorch、Caffe、MxNet 等深度学习框架采用作为卷积计算方法^[14]。im2col 将输入矩阵中滑动窗口内元素展开成行向量,将卷积核展开成列向量,分别存储在两个中间矩阵中,这两个矩阵的乘积即为输出矩阵,如图 2 所示。

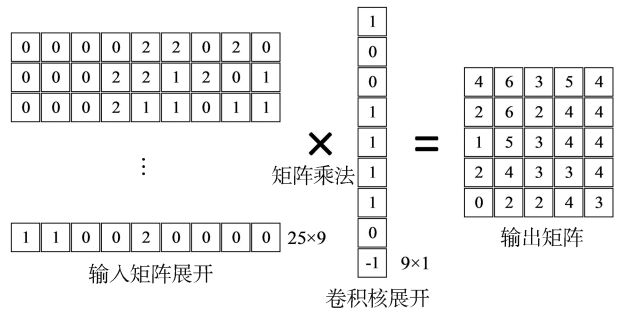


图 2 im2col+GEMM 卷积

基于 im2col+GEMM 的方法需要构造中间矩阵,空间复杂度为 $O(K^2CHW)$, 其中 K 、 C 、 H 、 W 分别表示卷积核高或宽、输入通道数、输出矩阵高、输出矩阵宽。对于 3x3 的卷积核,输入矩阵中每个元素被复制 9 次,即输入矩阵转换后的中间矩阵内存使用量变为原来的 9 倍。基于 im2col+GEMM 改进的方法,如 MEC^[15]、MFCA^[16]等,有效降低了中间矩阵的

大小,但相较于直接卷积方法仍需要大量额外内存空间存放中间矩阵,因此并不适用于内存受限设备。

(3) 快速卷积:该类方法主要包括 FFT (Fast Fourier Transform)^[17] 和 Winograd^[18] 方法,通过将输入矩阵和卷积核映射到对应的 FFT 空间和 Winograd 空间,将计算转换成对应元素相乘,再将结果逆线性变换映射到原空间即可得到卷积计算输出,从而减少卷积计算的计算量。然而此类算法由于其内在的数值不稳定性,导致计算精度降低,仅适用于单位步长和卷积核小的情况。文献[19]在 Winograd 的基础上提出使用超线性多项式来构造变换矩阵,提高了计算精度,缓解数值不稳定性。但基于快速卷积的方法仍需额外的内存存储输入矩阵和卷积核映射后的空间以及映射空间中的计算结果,同样也不适用于内存受限设备。

此外,内存共享优化^[20]被广泛用于 TensorFlow、Caffe、MXNet 等深度学习框架,通过记录卷积神经网络中间各层输出的大小和生命周期,回收不再使用中间结果的内存并用于其他层,从而降低卷积神经网络的内存消耗。内存共享优化属于卷积神经网络中不同层之间的内存优化,并不涉及层内卷积计算方法,本文方法属于层内的卷积计算优化方法,两者可以组合使用,进一步优化内存使用量,即混合内存重用策略^[21]。

综上所述,现有卷积计算方法主要对卷积计算速度进行优化,通过使用额外的内存空间加速卷积计算过程,在此基础上研究降低卷积内存使用量方法,但对于内存资源受限的设备来说,无法承担这种优化带来的额外内存开销。针对上述问题,本文提出一种面向内存资源受限设备的新型卷积计算方法,与直接卷积方法类似,根据卷积算子的定义直接进行卷积计算,但计算结果存放在输入矩阵中,从而降低卷积计算的内存使用量;与 im2col+GEMM 卷积和快速卷积不同,本文方法无需构造中间矩阵,避免大量的内存使用,大幅降低卷积计算的内存使用量。

2 算法实现

针对现有卷积计算方法内存使用量大,难以满足内存受限设备的内存约束的问题,本节将对卷积计算内存使用量进行优化,阐述通过复用输入矩阵内存从而降低卷积计算内存使用量的一种新型卷积计算方法,并给出以下几种卷积计算的具体实现方法。

2.1 标准卷积

该算法对输入矩阵滑动窗口对应的部分与卷积

核直接进行卷积计算,并将计算结果存放在输入矩阵中,避免或减少分配输出矩阵的内存。如图 3 所示,对于输入矩阵 $I(i_h \times i_w \times i_c = 6 \times 6 \times 1)$,卷积核 $K(k_h \times k_w \times k_c = 3 \times 3 \times 1)$,其中下标 h, w, c 分别表示对应矩阵高、宽和通道数。滑动窗口每次滑动计算完结果后,窗口左上角元素在后续计算中不会被再次使用,因此可将计算结果存放在对应窗口左上角位置,继续滑动窗口直至计算完全部输入矩阵,此时输入矩阵中存放着输出矩阵的内容,输入矩阵变为输出矩阵,不再需要额外分配内存存放输出矩阵,从而降低卷积过程内存使用量。图中浅灰色部分表示尚未计算的区域,深灰色部分表示正在计算的区域。

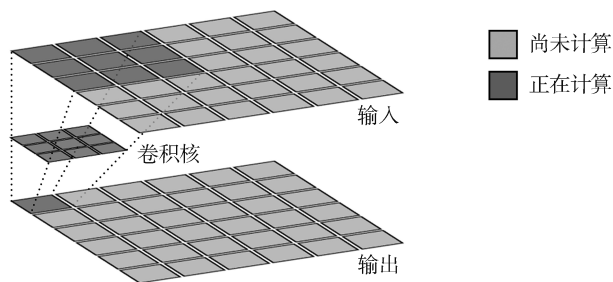


图 3 标准卷积计算方法示意图

在更一般的情况下,若输入矩阵为 $I(i_h \times i_w \times i_c)$,卷积核为 $K(k_h \times k_w \times k_c)$,输出矩阵为 $O(o_h \times o_w \times o_c)$ 时,可以分为两种情况讨论,用 Mem 表示矩阵占用内存的大小。

情况一: $\text{Mem}_I \geq \text{Mem}_O$, 此时输入矩阵可以存放全部计算结果,按照如下步骤计算卷积。

① 分配内存空间 m , 大小为 $h \times o_w \times o_c (h \geq \lceil k_h/2 \rceil)$, 临时缓存计算结果,如图 4(a) 所示。

② 计算输入矩阵中深灰色行卷积,计算结果存放在 m 中,如图 4(b) 所示,其中第一行虚线框表示卷积计算时 padding 填充部分示意,并不占据实际内存空间。

③ 此时输入矩阵中白色行的输入数据将不再被后续计算使用,将 m 中对应行的计算结果复制到该位置,如图 4(c) 所示,白色部分存放卷积计算结果。值得注意的是,当 $k_h > 1$ 时,输入矩阵中深灰色行除了参与以当前行为卷积中心的计算外,还会参与以相邻行为卷积中心的计算,因此不能直接将深灰色行的输入数据直接覆盖。

④ 重复步骤②、③,如图 4(d)、(e) 所示,直至计算完全部输入矩阵。

⑤ 计算结束后的结果如图 4(f) 所示,此时卷积

后的结果全部存放在输入矩阵中,卷积计算完成。

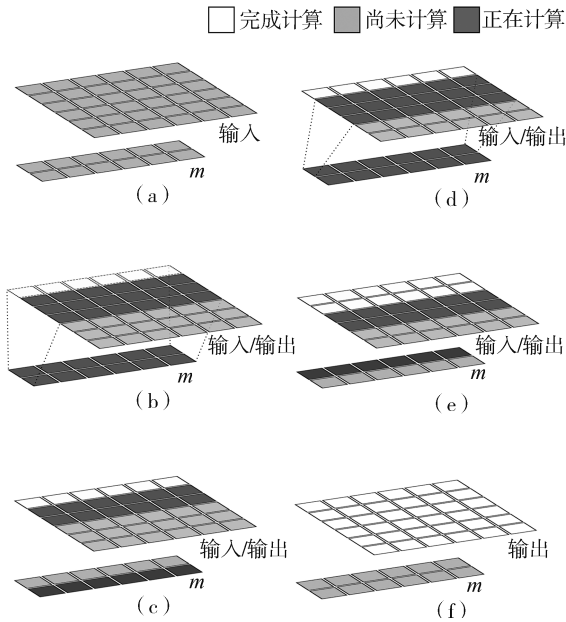


图 4 标准卷积计算方法情况一

情况二: $\text{Mem}_I < \text{Mem}_O$, 此时输出矩阵通道数大于输入矩阵通道数,按照如下步骤计算卷积。

① 分配内存空间 m , 大小为 $h \times o_w \times o_c (h \geq \lceil k_h/2 \rceil)$; 分配内存空间 M , 大小为 $\text{Mem}_O - \text{Mem}_I$, 如图 5(a)所示。

② 计算输出矩阵比输入矩阵多出来的通道,采用直接计算方法,计算结果存放在 M 中,如图 5(b)所示。

③ 忽略多出来的通道数和内存空间 M , 按照情况一的步骤②~④执行,如图 5(c)所示。

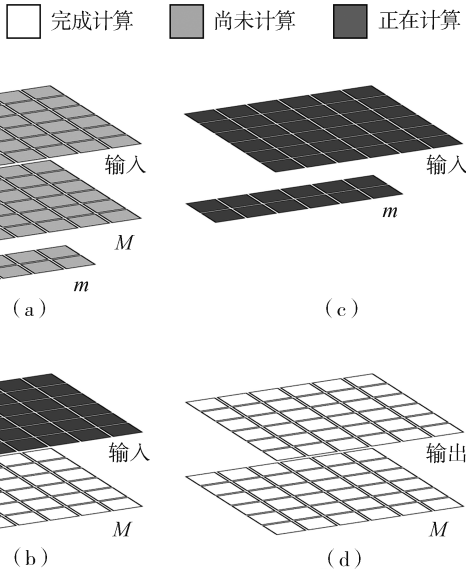


图 5 标准卷积计算方法情况二

④ 最终计算结果结果如图 5(d)所示,此时计算结果分为两部分存放,分别存放在输入矩阵和内存空间 M 中,卷积计算完成。

标准卷积计算方法的算法描述如算法 1 所示。

算法 1 标准卷积计算方法

输入:输入矩阵 I

输出:输出矩阵 O

1. if $\text{Mem}_I \geq \text{Mem}_O$
2. 分配内存空间 m
3. for line in I :
4. 计算卷积结果存放在内存空间 m 中
5. 释放 line 中输入数据
6. 将数据从内存空间 m 中复制到 line 中
7. end for
8. return $O = I$
9. else
10. 分配内存空间 m, M
11. 计算卷积结果存放在内存空间 M 中
12. for line in 尚未计算的 I :
13. 计算卷积结果存放在内存空间 m 中
14. 释放 line 中输入数据
15. 将数据从内存空间 m 中复制到 line 中
16. end for
17. return $O = [I, M]$
18. end if

2.2 深度卷积

深度卷积的输入通道个数与卷积核数量一致,输入通道与对应卷积核进行卷积计算,结算结果为对应输出通道,输入输出通道数保持一致。深度卷积计算可以看作多个通道数为 1 的标准卷积组合,因此可以使用上述提出标准卷积的计算方法。使用深度卷积时,输入通道数和输出通道数相等,输出尺寸小于或等于输入尺寸,因此并不需要分配内存空间 M , 即深度可分离卷积只需要采用标准卷积中情况一的计算方法。

2.3 点卷积

点卷积可以看作标准卷积中 $k_h = k_w = 1$ 的情况。点卷积计算不涉及输入矩阵中相邻行列的值,输入矩阵中对应位置数据计算完后将不再使用,因此,与标准卷积计算不同,点卷积计算时不再额外分配内存空间 m , 计算结果可以直接存放在输入矩阵中。点卷积计算方法如图 6 所示,左上角元素与点卷积核计算后计算结果直接存放在原来位置,图中输入输出矩阵为同一块内存空间。当 $\text{Mem}_I < \text{Mem}_O$ 时,仍需分配内存空间 M 存放额外的输出结果,其大小为 $\text{Mem}_O - \text{Mem}_I$ 。

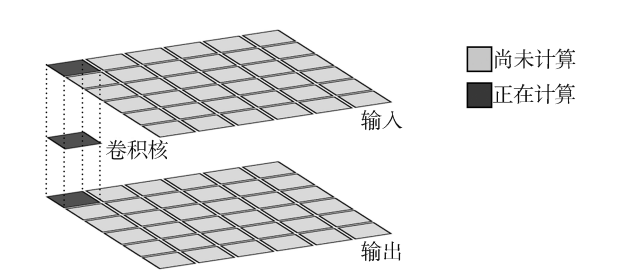


图 6 点卷积计算方法

除了上述几种卷积计算外,本节还提出一种降低卷积神经网络中池化计算内存使用量的方法。池化计算方法如图 7 所示,上层矩阵为输入矩阵,下层为输出矩阵,池化窗口每次移动计算后的结果依次存放在输出矩阵对应的位置上。由于计算后输入空间中对应区域的数据将不再被后续计算用到,即图中白色区域,因此该区域可以依次存放其他位置的池化结果。值得注意的是,图 7 中上下两层矩阵实际上是同一块内存不同时间上的状态,在池化计算前代表输入矩阵,池化计算结束后代表输出矩阵,因此本节提出的池化计算方法不需要使用额外内存。

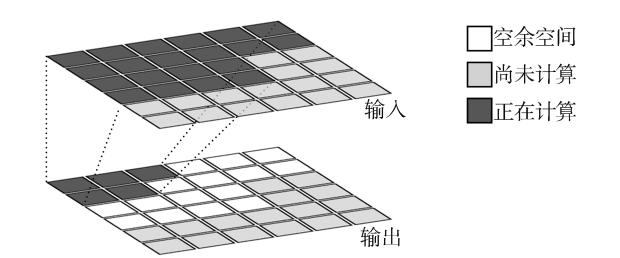


图 7 池化计算方法

3 实验分析

实验测试硬件平台选择在内存资源受限的微控制器上进行,选用微控制器的型号为 STM32H750VBT6,主频为 480 MHz,测试内存包括 DTCM(Data Tightly-Coupled Memory)和 SRAM,其中 DTCM 大小为 128 KB,与微控制器内部 CPU 直接连接,访存速度较快;SRAM 大小为 512 KB,通过 AXI 总线与 CPU 相连,访存速度较慢。使用 C++实现本文提出的卷积计算方法和其他对比方法。由于这类内存资源受限的设备通常没有 GPU 等硬件加速功能,因此基于 im2col+GEMM 方法的卷积实验和其他方法均使用微控制器内部 CPU 进行计算。

3.1 实验设置

实验测试数据包括单个卷积测试和 LeNet^[22]卷积神经网络测试,并分别对其进行内存使用量分析

和性能分析,验证本文提出方法的有效性。其中单个卷积测试集参考文献[15]给出的测试集,测试集信息如表 1 所示。

表 1 测试集信息

卷积	输入矩阵 $i_h \times i_w \times i_c$	卷积核 $k_h \times k_w \times o_c$
CV1	7×7×64	3×3×128
CV2	14×14×32	3×3×64
CV3	28×28×16	3×3×32
CV4	56×56×8	3×3×16
CV5	112×112×4	3×3×8
CV6	224×224×1	3×3×2
CV7	16×16×32	5×5×64
CV8	32×32×16	5×5×32
CV9	64×64×8	5×5×16
CV10	64×64×4	1×1×12
CV11	128×128×3	1×1×4
CV12	256×256×1	1×1×1

3.2 内存使用量分析

单个卷积测试集内存开销大小如表 2 所示,其中内存使用量包括卷积计算过程中的额外内存使用量和输出矩阵的内存使用量,不包含输入矩阵和卷积核的内存使用量, M_{im2col} 、 M_{MEC} 、 $M_{\text{direct conv}}$ 和 M_{ours} 分别表示 im2col+GEMM、MEC、直接卷积和本文方法内存使用量大小,单位为字,即单个 float32 类型参数的内存大小。

表 2 单个卷积计算内存使用量对比

卷积	M_{im2col}	M_{MEC}	$M_{\text{direct conv}}$	M_{ours}
CV1	17 600	9 920	3 200	1 344
CV2	50 688	25 344	9 216	4 480
CV3	118 976	56 576	21 632	10 752
CV4	256 608	119 232	46 656	23 296
CV5	532 400	244 640	96 800	48 384
CV6	542 124	247 752	98 568	49 280
CV7	124 416	39 936	9 216	3 328
CV8	338 688	96 768	25 088	11 392
CV9	77 7600	211 200	57 600	27 712
CV10	65 536	65 536	49 152	33 536
CV11	114 688	114 688	65 536	16 896
CV12	131 072	131 072	65 536	256

从表 2 中可以看出,本文方法明显优于其他几种卷积方法,大幅减少卷积的内存使用量。对比其他几种方法,本文方法的平均内存使用量分别下降 89.29%、82.60%和 57.15%,减少的内存使用量主要是由于对输入矩阵内存空间进行复用,减少分配输出矩阵内存大小,同时避免构造中间矩阵使用量额外内存。以 CV1 为例,im2col+GEMM 方法需要构造输入矩阵的中间矩阵,内存大小为 14 400 字 ($o_h \times o_w \times k_h \times k_w \times i_c$),分配输出矩阵,内存大小为 3 200 字 ($o_h \times o_w \times o_c$),共计需要 17 600 字内存;MEC 方法需要构造输入矩阵的中间矩阵,内存大小为 6 720 字 ($o_h \times k_h \times i_h \times i_c$),分配输出矩阵,内存大小为 3 200 字,共计需要 9 920 字内存;直接卷积方法无需构造中间矩阵,仅需分配 3 200 字的输出矩阵;本文方法需要分配内存空间 m ,大小为 1 280 字 ($\lceil k_h/2 \rceil \times o_w \times o_c$),此外还需要分配内存空间 M ,大小为 64 字 ($\text{Mem}_o - \text{Mem}_i$),共计需要 1 344 字内存。为方便对比,对表 2 中几种方法内存的使用量用直方图进行比较,以 im2col+GEMM 算法的内存使用量为基准,结果如图 8 所示。

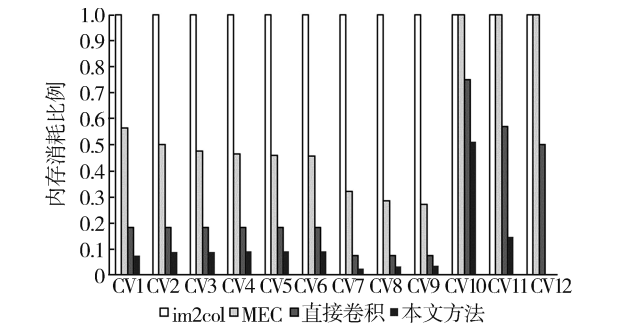


图 8 几种卷积方法的内存开销柱状图比较

除了对单个卷积进行实验分析外,本文还对经典卷积神经网络 LeNet 进行实验分析。LeNet 是一种用于数字识别的卷积神经网络,其结构简单,内存消耗少,方便对不同的卷积计算方法进行分析。表 3 为 LeNet 卷积神经网络结构参数及不同卷积计算方法的内存开销对比。

由于卷积神经网络输入为图片,第一层卷积无法使用输入空间存放卷积结果,因此本文方法在第一次卷积时和直接卷积采用相同的方式,分配输出矩阵,大小为 4 704 字 ($o_h \times o_w \times o_c$)。池化计算采用本文提出的池化计算方法,因此该网络中池化计算并不使用额外的内存。从表 3 中可以看出对于 LeNet 卷积神经网络,本文方法相较于上述其他几种方法内存使用量分别下降 89.90%、82.21%和 28.07%。

表 3 LeNet 卷积神经网络的内存使用量对比

步骤	输入尺寸	M_{im2col}	M_{MEC}	$M_{\text{direct conv}}$	M_{ours}
conv1 kernel: 5×5 stride:1	32×32×1	24 454	9 334	4 704	4 704
pool1 kernel: 2×2 stride:2	28×28×6	1 176	1 176	1 176	0
conv2 kernel: 5×5 stride:1	14×14×6	31 400	21 600	1 600	904
pool2 kernel: 2×2 stride:2	10×10×16	400	400	400	0
fc1	5×5×16	120	120	120	120
fc2	120	84	84	84	84
fc3	84	10	10	10	10
总计		57 644	32 724	8 094	5 822

3.3 性能分析

为验证本文算法对卷积计算性能的影响,本文还对不同卷积计算方法的计算时间实验测试,分别对单个卷积和 LeNet 卷积神经网络进行实验测试,测试结果分别如表 4 和表 5 所示。

表 4 单个卷积计算时间对比

卷积	T_{im2col}	T_{MEC}	$T_{\text{direct conv}}$	T_{ours}
CV1	151.6 *	156.5 *	186.2 *	132.2 *
CV2	244.0	224.5 *	269.7 *	195.6 *
CV3	—	290.7	344.9	235.9 *
CV4	—	—	389.6	324.3
CV5	—	—	—	406.9
CV6	—	—	—	213.8
CV7	—	797.3	747.3 *	538.6 *
CV8	—	977.6	1 121.2	766.6 *
CV9	—	—	1 318.0	1 075.4
CV10	24.2	26.8	31.6	35.6
CV11	—	—	36.1	41.4
CV12	—	—	—	32.2

表 4 中 T_{im2col} 、 T_{MEC} 、 $T_{\text{direct conv}}$ 和 T_{ours} 分别表示 im2col+GEMM、MEC、直接卷积和本文方法计算单个卷积的时间,时间单位为 ms。对于测试集中单个卷积计算内存使用量小于 128 KB 的卷积,放在访存速度较快的 DTCM 内存中计算,即表 4 中标记“*”的位置;若卷积计算内存使用量大于 512 KB 则无法

在该设备上进行计算,在表 4 中以“—”表示;其他情况下均在 SRAM 内存中进行卷积计算。虽然本文方法需要额外复制数据造成时间上的开销,但得益于使用较少的内存提高缓存命中率和更有效利用 DTCM 内存,从表 4 中可以看出,实际平均计算时间相较于其他方法较快。此外其他卷积计算方法内存使用量部分超出实验设备限制,无法完成全部计算,而本文方法能够完成测试集中全部卷积计算,为内存资源受限设备运行卷积神经网络提供更多的选择。

表 5 LeNet 卷积神经网络的计算时间对比 ms				
步骤	M_{im2col}	M_{MEC}	$M_{direct\ conv}$	M_{ours}
conv1	14.89	11.25	18.82 *	20.60 *
pool1	0.21	0.21	0.18 *	0.20 *
conv2	24.65	21.75	25.67 *	20.39 *
pool2	0.07	0.07	0.06 *	0.08 *
fc1	3.82	3.82	3.48 *	3.48 *
fc2	0.81	0.81	0.73 *	0.73 *
fc3	0.07	0.07	0.07 *	0.07 *
总计	44.52	37.98	49.01 *	45.55 *

表 5 为 LeNet 卷积神经网络各层计算时间的对比,单位为 ms,其中直接卷积和本文方法内存使用量均小于 128 KB,在 DTCM 中进行计算,用“*”表示;im2col+GEMM 和 MEC 方法内存使用量均大于 128 KB,在 SRAM 中进行计算。

综合表 3 和表 5 可以看出,虽然使用本文方法计算 LeNet 卷积神经网络相较于 im2col+GEMM 和 MEC 方法计算时间分别增加 2.31%和 19.93%,但内存使用量分别下降 89.90%和 82.21%;而对于直接卷积方法,本文方法计算时间在下降 6.98%的同时,内存使用量下降 28.07%,优势较为明显。此外本文提出的池化层计算方法较标准池化计算方法计算时间增加 14.29%,但总计仅增加 0.04 ms 的计算时间,对整个卷积神经网络而言可以忽略不计。综合以上实验分析,验证了本文提出的卷积计算方法的有效性,大幅降低了卷积计算的内存使用量。

4 结束语

本文提出一种面向内存资源受限设备的卷积计算方法,卷积计算时对输入矩阵计算后不再使用的内存空间进行复用存放计算结果,从而减少分配新

的内存空间存放计算结果,达到降低内存使用量的目的,使得在内存资源受限的设备上可以运行更复杂的卷积神经网络。实验结果验证了本文方法的有效性,相较于其他方法均大幅降低内存使用量,对于一些内存使用量较大的卷积,其他方法均无法在有限的内存中完成卷积计算,而本文方法可以完成卷积计算,对于卷积神经网络在内存受限设备上的部署运行具有重要意义。

参考文献:

[1] LECUN Y, BENGIO Y, HINTON G. Deep learning [J]. Nature, 2015, 521(7553): 436-444.

[2] 张索非, 冯烨, 吴晓富. 基于深度卷积神经网络的目标检测算法进展[J]. 南京邮电大学学报(自然科学版), 2019, 39(5): 72-80.

ZHANG Suofei, FENG Ye, WU Xiaofu. Recent advances on object detection using deep CNNs: an overview [J]. Journal of Nanjing University of Posts and Telecommunications (Natural Science Edition), 2019, 39(5): 72-80.(in Chinese)

[3] SHEN Z C, HOWARD N, NUNEZ-YANEZ J. Big-little adaptive neural networks on low-power near-subthreshold processors[J]. Journal of Low Power Electronics and Applications, 2022, 12(2): 28.

[4] BADIDI E. Edge AI and blockchain for smart sustainable cities: promise and potential [J]. Sustainability, 2022, 14(13):7609-7638.

[5] LOUKATOS D, LYGKOURA K A, MARAVEAS C, et al. Enriching IoT modules with edge AI functionality to detect water misuse events in a decentralized manner [J]. Sensors, 2022, 22(13): 4874-4893.

[6] DAVID R, DUKE J, JAIN A, et al. TensorFlow lite micro: embedded machine learning on TinyML systems [EB/OL]. [2022-04-21]. <https://arxiv.org/abs/2010.08678>.

[7] CAPOTONDI A, RUSCI M, FARISELLI M, et al. CMix-NN: mixed low-precision CNN library for memory-constrained edge devices [J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2020, 67(5): 871-875.

[8] HOWARD A G, ZHU M L, CHEN B, et al. MobileNets: efficient convolutional neural networks for mobile vision applications [C]//IEEE Conference on Computer Vision and Pattern Recognition.2017:432-445.

[9] ZHANG X, ZHOU X, LIN M, et al. Shufflenet: an extremely efficient convolutional neural network for mobile devices [C]// IEEE Conference on Computer Vision and Pattern Recognition. 2018:6848-6856.

- [10] PARK E, KIM D, KIM S, et al. Big/little deep neural network for ultra low power inference [C] // International Conference on Hardware/Software Codesign and System Synthesis. 2015:124-132.
- [11] GEORGANAS E, AVANCHA S, BANERJEE K, et al. Anatomy of high-performance deep learning convolutions on SIMD architectures [C] // IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. 2018:830-841.
- [12] ZHANG J, FRANCHETTI F, LOW T M. High performance zero-memory overhead direct convolutions [EB/OL]. [2022-04-21]. <https://arxiv.org/abs/1809.10170>.
- [13] 李爽, 赵荣彩, 王磊. 面向申威 1621 通用矩阵乘算法的实现与优化 [J]. 计算机科学, 2021, 48(S2): 699-704, 718.
LI Shuang, ZHAO Rongcai, WANG Lei. Implementation and optimization of Sunway 1621 general matrix multiplication algorithm [J]. Computer Science, 2021, 48(S2): 699-704, 718. (in Chinese)
- [14] 黄春, 姜浩, 全哲, 等. 面向深度学习的批处理矩阵乘法设计与实现 [J]. 计算机学报, 2022, 45(2): 225-239.
HUANG Chun, JIANG Hao, QUAN Zhe, et al. Design and implementation of batched GEMM for deep learning [J]. Chinese Journal of Computers, 2022, 45(2): 225-239. (in Chinese)
- [15] CHO M, BRAND D. MEC: memory-efficient convolution for deep neural network [EB/OL]. [2022-04-21]. <https://arxiv.org/abs/1706.06873>.
- [16] 方玉玲, 陈庆奎. 基于矩阵转换的卷积计算优化方法 [J]. 计算机工程, 2019, 45(7): 217-221, 228.
FANG Yuling, CHEN Qingkui. Convolution calculation optimization method based on matrix transformation [J]. Computer Engineering, 2019, 45(7): 217-221, 228. (in Chinese)
- [17] ZHANG Y L, LI X M. Fast convolutional neural networks with fine-grained FFTs [C] // ACM International Conference on Parallel Architectures and Compilation Techniques. 2020: 255-265.
- [18] 童敢, 黄立波. Winograd 快速卷积相关研究综述 [J]. 计算机科学与探索, 2022, 16(5): 959-971.
TONG Gan, HUANG Libo. Review of winograd fast convolution technique research [J]. Journal of Frontiers of Computer Science and Technology, 2022, 16(5): 959-971. (in Chinese)
- [19] BARABASZ B, GREGG D. Winograd convolution for DNNs: beyond linear polynomials [C] // International Conference of the Italian Association for Artificial Intelligence. 2019:307-320.
- [20] CHEN T Q, XU B, ZHANG C Y, et al. Training deep nets with sublinear memory cost [EB/OL]. [2022-04-21]. <https://arxiv.org/abs/1604.06174>.
- [21] 刘博. 深度学习系统内存管理和通信优化关键技术研究 [D]. 武汉: 华中科技大学, 2020.
LIU Bo. Research on key technologies of memory management and communication optimization for deep learning system [D]. Wuhan: Huazhong University of Science and Technology, 2020. (in Chinese)
- [22] 王济民, 魏怡, 周宇, 等. 基于 LeNet-5 卷积神经网络和颜色特征的限速标志识别 [J]. 计算机科学, 2021, 48(S2): 345-350.
WANG Jimin, WEI Yi, ZHOU Yu, et al. Speed limit sign recognition based on LeNet-5 CNN and color feature [J]. Computer Science, 2021, 48(S2): 345-350. (in Chinese)

(责任编辑:潘雪松)