# Cambricon-X: An Accelerator for Sparse Neural Networks

——国防科大2020年高性能评测与优化课程小组讨论

**第六组：郭莉娜、张新新、席闻**

指导：龚春叶、甘新标、杨博

# 一、需求分析

# 一、需求分析

- 神经网络的大规模应用使得NN加速器逐渐兴起；
- 由于密集的计算和访存，使得现有NN加速器处理大规模神经网络成为了挑战；
- 在大量裁剪后现有加速器不一定能改善性能和能耗；

# 二、技术背景与动机

# 神经网络简介

效果最好的神经网络

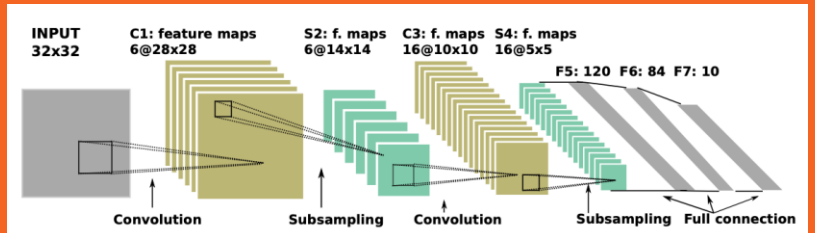


Fig. 2. A typical CNN: LeNet-5 [25].

效果最好的神经网络有卷积神经网络和深度神经网络。通常一个CNN包括四种类型的Layer：

1. 卷积层
2. 池化层
3. 分类层
4. 正则化层

神经网络的突触数量越来越大，使得计算量和访存越来越密集！

# 神经网络简介

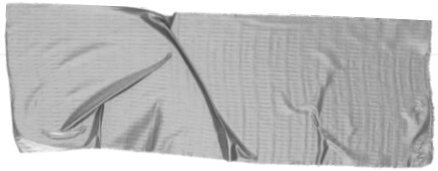
稀疏神经网络

最好的裁剪效果是由Han等人于2015年提出的，主要包括三个步骤：

1. training connectivity
2. pruning connection
3. training weight

重复以上步骤直到无法裁剪从而获得高压缩率。

通过上述算法，典型的NN平均可以压缩12%而无精确度损失。



# 动机

虽然通过裁剪算法，大量的运算和访存减少了，但是现有的硬件平台（包括CPU、GPU、FPGA和自定义的硬件）并不能从中受益，这是因为缺少专门为这个非常规的、稀疏的神经网络的硬件支持。

- → **CPU平台**
  除了LeNet-5，稀疏神经网络的表现比他们稠密版本的差；
- → **GPU平台**
  平均的性能增益只有23.43%
- → **原因**
  固有的带宽限制和经常性的非计算开支。

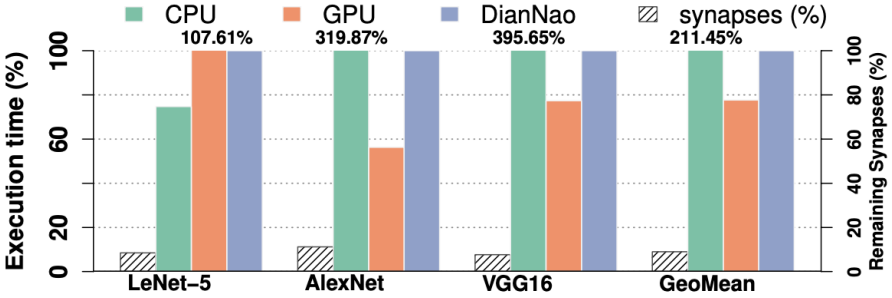



Fig. 3. The speedup of sparse NN vs. dense NN on CPU, GPU and DianNao.

通过以上观察，有必要构建一个高效的架构充分利用现代稀疏神经网络非常规和稀疏的特性

# 三、加速器设计

# 总体方案

- 控制处理器(CP)
- 缓冲控制器(BC)
  - 索引单元(indexing unit)
- 神经元缓冲区(NBin和NBout)
- 直接内存访问模块(DMA)
- 计算单元(CU)
  - Tn个处理单元(PEs)
  - 拓扑连接Fat-tree防止线路拥挤
- 16-bit运算单元

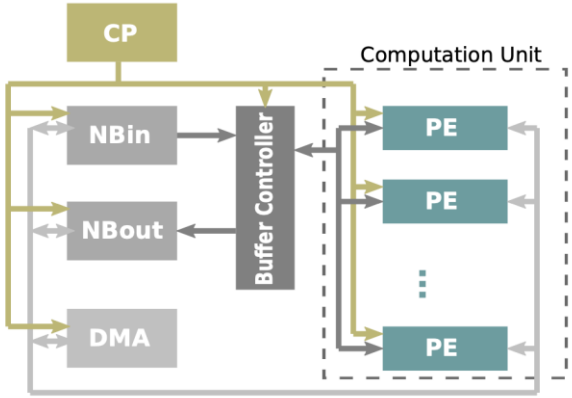



Fig. 4. Accelerator architecture.

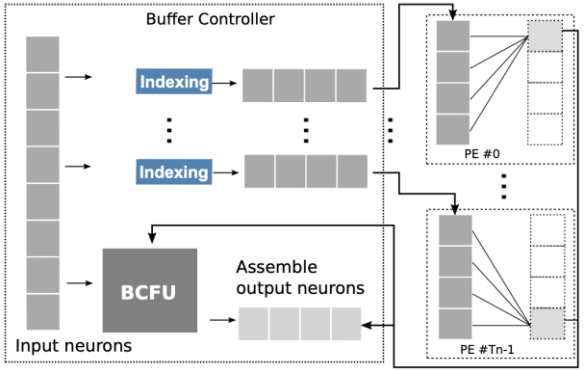



Fig. 5. Buffer controller architecture.

# 计算单元

用于高效计算神经网络的核心操作，

即：使用多PE进行向量乘加操作。

- 神经元突触缓冲(SB)

SB用于存储分布式突触

- PE神经网络功能性单元(PEFU)

PEFUs主要用于神经网络的乘法和加法运算

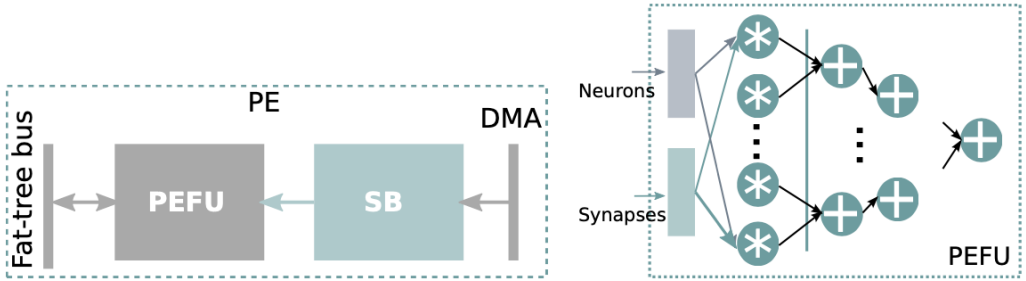



Fig. 6. (a) The architecture of the PE. (b) The architecture of the PEFU.

# 缓冲控制器

缓冲控制器用于将必要的神经元传输到PE中，并在PE上安排计算，执行较少的计算密集型操作。

- BCFU

BCFU主要用于储存IM选择的神经元

- IM

IM是加速器的关键部件，它用于索引具有不同稀疏度的稀疏神经网络所需的神经元。

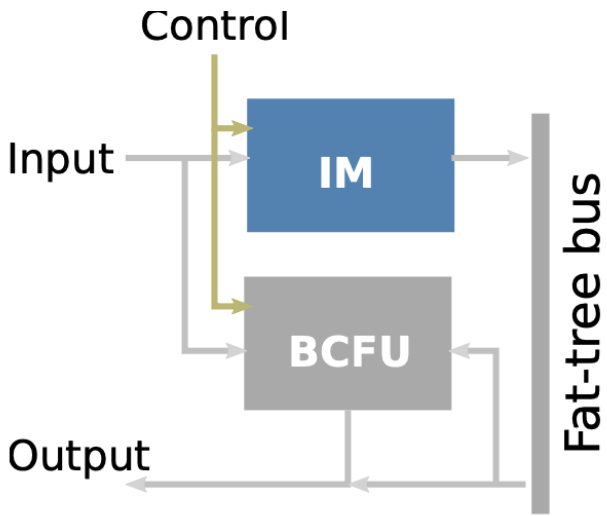



Fig. 8. The architecture of the buffer controller.

## 控制处理器

CP的设计目的是为了高效灵活地控制各种指令的执行。这些指令用于数据组织、执行协调和内存访问等，并存储在一个小的指令缓冲区中。为了减轻最终用户的编程负担，作者提供C++中的编译器来生成高效的指令。

## 神经元缓冲区

NB包括NBin和NBout，分别用于存储输入和输出神经元：输入神经元从NBin中选择，然后发送给所有的PEs进行计算，输出神经元在计算后收集到NBout。存储在NBs中的神经元排列有序，不考虑稀疏网络的各种连接模式。

# 互联

我们采用Fat-tree互连拓扑结构，以提高它们之间的数据移动效率。使用Fat-tree互连有两个原因：

- 与其他非树互连拓扑相比，使用Fat-tree可以避免由于BC和PE之间的不平衡延迟而导致的长关键路径；
- 与其他树状互连拓扑相比，Fat-tree可以提供私有连接来缓解网络拥塞，因为发送到不同PE的数据是独立的。

# 通信

片外存储器和片内缓冲器（包括NBin、NBout和SB）之间的数据通信是通过直接存储器存取（DMA）实现的。为了平衡不同PEs的执行，避免内存访问的拥塞，我们首先将所需的突触分成块。然后，在很短的时间内，内存访问端口一次只能分配给一个PE，因此每个PE在此期间只能加载几个块。在这种情况下，每个PE都会有一些突触（如果不是全部的话）在不同的周期执行相应的计算。

# 映射

层的代表性类型包括卷积层、池化层、分类器和正则化层。卷积层通过将多个输入特征映射卷积成共享或私有核来构造输出特征映射，占整个网络处理的85%左右。特征映射的数据按映射id的顺序存储在片上缓冲区中，在卷积层的计算过程中，所有输出神经元将完成与BC的NBin数据相关的计算，然后替换为新的输入神经元，最大限度地重用NBin中的输入神经元，减少片外存储器的访问。

# 编程模型

- Libraray based Programming

为了减轻编程负担，我们为加速器提出了一个基于库的编程模型。其基本思想是提供一组高级（例如，C/C++级）库函数，每个函数对应于一个基本的神经网络操作，以便用户可以直接用高级语言调用加速器。

- Programing Framework

为了获得性能可移植性，我们还将实现的编程库集成到广泛使用的深度学习框架中，如Caffe[30]。因此，最终用户可以直接利用Caffe的接口（即网络配置文件），而无需修改其代码。

# 四、实验结果

# 性能

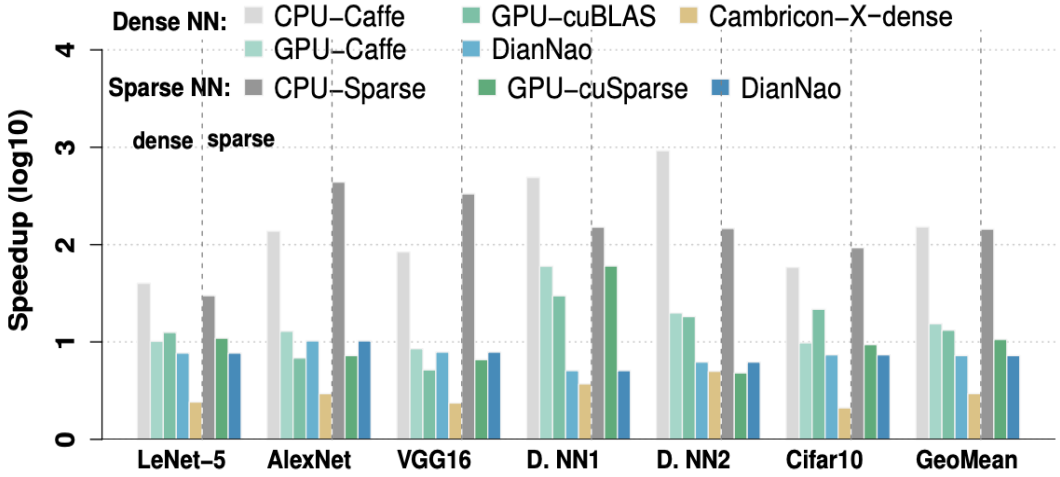
- 稠密表示

在评估的基准上，Cambricon-X平均比CPU-Caffe、GPU-Caffe和GPU-cuBLAS分别快51.55x、5.20x和4.94x

- 稀疏表示

Cambricon-X比CPU-Caffe、GPU-Caffe和GPU-cuBLAS快151.82x、15.32x和13.18x

与DianNao相比，Cambricon-X仍然达到了7.23x的加速，这很好地体现了Cambricon-X的效率



# 能耗

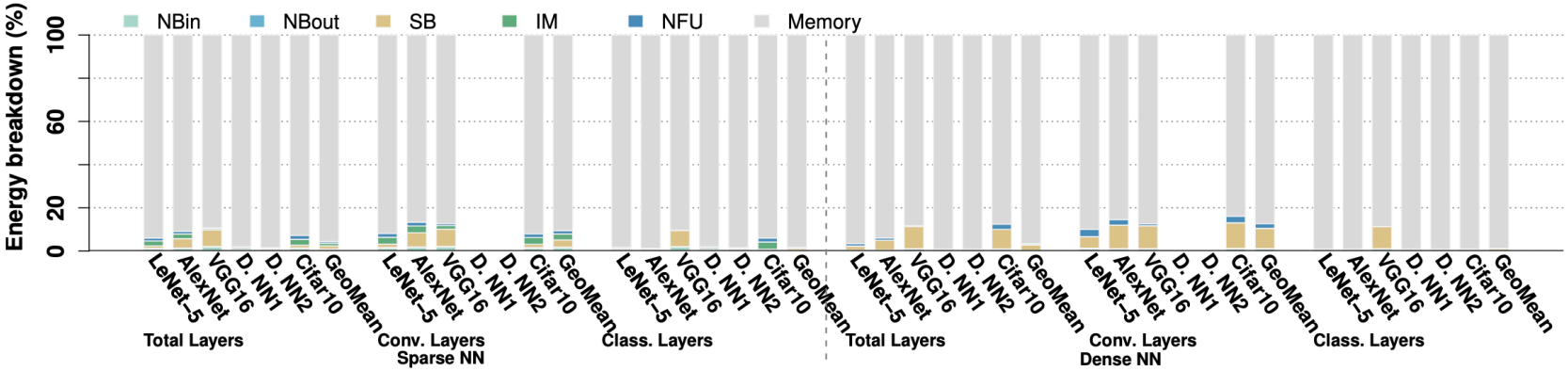



Fig. 18. Energy breakdown with memory accesses.

图12作者报告了GPU、DianNao和Cambricon-X在所有基准上的能耗比较，其中还包括了芯片外内存访问的能耗。与GPU平台相比，Cambricon-X在密集和稀疏网络下平均分别实现37.79x和29.43x的能效提升。平均而言，与DianNao相比，Cambricon-X在密集和稀疏网络下都能达到6.43倍的能效。作者观察发现，本加速器在AlexNet网络中，会超越GPU和DianNao获得最佳性能。其主要原因是由于AlexNet中卷积层的内核尺寸大于其他网络，从而大大提高了内存效率。此外，Cambricon-X在稠密表示中比DianNao减少了1.70倍的能耗，这表明Cambricon-X在处理稠密网络时也很节能。

# 五、分析

# 相关工作

通过上述对加速器结构的讲解，以及对其实现的效果评估，发现该加速器存在两个局限。

- 硬件实现较比较传统的加速器结构复杂，在实现上会困难点
- 在运行中，对于片外存储器访问的能耗问题严重的原因。

对于第一点，有没有集成度更高的硬件代替，避免从零开始搭建，或者有没有其他简化的结构但能实现一样的功能。

对于第二点，进一步了解为什么能耗会高，文中没有做出解释，我们可以探讨研究，有没有办法改善这种能耗。

# 谢谢