课程项目：基于自动代码生成框架实现神经网络
PART I

April 8, 2019

1 背景介绍

深度学习(deep learning)在近年得到了长足的发展，用于构建和计算深度学习网络的框架也层出不穷，如caffe,pyTorch,TensorFlow等等。然而，当今主流的深度学习框架都需要依赖硬件相关的手工优化的运算库实现，如在CPU上使用Intel MKL, 在NVIDIA GPU上使用cuDNN等。但是，手工优化的运算库开发周期长，难度大，为在新兴硬件上（如FPGA）部署深度学习算法带来了困难；同时，这些运算库在设计时考虑的是通用情景，对于特定输入的规模缺少定制化的优化。

为了进一步提高计算性能和对硬件适配的灵活性，学界涌现了许多高层次语言编程框架，如Halide, TVM, Tensor Comprehension等。它们能做到对特定的硬件进行动态代码生成，利用编译技术，从算子的数学描述，经过中间表示(Intermediate Representation)的多次变换，最终生成源代码(C、CUDA等)或可执行程序。

2 目的与任务

本课程项目分成多个部分，要求使用TVM框架，对深度学习中的各种算子进行实现和调度，并让同学们在此过程中了解编译技术在深度学习中的运用。本部分（Part I）的任务包括：

1. 学习TVM的基本概念，学习如何使用TVM表达各种数学运算
2. 学习深度学习中的各种基本算子，如卷积、ReLU、池化(Pooling)等
3. 用TVM实现卷积神经网络中的卷积层算子、ReLU算子，2x2池化层算子、flatten操作，全连接层算子，以及相应的反向传播算子，进行单元测试

3 具体要求

3.1 实验环境：

Linux, Python-3.6, tvm-v0.5。

3.2 技术路线：

1. 安装TVM并学习TVM教程. TVM主页为https://tvm.ai/，TVM的安装包和使用教程都在主页上提供。虽然TVM支持Windows, Mac, linux环境的安装，但我们推荐使用Linux环境(这也是最终的测评环境)。

2. 根据我们在第6节中提供的数学定义，实现五个算子及其反向传播：二维卷积，ReLU，2x2池化，flatten以及全连接。为方便测评，对算子的要求必须严格遵循我们提供的数学定义(包括数据的分布，如我们提供的是NHWC卷积，如果你提交了NCHW的卷积操作，将导致扣分)。

3.3 提交要求：

请提交写有这些算子实现的python源文件，要求严格按照我们提供的模板实现，文件名为学号（如同学小天的学号为160001XXXX，则他提交的文件名为160001XXXX.py）。提交网址为http://xzc.cn/eSlkSHS6ZH,提交需注明学号，允许提前提交，可重复提交，有效提交是最后一次提交，超过截止日期的提交无效。

3.4 代码标准：

1. 提交实现所规定的算子的python源文件，要求五个算子及其反向传播实现在一个文件中，有充足的注释且清晰指明每个算子定义的位置；
2. 要求每个算子单独实现在一个函数中，函数的输入为算子的输入张量，输出为结果张量(参照第4节中的样例)。
3. 请确保实现的函数名、参数及返回值（包括顺序）定义与我们提供的模板完全一致，否则在测评时会因无法找到对应算子而无法计分。模板文件请在教学网下载。

3.5 评分标准:

本部分项目满分100分，每个算子及其反向传播共占20分。测评正确性方法为我们给定随机输入数据，验证输出是否正确，每个算子（前向和反向分开计算）各有10个测试例子，每通过一个得一分，全部通过得100分，比较输出数据内容是否正确。

要求每个同学独立完成，禁止抄袭，提交的代码将通过查重系统检验，发现抄袭现象将直接判处0分。

4 样例

为了帮助更好理解如何完成任务，我们用矩阵加法说明这个过程。矩阵加法的数学定义为：

$$Output[i,j] = A[i,j] + B[i,j] \tag{1}$$

根据数学定义，使用tvm提供的张量描述语言编写的程序为：

```
# example of matrix add
def matrix_add(A, B):
    # A and B are input matrices
    M, N = A.shape   # the shape of matrices
    C = tvm.compute((M, N), lambda i, j: A[i, j] * B[i, j], name="C")
    # return the output tensor's op, and the intermediate buffers in a list
    return C
```

对于本次project中要求的算子，其函数名、参数和返回值定义请见提供的模板文件。

5 提示

- TVM的tutorial中有很多有价值内容可供参考
- 有的操作如果用tvm中的特殊语句可能会很简单，如使用tvm.if_then_else完成padding操作
- 在学习深度学习算子时，可以先查找相关技术博客理解各个算子进行的操作（如参考文献中的[3][4]），然后再查看第6节中的数学定义
- 提供的模板文件中已经确定了参数和返回值，请按照其规定补充代码。
- 本地测试时可以使用已有的框架如PyTorch验证自己的算子是否能输出正确结果。

6 算子数学定义

6.1 二维卷积

卷积层的数学定义为：

$$Output[n,h,w,o]=\sum_{rx=0}^{H_k}\sum_{ry=0}^{W_k}\sum_{rc=0}^{C}Image[n,h+rx,w+ry,rc]\times Filer[o,rc,rx,ry] \quad (2)$$

这里$0\le n<N,0\le h<H-(H_k-1),0\le w<W-(W_k-1),0\le o<K$。其中$Image$输入图像，为四维张量，第一维为batch_size，大小为N，第二维和第三维是图像高度和宽度，大小分别为H,W。第四维为通道(channel)数，大小为C。$Filter$为卷积核，也是四维张量，第一维是输出图像的通道数，大小为K，第二维是输入通道数，大小为C，第三维和第四维是卷积核的高度和宽度，大小为H_k,W_k数。我们不考虑padding和stride > 1情形。

卷积层的反向传播运算本质上还是卷积运算，只是略有差别，对卷积核的导数的数学定义为：

$$PFilter[o,c,h,w]=\sum_{rn=0}^{N}\sum_{rx=0}^{H_1}\sum_{ry=0}^{W_1}Image[rn,h+rx,w+ry,c]\times POutput[rn,rx,ry,o] \quad (3)$$

这里$H_1=H-(H_k-1),W_1=W-(W_k-1),0\le o<K,0\le c<C,0\le h<H_k,0\le w<W_k$，$POutput$是最终的loss对于$Output$张量的导数，也是四维张量(这种保证是由于loss函数的结果一定是标量，这是绝大多数深度学习框架所采用的规定)。对输入图像的导数的数学定义为：

$$PImage[n,h,w,c]=\sum_{rx=0}^{H_k}\sum_{ry=0}^{W_k}\sum_{ro=0}^{K}Filter[ro,c,H_k-rx-1,W_k-ry-1]\times ZPOutput[n,h+rx,w+ry,ro] \quad (4)$$

这里$0\le n<N,0\le h<H,0\le w<W,0\le c<C$，注意卷积核被反转后再做的卷积操作，而且并不是在$POutput$张量上直接进行卷积，必须先对$POutput$张量进行zero-padding操作得到$ZPOutput$，padding方法为在$POutput$张量四周围一圈0，具体参考图1。

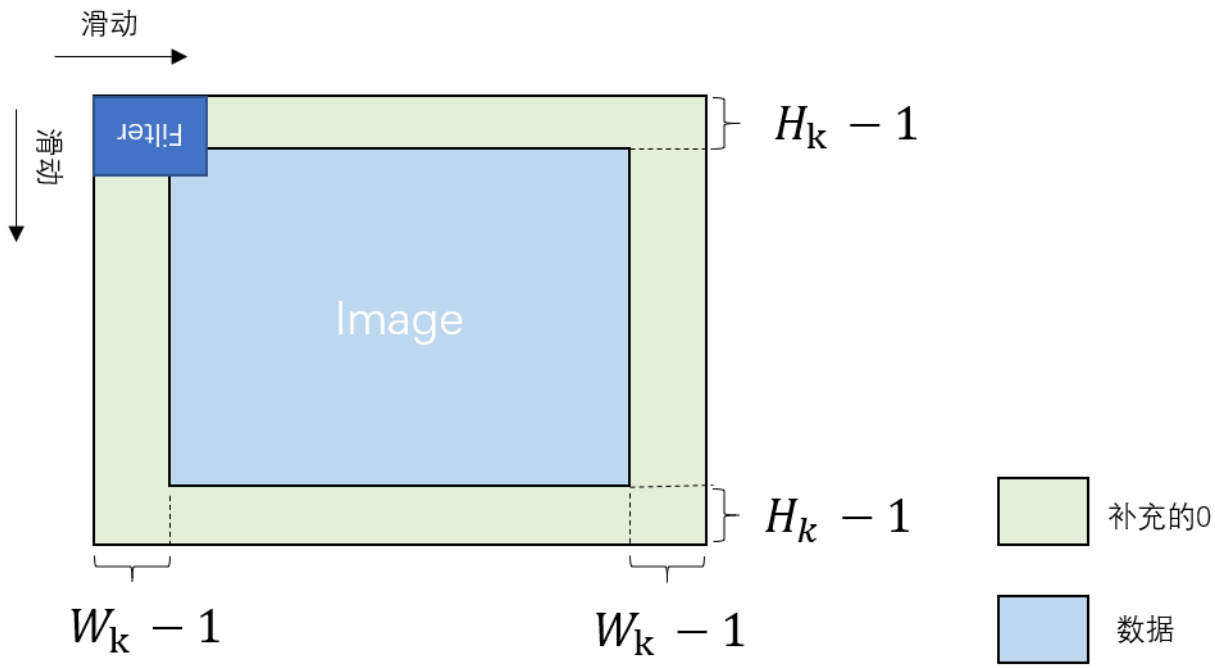

Figure 1: 将POutput补充0后与反转的Filter进行卷积操作

6.2 ReLU

ReLU层数学定义为：

$$Output[n,i,j,c] = max(0, Image[n,i,j,c]) \tag{5}$$

各个维度的大小及含义参考卷积描述，这里不再赘述。

ReLU层反向传播数学定义为：

$$PImage[n,i,j,c] = POutput[n,i,j,c]\ if\ Image[n,i,j,c] > 0.0\ else\ 0.0 \tag{6}$$

6.3 2x2池化

2x2池化算子数学定义为：

$$Output[n,i,j,c] = max_{0 \le rx < 2, 0 \le ry < 2} Image[n, i*2+rx, j*2+ry, c] \tag{7}$$

这里注意$0 \le i < H/2, 0 \le j < W/2$

其反向传播数学定义为：

$$\begin{aligned} PImage[n,i,j,c] =& POutput[n,i//2,j//2,c] \\ & if\ [n,i,j,c] == argmax_{0 \le rx < 2, 0 \le ry < 2} Image[n, (i//2)*2+rx, (j//2)*2+ry, c] \\ & else\ 0.0 \end{aligned} \tag{8}$$

这里//代表整除。我们这里也仅仅考虑2x2池化。

6.4 flatten操作

flatten操作的数学定义为：

$$Output[n,i] = Image[n, i//(W*C), (i//C)\%W, i\%C] \tag{9}$$

其反向传播数学定义为：

$$PImage[n,i,j,c] = POutput[n,(i*W+j)*C+c] \tag{10}$$

6.5 全连接

全连接算子本质上是矩阵乘法，其数学定义为：

$$Output[n,i] = \sum_{k=0}^{K} Input[n,k] \times Weight[k,i] \tag{11}$$

这里的$Weight$是权重张量，$Input$和$Output$的第一维都是batch_size。我们没有考虑偏置项。

其对权重$Weight$的反向传播数学定义为：

$$PWeight[k,i] = \sum_{n=0}^{N} Input[n,k] \times POutput[n,i] \tag{12}$$

对称地，对输入$Input$的反向传播数学定义为：

$$PInput[n,k] = \sum_{i=0}^{M} POutput[n,i] \times Weight[k,i] \tag{13}$$

7 参考文献

1 TVM官网，https://tvm.ai/

2 Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Meghan Cowan, Haichen Shen, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, Arvind Krishnamurthy "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning" OSDI'18

3 卷积算子的反向传播：https://medium.com/@2017csm1006/forward-and-backpropagation-in-convolutional-neural-network-4dfa96d7b37e

4 理解CNN卷积层与池化层计算：https://blog.51cto.com/gloomyfish/2108390