低功耗深度学习加速器研究

摘要：深度学习神经网络（DNNs）在许多领域有越来越多的应用，然而深度学习神经网络需要很强的计算能力并且对硬件资源占用极大，所以它很难嵌入智能手机、智能手表和机器人等移动设备。FPGA是由于其开发成本低、可重复性强、易于修改、周期短等特点，已然成为了最有可能加速深度学习神经网络的平台之一，但是有限的带宽和片上存储资源限制了加速性能。为了解决这些问题，构建出一个能根据用Caffe定义的高层次描述的网络结构和所使用的FPGA开发板，自动生成可综合加速器的框架。它首先把高层次DNN描述文件转换用它内部指令集描述的微数据流图。它的编译器将DNN操作分块、分批地执行，以最大化复用数据和最大化利用FPGA的内存和其它资源。（通过对卷积层，和池化层算法的优化和硬件的再分配？？？进一步提高了性能

结果对比？多少倍的提高 ？功耗降低？

关键词：DNN、FPGA、加速器、功耗。

Abstract:

Keywords: DNN、FPGA、accelerator、power consumption.

设计说明：

Hsarma开发一套生成DNN加速器的框架。根据他所做的工作搭建实验平台，并测试性能功耗，在他的基础上改进

设计标准：？

设计原则：？

技术资料：？Xilinx ug1144 957 Zedboard软硬件协同设计（书）？

目录

引言：现阶段的深度学习神经网络还是以大量的计算能耗，大量占用内存资源为代价来达到较高的准确率，然而，目前，嵌入式视觉系统广泛应用于自动半自动驾驶、消费电子（相机和移动电话）、电动玩具，它们硬件资源少，电池容量小且要求准确率高和实时性，不能直接运行深度学习网络，需要一个低功耗、轻量化、低成本的系统。尽管ASIC以其性能和高效率非常适合于DNNs，但是定制的ASIC不能适用于不断变化的DNN模型，而且ASIC具有高的非重复性成本和长的开发周期。而FPGA兼具高性能、通用性、重复性强。之前也有用FPGA实现DNN加速：研究[ 1][ 2][ 3][6][14]着重于计算单元的优化，他们要么忽视了对外部存储的进行操作的能力，要么把加速器直接连接到外部存储。研究[12]通过数据复用来减少对外部数据的访问，但是这种方法不能实现总体性能的提高，而且对神经网络中不相同的层都需要重新配置FPGA， Chen Zhang。[4]使用多种优化技术比如loop tiling量化分析它的对计算能力和带宽的需求，roofline model[ ]（一种减少外部数据访问，增大内部数据复用来减小带宽对计算性能的限制，进而充分使用所有的硬件计算资源的模型）确定一种最大化利用现有的FPGA资源获得更高性能的解决方案，而且对于不同的神经网络层不需要重新配置FPGA就能实现加速，但是对于不同的神经网络结构不同的FPGA都需要重新手动计算确定解决方案。[28] 提出了一个可伸缩的硬件结构用于加速大规模多个神经网络层的视觉系统，基于大量并行的特征提取阵列？但仅适用于图像检测领域，且需要多个FPGA的支持。

Zynqnet[7 ] 用 Netscope CNN Analyzer分析比较不同的CNN拓扑结构，最后选出SqueezeNet作为基本神经网络结构，因为很少的参数设置非常适合FPGA的片上SRAM，很短的训练周期和简洁的结构使得后期优化变得容易。Zynqnet CNN在Caffe框架下用GUP训练，高层次综合后生成ZynqNet FPGA Accelerator。但？？？

FPGA不能像ASIC通过增加面积来增加存储空间，增加带宽来增加存储访问速度，加速器要对不同的DNN在不同的FPGA上都能实现加速，但DNNs的超高存储访问、不同DNN模型的体量变化大和运算操作数量变化范围大，使得DNN在FPGA上加速变得十分困难。

所以，为了克服FPGA在片上存储的限制，要协同优化加速器结构和相应的执行表？

因此建立了一个叫做DNNWEAVER的框架，它能根据不同的DNN模型和不同的FPGA开发板生成可综合的加速器。

贡献：

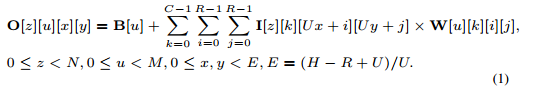
实验结果：

下面分几个部分：

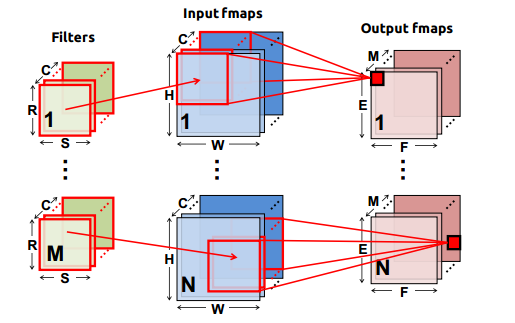
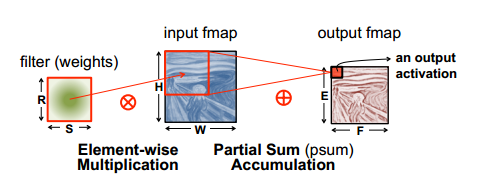
DNN理论基础：

DNN以CNN为基础，加入池化层（Pooling layer），内积层（Inner product）、激活层（Activation layer）、标准化层（Normalization layer）、

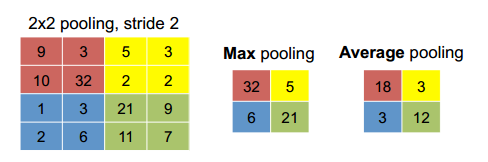
卷积层：如图所示，通过将一个权值滤波器在输入上面滑动并与相应位置上的输入值相乘，所有的积再相加得到这个滤波器当前所在中心位置的输出，二维权值滤波器以一定的步长滑过输入特征图产生的所有输入值按位置排列形成的阵列就是二维输出特征图，二维权值滤波器一般为1 x 1 、3 x 3、7 x 7等奇数，所以输出值的位置一般在滤波器的中心。高维卷积的输入是多个二维特征图堆叠而成的，每一个特征图称为一个通道，每一个通道都和相应的高维滤波器的一层卷积，再把所有通道的卷积值相加，得到输出特征图的一个通道，多个高维滤波器通过相同的运算能产生相等数量的输出通道，再把这些通道堆叠起来生成输出特征图。



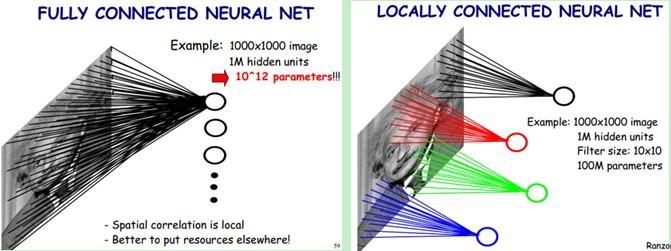
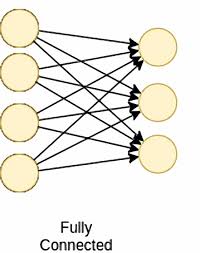
O I W B 是输出特征图矩阵、输入特征图矩阵、滤波器矩阵、偏差矩阵。

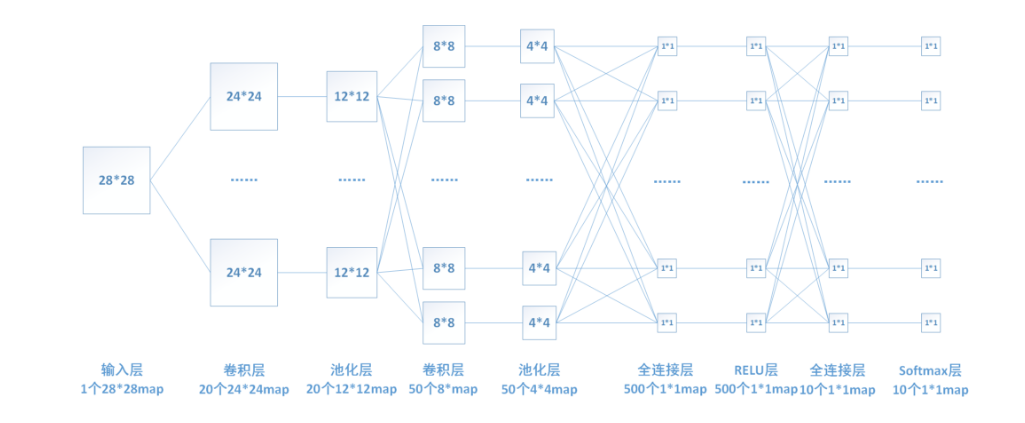


池化层：池化层让网络更加稳定，不受微扰的影响，还能缩小网络的大小。如图，在输入特征图的每一个通道，用一定大小的方框，（比如2x2），以一定的规则（比如，选取方框中的平均值、最大值、最小值）得到一个输出值替代方框中的所有输入值，类似于卷积，当方框滑过整个输入通道时，得到输出通道，所有输出通道堆叠形成输出特征图。

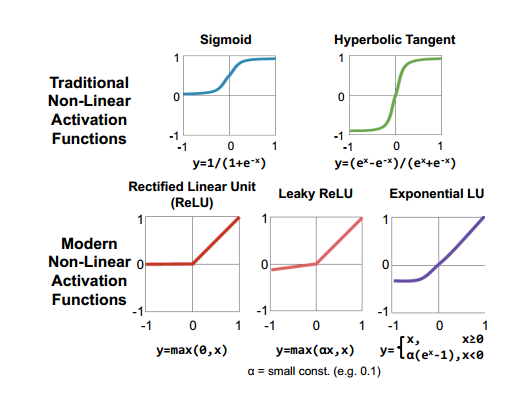


内积层：在Caffe中全连接层（Fully Connected ）又被称为内积层（Inner product）,顾名思义，内积就是做输入向量和权重矩阵的内积，本质是由一个特征空间线性变换到另一特征空间，输出层的每一个元素都会受到所有输入元素的影响。在DNN中，内积常常出现在最后几层，用于对前面设计的特征做加权和。





激活层：在卷积和内积计算之后往往有非线性激活函数，如果没有非线性激活函数或激活函数是线性的，那么多个内积层可以用一个层来代替，失去了深度学习神经网络在更高层次提取特征的意义。非线性化函数有传统的Sigmoid、Tanh和ReLU，其中ReLU由于其实现方式简单能够快速训练网络，在DNN中有广泛运用，为了提高准确率，ReLU也有许多变种，比如Leaky ReLU、Parametric ReLU和Exponential LU。



标准化：为了加速训练速度和提高准确率，必须控制输入值的分布。标准化先计算出滑动窗口中元素的平方和，再将平方和非线性化，然后乘以输入元素得到相应的输出

ZedBoard：

ZedBoard是一个基于Xilinx ZynqTM-7000全可编程SoC的开发平台，由双Corex-A9处理系统（PS）和8，5000可编程逻辑（PL）单元组成，丰富的板上接口和易于扩展的性质使得它成为理想的开发平台。

部分特性：

Memory

o 512 MB DDR3 (128M x 32)

o 256 Mb QSPI Flash

Interfaces

o USB-JTAG Programming using Digilent SMT1-equivalent circuit

 Accesses PL JTAG

 PS JTAG pins connected through PS Pmod

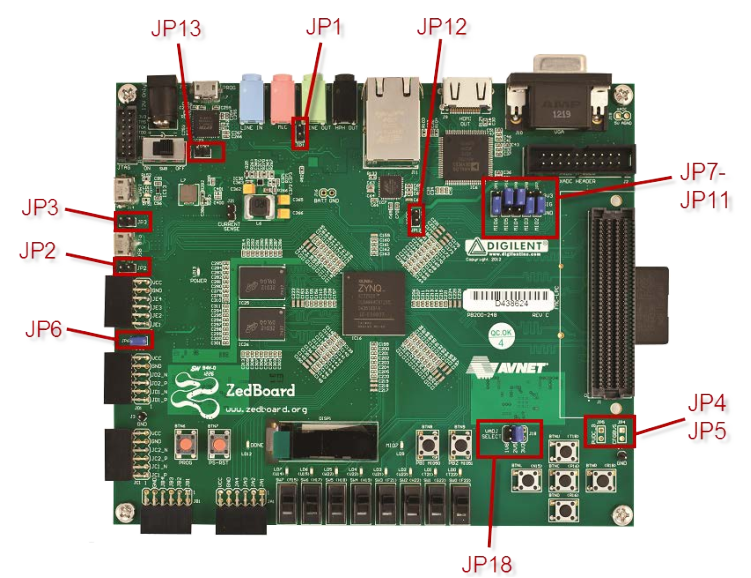
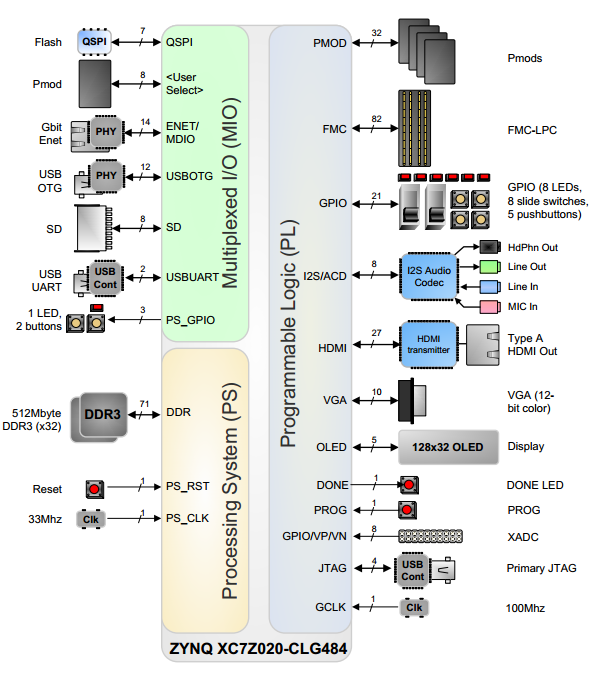
USB 2.0 FS USB-UART bridge

SD Card

On-board Oscillators

o 33.333 MHz (PS)

o 100 MHz (PL)

参考文献：

[1] D. Aysegul, J. Jonghoon, G. Vinayak, K. Bharadwaj,C. Alfredo, M. Berin, and C. Eugenio. Accelerating deep neural networks on mobile processor with embedded programmable logic. In NIPS 2013. IEEE, 2013

[2] S. Cadambi, A. Majumdar, M. Becchi, S. Chakradhar, and H. P. Graf. A programmable parallel accelerator for learning and classification. In Proceedings of the 19th international conference on Parallel architectures and compilation techniques, pages 273{284. ACM, 2010.

[3] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi. A dynamically configurable coprocessor for convolutional neural networks. In ACM SIGARCH Computer Architecture News, volume 38, pages 247{257. ACM, 2010.

[4] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In FPGA, 2015

[6] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun. Cnp: An fpga-based processor for convolutional networks. In Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on, pages 32{37. IEEE, 2009.

[7]

[12] M. Peemen, A. A. Setio, B. Mesman, and H. Corporaal. Memory-centric accelerator design for convolutional neural networks. In Computer Design (ICCD), 2013 IEEE 31st International Conference on, pages 13{19. IEEE, 2013.

[14] M. Sankaradas, V. Jakkula, S. Cadambi, S. Chakradhar, I. Durdanovic, E. Cosatto, and H. P. Graf. A massively parallel coprocessor for convolutional neural networks. In Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference

[28] Clément Farabet, Berin Martini, Polina Akselrod, Selçuk Talay, Yann LeCun, and Eugenio Culurciello. Hardware accelerated convolutional neural networks for synthetic vision systems. In ISCAS, 2010.