计算机科学与探索
Journal of Frontiers of Computer Science and Technology
1673-9418/2022/16(05)-0959-13
doi: 10.3778/j.issn.1673-9418.2110036

Winograd快速卷积相关研究综述

童　敢,黄立波+
国防科技大学 计算机学院,长沙 410073
+通信作者 E-mail: libohuang@nudt.edu.cn

摘　要:卷积神经网络(CNN)已经被广泛应用到各个领域并发挥了重要作用。卷积算子是卷积神经网络的基础组件,同时也是最耗时的部分。近年来,研究者提出了包括基于FFT和Winograd的若干种快速卷积算法。其中Winograd卷积因大幅减少了卷积中乘法操作且占用内存更小而迅速成为小卷积核的卷积算子上快速卷积实现的首选。但目前相关工作聚焦于算法的一般化、拓展和各类体系结构上的实现,还没有研究者对Winograd卷积算法作系统性的总结。为了给后续研究者提供详细的参考依据,对Winograd卷积引入以来的相关工作进行了总结。首先阐述了Winograd最小滤波算法及Winograd卷积的引入,介绍了Winograd卷积的一般化与拓展,并对比了现有实现之间的差异;从稀疏剪枝、低精度与量化、数值稳定性这三方面介绍了Winograd卷积的优化工作,并详细介绍了相关具体方法的优缺点;对各类体系结构上的实现和优化进行了分类总结,比较了各平台上实现可用的通用优化方法,并介绍了Winograd卷积的实际应用;最后对内容进行了简要总结,分析了现有研究的局限性,并对未来可能的方向进行了初步展望。

关键词: Winograd卷积;快速卷积算法;卷积神经网络(CNN);卷积优化
文献标志码:A　　**中图分类号:**TP183

Review of Winograd Fast Convolution Technique Research

TONG Gan, HUANG Libo+
School of Computer, National University of Defense Technology, Changsha 410073, China

Abstract: Convolutional neural networks (CNN) have been widely used in various fields and have played an important role. Convolution operator is the basic component of CNN, and it is also the most time-consuming part. In recent years, researchers have proposed several fast convolution algorithms including FFT and Winograd. Among them, Winograd convolution has quickly become the first choice for fast convolution implementation on convolution operators with small convolution kernels, because it greatly reduces the multiplication operations in convolution and occupies less memory. Related work focuses on the generalization, extension and implementation on various architectures of the Winograd convolution, but there are no researchers who have systematically summarized the Winograd convolution algorithm. This paper aims to provide detailed reference for follow-up researchers, and summarizes all related work since the introduction of Winograd convolution. Firstly, the introduction of Winograd minimum filtering algorithm and Winograd convolution is described, the generalization and extension of Winograd convolution are introduced, and the detailed differences between existing implementations are also listed. The optimization of Winograd convolution is introduced from the three aspects of sparse pruning, low precision and quantization, and numerical stability, and the advantages and disadvantages of the specific methods are elaborated.

基金项目:国家自然科学基金(61872374)。
This work was supported by the National Natural Science Foundation of China (61872374).
收稿日期:2021-10-18　　**修回日期:**2022-01-05

The implementations and optimizations of various architectures are classified and summarized, the general optimization methods available for implementation on each platform are compared, and the practical application of Winograd convolution is also introduced. Finally, a brief summary of the content is made, the limitations of existing research are analyzed, and a preliminary outlook for the possible future directions is made.

Key words: Winograd convolution; fast convolution algorithm; convolutional neural network (CNN); convolution optimization

卷积神经网络(convolutional neural network,CNN)在计算机视觉、自然语言处理等任务上应用广泛。越来越多的研究尝试加速CNN的训练和推理,使用快速卷积算子就是其中的重要方法。快速卷积算子包括快速傅里叶变换(fast Fourier transformation,FFT)卷积[1]和Winograd卷积[2],这类卷积通过把输入特征映射和卷积核线性变换到相应的空间,将原来的运算转换为对应位相乘,运算结果再经过逆线性变换即可得到原特征映射空间的输出。

在“变换-运算-逆变换”的过程中,乘法运算的次数比直接卷积有可观的减少,而代价则是加法运算次数的增加。在绝大多数现代处理器上,加法的执行效率远高于乘法,因此可以使用快速卷积算子来提高模型执行效率。由于FFT变换是映射到复数空间,Winograd卷积运算过程中对内存的占用只需FFT卷积的一半,使其迅速成为最流行的快速卷积算子。

但是直接应用Winograd卷积存在很多挑战。首先,基本的Winograd卷积适用范围有限,仅可在单位步长、小卷积核的二维卷积上应用,在大卷积核上应用则会有数值不稳定的情况[3]。其次,由于线性变换和逆线性变换的复杂性,快速卷积算子在特定平台上的优化难以实现,比如利用并行性和数据局部性[4]。此外,Winograd卷积与以剪枝和量化为代表的网络压缩技术难以直接结合,因此不易在算力不足和有能耗限制的平台上部署实现[5]。针对这些问题,研究者做了大量的工作,但至今还未有公开的文章对相关工作进行系统性的总结。为给后续研究者提供参考,本文从算法拓展、算法优化、实现与应用三方面综述Winograd的发展,并对未来可能的研究方向做出展望。

1 Winograd卷积原理

Winograd于1980年提出了有限脉冲响应(finite impulse response,FIR)滤波的最小滤波算法[6]。最小滤波算法指出,由 r 拍的FIR滤波器生成 m 个输出,即 $F(m,r)$,需要的最少乘法数量 $\mu(F(m,r))$ 为 $m+r-1$ 。以 $F(2,3)$ 为例,涉及到的乘法数量为 $\mu(F(2,3))=2+3-1=4$,从6次降低到了4次。

2015年,Winograd最小滤波算法初次被应用在CNN中[2],利用减少的乘法次数提升卷积算子性能。如果用矩阵的形式表示Winograd最小滤波算法,则可以得到:

$$\boldsymbol{Y}=\boldsymbol{A}^{\mathrm{T}}[(\boldsymbol{Gg})\odot(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{d})]$$

其中, $\boldsymbol{g}$ 为滤波器向量, $\boldsymbol{d}$ 为输入数据向量, $\boldsymbol{Y}$ 为输出数据向量, $\boldsymbol{G}$ 表示滤波器变换矩阵, $\boldsymbol{B}^{\mathrm{T}}$ 表示数据变换矩阵, $\odot$ 表示矩阵的对应位相乘(Hadamard积), $\boldsymbol{A}^{\mathrm{T}}$ 表示输出变换矩阵。通过嵌套一维最小滤波算法 $F(m,r)$,可以得到二维的最小滤波算法 $F(m\times m,r\times r)$:

$$\boldsymbol{Y}=\boldsymbol{A}^{\mathrm{T}}[(\boldsymbol{GgG}^{\mathrm{T}})\odot(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{dB})]\boldsymbol{A}$$

二维最小滤波算法所需乘法数为 $(m+r-1)^2$,而原始卷积算法需要乘法数为 $m\times m\times r\times r$ 。对于 $F(2\times 2,3\times 3)$ 而言,乘法次数从36降低到了16,减少了55.6%。

根据二维矩阵形式,可以自然地将Winograd卷积分为四个分离的阶段:输入变换(input transformation,ITrans)、卷积核变换(kernel transformation,KTrans)、对应位相乘(element-wise matrix multiplication,EWMM)和输出变换(output transformation,OTrans),如图1所示。

对于二维卷积算子,需要先将卷积输入划分为相互重叠的 $(m+r-1)\times(m+r-1)$ 的切片,切片之间有 $r-1$ 的重叠部分。实验表明, $F(2\times 2,3\times 3)$ 在多个卷积上的实现性能超过了NVIDIAcuDNN[2],而且使用的内存大小远低于FFT卷积。

2 Winograd卷积的一般化和拓展

2.1 Winograd卷积的一般化

基本的Winograd卷积仅支持 $r=3$ 和 $r=2$ 的二维卷积算子,且切片大小不超过6,无法满足现代CNN中丰富的卷积算子类型,需要对其进行一般化。Winograd卷积的一般化主要分为四个方向,分别是支持任意维度、支持任意切片大小、支持任意常规卷积、支持特殊卷积。

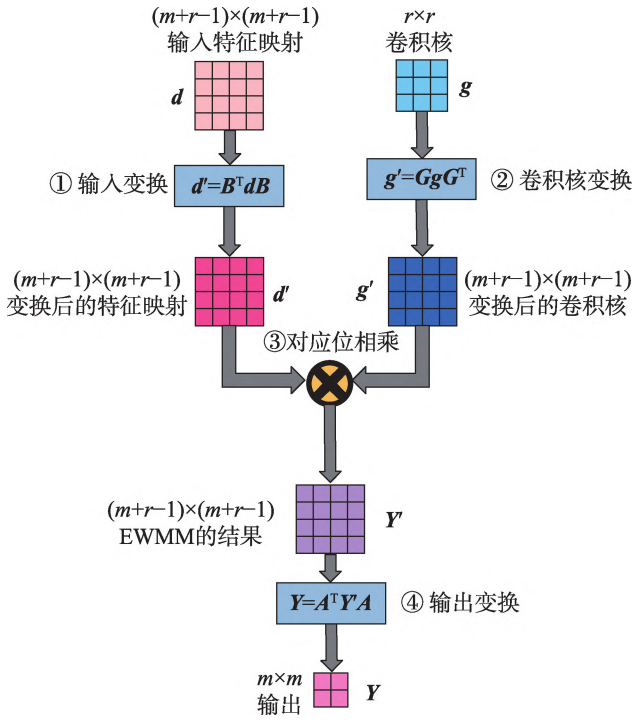

图1　Winograd卷积的四个阶段

Fig.1　Four stages of Winograd convolution

三维卷积是三维CNN的主要组件，常用于处理空间相关的信息。通过对一维Winograd卷积进行嵌套，可以得到其二维形式，重复进行嵌套则可以得到任意维度的Winograd卷积。Budden等[7]给出了N维Winograd卷积的一般形式，并将二维Winograd卷积视为特殊情况在CPU上实现，但并未实现三维的情况。其他研究者[8-12]使用了同样的嵌套方法，并针对特定平台完成了三维Winograd卷积的实现。由于不同维度上算法的实现有统一性，Shen等[13-14]提出了二维、三维统一的现场可编程逻辑门阵列（field programmable gate array，FPGA）模板实现。Deng等[15]提出了可变分解方法，支持三维卷积的同时也支持了非单位步长的卷积。

更大的切片大小会减少切片之间的重叠部分，但同时也会带来更大的数值误差，因此在对精度要求不太严格的场合会直接使用更大的切片尺寸以提升性能。大尺寸的卷积核也在卷积网络模型中经常出现，通常为了保持Winograd卷积的精度，这里卷积也会被替换为小尺寸的卷积。Lu等[16]在FPGA上评估了大尺寸切片分别在$r=3$和$r=5$下的精度情况，实验表明小的切片尺寸在$r=3$时可以保持模型的高精度。Huang等[17]也完成了类似的工作。Mazaheri等[18]则基于符号编程构建了支持不同硬件后端的实现，同时也支持不同尺寸的切片。

此类直接实现的方法会在大尺寸切片和大尺寸卷积核上显著损失精度，因此将这类卷积分解为更小的卷积成为了研究者常用的方法。Yang等[19]使用分解方法统一了常规卷积、depth-wise卷积以及分组卷积，而大切片尺寸[20]、大卷积核卷积[21-25]和非单位步长卷积[15,20,22-24,26-28]也都可以通过分解方法转换为基本的Winograd卷积。常用的分解单元实现包括$F(2\times2,3\times3)$、$F(2\times3,3\times3)$、$F(3\times2,3\times3)$、$F(3\times3,3\times3)$等。Liu等[29]同样基于分解方法，在FPGA上实现了使用相同资源支持任意卷积核大小的Winograd卷积。利用大卷积核上计算的对称性，Sabir等[30]使用近似计算技术支持了$r=5$大小的卷积核。

包括空洞卷积和转置卷积在内的特殊卷积常用于图像分割、超分辨率等领域。空洞卷积的Winograd形式被提出用于支持扩张为2和4的情况[31]，原理是扩张输入变换矩阵的规模。Shi等[26]通过预定义的分解和交织操作将转置卷积转换为多个基本卷积，从而实现了对转置卷积的支持。

总结Winograd卷积一般化研究工作相关文献如表1所示。通过结合嵌套方法和分解方法，理论上可以实现CNN中所有卷积的Winograd卷积形式。

2.2　Winograd卷积的拓展

除了一般化到各类卷积，还有一些研究尝试拓展Winograd卷积本身的线性变换。Winograd算法族首先将输入的切片和卷积核线性变换到Winograd域，执行Hadamard积之后再逆变换回特征映射域。对于指定的卷积核和切片尺寸，线性变换矩阵$\boldsymbol{A}$、$\boldsymbol{G}$、$\boldsymbol{B}$是给定的。卷积可以表示为多项式乘法，将卷积核$\boldsymbol{g}$和输入向量$\boldsymbol{d}$的元素分别映射到多项式$g(x)$和$d(x)$的系数，则输出向量$\boldsymbol{y}$（$\boldsymbol{g}$和$\boldsymbol{d}$的卷积）的元素等于多项式$y(x)=g(x)d(x)$的系数。Winograd卷积算法族基于多项式上的中国剩余定理（Chinese remainder theorem，CRT）对不可约且互质的多项式同余系统内的多项式取余即可得到卷积输出，对同余方程组进行求解即根据多项式的系数得到线性变换矩阵的具体解[6]。因此，对Winograd卷积的拓展可以从两个角度入手：一是使用不同的变换，将运算映射到不同的域；二是采用不同的变换矩阵生成多项式。

Barabasz等[33]将Winograd卷积算法中使用的卷积多项式拓展为高阶多项式，实验表明使用二阶多项式会显著降低误差，但同时也会增加乘法次数，因此需要在乘法次数和浮点数精度之间做权衡。Ju等[34]

表1 Winograd卷积的一般化

Table 1 Generalization of Winograd convolution

文献	嵌套方法	分解方法	其他方法	维度	大切片	大卷积核	跨步卷积	空洞卷积	转置卷积	备注
[2]	—	—	—	一/二维	—	—	—	—	—	引入Winograd卷积
[7]	√	—	—	*N* 维	—	—	—	—	—	仅实现了CPU上的二维
[8]	√	—	—	三维	—	—	—	—	—	FPGA实现
[9]	√	—	—	三维	—	—	—	—	—	GPU实现
[10]	√	—	—	*N* 维	√	√	—	—	—	覆盖了CPU上各类常规卷积
[11]	√	—	—	三维	—	—	—	—	—	在FPGA上实现了动作识别
[12]	—	√	—	三维	—	—	—	—	—	向量DSP上的实现
[13-14]	√	—	—	二/三维	—	—	—	—	—	统一了FPGA上的多维实现
[15]	—	√	—	二/三维	—	√	√	—	—	可变分解方法
[16]	—	—	—	二维	√	√	—	—	—	在FPGA上进行了全面评估
[17]	—	—	—	二维	√	—	—	—	—	FPGA上大切片的精度测试
[18]	—	√	√	二维	√	√	—	—	—	基于符号编程兼容各种硬件
[19]	—	√	—	二维	—	√	√	—	—	支持depth-wise和分组卷积
[20]	√	√	—	二维	√	√	√	—	—	FPGA实现，分解为 $F(2^2,2^2)/F(3^2,3^2)/F(4^2,4^2)/F(6^2,3^2)$
[21]	—	√	—	二维	—	√	—	—	—	在FPGA上实现了不同尺寸卷积核的Winograd模块
[22]	√	√	—	一/二/三维	—	√	√	—	—	FPGA和GPU上均有实现
[23]	—	√	—	二维	—	√	√	—	—	GPU实现，形式化了分解方法
[24]	—	√	—	二维	√	√	√	—	—	FPGA实现，分解为 $F(4^2,3^2)$ 或 $F(4^2,2^2)$
[27]	—	√	—	二维	—	√	√	—	—	支持步幅为2或3的卷积
[29]	—	√	—	二维	—	√	—	—	—	可变分解方法
[30]	—	—	√	二维	—	√	—	—	—	利用近似计算将 $r=5$ 的卷积核对称因子化为 $r=3$ 的卷积核
[31]	—	—	√	二维	—	—	√	√	—	方法为拓展 $\boldsymbol{B}$ 矩阵的规模
[32]	—	√	—	二维	—	√	√	—	√	在FPGA上实现了实时超分

提出的双线性多项式与此方法原理相同，保持了大卷积核上的数值稳定性。而Meng等[35]将多项式乘法拓展到复数域，利用共轭复数乘法的对称性可以进一步减少乘法数量。Liu等[36]提出将余数系统（residual number system，RNS）引入到Winograd卷积，通过取余的操作实现Winograd卷积的量化操作，进一步支持更大的输入切片尺寸而不会引入显著误差。Xu等[25]创新地引入了费马数变换（Fermat number transformation，FNT），使用这种变换一方面可以确保中间运算结果均为无符号数，另一方面还将所有的计算都简化为移位和加法操作，有利于在FPGA等设备上实现。

根据最小滤波算法的描述，Winograd卷积已经达到了最少的乘法运算次数。FFT卷积由于使用了傅里叶变换，在乘法次数和内存占用上远高于Winograd卷积，但保持了很好的精度。探索不同的变换可能会引入更多的乘法次数，但若能保持模型精度也是可选的实现技术。而生成多项式的选择则是在不增加乘法次数的前提下减少了精度损失，且无需修改算法，因此在应用上更具有现实意义。

2.3 其他

Winograd卷积还被用于和Strassen算法结合[37]。Strassen算法[38]是一种减少矩阵运算次数的算法。虽然有研究指出Strassen算法减少的运算远小于Winograd算法[2]，但该工作[37]将在Strassen算法中使用的卷积替换为Winograd卷积，结合了两者带来的运算减少实现了更进一步的优化。Winograd卷积也被应用在加法神经网络上[39]，用加法代替乘法，保持了相当的性能且降低了功耗。

3　Winograd卷积的优化

3.1　剪枝和利用稀疏性

剪枝是CNN优化中常用的有效技术。剪枝主要用于对CNN中卷积算子的权值进行修剪，对输出影响很小的权值会被置零。剪枝后的卷积核成为稀疏张量，这带来了两点好处：一是按照特定的压缩格式存储稀疏的卷积核张量权值可以减少内存使用；二是稀疏张量中大量元素为0，因此可以减少卷积的计算量。对于卷积层和全连接层的卷积可以将参数减少90%以上。但在Winograd卷积上直接应用剪枝是有困难的，因为稀疏的卷积核在变换到Winograd域后又会变回稠密矩阵，这违背了剪枝的初衷。

Liu等[5]首先提出在Winograd卷积和FFT卷积上应用剪枝，在卷积核变换之后引入剪枝以得到稀疏的Winograd域卷积核。他们在后续的研究[40]中又将线性整流单元（rectified linear unit，ReLU）置于输入变换之后，结合稀疏的卷积核进一步提升了稀疏性，如图2所示，Wang等[41]也使用了相同的剪枝方法。Li

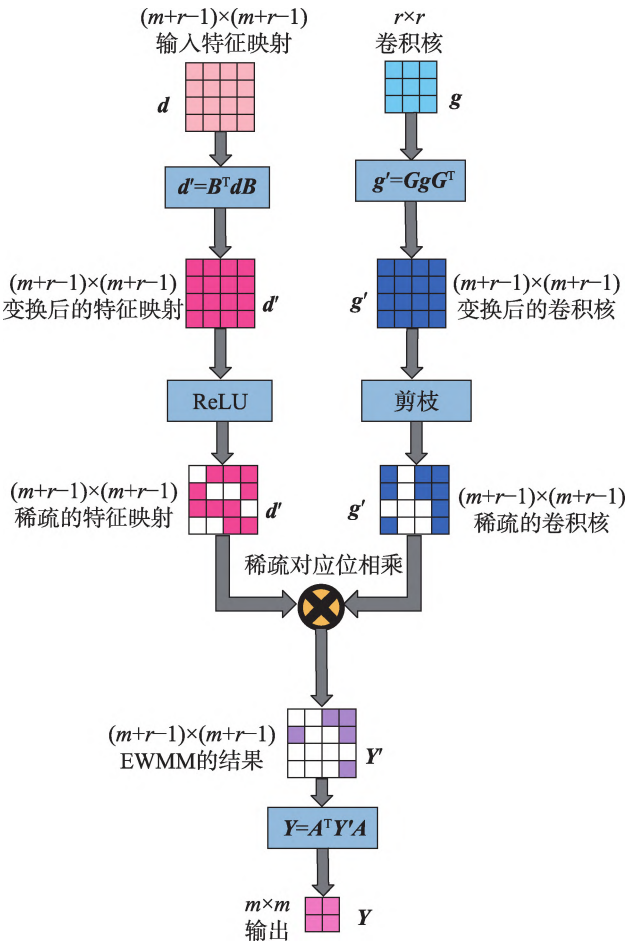

图2　在Winograd卷积中应用ReLU以实现剪枝

Fig.2　Pruning by applying ReLU in Winograd convolution

等[42]提出在本地学习剪枝系数，减少了剪枝带来的精度损失。Lu等和Shi等[26,43]也分别在卷积核变换后引入了剪枝。Yu等[44]指出，添加ReLU的方法改变了网络结构，重训练的代价也大，因此他们提出在输入变换前进行结构化剪枝传递稀疏性，同时对卷积核进行剪枝。为了兼顾速度和准确性，Zheng等[45]在Liu等工作[40]的基础上，提出了动态学习批量的大小。

在对稀疏性的利用上，包括计算优化和模型压缩两方面。Park等[46]提出Zero-Skip技术，软硬件上分别实现在EWMM阶段计算时跳过零权值，是经典的计算优化方法。Choi等[47]首次提出了利用Winograd卷积剪枝的模型压缩方法，区别于Liu等工作[40]，他们使用池化代替了ReLU。Yang等[48]提出了一种规则的剪枝模式，以优化模型的压缩，Wang等[49]则是提出了一种新的编码方式来确保压缩与解压缩。

Winograd卷积的剪枝技术主要相关工作的总结与分析如表2所示。

表2　Winograd卷积中的剪枝

Table 2　Pruning in Winograd convolution

文献	方法	优点	缺点
[5]	在卷积核变换后引入剪枝	取代了特征向量空间的剪枝，变换不会恢复稠密	需要在Winograd域重新训练
[40]	把ReLU层置于输入变换后	赋予了特征向量稀疏性，和卷积核的剪枝结合加强了EWMM阶段的稀疏性	对网络结构进行了重构
[44]	在输入变换前结构化剪枝	将特征向量空间的稀疏性转移到了Winograd域	在特征向量空间和Winograd域都需要重新训练

3.2　低精度与量化

Winograd卷积也可以与量化结合，牺牲精度以换取更小的模型和更快的运算速度。Zhuge等[50]首先使用了8位定点数的Winograd卷积，但与16位定点数相比误差显著。Zhang等[51]使用的精度与他们一致，但引入了细粒度调度，因此在性能上有显著提升。Meng等[35]在变换后的卷积核上应用量化，使用精度缩放技术量化到INT8精度上。Liu等[36]在把Winograd卷积拓展到余数系统的同时也使得可以将卷积量化到INT8精度。Ye等[52]实现了12位的混合卷积FPGA实现，还有一些研究[25,53-56]也使用了8位的量化精度。Han等[57]还进一步探索了ARMCPU上的2~8位量化精度。Li等[58]直接在Winograd域插入线性量化，对2~8位的量化精度进行了全面评估。他们

的实验表明，8位以下的量化精度会带来不可忽视的模型精度下降。

对于低精度和量化带来的精度损失，Fernandez等[59]提出通过学习训练减少INT8精度Winograd卷积的误差，与剪枝技术中的重训练原理相同。而Ahmad等[60]提出对精度损失建模，为特征映射和卷积核使用不同的量化级别。Barabasz[61]用勒让德基多项式取代Winograd变换中的规范基多项式，提出基于基变技术的9位量化精度Winograd卷积，维持了数值稳定性。Sabir等[30]在特征映射切片上应用量化，应用粒子群优化技术找到量化的阈值以保持精度。可以根据他们的工作，总结缓解量化带来的精度损失的方法如表3所示。

表3 缓解量化Winograd卷积精度损失的方法

Table 3 Methods to alleviate accuracy loss of quantization in Winograd convolution

方法	适用场合
重新训练	不限
调整量化级别	量化操作的参数可调整
修改Winograd变换	新的变换直接变换到量化空间

3.3 数值稳定性

Winograd卷积在初期只应用在3×3的卷积核和小的输入切片上，原因在于Winograd卷积计算中内在的数值不稳定性。在更大的卷积核或输入切片上，Winograd变换的多项式系数呈指数增长，这种不平衡会反映在变换矩阵的元素上，造成很大的相对误差。Vincent等[3]指出这种数值不稳定性的来源是变换中大尺寸的范德蒙德矩阵，提出精心挑选出最小指数增长的值相应的多项式，同时对变换矩阵进行缩放以缓解数值不稳定性。

Barabasz团队从数学的角度在Winograd卷积数值稳定性的维持上做了大量工作。他们首先提出使用超线性多项式来构造Winograd变换矩阵[33]，在运算次数和计算精度之间进行平衡，Ju等[34]提出的双线性方法与他们的想法一致；之后他们又提出基变技术[61]，通过额外引入一个正则化变换矩阵，在实现了量化的同时保持了数值稳定性；他们进一步的研究表明，线性变换过程中浮点数乘加的运算顺序会影响到结果的准确性[62]，这同样是由于变换矩阵中元素的指数级不平衡导致的，通过霍夫曼编码运算顺序就可以减少这种误差，从而允许更大的切片尺寸和卷积核尺寸。

在前文提到的工作中，也有很多研究尝试优化Winograd卷积的数值稳定性，总结相关工作的主要思路有四点：

（1）将卷积拆分为小切片或小卷积核的Winograd卷积，如分解方法；

（2）对Winograd变换作出修改映射到精度更高的空间，如使用超线性多项式；

（3）选择变换矩阵生成多项式中相对误差更小的，如使用变换矩阵最大最小元素比值最小的；

（4）更改计算中乘累加的顺序，优先累加乘积更小的结果。

其中分解方法已经大量使用于研究中，而选择变换矩阵的方法由于无需修改Winograd卷积算法本身也具备直接应用的可行性。而超线性变换会突破现有Winograd卷积实现的架构，在数学上论证该方法优越性之前距离实际应用还有一定的距离。

4 Winograd卷积的实现、优化与应用

4.1 实现

Winograd卷积带来的高性能使得研究者们迅速将其部署到各类平台，除了CPU、GPU等，还包括对效率和功耗有严格要求的FPGA平台、移动端和边缘计算设备。对实现了Winograd卷积的实现进行统计，得到特定平台上的研究占比如图3所示。

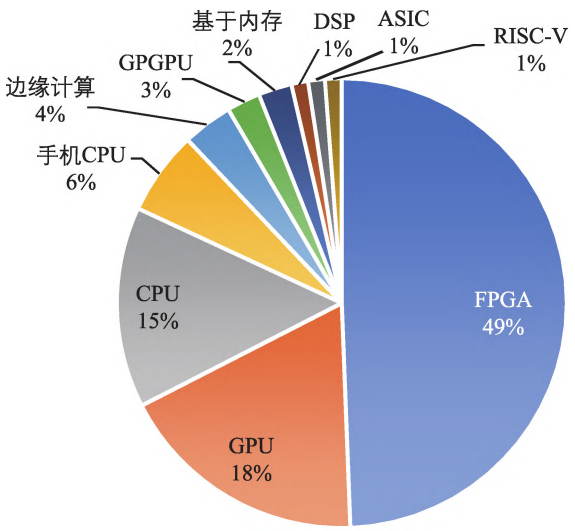

图3 Winograd卷积实现的平台分布情况

Fig.3 Platform distribution of Winograd convolution implementation

从图中可以看到，在FPGA上的Winograd卷积实现几乎达到了总数的一半。这个具有统治地位的

比值一方面说明Winograd卷积在硬编码的FPGA平台上更容易实现并发挥优势,另一方面也说明越来越多的人工智能应用在向低功耗的平台上部署。与之类似的是移动端[36,54,59,63-64]和边缘计算[65-66]上的实现,在这类平台上计算资源十分有限,对功耗的要求也更高,因此Winograd卷积带来的乘法数量减少是有重大收益的。CPU端和GPU端具备很强的算力,通常用于神经网络的训练,因此相关工作也集中在对训练速度的优化上。而也有部分供应商仍在使用云服务器上的CPU和GPU为用户提供人工智能服务,因此相关研究也具备一定的市场。GPGPU-Sim作为模拟GPU的软件平台,可以模拟对GPU硬件的修改,在GPGPU-Sim上的研究[46,67]可以为SIMT体系结构的设计提供方向。

对于上述传统平台,部分研究实现了相应的深度学习框架,在其中集成了Winograd卷积以提升模型执行效率。Perkins[4]实现的Cltorch是基于OpenCL实现的硬件无关的后端平台。Xiao等[68]实现了一个Caffe模型到FPGA映射的工具,基于动态规划选择是否应用Winograd卷积。Dicecco等[69]设计了CPU-FPGA异构平台上的开源后端框架,但仅支持单位步长的Winograd卷积。Demidovskij等[70]还实现了面向Intel硬件的、支持Winograd卷积的软件栈,面向包括CPU、集显、神经计算棒等生成高效负载。

值得注意的是,除了上述传统平台,还有一些研究尝试将Winograd卷积部署在其他非传统平台上。比如基于内存的计算平台,Lin等[71]在ReRAM上实现了Winograd卷积,基于切片提高了数据重用,而Ghaffar等[53]则基于DRAM架构实现了8-bit的量化卷积。随机计算和近似计算也用于实现Winograd卷积[28,72-73]。Chen等[12]实现了向量DSP(domain specific processor)上的三维Winograd卷积。而第五代精简指令集计算机(reduced instruction set computer-V,RISC-V)指令集作为新流行的开源指令集,同样吸引了部分研究者在相关平台上部署人工智能应用。Wang等[74]通过拓展一条 $F(2\times2,3\times3)$ 的卷积指令并新增计算模块,在一个开源RISC-V框架上实现了Winograd卷积。

4.2　优化

性能是部署在特定平台上必须要考虑的事情,对于不同的平台,研究者们采用的优化方法也大相径庭,现分平台总结相关优化技术如下。

4.2.1　CPU

Heinecke等[75]将即时(just-in-time,JIT)编译优化技术用于加速x86 CPU架构上直接卷积和Winograd卷积在小卷积核上的实现,在编译过程中提前计算卷积核调用时的地址偏移量。Ragate[76]依赖编译器的自动向量化,可将计算阶段转换为批量矩阵乘法(batched general matrix multiplication,BGEMM),利用CPU的高级向量扩展(advanced vector extensions,AVX)指令集实现性能提升。Jia等[10]提出了CPU上的自定义数据布局,同样利用CPU上的向量化指令实现高效访存。在数据重用方面,Gelashvili等[77]利用CPU的L3 Cache驻留卷积核实现了对卷积核的重用,但无法支持通道数过大的卷积,而Wu等[78]利用Winograd卷积中的相似性也可以实现CPU上的深度数据重用。

4.2.2　GPU

在GPU上同样可以将Winograd卷积的计算阶段转换为批量矩阵乘法,然后调用高效的矩阵乘法实现。Lan等[8]和Wang等[9]分别在GPU上实现了三维Winograd卷积,但前者的计算阶段直接调用了cuBLAS中的矩阵乘法实现,后者则是手动编写了特定的实现。Hong等[79]在大规模GPU集群上利用Winograd卷积的数据并行性和切片内并行性实现了多维并行训练。Jia等[80]利用MegaKernel技术将Winograd卷积的四个阶段融合,同时利用精心设计的任务映射算法可在GPU上达成显著的性能提升。另外,Yan等[81]将源代码和汇编(source and assembly,SASS)级别的汇编器优化用于优化Winograd卷积,通过合并全局访存并使共享访存无冲突,利用缓存设计流水线,提高计算强度,还利用常规寄存器填补了谓词寄存器不足的缺陷。

4.2.3　FPGA

与CPU和GPU不同的是,FPGA上没有高效的神经网络计算库可供直接调用,但可定制的特性给予了FPGA上进行优化更多可能性。Cariow等[82]首先研究了Winograd卷积硬件实现的最小需求,并在FPGA上实现了Winograd卷积的基本模块。在数据重用方面,Aydonat等[83]利用流缓冲区暂存所有的中间特征映射实现了高能耗比的FPGA实现,而Lu等[16]设计了线缓存结构来暂存特征映射并重用不同切片的数据,并在后续工作[43]中针对稀疏的情况进行了优化。

由于FPGA可定制化的特性,对计算资源的充分利用是优化的重点。一些研究[13-15]统一了二维和三维的Winograd,构建了FPGA上的统一模板。另一些

研究[26,84-86]聚焦于统一Winograd卷积和矩阵乘的实现,以最大化模块的可重用性。对硬件实现方案进行系统评估才能最大化资源利用率并提升计算效率。Ahmad等[87]和Liu等[29]还对FPGA上实现Winograd卷积进行了全面的设计空间探索。此外,还有其他工作也在FPGA上实现了Winograd卷积并对设计空间进行了探索[17,52,60,88-90],他们还在设计高效数据布局等方面进行了大量研究。

各个平台上的优化虽然方法各异,但相互有共同之处。例如,CPU和GPU均为多级内存层级体系结构,因此均可利用各级缓存驻留数据实现数据重用。三个平台上的优化方法总结比较如表4所示。

表4 不同平台上的性能优化方法

Table 4 Performance optimization methods on different platforms

优化方法	CPU	GPU	FPGA
调用计算库	√	√	—
利用向量化指令	√	√	—
数据重用	√	√	√
自定义数据布局	√	√	√
自定义计算模块	—	—	√
软流水	√	√	—
硬流水	—	—	√
算法自适应	√	√	√

4.3 应用

Winograd卷积提出的初衷是为了实现高性能卷积,但由于其内在的数值不稳定性,初期仅有小卷积核上应用Winograd卷积。CPU上[91-92]和GPU上[93]均有对FFT和Winograd的性能进行的比较,他们的结论是一致的,即FFT在大尺寸卷积核的卷积上性能更好,而Winograd卷积适用于小尺寸卷积。而Zlateski等[91-92]指出,随着CPU的内存带宽越来越大,Winograd卷积的性能优势也会减少。不过随着拓展和优化的进一步深入,Winograd卷积也成为了最适合小尺寸卷积核的快速卷积实现,在各神经网络计算库和深度学习编译框架上均有实现。

Winograd卷积旨在加速卷积以提高CNN模型的执行效率,对实时性有要求的场景都可以尝试使用。Zhuge等[50]利用混合卷积实现了人脸识别系统。Lou等[11]基于三维Winograd卷积实现了用于动作识别的加速器。Shi等[32]和Yen等[94]分别将其用于实时超分辨率,但在上采样上有区别,前者使用的是转置卷积的Winograd实现,而后者使用shuffle层代替了转置卷积。Yao等[54]还实现了穿戴设备上的语音识别加速器,应用了一维的8 bit整数Winograd卷积网络。Winograd卷积的应用不止于此,未来还可以有更多对实时性有要求的人工智能应用Winograd卷积实现,尤其是在移动端、物联网和边缘计算设备上。

5 总结与展望

Winograd卷积是当前应用最广泛的快速卷积算子。从引入到CNN至今,其使用范围随着研究的深入逐渐覆盖了现代CNN中的各类卷积,与剪枝、量化等技术的结合也走向成熟。在各种平台深度学习框架和神经网络库中均已集成Winograd卷积,可以为各类硬件平台生成高效的工作负载。

这里对未来的研究方向给出几点展望。在算法本身的优化方面,数学方法仍然是突破Winograd卷积局限性的根本方法,由于其内在的最小乘法次数属性,有望在未来的研究中基本取代现有的基于一般矩阵乘的卷积。现已有从数学角度解决数值稳定性的方法,但由于引入了新的计算机制或额外的步骤,在各平台上还没有高效的实现,对硬件友好优化方法的研究可能会是后续研究的重点方向。在实现与应用方面,FPGA平台上可以轻松为Winograd卷积定制软硬件协同的实现,但现有FPGA实现对数值稳定性的关注太少。FPGA实现具备很高的灵活性,可参照相关优化方法率先部署更快更精确的Winograd卷积。由于Winograd卷积数据流的内在复杂性,在CPU、GPU这类通用计算平台上,如何利用好算力和内存层级还有待进一步研究。比如Winograd四个阶段现大多实现为四个分离的计算核,CPU上已经有研究尝试利用L3Cache进行融合。但GPU上尝试的融合属于任务调度层面的融合,利用高速缓存的融合还未有相关研究。GPU等设备近年都引入了类似TensorCore的高性能运算单元,但Winograd卷积相关研究均未利用这类部件,因此对新硬件特性的利用也可以成为另一个突破口。此外,在非常规平台的实现明显滞后于理论,比如基于内存的计算平台、开源RISC-V框架上的实现还局限于小卷积核,下一步可以尝试在这类平台上实现更一般化的Winograd卷积。

参考文献:

[1] MATHIEU M, HENAFF M, LECUN Y. Fast training of convolutional networks through FFTs[C]//Proceedings of the 2014 International Conference on Learning Representa-

tions, Banff, Apr 14-16, 2014: 1-9.

[2] LAVIN A, GRAY S. Fast algorithms for convolutional neural networks[C]//Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, Jun 27-30, 2016. Washington: IEEE Computer Society, 2016: 4013-4021.

[3] VINCENT K, STEPHANO K, FRUMKIN M, et al. On improving the numerical stability of Winograd convolutions [C]//Proceedings of the 5th International Conference on Learning Representations, Toulon, Apr 24-26, 2017: 1-4.

[4] PERKINS H. Cltorch: a hardware-agnostic backend for the torch deep neural network library, based on OpenCL[J]. arXiv:1606.04884, 2016.

[5] LIU X, TURAKHIA Y. Pruning of Winograd and FFT based convolution algorithm: CS231n-117[R/OL]. California: Stanford University, 2016. http://cs231n.stanford.edu/reports/2016/pdfs/117_Report.pdf.

[6] WINOGRAD S. Arithmetic complexity of computations [M]. Philadelphia: SIAM, 1980.

[7] BUDDEN D, MATVEEV A, SANTURKAR S, et al. Deep tensor convolution on multicores[C]//Proceedings of the 34th International Conference on Machine Learning, Sydney, Aug 6-11, 2017: 615-624.

[8] LAN Q, WANG Z, WEN M, et al. High-performance implementation of 3D convolutional neural networks on a GPU[J]. Computational Intelligence and Neuroscience, 2017: 8348671.

[9] WANG Z, LAN Q, HE H, et al. Winograd algorithm for 3D convolution neural networks[C]//LNCS 10614：Proceedings of the 26th International Conference on Artificial Neural Networks, Alghero, Sep 11-14, 2017. Cham: Springer, 2017: 609-616.

[10] JIA Z, ZLATESKI A, DURAND F, et al. Optimizing N-dimensional, Winograd-based convolution for manycore CPUs[C]//Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Vienna, Feb 24-28, 2018. New York: ACM, 2018: 109-123.

[11] LOU M, LI J, WANG G, et al. AR-C3D: action recognition accelerator for human-computer interaction on FPGA[C]// Proceedings of the 2019 IEEE International Symposium on Circuits and Systems，Sapporo, May 26-29, 2019. Piscataway: IEEE, 2019: 1-4.

[12] CHEN W, WANG Y, YANG C, et al. Hardware acceleration implementation of three-dimensional convolutional neural network on vector digital signal processors[C]//Proceedings of the 2020 4th International Conference on Robotics and Automation Sciences, Wuhan, Jun 12-14, 2020. Piscataway: IEEE, 2020: 122-129.

[13] SHEN J, HUANG Y, WANG Z, et al. Towards a uniform template-based architecture for accelerating 2D and 3D CNNs on FPGA[C]//Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, Feb 25-27, 2018. New York: ACM, 2018: 97-106.

[14] SHEN J, HUANG Y, WEN M, et al. Toward an efficient deep pipelined template-based architecture for accelerating the entire 2-D and 3-D CNNs on FPGA[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(7): 1442-1455.

[15] DENG H, WANG J, YE H, et al. 3D-VNPU: a flexible accelerator for 2D/3D CNNs on FPGA[C]//Proceedings of the 29th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, Orlando, May 9-12, 2021. Piscataway: IEEE, 2021: 181-185.

[16] LU L, LIANG Y, XIAO Q, et al. Evaluating fast algorithms for convolutional neural networks on FPGAs[C]//Proceedings of the 25th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, Napa, Apr 30-May 2, 2017. Washington: IEEE Computer Society, 2017: 101-108.

[17] HUANG Y, SHEN J, WANG Z, et al. A high-efficiency FPGA-based accelerator for convolutional neural networks using Winograd algorithm[J]. Journal of Physics: Conference Series, 2018, 1026(1): 012019.

[18] MAZAHERI A, BERINGER T, MOSKEWICZ M, et al. Accelerating Winograd convolutions using symbolic computation and meta-programming[C]//Proceedings of the 15th EuroSys Conference 2020, Heraklion, Apr 27-30, 2020. New York: ACM, 2020: 1-14.

[19] YANG C, WANG Y, WANG X, et al. WRA: a 2.2-to-6.3 TOPS highly unified dynamically reconfigurable accelerator using a novel Winograd decomposition algorithm for convolutional neural networks[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2019, 66(9): 3480-3493.

[20] JIANG J, CHEN X, TSUI C Y. A reconfigurable Winograd CNN accelerator with nesting decomposition algorithm for computing convolution with large filters[J]. arXiv:2102.13272, 2021.

[21] CARIOW A, CARIOWA G. Minimal filtering algorithms for convolutional neural networks[M]//Reliability Engineering and Computational Intelligence. Berlin, Heidelberg: Springer, 2021.

[22] YEPEZ J, KO S B. Stride 2 1-D, 2-D, and 3-D Winograd for convolutional neural networks[J]. IEEE Transactions on Very Large-Scale Integration Systems, 2020, 28(4): 853-863.

[23] HUANG D, ZHANG X, ZHANG R, et al. DWM: a decomposable Winograd method for convolution acceleration[C]// Proceedings of the 34th AAAI Conference on Artificial Intelligence, the 32nd Innovative Applications of Artificial

Intelligence Conference, the 10th AAAI Symposium on Educational Advances in Artificial Intelligence, New York, Feb 7-12, 2020. Menlo Park: AAAI, 2020: 4174-4181.

[24] YANG C, WANG Y, WANG X, et al. A stride-based convolution decomposition method to stretch CNN acceleration algorithms for efficient and flexible hardware implementation[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 67(9): 3007-3020.

[25] XU W, ZHANG Z, YOU X, et al. Reconfigurable and low-complexity accelerator for convolutional and generative networks over finite fields[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39 (12): 4894-4907.

[26] SHI F, LI H, GAO Y, et al. Sparse Winograd convolutional neural networks on small-scale systolic arrays[J]. arXiv: 1810.01973, 2018.

[27] PAN J, CHEN D. Accelerate non-unit stride convolutions with Winograd algorithms[C]//Proceedings of the 26th Asia and South Pacific Design Automation Conference, Tokyo, Jan 18-21, 2021. New York: ACM, 2021: 358-364.

[28] LENTARIS G, CHATZITSOMPANIS G, LEON V, et al. Combining arithmetic approximation techniques for improved CNN circuit design[C]//Proceedings of the 27th IEEE International Conference on Electronics, Circuits and Systems, Glasgow, Nov 23-25, 2020. Piscataway: IEEE, 2020: 1-4.

[29] LIU X, CHEN Y, HAO C, et al. WinoCNN: kernel sharing Winograd systolic array for efficient convolutional neural network acceleration on FPGAs[C]//Proceedings of the 32nd IEEE International Conference on Application-Specific Systems, Architectures and Processors, Jul 7-9, 2021. Piscataway: IEEE, 2021: 258-265.

[30] SABIR D, HANIF M A, HASSAN A, et al. TiQSA: workload minimization in convolutional neural networks using tile quantization and symmetry approximation[J]. IEEE Access, 2021, 9: 53647-53668.

[31] KIM M, PARK C, KIM S, et al. Efficient dilated-Winograd convolutional neural networks[C]//Proceedings of the 2019 IEEE International Conference on Image Processing, Taipei, China, Sep 22-25, 2019. Piscataway: IEEE, 2019: 2711-2715.

[32] SHI B, TANG Z, LUO G, et al. Winograd-based real-time super-resolution system on FPGA[C]//Proceedings of the 2019 International Conference on Field-Programmable Technology, Tianjin, Dec 9-13, 2019. Piscataway: IEEE, 2019: 423-426.

[33] BARABASZ B, GREGG D. Winograd convolution for DNNs: beyond linear polynomials[C]//LNCS 11946: Proceedings of the XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Nov 19-22, 2019. Cham: Springer, 2019: 307-320.

[34] JU C, SOLOMONIK E. Derivation and analysis of fast bilinear algorithms for convolution[J]. SIAM Review, 2020, 62 (4): 743-777.

[35] MENG L, BROTHERS J. Efficient Winograd convolution via integer arithmetic[J]. arXiv:1901.01965, 2019.

[36] LIU Z G, MATTINA M. Efficient residue number system based Winograd convolution[C]//LNCS 12364: Proceedings of the 16th European Conference on Computer Vision, Glasgow, Aug 23-28, 2020. Cham: Springer, 2020: 53-68.

[37] ZHAO Y, WANG D, WANG L. Convolution accelerator designs using fast algorithms[J]. Algorithms, 2019, 12(5): 112.

[38] STRASSEN V. Gaussian elimination is not optimal[J]. Numerische Mathematik, 1969, 13(4): 354-356.

[39] LI W, CHEN H, HUANG M, et al. Winograd algorithm for Adder-Net[J]. arXiv:2105.05530, 2021.

[40] LIU X, POOL J, HAN S, et al. Efficient sparse-Winograd convolutional neural networks[J]. arXiv:1802.06367, 2018.

[41] WANG H, LIU W, XU T, et al. A low-latency sparse-Winograd accelerator for convolutional neural networks[C]// Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing, Brighton, May 12-17, 2019. Piscataway: IEEE, 2019: 1448-1452.

[42] LI S, PARK J, TANG P T P. Enabling sparse Winograd convolution by native pruning[J]. arXiv:1702.08597, 2017.

[43] LU L, LIANG Y. SpWA: an efficient sparse Winograd convolutional neural networks accelerator on FPGAs[C]// Proceedings of the 55th Annual Design Automation Conference, San Francisco, Jun 24-29, 2018. New York: ACM, 2018: 1-6.

[44] YU J, PARK J, NAUMOV M. Spatial-Winograd pruning enabling sparse Winograd convolution[J]. arXiv:1901.02132, 2019.

[45] ZHENG S, WANG L, GUPTA G. Efficient ensemble sparse convolutional neural networks with dynamic batch size[C]// Proceedings of the 5th International Conference on Computer Vision and Image Processing, Prayagraj, Dec 4-6, 2020. Cham: Springer, 2020: 262-277.

[46] PARK H, KIM D, AHN J, et al. Zero and data reuse-aware fast convolution for deep neural networks on GPU[C]// Proceedings of the 11th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Pittsburgh, Oct 1-7, 2016. New York: ACM, 2016: 1-10.

[47] CHOI Y, EL-KHAMY M, LEE J. Jointly sparse convolutional neural networks in dual spatial-Winograd domains [C]//Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing, Brighton, May 12-17, 2019. Piscataway: IEEE, 2019: 2792-2796.

[48] YANG T, LIAO Y, SHI J, et al. A Winograd-based CNN accelerator with a fine-grained regular sparsity pattern[C]// Proceedings of the 30th International Conference on Field-

Programmable Logic and Applications, Gothenburg, Aug 31-Sep 4, 2020. Piscataway: IEEE, 2020: 254-261.

[49] WANG X, WANG C, CAO J, et al. WinoNN: optimizing FPGA-based convolutional neural network accelerators using sparse Winograd algorithm[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39 (11): 4290-4302.

[50] ZHUGE C, LIU X, ZHANG X, et al. Face recognition with hybrid efficient convolution algorithms on FPGAs[C]//Proceedings of the 2018 on Great Lakes Symposium on VLSI, Chicago, May 23-25, 2018. New York: ACM, 2018: 123-128.

[51] ZHANG W, LIAO X, JIN H. Fine-grained scheduling in FPGA-based convolutional neural networks[C]//Proceedings of the 2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics, Chengdu, Apr 10-13, 2020. Piscataway: IEEE, 2020: 120-128.

[52] YE H, ZHANG X, HUANG Z, et al. HybridDNN: a framework for high-performance hybrid DNN accelerator design and implementation[C]//Proceedings of the 2020 57th ACM/IEEE Design Automation Conference, San Francisco, Jul 20-24, 2020. Piscataway: IEEE, 2020: 1-6.

[53] GHAFFAR M M, SUDARSHAN C, WEIS C, et al. A low power in-dram architecture for quantized CNNs using fast Winograd convolutions[C]//Proceedings of the 2020 International Symposium on Memory Systems, Washington, Sep 28-Oct 1, 2020. New York: ACM: 158-168.

[54] YAO Y, LI Y, WANG C, et al. INT8 Winograd acceleration for Conv1D equipped ASR models deployed on mobile devices[J]. arXiv:2010.14841, 2020.

[55] CAO Y, SONG C, TANG Y. Efficient LUT-based FPGA accelerator design for universal quantized CNN inference [C]//Proceedings of the 2nd Asia Service Sciences and Software Engineering Conference, Macau, China, Feb 24-26, 2021. New York: ACM, 2021: 108-115.

[56] WU D, FAN X, CAO W, et al. SWM: a high-performance sparse-Winograd matrix multiplication CNN accelerator[J]. IEEE Transactions on Very Large Scale Integration Systems, 2021, 29(5): 936-949.

[57] HAN Q, HU Y, YU F, et al. Extremely low-bit convolution optimization for quantized neural network on modern computer architectures[C]//Proceedings of the 49th International Conference on Parallel Processing, Edmonton, Aug 17-20, 2020. New York: ACM, 2020: 1-12.

[58] LI G, LIU L, WANG X, et al. Searching for Winograd-aware quantized networks[J]. arXiv:2002.10711, 2020.

[59] FERNANDEZ-MARQUES J, WHATMOUGH P N, MUNDY A, et al. Searching for Winograd-aware quantized networks [C]//Proceedings of Machine Learning and Systems 2020, Austin, Mar 2-4, 2020: 16.

[60] AHMAD A, PASHA M A. FFConv: an FPGA-based accelerator for fast convolution layers in convolutional neural networks[J]. ACM Transactions on Embedded Computing Systems, 2020, 19(2): 1-24.

[61] BARABASZ B. Quantized Winograd/Toom-cook convolution for DNNs: beyond canonical polynomials base[J]. arXiv: 2004.11077, 2020.

[62] BARABASZ B, ANDERSON A, SOODHALTER K M, et al. Error analysis and improving the accuracy of Winograd convolution for deep neural networks[J]. ACM Transactions on Mathematical Software, 2020, 46(4): 1-33.

[63] MAJI P, MUNDY A, DASIKA G, et al. Efficient Winograd or Cook-Toom convolution kernel implementation on widely used mobile CPUs[C]//Proceedings of the 2019 2nd Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications, Washington, Feb 17, 2019. Piscataway: IEEE, 2019: 1-5.

[64] LAN H, MENG J, HUNDT C, et al. FeatherCNN: fast inference computation with TensorGEMM on ARM architectures[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 31(3): 580-594.

[65] XYGKIS A, PAPADOPOULOS L, MOLONEY D, et al. Efficient Winograd-based convolution kernel implementation on edge devices[C]//Proceedings of the 55th Annual Design Automation Conference, San Francisco, Jun 24-29, 2018. New York: ACM, 2018: 1-6.

[66] MAHALE G, UDUPA P, CHANDRASEKHARAN K K, et al. WinDConv: a fused datapath CNN accelerator for power-efficient edge devices[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39 (11): 4278-4289.

[67] JEON H, LEE K, HAN S, et al. The parallelization of convolution on a CNN using a SIMT based GPGPU[C]// Proceedings of the 2016 International SoC Design Conference, Jeju, Oct 23-26, 2016. Piscataway: IEEE, 2016: 333-334.

[68] XIAO Q, LIANG Y, LU L, et al. Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on FPGAs[C]//Proceedings of the 54th Annual Design Automation Conference, Austin, Jun 18-22, 2017. New York: ACM, 2017: 1-6.

[69] DICECCO R, LACEY G, VASILJEVIC J, et al. Caffeinated FPGAs: FPGA framework for convolutional neural networks [C]//Proceedings of the 2016 International Conference on Field-Programmable Technology, Xi'an, Dec 7-9, 2016. Piscataway: IEEE, 2016: 265-268.

[70] DEMIDOVSKIJ A, GORBACHEV Y, FEDOROV M, et al. OpenVINO deep learning workbench: comprehensive analysis and tuning of neural networks inference[C]//Proceedings of the 2019 IEEE/CVF International Conference on Computer

Vision Workshop, Seoul, Oct 27-28, 2019. Piscataway: IEEE, 2019: 783-787.

[71] LIN J, LI S, HU X, et al. CNNWIRE: boosting convolutional neural network with Winograd on ReRAM based accelerators[C]//Proceedings of the 2019 on Great Lakes Symposium on VLSI, Tysons Corner, May 9-11, 2019. New York: ACM, 2019: 283-286.

[72] WANG H, ZHANG Z, YOU X, et al. Low-complexity Winograd convolution architecture based on stochastic computing[C]//Proceedings of the 23rd IEEE International Conference on Digital Signal Processing, Shanghai, Nov 19-21, 2018. Piscataway: IEEE, 2018: 1-5.

[73] GONG Y, LIU B, GE W, et al. ARA: cross-layer approximate computing framework based reconfigurable architecture for CNNs[J]. Microelectronics Journal, 2019, 87: 33-44.

[74] WANG S, ZHU J, WANG Q, et al. Customized instruction on RISC-V for Winograd-based convolution acceleration [C]//Proceedings of the 32nd IEEE International Conference on Application-Specific Systems, Architectures and Processors, Jul 7-9, 2021. Piscataway: IEEE, 2021: 65-68.

[75] HEINECKE A, GEORGANAS E, BANERJEE K, et al. Understanding the performance of small convolution operations for CNN on Intel architecture[C]//Proceedings of ACM SC17 Conference. New York: ACM, 2017: 1-2.

[76] RAGATE S N. Optimization of spatial convolution in ConvNets on Intel KNL[D]. Knoxiville: University of Tennessee, 2017.

[77] GELASHVILI R, SHAVIT N, ZLATESKI A. L3 fusion: fast transformed convolutions on CPUs[J]. arXiv:1912.02165, 2019.

[78] WU R, ZHANG F, ZHENG Z, et al. Exploring deep reuse in Winograd CNN inference[C]//Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Feb 27-Mar 3, 2021. New York: ACM, 2021: 483-484.

[79] HONG B, RO Y, KIM J. Multi-dimensional parallel training of Winograd layer on memory-centric architecture[C]// Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture, Fukuoka, Oct 20-24, 2018. Washington: IEEE Computer Society, 2018: 682-695.

[80] JIA L, LIANG Y, LI X, et al. Enabling efficient fast convolution algorithms on GPUs via MegaKernels[J]. IEEE Transactions on Computers, 2020, 69(7): 986-997.

[81] YAN D, WANG W, CHU X. Optimizing batched Winograd convolution on GPUs[C]//Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, San Diego, Feb 22-26, 2020. New York: ACM, 2020: 32-44.

[82] CARIOW A, CARIOWA G. Hardware-efficient structure of the accelerating module for implementation of convolutional neural network basic operation[J]. arXiv:1811.03458, 2018.

[83] AYDONAT U, O'CONNELL S, CAPALIJA D, et al. An OpenCL™ deep learning accelerator on Arria 10[C]//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, Feb 22-24, 2017. New York: ACM, 2017: 55-64.

[84] KALA S, MATHEW J, JOSE B R, et al. UniWiG: unified Winograd-GEMM architecture for accelerating CNN on FPGAs[C]//Proceedings of the 32nd International Conference on VLSI Design and 18th International Conference on Embedded Systems, Delhi, Jan 5-9, 2019. Piscataway: IEEE, 2019: 209-214.

[85] KALA S, JOSE B R, MATHEW J, et al. High-performance CNN accelerator on FPGA using unified Winograd-GEMM architecture[J]. IEEE Transactions on Very Large Scale Integration Systems, 2019, 27(12): 2816-2828.

[86] KALA S, NALESH S. Efficient CNN accelerator on FPGA [J]. IETE Journal of Research, 2020, 66(6): 733-740.

[87] AHMAD A, PASHA M A. Towards design space exploration and optimization of fast algorithms for convolutional neural networks (CNNs) on FPGAs[C]//Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition, Florence, Mar 25-29, 2019. Piscataway: IEEE, 2019: 1106-1111.

[88] PODILI A, ZHANG C, PRASANNA V. Fast and efficient implementation of convolutional neural networks on FPGA [C]//Proceedings of the 28th IEEE International Conference on Application-Specific Systems, Architectures and Processors, Seattle, Jul 10-12, 2017. Washington: IEEE Computer Society, 2017: 11-18.

[89] VEMPARALA M R, FRICKENSTEIN A, STECHELE W. An efficient FPGA accelerator design for optimized CNNs using OpenCL[C]//Proceedings of the 2019 International Conference on Architecture of Computing Systems. Cham: Springer, 2019: 236-249.

[90] BAI Z, FAN H, LIU L, et al. An OpenCL-based FPGA accelerator with the Winograd's minimal filtering algorithm for convolution neuron networks[C]//LNCS 11479: Proceedings of the 32nd International Conference, Copenhagen, May 20-23, 2019. Cham: Springer, 2019: 277-282.

[91] ZLATESKI A, JIA Z, LI K, et al. FFT convolutions are faster than Winograd on modern CPUs, here is why[J]. arXiv: 1809.07851, 2018.

[92] ZLATESKI A, JIA Z, LI K, et al. The anatomy of efficient

FFT and Winograd convolutions on modern CPUs[C]//Proceedings of the 2019 ACM International Conference on Supercomputing, Phoenix, Jun 26-28, 2019. New York: ACM, 2019: 414-424.

[93] KIM H, NAM H, JUNG W, et al. Performance analysis of CNN frameworks for GPUs[C]//Proceedings of the 2017 IEEE International Symposium on Performance Analysis of Systems and Software, Santa Rosa, Apr 24-25, 2017. Washington: IEEE Computer Society, 2017: 55-64.

[94] YEN P W, LIN Y S, CHANG C Y, et al. Real-time super resolution CNN accelerator with constant kernel size Winograd convolution[C]//Proceedings of the 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems, Genova, Aug 31-Sep 2, 2020. Piscataway: IEEE, 2020: 193-197.

童敢（1995—），男，安徽宣城人，硕士研究生，助理工程师，主要研究方向为面向计算机体系结构的CNN加速。

TONG Gan, born in 1995, M.S. candidate, associate engineer. His research interest is CNN acceleration for computer architecture.

黄立波（1983—），男，博士，副研究员，硕士生导师，CCF高级会员，主要研究方向为计算机体系结构。

HUANG Libo, born in 1983, Ph.D., associate researcher, M.S. supervisor, senior member of CCF. His research interest is computer architecture.

2022 CCF全国高性能计算学术年会征文通知

由中国计算机学会主办，中国计算机学会高性能计算专业委员会、齐鲁工业大学（山东省科学院）共同承办，山东省计算中心（国家超级计算济南中心）、济南超级计算技术研究院、北京并行科技股份有限公司共同协办的“2022 CCF全国高性能计算学术年会（CCF HPC CHINA 2022）”将于2022年9月23日至25日在济南.山东国际会展中心召开。全国高性能计算学术年会是中国一年一度高性能计算领域的盛会，为相关领域的学者提供交流合作、发布最前沿科研成果的平台，将有力地推动中国高性能计算的发展。

征文涉及的领域包括但不限于：高性能计算机体系结构、高性能计算机系统软件、高性能计算环境、高性能微处理器、高性能计算机应用、并行算法设计、并行程序开发、大数据并行处理、科学计算可视化、云计算和网格计算相关技术及应用，State of Practice最佳实践，以及其他高性能计算相关领域。本次大会增设“超算最佳应用”Track，评选2022年度超算最佳应用。会议录用的中文论文将分别推荐到《计算机研究与发展》(EI)、《计算机学报》(EI)、《计算机科学与探索》（正刊）、《计算机工程与科学》（正刊）、《计算机科学》（正刊）、《国防科技大学学报》（EI 正刊）和《数据与计算发展前沿》（正刊）等刊物上发表，英文论文推荐到CCF Transactions on High Performance Computing（CCF THPC）、Algorithms或拟由Springer出版。会议还将评选优秀论文和优秀论文提名奖各5名。

投稿须知：本届大会接收中英文投稿。作者所投稿件必须是原始的、未发表的研究成果、技术综述、工作经验总结或技术进展报告。请登录https://easychair.org/conferences/?conf=hpcchina2022的会议投稿系统链接进行投稿，首次登录请注册。

投稿要求：论文模版下载地址为https://gitee.com/hpcchina/template。其中，中文/英文word模版为word-cn-en.doc，中文/英文latex模版为latex-cn-en.zip。“超算最佳应用”Track请单独使用best-application-latex-cn.zip中文模版或者best-application-latex-en.zip英文模版。

会议将邀请知名院士、学者做大会特邀报告，举行学术报告和分组交流，还将进行高性能计算专题研讨、高性能计算新技术与新产品展示等活动，并同期现场举办“PAC2022全国并行应用挑战赛”决赛。本次会议邀请了国内外知名超算中心主任参加，并举行形式多样、不同主题的论坛研讨。从中您能了解到国内、外高性能计算的最新动态，获取对您个人的职业发展有益的各类信息。欢迎从事高性能计算及相关研究的同仁踊跃投稿。

论文提交截止日期：2022年7月01日
论文录用通知日期：2022年8月15日
正式论文提交日期：2022年8月20日

联 系 人：袁良、李希代
联系电话：010-62600662
电子邮箱：hpcchina@gmail.com，lixidai@ict.ac.cn