智能系统学报
CAAI TRANSACTIONS ON INTELLIGENT SYSTEMS

一种高效的稀疏卷积神经网络加速器的设计与实现

余成宇， 李志远， 毛文宇， 鲁华祥

引用本文:
余成宇, 李志远, 毛文宇, 等. 一种高效的稀疏卷积神经网络加速器的设计与实现[J]. 智能系统学报, 2020, 15(2): 323–333.
YU Chengyu, LI Zhiyuan, MAO Wenyu, et al. Design and implementation of an efficient accelerator for sparse convolutional neural network[J]. *CAAI Transactions on Intelligent Systems*, 2020, 15(2): 323–333.

在线阅读 View online: https://dx.doi.org/10.11992/tis.201902007

DOI: 10.11992/tis.201902007
网络出版地址: http://kns.cnki.net/kcms/detail/23.1538.TP.20191205.1449.006.html

一种高效的稀疏卷积神经网络加速器的设计与实现

余成宇[1,2], 李志远[1,2], 毛文宇[1], 鲁华祥[1,2,3,4]

（1. 中国科学院 半导体研究所, 北京 100083; 2. 中国科学院大学, 北京 100089; 3. 中国科学院 脑科学与智能技术卓越创新中心, 上海 200031; 4. 半导体神经网络智能感知与计算技术北京市重点实验室, 北京 100083）

摘　要: 针对卷积神经网络计算硬件化实现困难的问题, 之前大部分卷积神经网络加速器的设计都集中于解决计算性能和带宽瓶颈, 忽视了卷积神经网络稀疏性对加速器设计的重要意义, 近来少量的能够利用稀疏性的卷积神经网络加速器设计也往往难以同时兼顾计算灵活度、并行效率和资源开销。本文首先比较了不同并行展开方式对利用稀疏性的影响, 分析了利用稀疏性的不同方法, 然后提出了一种能够利用激活稀疏性加速卷积神经网络计算的同时, 相比于同领域其他设计, 并行效率更高、额外资源开销更小的并行展开方法, 最后完成了这种卷积神经网络加速器的设计并在 FPGA 上实现。研究结果表明: 运行 VGG-16 网络, 在 ImageNet 数据集下, 该并行展开方法实现的稀疏卷积神经网络加速器和使用相同器件的稠密网络设计相比, 卷积性能提升了 108.8%, 整体性能提升了 164.6%, 具有明显的性能优势。

关键词: 卷积神经网络; 稀疏性; 嵌入式 FPGA; ReLU; 硬件加速; 并行计算; 深度学习
中图分类号: TN4　　**文献标志码**: A　　**文章编号**: 1673−4785(2020)02−0323−11

中文引用格式: 余成宇, 李志远, 毛文宇, 等. 一种高效的稀疏卷积神经网络加速器的设计与实现 [J]. 智能系统学报, 2020, 15(2): 323–333.
英文引用格式: YU Chengyu, LI Zhiyuan, MAO Wenyu, et al. Design and implementation of an efficient accelerator for sparse convolutional neural network[J]. CAAI transactions on intelligent systems, 2020, 15(2): 323–333.

Design and implementation of an efficient accelerator for sparse convolutional neural network

YU Chengyu[1,2], LI Zhiyuan[1,2], MAO Wenyu[1], LU Huaxiang[1,2,3,4]

(1. Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China; 2. University of Chinese Academy of Sciences, Beijing 100089, China; 3. Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai 200031, China; 4. Semiconductor Neural Network Intelligent Perception and Computing Technology Beijing Key Lab, Beijing 100083, China)

Abstract: To address the difficulty experienced by convolutional neural networks (CNNs) in computing hardware implementation, most previous designs of convolutional neural network accelerators have focused on solving the computation performance and bandwidth bottlenecks, while ignoring the importance of CNN sparsity to accelerator design. Recently, it has often been difficult to simultaneously achieve computational flexibility, parallel efficiency, and resource overhead using the small number of CNN accelerator designs capable of utilizing sparsity. In this paper, we first analyze the effects of different parallel expansion methods on the use of sparsity, analyze different methods that utilize sparsity, and then propose a parallel expansion method that can accelerate CNNs with activated sparsity to achieve higher parallelism efficiency and lower additional resource cost, as compared with other designs. Lastly, we complete the design of this CNN accelerator and implemented it on FPGA. The results show that compared with a dense network design using the same device, the acceleration performance of the VGG-16 network was increased by 108.8% and its overall performance was improved by 164.6%, which has obvious performance advantages.

Keywords: convolutional neural network; sparsity; embedded FPGA; ReLU; hardware acceleration; parallel computing; deep learning

近年来, 由于大数据时代海量数据的获取以及计算机性能的显著提升, 以卷积神经网络为代表的深度学习算法在许多领域体现出了巨大的优越性。在计算机视觉方向, 深度学习的方法已经

收稿日期: 2019−02−14.　**网络出版日期**: 2019−12−06.
基金项目: 国家自然科学基金项目 (61701473); 中国科学院 STS 计划项目 (KFJ-STS-ZDTP-070); 中国科学院国防科技创新基金项目 (CXJJ-17-M152); 中国科学院战略性先导科技专项 (A 类)(XDA18040400); 北京市科技计划项目 (Z181100001518006).
通信作者: 毛文宇. E-mail: maowenyu@semi.ac.cn.

获得了超过人类的认知和分类水平[1]。然而像典型的分类网络 VGG-16，需要 15.5 G 次乘加操作和 138 M 的参数量[2]，巨大的计算量和参数量使得卷积神经网络的实际应用困难重重。

如何加速卷积神经网络计算，近年来一直是个活跃的研究领域。文献 [3] 研究了卷积神经网络的加速基本方法，并分析了加速设计的性能瓶颈。文献 [4] 比较了各种典型的并行展开方法，提出了适合实时卷积神经网络计算的数据流结构。文献 [5] 设计了多个卷积层处理器，分别加速不同的卷积层。文献 [6] 比较了不同的卷积计算展开方式，并提出了有效的数据复用和资源展开方法。

然而仅从计算性能和带宽瓶颈上入手对加速卷积神经网络计算来说还远远不够，有效利用神经网络本身的稀疏性，能够从算法层面大大加快网络计算速度[7]。卷积神经网络的稀疏性体现在激活结果和网络参数两个方面。

文献 [8] 利用激活稀疏性压缩了激活结果中的零值，节省了传输带宽开销并降低了计算功耗。文献 [9] 利用激活稀疏性对卷积计算激活结果压缩编码，在次层计算中跳过激活输入中的零值降低了卷积神经网络计算量。文献 [10] 利用参数稀疏性，将激活输入中权重无效的输入剔除，降低了卷积神经网络计算量。文献 [11] 同时利用激活稀疏性和参数稀疏性，实时选择激活输入中的有效值广播给所有 PE，PE 独立索引，将有效的激活输入和权重的乘积累加到对应位置完成卷积计算，有效降低了网络计算量和参数量。文献 [12] 同时利用激活稀疏性和参数稀疏性，PE 内使用笛卡尔乘积的方法，完成稀疏激活输入与稀疏权重的乘积，再将乘积结果还原到对应位置进行累加完成卷积计算，有效降低了网络计算量和参数量。

尽管这些方法都能在一定程度上利用卷积神经网络的稀疏性优化计算，但是我们注意到它们很难兼顾计算灵活性、并行效率和资源开销。文献 [8] 虽然利用激活稀疏性压缩了激活结果，但受制于计算方式的规整，实际上并没有将它用于降低计算量。文献 [9-10]、[12] 没有对并行计算负载失衡导致的性能损失问题进行优化，进行卷积计算时平均有 20%~40% 的性能损失在负载失衡上[12]。文献 [11] 只对输出特征图方向进行了并行展开，在计算输出特征图较少的卷积层时不能完全利用计算性能，并行效率受到明显影响。文献 [10-12] 为了利用参数稀疏性，每个 PE 都需要庞大的逻辑和存储空间配套，这不利于大规模并行，尤其是在 FPGA 设备上不能将乘法器资源充分利用，并且这些设计对于计算量密集的卷积计算，最终体现出的加速效果也一般，性价比并不高。

针对这些问题，提出了一种利用稀疏性加速计算同时并行效率极高、额外资源开销较小的并行展开方法，设计并实现了能够高效利用稀疏性加速计算的卷积神经网络加速器。对于卷积层，我们通过使用内核脉动阵列展开与特征图平铺展开结合的并行展开方式，在有效利用激活稀疏性减少计算量加速计算的同时，降低了并行计算负载失衡对加速效果的影响；对于全连接层，我们选择使用压缩稀疏列的计算方式，同时利用激活稀疏性和参数稀疏性，明显的降低了参数量和计算量。

1　卷积神经网络的稀疏性

自然界中的神经系统本身是稀疏的[13]，卷积神经网络本身也是稀疏的。卷积神经网络的稀疏性体现在激活结果和网络参数的两个方面。

卷积神经网络的激活结果是稀疏的。卷积神经网络使用非线性激活函数模仿了生物中神经元的放电方式[14]，近年来被广泛使用的 ReLU 激活函数将所有负值激活为零，给卷积神经网络的激活结果带来了显著的稀疏性[15]。例如 AlexNet 中卷积层有 19%~63% 的激活结果是 0[16]，而对 0 进行乘累加对计算结果没有任何影响，跳过这部分的无效计算，就能给卷积神经网络加速器带来明显的性能提升。

卷积神经网络的网络参数是冗余的。卷积神经网络使用了大量的参数来保证网络足够的复杂度和表现力，然而大量实验证明，增加参数量并不一定能提升网络效果，甚至会带来网络的过拟合[17]。特别地，虽然卷积神经网络的全连接层占用了绝大部分的参数量，却仅贡献了非常小的计算量。例如 VGG-16 中全连接层使用了 89.4% 的参数，却仅仅占用了 0.8% 的计算量。实验证明全连接层的连接是非常稀疏的，能够在不损失网络精度的情况下，通过剪枝移除超过 90% 的全连接层连接[18]。

如图 1 所示，实验随机选取了 ImageNet 数据集中的 500 张图，直接使用 VGG-16 网络不做任何优化的原始基础参数，统计了每一层的激活输入稀疏性。实验结果表明，对比 VGG-16 基准网络模型，在理想情况下利用激活稀疏性能够降低 50.92% 的计算量。

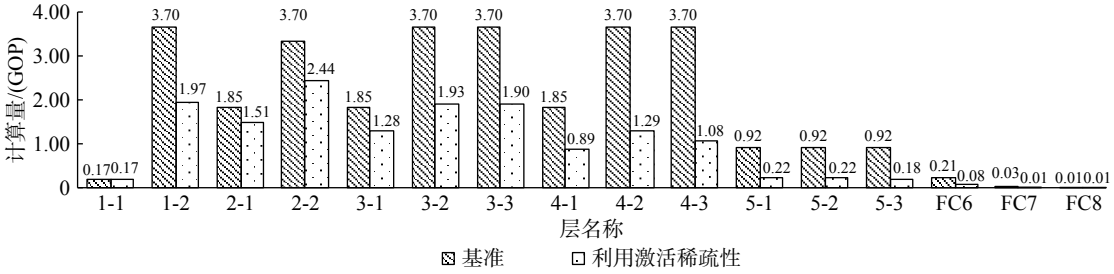


图 1　利用激活稀疏性前后 VGG-16 网络每层计算量的变化

Fig. 1　Changes of the computation amount of VGG-16 network layer before and after utilizing the sparsity of activation

使用动态精度量化的方法[19], 将参数与激活结果均量化为 8 bit, 并对全连接层进行了剪枝, 比较了网络剪枝前后的参数量。实验结果如图 2 所示。实验结果表明, 全连接层占据了大部分的参数量, 对全连接层进行稀疏剪枝能够降低 85.23% 的参数量。以上两个实验证明了, 即使不对网络结构进行任何额外的优化, 利用稀疏性对卷积神经网络计算的提升也非常明显。如何有效利用稀疏性优化计算是当前的卷积神经网络加速器设计必须要认真考虑的问题。

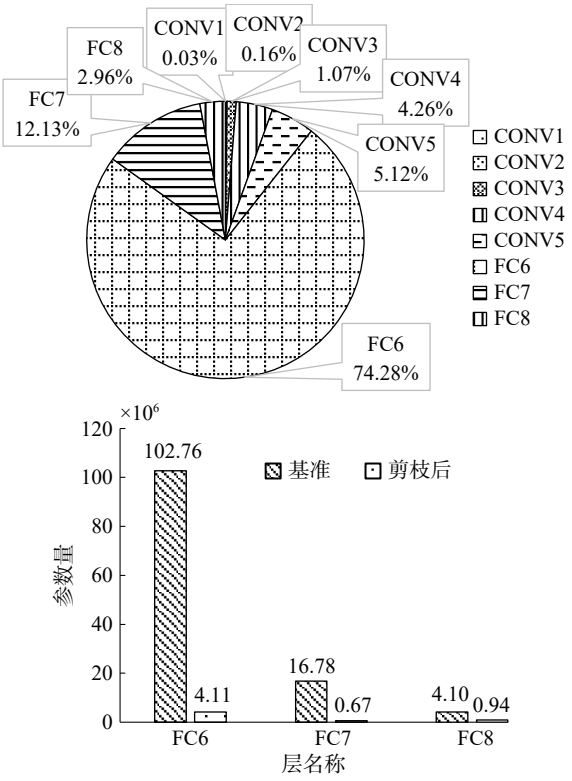


图 2　VGG-16 全连接层参数量比重与剪枝前后的变化

Fig. 2　Proportion and Changes of the parameters amount of VGG-16 full connected layer before and after pruning

2　算法理论设计

典型的卷积神经网络卷积计算如图 3 所示。输出特征图的行数为 R、列数为 C, 输入特征图的尺寸考虑了边界补零, 输入特征图通道深度为 N, 输出特征图的数量为 M, 内核行方向为 K_x, 内核列方向为 K_y。进行卷积操作时, N 个大小为 $K_x \times K_y$ 的窗口同时在输入特征图上滑动, 每次获取一个大小为 $K_x \times K_y \times N$ 的张量, 分别与 M 个输出特征图的权重卷积。卷积的 M 个计算结果为输出特征图上的一个像素点, 通过依次滑动窗口, 最终获得完整的输出特征图。卷积计算的伪代码如算法 1 中表示。

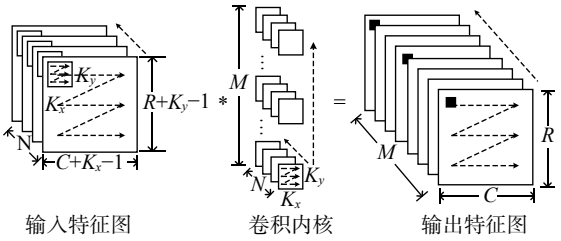


图 3　卷积层计算示意

Fig. 3　Schematic diagram of convolution computing

算法1　典型的卷积计算伪代码

```
for(m = 0; m < M; m++){                    //输出特征图
  for(n = 0; n < N; n++){                  //输入特征图
    for(r = 0; r < R; r++){                //特征图的行
      for(c = 0; c < C; c++){              //特征图的列
        for(i = 0; i < Kx; i++){           //内核行方向
          for(j = 0; j < Ky; j++){         //内核列方向
            O(c,r)(m) += K(i,j)(m,n) × I(c+i,r+j)(n);  //乘累加
}}}}}}
```

利用稀疏性加速卷积神经网络计算会给硬件上带来两个问题: 首先, 稀疏的卷积计算是不平衡的, 而并行计算时负载失衡会导致计算单元利用率下降, 损失性能, 在并行队列数量较多、队列长度不够长时, 负载失衡的影响会非常明显; 其次, 对激活输入和权重的判断或选择会增加逻辑复杂度, 可能会带来不能接受的面积开销, 影响并行展开规模。

用系统并行自由度来表征并行计算的负载失衡。如图 4 所示, 大部分卷积计算是以窗口滑动

为时间基本单位控制时序的，窗口滑动之间需要等待所有并行化方向上的计算队列计算完成，所以系统并行自由度越高，负载失衡造成的性能损失就越严重。对于完全规则设计的卷积计算模型，所有并行计算队列的计算时间是相同的，这样的系统的并行自由度为 1，而增加对权重 $K_{(i,j)}^{(m,n)}$ 和对激活输入 $I_{(c+i,r+j)}^{(n)}$ 的选择会使子队列长度不均等，增加系统并行自由度。

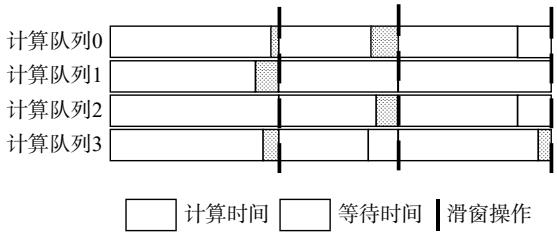


图 4　并行计算性能损失示意

Fig. 4　Schematic diagram of parallel computing performance losses

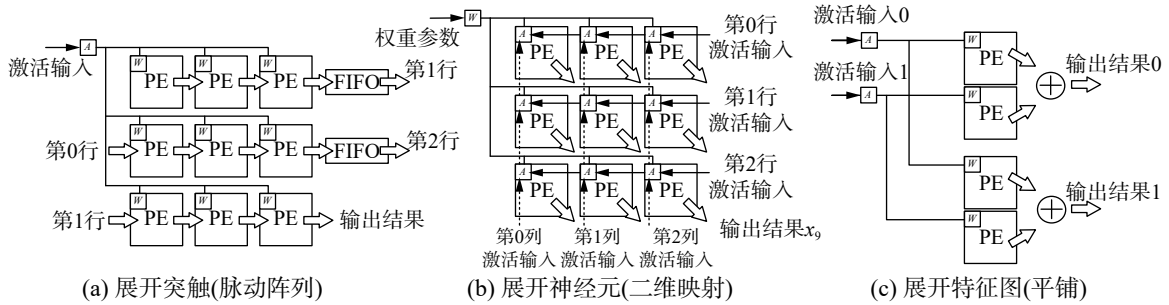
卷积计算硬件化设计的关键在于如何兼顾数据共享、复用和计算的灵活性，对于稀疏的卷积计算，需要最大化数据共享来降低并行计算时负载失衡导致的性能损失。本文分别从并行展开方式和利用稀疏性方式进行分析比较，确定并行效率极高、额外资源开销较小的稀疏卷积神经网络加速器设计。

2.1　并行展开方式

如图 5 所示，总体来说，卷积神经网络加速器设计的卷积计算的并行展开方式可以分为 3 种：展开突触 (K_x、K_y)、展开神经元 (R、C)、展开特征图 (M、N)。大部分卷积神经网络加速器的设计都是按照这 3 种并行展开方式的其中的一种或者多种展开的[4]。

对突触进行并行展开通常使用脉动阵列[20]的方式。如图 5(a) 所示，图中 K_x=K_y=3，PE 缓存权重，按照内核行方向 K_x 从左到右级联，每次计算时 PE 将乘法运算结果与左边相邻 PE 输入的值做加法运算送往右边相邻 PE，每行最右边 PE 通过一个深度为 R 的 FIFO 连接到次行，脉动阵列的输入为 0，通过全部 9 个 PE 之后计算完成全部内核方向的计算。这种并行展开方式共享了激活输入，复用了权重。它利用激活稀疏性的系统并行自由度为 1，利用参数稀疏性的系统并行自由度为 K_x×K_y，同时利用两种稀疏性的系统并行自由度为 K_x×K_y。

对神经元进行并行展开通常使用二维映射的方式，如图 5(b) 所示，特征图的行方向 R 和列方向 C 被分块为 T_r 与 T_c，图中 T_r=T_c=3，计算开始时，T_r×T_c 的激活输入被初始化到 T_r×T_c 个 PE 上，权重 $K_{(i,j)}^{(m,n)}$ 广播到每个 PE 完成乘累加计算，每个 PE 分别缓存计算结果，每次计算时 PE 向左或者向上将激活输入传递给相邻 PE，依次按照内核行方向 K_x 和列方向 K_y 输入权重并传递激活输入，经过 K_x×K_y 个周期，计算完成，每个 PE 完成了对应神经元的全部内核方向的乘累加计算。这种并行展开方式共享了权重，复用了激活输入。它利用激活稀疏性的系统并行自由度为 T_r×T_c，利用参数稀疏性的系统并行自由度为 1，同时利用两种稀疏性的系统并行自由度为 T_r×T_c。



(a) 展开突触(脉动阵列)　(b) 展开神经元(二维映射)　(c) 展开特征图(平铺)

图 5　并行展开方式示意

Fig. 5　Schematic diagram of parallel expansion mode

对特征图进行并行展开通常使用平铺的方式。如图 5(c) 所示，输入特征图 N 和输出特征图 M 被分块为 T_n 与 T_m，图 5 中 T_n=1，T_m=2，对应相同输入特征图的乘法器共享激活输入，对应相同输出特征图的乘法器将计算结果经由一个加法树累加输出。这种并行展开方式共享了激活输入。它利用激活稀疏性的系统并行自由度为 T_n，利用参数稀疏性的系统并行自由度为 T_m，同时利用两种稀疏性的系统并行自由度为 T_n×T_m。

假设同时利用权重 $K_{(i,j)}^{(m,n)}$ 和激活输入 $I_{(c+i,r+j)}^{(n)}$ 的稀疏性，由上文的分析可以看出，无论从什么方向对计算进行并行化，都会增加系统并行自由度，系统并行自由度等于 PE 展开数量。可以预见在一个没有经过特殊均衡化处理的网络上，由

于负载失衡造成的性能损失将是非常严重的。与此同时, 只要利用参数稀疏性, 这 3 种并行展开方式目前的数据共享与复用都会遭到不同程度的破坏, 每个 PE 都需要独立的选择与索引逻辑, 这导致单位 PE 面积可能会非常巨大, 不利于大规模并行。尤其是目前看来, 对于计算量密集的卷积计算, 通过利用参数稀疏性节省的计算量可能还不能抵消资源上的开销, 这是得不偿失的。

基于以上这些考虑, 在卷积层计算时仅利用了激活输入 $I^{(n)}_{(c+i,r+j)}$ 的稀疏性。如图 6 所示, 首先使用平铺的方式对输出特征图方向部分并行展开了 T_m 长度, 这 T_m 个 PE 组成一个共享激活输入的 PE 组, 然后将 $K_x \times K_y$ 个这样的 PE 组按突触方向使用脉动阵列的方式并行展开。使用这样的并行展开方式, 展开了 $T_m \times K_x \times K_y$ 个 PE, 而整个系统仅使用了 1 个逻辑单元负责有效激活输入选择, 就完成了利用激活稀疏性加速计算的功能, 资源开销极小, 并且系统并行自由度仅为 1,负载失衡造成的性能损失最低。对应的卷积层计算伪代码如算法 2 所示。

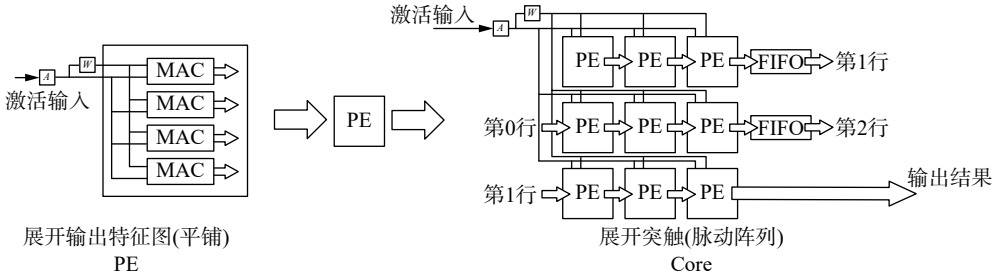


图 6　本设计使用的并行展开方式示意

Fig. 6　Schematic diagram of parallel expansion mode in our design

算法2　卷积层计算伪代码

```
for(m = 0;m < M;m+ = T_m){
K = load_weight(m);
  for(c = 0;c < C;c++){
  for(r = 0;r < R;r++){
    for(n = 0;n < N;n++){
    if(I^(tn)_(r,c+j) ≠ 0){
    for(j = 0;j < K_y;j++){
    #unroll
    for(i = 0;i < K_x;i++){}{
    #unroll
      for(tm = m;tm < min(m + T_m,M);tm++){
      #unroll
  S^(tm)_(c,r)[i][j]+ = K^(tm,n)_(i,j) × I^(n)_(c,r);
    }}}}}}
  O^(tm)_(c,r) = Systolic(Σ_{j=0}^{Ky-1} Σ_{i=0}^{Kx-1} S^(tm)_(c+i-Kx,r+j-Ky)[i][j]);
}}}
```

注意到现代卷积神经网络的卷积核几乎只有 3×3 和 1×1, 并且对于大于 3×3 的卷积核 (例如 5×5) 可以使用多次 3×3 卷积替代[2]。类似的, 每个 PE 组分别独立计算输出特征图即可以将 3×3 卷积工作模式转换成 1×1 卷积工作模式。所以本文提出的这种并行展开方式展开 3×3 的卷积核是可以适应于大部分卷积神经网络的。

全连接层参数量巨大, 片上片下传输带宽是全连接层计算瓶颈, 不做任何算法优化在硬件上完成全连接层计算是效率极低的, 并且本身全连接层稀疏性非常高 (对于 VGG-16, FC-6 的有效参数密度为 4%), 计算队列的长度特别长 (VGG-16, 不分块情况下 FC-6 的激活队列长度为 25 088), 足以忽略负载失衡造成的性能损失, 非常适合利用参数稀疏性进行计算, 所以对全连接层计算使用压缩稀疏列 (CSC) 方式同时使用了激活稀疏性和参数稀疏性。全连接层计算是简单的矩阵乘法, 使用平铺的方式并行展开输出特征图效率最高, 此时系统并行自由度为 T_m。对应的全连接层计算伪代码如算法 3 所示。

算法3　全连接层计算伪代码

```
for(m = 0;m < M;m+ = T_m){
  for(n = 0;n < N;n+ = T_n){
  K = load_weight(m,n);
  for(tn = n;tn < min(n + T_n,N);tn++){
  #pipeline
    if(I^(tn) ≠ 0){
    for(tm = m;tm < min(m + T_m,M);tm+ = Tmm){
    #parallelism
for(tmm = tm;tmm < min(tm + Tmm,M);tmm++){
      #pipeline
      if(K^(tmm,tn) ≠ 0){
      O^(tmm)+ = K^(tmm,tn) × I^(tn);
}}}}}}}
```

2.2　利用稀疏性方式

卷积神经网络稀疏性分为激活稀疏性和参数稀疏性两种, 利用稀疏性的方式直接决定了设计的复杂性和额外的资源开销量, 我们分别分析不同的利用稀疏性方式。

利用激活稀疏性主要可以分为压缩编码和多

路选择两种方式，如图 7 所示。压缩编码方式将激活结果分块，按顺序判断激活结果是否有效，记录有效激活结果的值和有效激活结果之间的距离。多路选择方式一次读入多个激活输入，按优先级从中选择有效的激活结果，输出激活结果的值和位置。压缩编码方式能够降低传输功耗，但不适合通道较浅的卷积层；多路选择方式不能降低传输功耗，但逻辑更简单，并且适应所有卷积层。

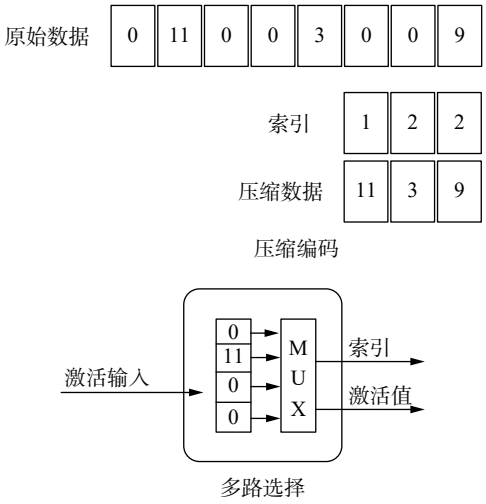


图 7　利用激活稀疏性的方式

Fig. 7　Method of using activate sparsity

文献 [8-9]、[12] 使用了压缩编码的方式，文献 [11] 使用了多路选择的方式。本文在上节中实际分析了 VGG-16 网络在 ImageNet 下的激活稀疏性在 50% 左右，使用压缩编码的方式能够降低 50% 的激活结果数据量，但与此同时对压缩编码的索引也是需要占用数据量的。几乎所有分类网络经过优化都能使用动态精度量化的方法量化成 8 bit 激活结果与 8 bit 权重[21]。如图 8 所示，随着激活结果量化程度的不断提高，对激活结果进行压缩编码的收益是在不断下降的。并且我们的设计充分靠虑了最大化共享激活输入，使用多路选择的方式消耗的激活压缩单元会非常少。基于资源开销和普适性的考虑，我们使用了多路选择的方式利用激活稀疏性。

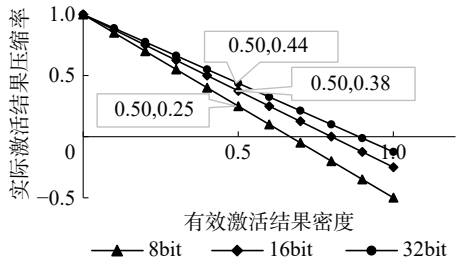


图 8　不同量化精度下 VGG-16 压缩编码实际压缩效率

Fig. 8　Actual compression efficiency of VGG-16 compression code with different quantization precision

利用参数稀疏性主要可以分为压缩稀疏行 CSR 和压缩稀疏列 CSC 两种方式[22]，如图 9 所示。压缩稀疏行 CSR，稀疏矩阵按照输出特征图分离成若干个独立计算，每个输出特征图共享输入特征图，并分别按照权重的有效性选择激活输入进行乘累加计算。压缩稀疏列 CSC，稀疏矩阵按权重方向分离成若干个计算队列，每个计算队列对应一个激活输入，计算时依次选择有效权重与激活结果相乘，并将乘积累加到对应输出特征图位置。

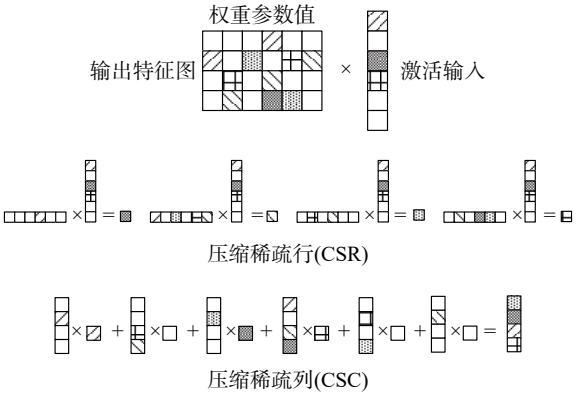


图 9　利用参数稀疏性方式

Fig. 9　Method of using parameter sparsity

CSR 模式需要完成两个稀疏矩阵的乘法，复杂性较高，文献 [12] 利用笛卡尔乘法在 CSR 模式完成了稀疏矩阵计算，文献 [10] 使用 CSR 模式仅利用了参数稀疏性。CSC 模式将计算分离成了若干个计算队列，选择有效队列就能利用激活稀疏性，更容易实现，文献 [11] 使用了 CSC 模式同时利用了激活稀疏性和参数稀疏性。使用 CSC 模式完成全连接层计算。

全连接层稀疏计算 PE 如图 10 所示。多路选择获得的有效激活输入和索引被送入 PE，通过一个存储权重指针的逻辑获得这个有效激活输入对应的有效权重的头地址和长度，然后从权重缓存中依次读出权重。权重由位置索引和值两部分组成，在乘累加缓存区域获得位置索引对应位置的乘累加值，与新产生的乘积累加，再送回乘累加缓存区域，使用一个旁路逻辑缓存乘累加结果，避免相邻周期对同一个位置读写导致的一个周期的时间惩罚。

2.3　与其他加速器设计的比较

我们分别比较了上述几种卷积神经网络加速器利用稀疏性的方式和并行展开方式，见表 1。表 1 中稀疏性 A 代表激活稀疏性，稀疏性 W 代表参数稀疏性，CSR 代表压缩稀疏行，CSC 代表压缩稀疏列，展开方式 S、N、F 分别代表突触 (synapse)、神经元 (neuron)、特征图 (feature map)。

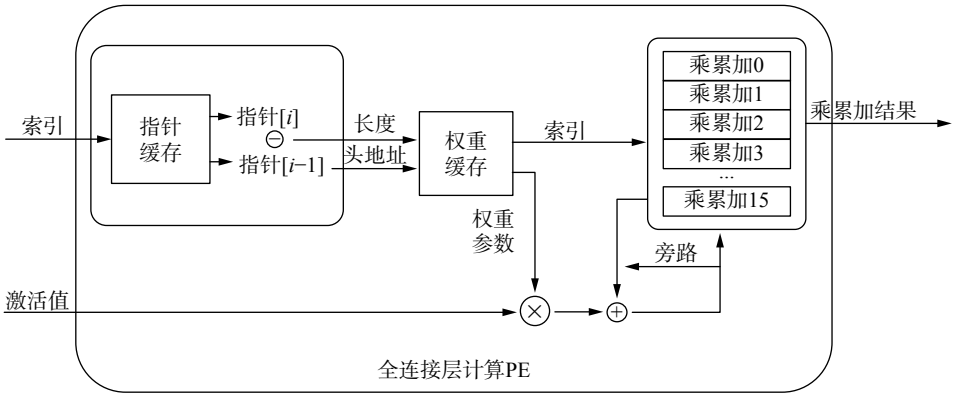


图 10　全连接层稀疏计算 PE 示意

Fig. 10　Schematic diagram of sparse computing PE for FC layer

表 1　并行展开方式和利用稀疏性的方式比较

Table 1　Compare of parallel expansion and method of using sparsity

	稀疏性	加速计算	A方式	W方式	展开方式	展开规模	并行自由度	单位MAC额外逻辑
Eyeriss[8]	A	否	压缩编码	—	SNF	$K_x \times R \times T_m$	R*	—
Cnvlutin[9]	A	是	压缩编码	—	NF	$T_r \times T_c \times T_m$	$T_r \times T_c$	$(T_m)^{-1}$
Cambricon-X[10]	W	是	—	CSR	NF	$T_r \times T_c \times T_m$	T_m	$(T_r \times T_c)^{-1}$
EIE[11]	A+W	是	多路选择	CSC	F	T_m	T_m	1
SCNN[12]	A+W	是	压缩编码	CSR	N**	$T_r \times T_c \times I \times F$**	$T_r \times T_c$	$(I \times F)^{-1}$**
本文(Conv)	A	是	多路选择	—	SF	$K_x \times K_y \times T_m$	1	$(K_x \times K_y \times T_m)^{-1}$
本文(FC)	A+W	是	多路选择	CSC	F	T_m	T_m	1

*Eyeriss并没有利用稀疏性加速计算, 所以系统并行自由度实际并没有影响性能。

**SCNN展开了$I \times F$的乘法器, $I \times F$=4×4, 使用笛卡尔乘积方法完成并行展开了突触与特征图进行计算。

从表 1 中可知, 几乎所有利用稀疏性加速卷积神经网络计算的加速器都受到了负载失衡的影响。文献 [11] 系统并行自由度仅受到参数稀疏度不平衡性影响, 这是因为它只对输出特征图方向进行了展开, 然而这种并行展开是有上限的, 当展开规模大于输出特征图数量时, 继续并行展开并不能提升计算性能。不考虑利用参数稀疏性的情况下, 和文献 [11] 相比, 在同样负载失衡损失条件下实现了 $K_x \times K_y$ 倍的并行展开规模和更小的单位乘法器资源开销。和同样仅利用激活稀疏性加速卷积计算的文献 [9] 相比, 本文实现了比它更低的负载失衡损失和更少的单位乘法器资源开销。综上所述, 本文提出的利用激活稀疏性加速卷积神经网络计算, 相比于同领域其他设计, 并行效率更高、额外资源开销更小。

3　硬件设计与实现

本文使用了具有 ARM+FPGA 的 Zynq 系列异构计算体来设计卷积神经网络加速器。ARM 处理器部分被称为 PS 端, FPGA 被称为 PL 端, 两者之间通过多种接口互联。PS 端即 ARM 处理器部分, 我们运行了 linux 操作系统用于从文件中读取权重和输入、测量时间并控制 DMA(direct memory access, 直接内存存取) 等各个模块进程。PL 端即 FPGA 部分, 我们使用 HLS 工具设计了共用端口的卷积计算模块与全连接计算 IP 核、可配置工作模式的输入网络和输出网络、非线性激活 ReLU 模块、极大池化模块和数据整合模块, 使用 SDSoC 工具完成了 DMA 与 PL-PS 交互控制设计。PS 与 PL 之间连接有 4 个高速的 HP 端口和 4 个中速的 GP 端口用于传输数据。整体架构如图 11 所示。

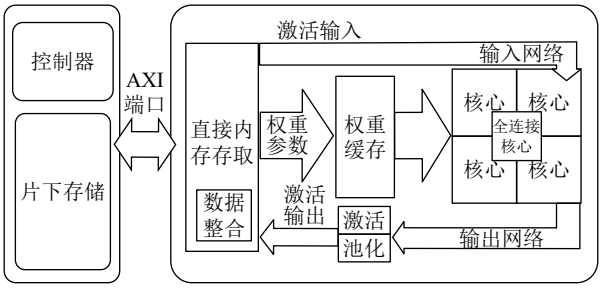


图 11　系统整体架构

Fig. 11　Schematic diagram of the overall architecture

网络参数和输入图片都被存储在文件系统中，在初始化时会读取到片下存储 DRAM，再通过 DMA 传输到卷积神经网络加速器中。在卷积计算过程，ARM 会控制卷积神经网络加速器按照规定顺序，以分块乒乓操作的形式读取权重并完成每一层网络的计算。计算完成之后 ARM 将卷积神经网络加速器计算结果与计算时间通过串口打印输出。

平铺展开设计的卷积神经网络加速器往往被认为对不同卷积层计算效率适应性不高，通过配置输入网络和输出网络的方式实现了多工作模式，这能让卷积神经网络加速器在计算不同卷积层时利用率更高。如图 12 所示，(a)代表所有核心共享激活输入，同时计算 1 组输入特征图和 4 组输出特征图，这种计算模式带宽压力最低，并行效率最高；(b)代表核心同时计算两个输入特征图两个输出特征图，这种计算模式适合卷积层输出特征图较少的情况；(c)代表所有核心计算不同的激活输入，这种计算模式适合输入特征图通道极深的情况；(d)代表核心左右分成两组分块计算，这种计算模式适合网络输入层尺寸极大的情况。通过多核心组合多种工作模式的方法，设计的卷积神经网络加速器能够支持最高 2 048 通道的输入特征图以及最大 1 024 尺寸的网络输入层计算。在计算 VGG-16 网络时，我们设计的卷积神经网络加速器每一层卷积计算的 PE 利用率均为 100%，没有因为适应性导致性能损失。

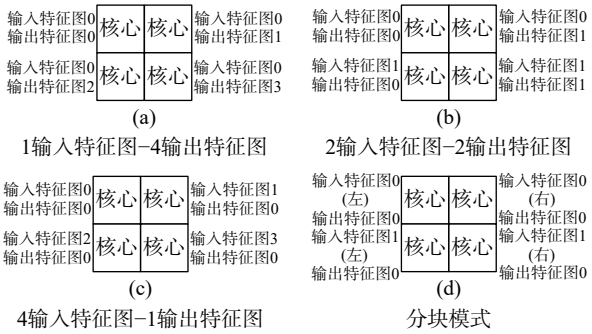


图 12　多种工作模式示意

Fig. 12　Schematic diagram of multiple working modes

使用了高级语言硬件设计工具链 vivado HLS 和 SDSoC 进行 FPGA 上的实现。首先在 vivado HLS 上完成计算 IP 核的设计与功能时序验证，再修改接口，移植到 SDSoC 上完成 IP 核读写接口与软硬件环境的设计，系统级验证测试带宽影响。所有功能和时序验证通过后使用 vivado 综合实现，得到实际资源报表和比特流 Bitstream。我们设计的卷积神经网络加速器资源使用情况如表 2 所示。工作频率保守的设置为 100 MHz，AXI 总线设置为 200 MHz。每个卷积核心使用了 32×9=288 个乘加器，75% 使用 DSP 实现，剩余的部分使用 LUT 实现，有 4 个这样的卷积核心；全连接层计算核心使用了 16 个乘加器，全部用 DSP 实现。

表 2　FPGA 资源使用情况

Table 2　Utilization report of FPGA

资源种类	使用量	总量	使用率/%
LUT	172 629	218 600	78.97
LUTRAM	19 782	70 400	28.10
FF	220 850	437 200	50.51
BRAM	474	545	86.97
DSP	880	900	97.78

生成的 linux 内核、可执行程序、Bitstream 文件最终打包，通过 SD 卡移植到开发板上实际测试。程序中调用了 SDSoC 提供的高精度时钟计数接口，计算完成后会将计算结果和每层所用时间打印到串口。

4　性能分析评估与比较

4.1　加速效果与并行展开方式比较

影响加速效果的最主要因素是激活输入的不平衡。局部上，多路选择需要一个周期，如果激活输入极为稀疏也要受到一个周期的时间惩罚；整体上，计算的整体时间取决于最长的计算时间队列，如果并行展开存在多个计算队列，负载失衡影响加速效果。基于这两个因素，我们使用 TensorFlow 环境，在 ImageNet 数据集下运行 VGG-16 网络，统计了不同的并行展开方式最终的加速效果，如图 13 所示。由于我们的设计单个核心内部无负载失衡损失 (1 IFM)，加速效果均比其他的并行展开方式优秀。相比于稠密卷积计算，设计的卷积神经网络加速器理论上能加速 1.92 倍。

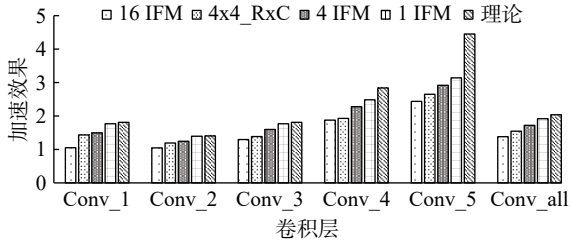


图 13　VGG-16 网络不同的并行展开方式加速效果比较

Fig. 13　Comparison of acceleration effects of VGG-16 networks with different parallel expansion modes

4.2　系统带宽分析

卷积神经网络加速器设计的一个重要考虑因素就是系统带宽。考虑到我们利用稀疏性加速了卷积计算, 计算速度更快, 系统带宽压力会比普通设计的加速器的带宽压力要高。分析了极限情况下 (计算加速 4 倍) 系统带宽情况, 如图 14 所示, 图中 IFM 代表输入特征图读操作, OFM 代表输出特征图写操作, OFMDM 代表输出特征图整合操作, W 代表权重读操作。

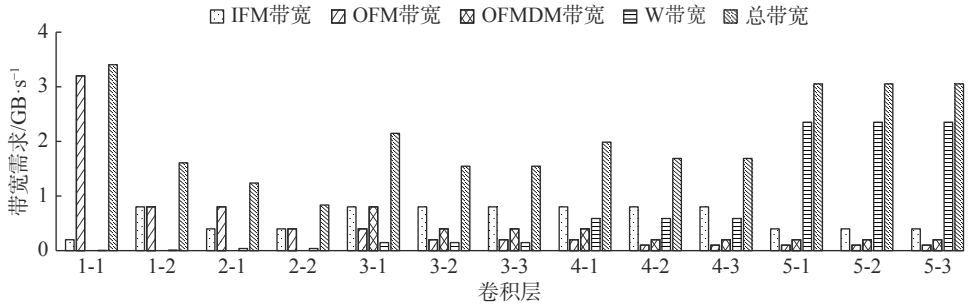


图 14　VGG-16 网络极限情况下系统带宽需求

Fig. 14　Limit case system bandwidth requirements of VGG-16 network

系统带宽瓶颈来自于两个方面, 我们使用的 DDR3 内存带宽上限为读写总共 4.2 GB/s, 对于 PS 与 PL 连接的 HP 端口, 每个端口读带宽 1.6 GB/s、写带宽 1.2 GB/s。本文分配了 3 个 HP 端口用于权重读取, 大部分情况下加速器都没有因为带宽瓶颈影响性能, 卷积计算各层带宽占用如图 14 所示。本文使用将激活结果量化为了 INT8, 并且对需要分块计算的卷积层使用多核心共享激活输入的方式降低了带宽压力。

4.3　计算时间与有效激活输入密度关系

本文测量了不同卷积层的实际有效激活输入密度和对应的计算时间, 如图 15 所示,其中计算时间根据各层不加速情况进行了归一化。从图 15 中可以看出, 加速器加速效果与有效激活输入密度在大部分区间成正比, 当理论激活输入密度靠近 0.25 时, 受到多路选择单元瓶颈影响 (从 4 路输入中选择一路), 即使激活输入密度更低加速效果也基本上不会增加。

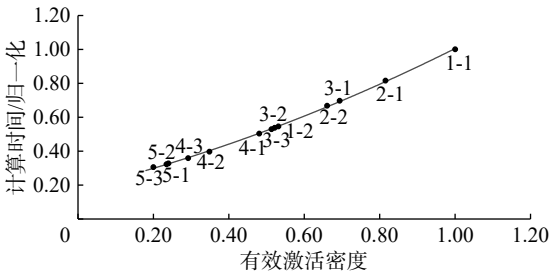


图 15　VGG-16 网络计算时间和有效激活输入密度关系

Fig. 15　Relationship between computing time and effective activation input density of VGG-16 network

表 3　稀疏卷积神经网络加速器 VGG-16 各层计算性能

Table 3　Sparse convolutional neural network accelerator VGG-16 calculation performance of each layer

层次	理论计算时间/ms	稠密计算时间/ms	稀疏计算时间/ms	计算量/GOP	稠密计算性能/GOP/s	稀疏计算性能/GOP/s
CONV1	16.81	21.52	12.82	3.87	180.00	302.13
CONV2	24.08	25.65	18.43	5.55	216.32	301.06
CONV3	40.14	44.13	25.67	9.25	209.57	360.33
CONV4	40.14	42.09	17.21	9.25	219.72	537.29
CONV5	12.04	12.52	4.15	2.77	221.68	668.72
CONV Total	133.22	145.90	78.28	30.69	210.37	392.10
FC Total	—	—	7.08	0.25	—	34.92
Total	—	—	85.36	30.94	—	362.48

表 4　与其他卷积神经网络加速器的 VGG-16 运行性能的比较

Table 4　Comparison of VGG-16 performance with other convolution neural network accelerators

	FPGA '16[23]	FPGA '16[24]	ICCAD '16[25]	ICCAD '16[25]	本设计
使用器件	Zynq XC7Z045	Stratix-V GSD8	Ultrascale KU060	Virtex690t	Zynq XC7Z045
量化精度	fixed 16 bit	fixed 8-16 bit	fixed 16 bit	fixed 16 bit	fixed 8-8 bit
MAC数量	780	1 963	1 058	2 833	1 152
工作频率/MHz	150	120	200	150	100
Conv性能/GOP·s^{-1}	187.80	136.50	310	488	392.10
FC性能/GOP·s^{-1}	1.20	—	173	170	34.92
全网络性能/GOP·s^{-1}	136.97	117.80	266	354	362.48
单位MAC卷积效率/%	80.26	28.97	73.25	57.42	170.18

*文献[25]中同时对全连接层计算了32个batch, 实际上单张图片FC性能分别为5.41和5.31 GOP/s

4.4　卷积神经网络加速器性能对比

本文完整地测量了卷积神经网络计算时间，并与同领域 FPGA 实现的卷积神经网络加速器设计的性能作了比较。本文使用多核心组合多种工作模式实现了更高的乘加器利用率，并利用稀疏性减少了卷积计算计算量。从表 3 中可以看出，在 VGG-16 网络、ImageNet 数据集环境下，本文设计的稀疏卷积神经网络加速器平均获得了 392.10 GOP/s 的卷积计算性能和 362.48 GOP/s 的整体网络性能。表 4 中显而易见的，和同领域 FPGA 实现的卷积神经网络加速器相比，本文的设计单位乘加器的卷积效率更高。与同器件实现的典型卷积神经网络加速器相比，本文设计的稀疏卷积神经网络加速器卷积性能提升了 108.8%，整体性能提升了 164.6%，优势显著。

5　结束语

卷积层庞大的计算量以及全连接层冗余的参数量为卷积神经网络硬件化实现带来了不小的挑战。本文分析了不同的并行展开方式和利用稀疏性的方式对并行效率和资源开销的影响，提出了相比于同领域其他设计，并行效率更高、额外资源开销更小的能够高效利用稀疏性加速卷积神经网络计算的并行展开方式，兼顾了并行效率和计算灵活度。本文设计的稀疏卷积神经网络加速器通过利用激活稀疏性和参数稀疏性的方式，降低了 VGG-16 网络 47.9% 的计算量和 85.23% 的参数量，极大地加快了卷积神经网络计算速度。

本文使用 Zynq XC7Z045 器件实现了 ImageNet 数据集下运行 VGG-16 网络 11.7 帧·s^{-1} 的计算性能，平均卷积性能 392.10 GOP·s^{-1}、整体网络性能 362.48 GOP·s^{-1}。和使用同样器件的同领域设计相比，本文的卷积性能提升了 108.8%，整体性能提升了 164.6%，具有明显的性能优势。

参考文献：

[1] RUSSAKOVSKY O, DENG Jia, SU Hao, et al. ImageNet large scale visual recognition challenge[J]. International journal of computer vision, 2014, 115(3): 211–252.

[2] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[C]// The 3rd International Conference on Learning Representations (ICLR2015).San Diego, CA, 2015.

[3] ZHANG Chen, LI Peng, SUN Guangyu, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks[C]//Proceedings of 2015 ACM/SIGDA International Symposium on Field-programmable Gate Arrays. New York, NY, USA, 2015.

[4] NATALE G, BACIS M, SANTAMBROGIO M D. On how to design dataflow FPGA-based accelerators for convolutional neural networks[C]//2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). Bochum, Germany, 2017.

[5] SHEN Yongming, FERDMAN M, MILDER P. Maximizing CNN accelerator efficiency through resource partitioning[C]//2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA). Toronto, ON, Canada, 2016.

[6] MA Yufei, CAO Yu, VRUDHULA S, et al. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks[C]//ACM/SIGDA International Symposium on Field-programmable Gate Arrays. Monterey, California, USA, 2017.

[7] SHI Shaohuai, CHU Xiaowen. Speeding up convolutional neural networks by exploiting the sparsity of rectifier units[J]. Computer vision and pattern recognition, 2017, 4:

1–7.

[8] CHEN Y H, KRISHNA T, EMER J S, et al. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. IEEE journal of solid-state circuits, 2017, 52(1): 127–138.

[9] ALBERICIO J, JUDD P, HETHERINGTON T, et al. Cnvlutin: ineffectual-neuron-free deep neural network computing[C]//2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). Seoul, South Korea, 2016.

[10] ZHANG Shijin, DU Zidong, ZHANG Lei, et al. Cambricon-X: an accelerator for sparse neural networks[C]//201649th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Taipei, China, 2016: 1−12.

[11] HAN Song, LIU Xingyu, MAO Huizi, et al. EIE: efficient inference engine on compressed deep neural network[J]. International symposium on computer architecture, 2016, 44(3): 243–254.

[12] Parashar A , RHU M , Mukkara A , et al. SCNN: An accelerator for compressed-sparse convolutional neural networks[C]//The 44th Annual International Symposium. TorontoCanada, 2017.

[13] OLSHAUSEN B A, FIELD D J. Sparse coding with an overcomplete basis set: a strategy employed by V1?[J]. Vision research, 1997, 37(23): 3311–3325.

[14] DAYAN P, ABBOTT L F. Theoretical neuroscience: computational and mathematical modeling of neural systems[M]. Cambridge, USA: The MIT Press, 2001.

[15] NAIR V, HINTON G E. Rectified linear units improve restricted Boltzmann machines[C]//Proceedings of the 27th International Conference on International Conference on Machine Learning. Haifa, Israel, 2010.

[16] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet classification with deep convolutional neural networks[C]//Proceedings of the 25th International Conference on Neural Information Processing Systems. Lake Tahoe, Nevada, 2012.

[17] HAN Song, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural networks[C]//Proceedings of the 28th International Conference on Neural Information Processing Systems. Montreal, Canada, 2015: 1135−1143.

[18] HAN Song, MAO Huizi, DALLY W J, et al. Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding[C]//International Conference on Learning Representations, San Juan, Puerto Rico,2016.

[19] MA Yufei, SUDA N, CAO Yu, et al. Scalable and modularized RTL compilation of convolutional neural networks onto FPGA[C]//201626th International Conference on Field Programmable Logic and Applications (FPL). Lausanne, Switzerland, 2016: 1−8.

[20] KU NG. Why systolic architectures[J]. IEEE Computer, 1982, 15(1): 300–309.

[21] GYSEL P, MOTAMEDI M, GHIASI S. Hardware-oriented approximation of convolutional neural networks[J]. Computer Vision and Pattern Recognition, 2016, 10: 1–8.

[22] DORRANCE R, REN Fengbo, MARKOVIĆ D, et al. A scalable sparse matrix-vector multiplication kernel for energy-efficient sparse-blas on FPGAs[C]//ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, California, USA, 2014: 161−170.

[23] QIU Jiantao, WANG Jie, YAO Song, et al. Going deeper with embedded FPGA platform for convolutional neural network[C]//Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, California, USA, 2016: 26−35.

[24] SUDA N, CHANDRA V, DASIKA G, et al. Throughput-optimized openCL-based FPGA accelerator for large-scale convolutional neural networks[C]//Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, California, USA, 2016.

[25] ZHANG Chen, FANG Zhenman, ZHOU Peipei, et al. Caffeine: towards uniformed representation and acceleration for deep convolutional neural networks[C]//Proceedings of the 35th International Conference on Computer-Aided Design. Austin, Texas, USA, 2016.

作者简介：

余成宇，硕士研究生，主要研究方向为算法硬件加速。

李志远，博士研究生，主要研究方向为计算机视觉。

毛文宇，助理研究员，主要研究方向为智能计算系统、人工智能算法、信号处理。主持国家自然科学基金项目 1 项，中科院创新基金项目 1 项，授权专利 1 项。发表学术论文 10 余篇。