

高性能计算程序设计基础 秋季 2021

提交格式说明

按照实验报告模板填写报告，需要提供源代码及代码描述至

<https://easyhpc.net/course/129>。实验报告模板使用 PDF 格式，命名方

式为高性能计算程序设计_学号_姓名。如果有问题，请发邮件至

jiangjzh6@mail2.sysu.edu.cn, liuyh73@mail2.sysu.edu.cn 询问细节。

任务 1:

通过 CUDA 实现通用矩阵乘法（Lab1）的并行版本，CUDA Thread

Block size 从 32 增加至 512，矩阵规模从 512 增加至 8192。

通用矩阵乘法（GEMM）通常定义为：

$$C = AB$$

$$C_{m,n} = \sum_{n=1}^N A_{m,n} B_{n,k}$$

输入：M, N, K 三个整数（512 ~8192）

问题描述：随机生成 M*N 和 N*K 的两个矩阵 A,B,对这两个矩阵做乘法得到矩阵 C。

输出：A,B,C 三个矩阵以及矩阵计算的时间

任务 2:

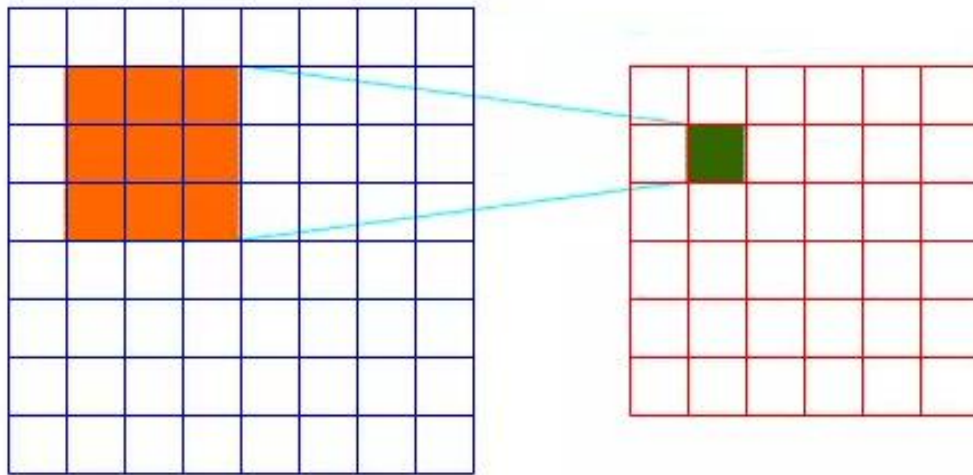
通过 NVIDIA 的矩阵计算函数库 CUBLAS 计算矩阵相乘，矩阵规模从 512 增加至 8192，并与任务 1 和任务 2 的矩阵乘法进行性能比较和分析，如果性能不如 CUBLAS，思考并文字描述可能的改进方法（参考《计算机体系结构-量化研究方法》第四章）。

CUBLAS 参考资料《CUBLAS_Library.pdf》，CUBLAS 矩阵乘法参考第 70 页内容。

CUBLAS 矩阵乘法例子，参考附件《matrixMulCUBLAS》

任务 3:

在信号处理、图像处理和其他工程/科学领域，卷积是一种使用广泛的技术。在深度学习领域，卷积神经网络(CNN)这种模型架构就得名于这种技术。在本实验中，我们将在 GPU 上实现卷积操作，注意这里的卷积是指神经网络中的卷积操作，与信号处理领域中的卷积操作不同，它不需要对 Filter 进行翻转，不考虑 bias。



任务一通过 CUDA 实现直接卷积（滑窗法），输入从 256 增加至 4096
或者输入从 32 增加至 512.

输入：Input 和 Kernel(3x3)

问题描述：用直接卷积的方式对 Input 进行卷积，这里只需要实现 2D, height*width, 通道 channel(depth)设置为 3, Kernel (Filter)大小设置为 3*3, 步幅(stride)分别设置为 1, 2, 3, 可能需要通过填充(padding)配合步幅(stride)完成 CNN 操作。注：实验的卷积操作不需要考虑 bias(b), bias 设置为 0.

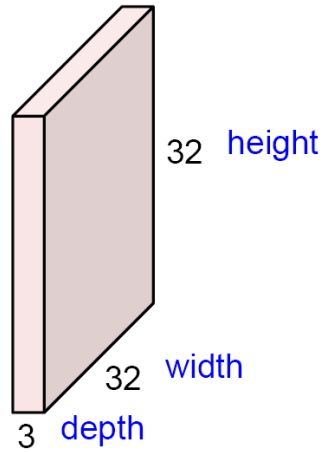
输出：输出卷积结果以及计算时间

以下是部分 CNN 操作的解释 ppt，具体参考附件中的人工智能课件。

1) CNN Input Image 举例

Convolution Layer

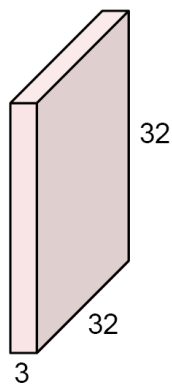
32x32x3 image (Input)



2) CNN Input Image 和 Kernel(Filter)

Convolution Layer

32x32x3 image



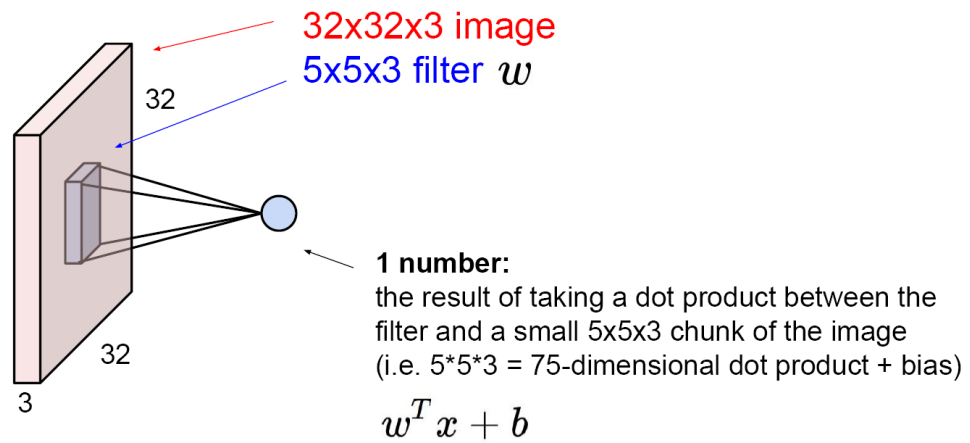
Filters always extend the full depth of the input volume

5x5x3 filter



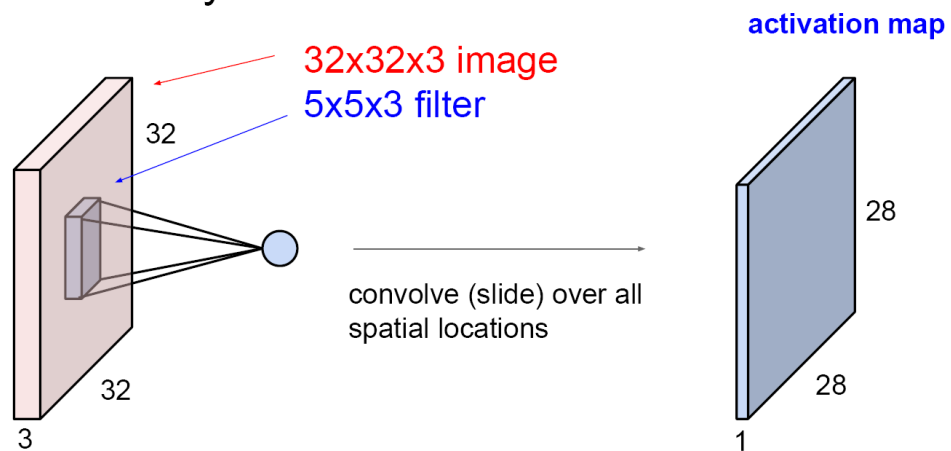
Convolve the filter with the image
i.e. "slide over the image spatially,
computing dot products"

Convolution Layer

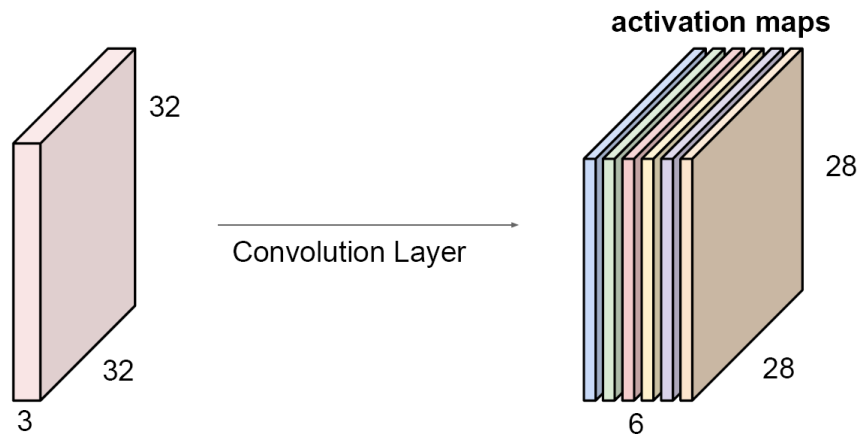


3)CNN 操作过程

Convolution Layer



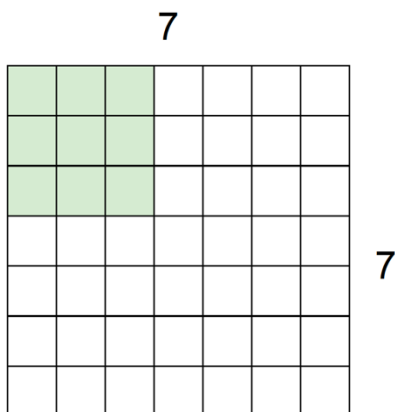
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a “new image” of size 28x28x6!

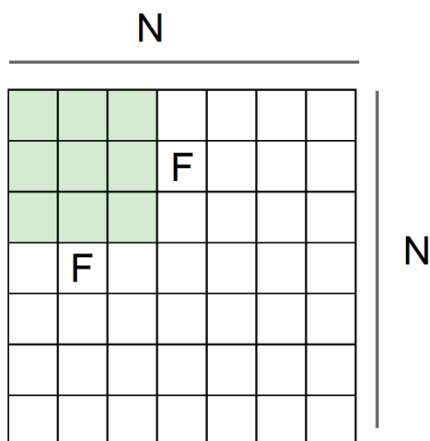
4) CNN 步幅 stride 和填充 padding

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.



Output size:
 $(N - F) / \text{stride} + 1$

e.g. $N = 7, F = 3$:
stride 1 $\Rightarrow (7 - 3) / 1 + 1 = 5$
stride 2 $\Rightarrow (7 - 3) / 2 + 1 = 3$
stride 3 $\Rightarrow (7 - 3) / 3 + 1 = 2.33$

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with **stride 1**

pad with 1 pixel border => what is the output?

7x7 output!

in general, common to see CONV layers with stride 1, filters of size $F \times F$, and zero-padding with $(F-1)/2$. (will preserve size spatially)

e.g. $F = 3 \Rightarrow$ zero pad with 1

$F = 5 \Rightarrow$ zero pad with 2

$F = 7 \Rightarrow$ zero pad with 3

任务 4:

使用 `im2col` 方法结合任务 1 实现的 GEMM（通用矩阵乘法）实现卷积操作。输入从 256 增加至 4096 或者输入从 32 增加至 512，具体实现的过程可以参考下面的图片和参考资料。

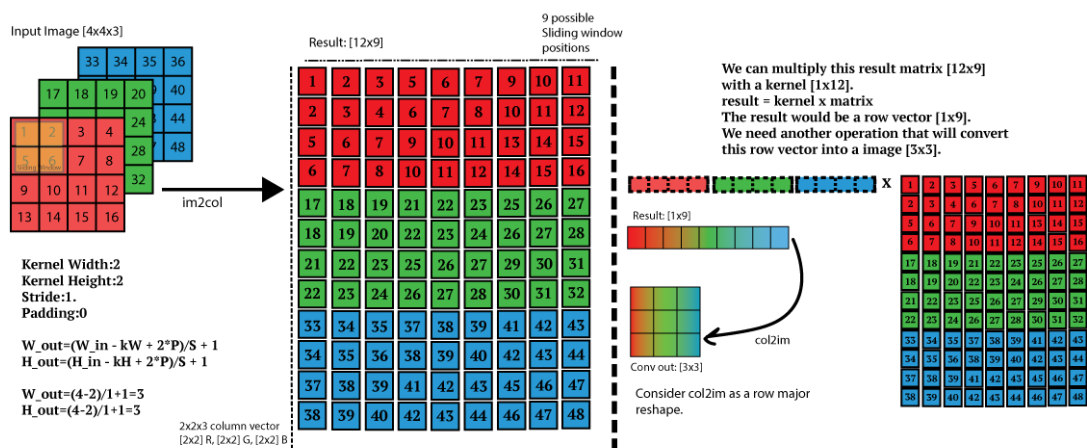
输入：Input 和 Kernel (Filter)

问题描述：用 `im2col` 的方式对 Input 进行卷积，这里只需要实现 2D, $\text{height} \times \text{width}$ ，通道 `channel(depth)` 设置为 3，Kernel (Filter) 大小设置为 3×3 。注：实验的卷积操作不需要考虑 `bias(b)`，`bias` 设置为 0，步幅(stride)分别设置为 1, 2, 3。

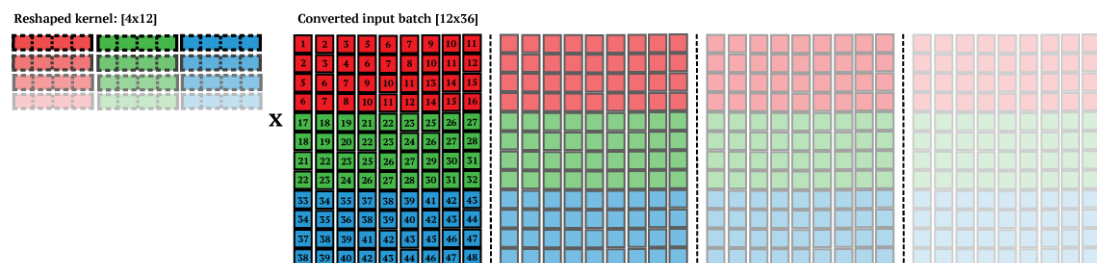
输出：卷积结果和时间。

Image to column operation (im2col)

Slide the input image like a convolution but each patch become a column vector.



We get true performance gain when the kernel has a large number of filters, ie: F=4 and/or you have a batch of images (N=4). Example for the input batch [4x4x3x4], convolved with 4 filters [2x2x3x2]. The only problem with this approach is the amount of memory



任务 5:

NVIDIA cuDNN 是用于深度神经网络的 GPU 加速库。它强调性能、易用性和低内存开销。

使用 cuDNN 提供的卷积方法进行卷积操作，记录其相应 Input 的卷积时间，与自己实现的卷积操作进行比较。如果性能不如 cuDNN，用文字描述可能的改进方法。

CNN 参考资料，见实验发布网站

斯坦福人工智能课件 Convolutional Neural Networks, by Fei-Fei Li & Andrej Karpathy & Justin Johnson

其他参考资料（搜索以下关键词）

[1] 如何理解卷积神经网络（CNN）中的卷积和池化

[2] Convolutional Neural Networks (CNNs / ConvNets)

<https://cs231n.github.io/convolutional-networks/>

[3] im2col 的原理和实现

[4] cuDNN 安装教程

[5] convolutional-neural-networks

<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>