

用于加速 CNN 定点数仿真的 MATLAB 函数库

姓名： 赵明心

当前版本： Release V1.0

日期： 2018 年 12 月 3 日

Copyright © 2018 Zhao Mingxin

E-mail : zhaomingxin17@semi.ac.cn

Homepage: <https://jackgittes.github.io>

目 录

第 1 部分 使用方法	3
§ 1.1 函数库组成	3
§ 1.1.1 加速的主要原理	3
§ 1.1.2 加速效果	4
§ 1.2 参数准备	4
§ 1.2.1 图像与 Feature Map	5
§ 1.2.2 权重参数	5
§ 1.2.3 常用函数格式	6
§ 1.3 快速搭建模型	6
§ 1.4 获取仿真结果	7
§ 1.5 完整流程	7
第 2 部分 细节与常见问题	9
§ 2.1 函数细节	9

第 1 部分

使用方法

MATLAB 自带的定点数对象 (fi) 运算效率相比 double、float 等数值类型要差很多，同时 fi 对象默认无法支持多核并行和 GPU 加速。在仿真规模较大的神经网络时，仿真时间往往很长，降低了验证定点量化方案正确性的效率。

为解决这个问题，同时保持跟现有的主流深度学习框架格式上兼容，设计了这个库函数。本说明假定读者具有基本的深度学习知识 (了解一般卷积神经网络的组成结构)。

第一部分将介绍以下几个主要内容：

- 函数库组成
- 参数准备
- 快速搭建模型
- 仿真并获取结果

§ 1.1 函数库组成

CNN 常用的操作在 nn 目录中，目前可以支持 MobileNet V1 版本的所有操作。

操作名称	说明	备注
Conv2d	用来实现二维卷积	
PointwiseConv2d	用来实现卷积核大小为 1 的卷积	通过调用 Conv2d 调用
DepthwiseConv2d	用来实现深度可分离卷积	
Pooling	用来实现 Pooling 操作	支持'MAX'、'AVG'
ReLU	激活函数，目前只支持 ReLU	

§ 1.1.1 加速的主要原理

库函数的设计参考了目前主流的深度学习框架 Caffe、Tensorflow 加速卷积操作的方法，先把卷积操作通过 im2col 映射为卷积核向量与图像矩阵的乘积，然后将矩阵进行分块处理，并把分块后的矩阵发送到 CPU 的不同计算核心上计算，最后将计算结果回收并重新整理成需要的形状。

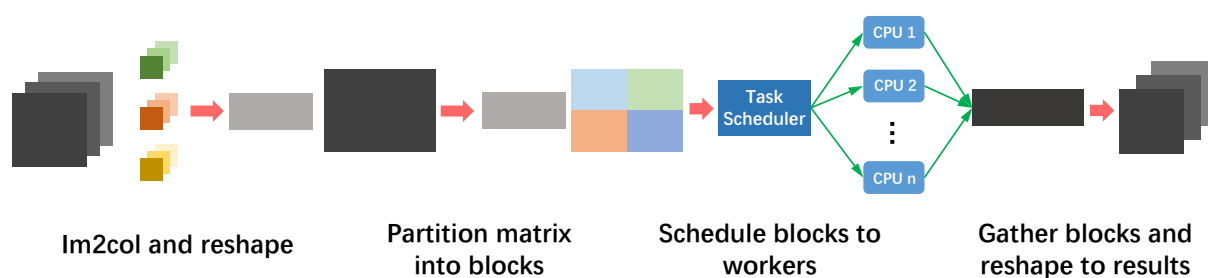


图 1.1-1 函数库的加速原理 (CPU)

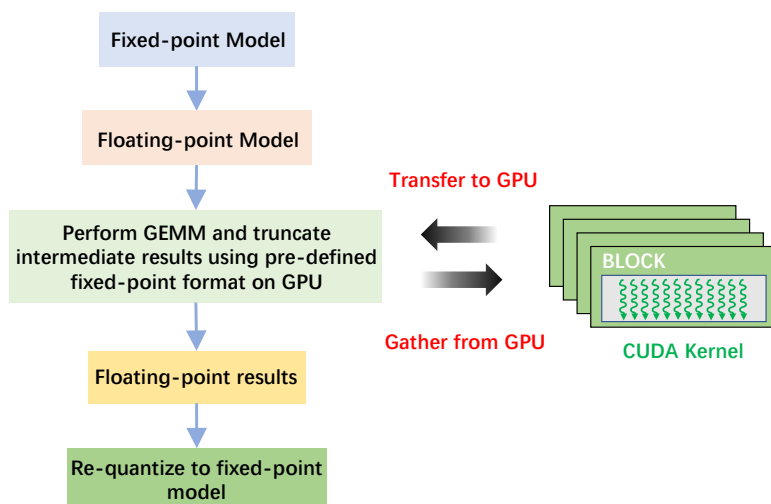


图 1.1-2 函数库的加速原理 (GPU)

通过把卷积转换为矩阵乘法避免大量循环操作，同时借助多核并行或 GPU 加速提高计算效率，从而提升 CNN 定点数仿真的速度，帮助快速验证定点化方案的正确性。

§ 1.1.2 加速效果

测试使用了 MobileNet V1 为目标神经网络，CPU 实现分别在四核 i5-7600HQ CPU+16.0GB RAM 的笔记本、六核 i5-8400 CPU+16.0GB RAM 的台式机、12 核 Xeon Centos 服务器和 32 核 Xeon Ubuntu 服务器上进行。GPU 实现在六核 i5-8400HQ CPU+16.0GB RAM+GTX 1050Ti 的笔记本进行。

MobileNet 实现方法	循环实现卷积	矩阵乘法	矩阵乘法 + 多核	矩阵乘法 + CUDA
耗时	1~2 hour(s)	3~6 min	50~90 sec	<10 sec

表 1.1-1 不同实现方法和运行时间

目前基于 CUDA 核的加速模块还在编写当中，表格中的测试结果只是用初步编写的 CUDA 函数测得的。

§ 1.2 参数准备

输入网络的图像、Feature Map 和网络权重参数应按照 MATLAB 多维数组的形式进行排列，具体如下。

§ 1.2.1 图像与 Feature Map

输入图像与 Feature Map 应该按照下图1.2-3的格式。但存在一些特例，例如 MATLAB 中，当数组的最后一个维度为 1(也即输入深度为 1) 的时候，MATLAB 会自动省去最后一个维度而变成 [Height, Width] 格式。遇到这样的情况也没有关系，函数会判断输入图像深度是否为 1 自行处理。

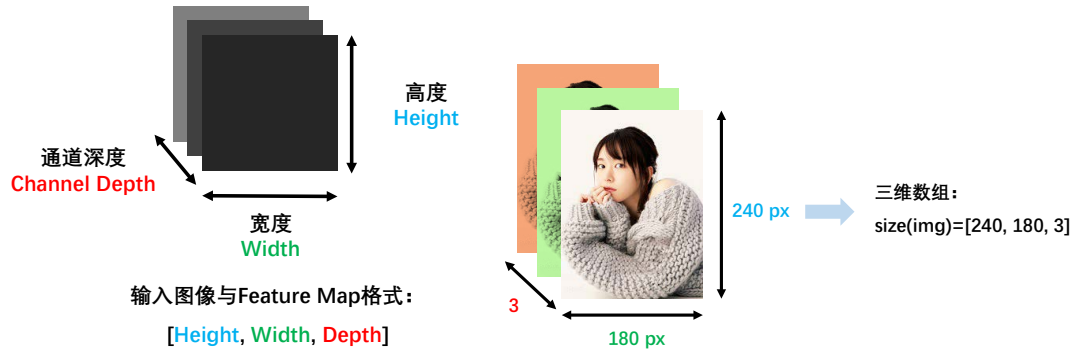


图 1.2-3 输入格式与示例

§ 1.2.2 权重参数

和输入图像的格式类似，参数格式也是一个多维数组，库函数使用了 TensorFlow 默认的 NHWC 格式，即第一个维度表示输入图像深度，中间两个维度为卷积核形状，最后一个维度为输出通道数。应时刻确保输入图像深度 Depth 和权重数组的 N 相等。

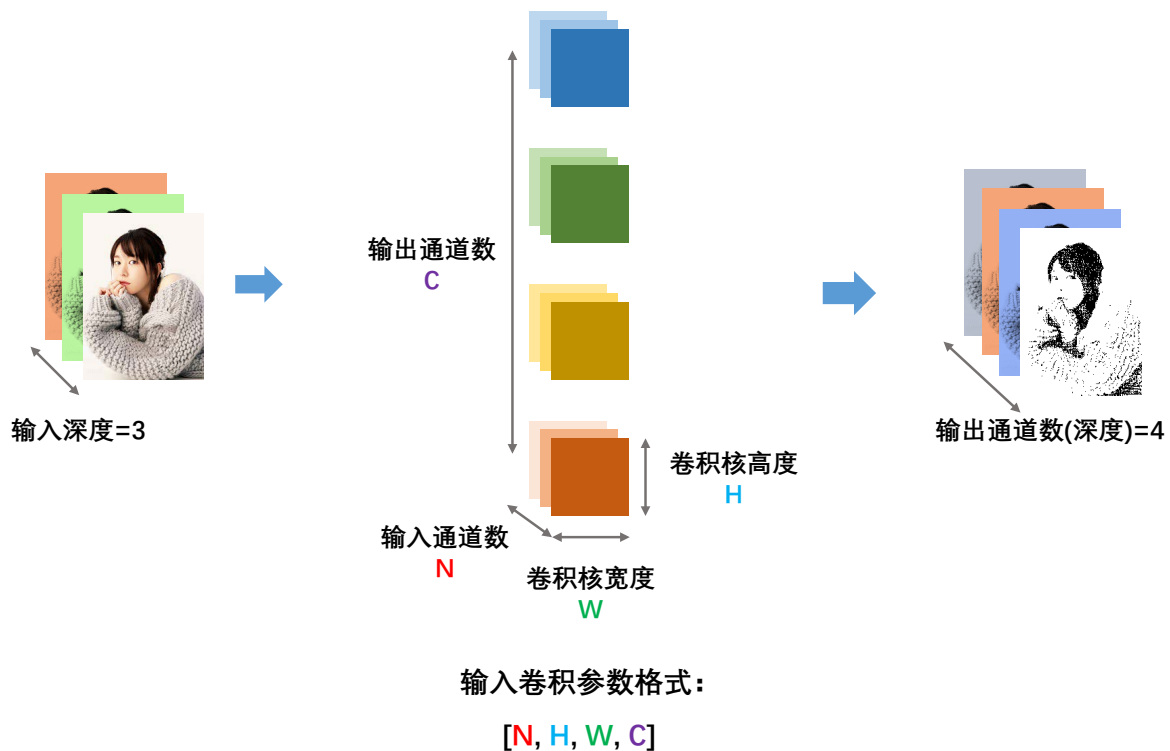


图 1.2-4 输入参数格式

§ 1.2.3 常用函数格式

- `nn.Conv2d(img, kernel, numeric_type, fimath, stride, padding_type)`

参数名称	说明
<code>img</code>	输入图像或 Feature Map
<code>kernel</code>	卷积核
<code>numeric_type</code>	定点数格式 (位宽、小数位宽度)
<code>fimath</code>	定点数运算格式
<code>stride</code>	步长
<code>padding_type</code>	'SAME' 或 'VALID', 注意大写

和 TensorFlow 兼容, 支持不等宽卷积核、不等宽步长、任意输入大小和深度。

- `nn.DepthwiseConv2d(img, kernel, numeric_type, fimath, stride, padding_type)`
同 `nn.Conv2d`。
- `nn.AddBias(img, bias, numeric_type, fimath)`

参数名称	说明
<code>img</code>	输入图像或 Feature Map
<code>bias</code>	偏置
<code>numeric_type</code>	定点数格式 (位宽、小数位宽度)
<code>fimath</code>	定点数运算格式

- `nn.Pooling(img, numeric_type, fimath, window_shape, pool_type, stride, padding_type)`

参数名称	说明
<code>img</code>	输入图像或 Feature Map
<code>numeric_type</code>	定点数格式 (位宽、小数位宽度)
<code>fimath</code>	定点数运算格式
<code>window_shape</code>	Pooling 窗口大小
<code>pool_type</code>	'MAX'、'AVG' 或者自定义
<code>stride</code>	步长
<code>padding_type</code>	'SAME' 或 'VALID', 注意大写

和 TensorFlow 兼容, 支持不等宽窗口、步长、任意输入大小和深度。

- `nn.ReLU(img)`
省略。

§ 1.3 快速搭建模型

库函数仿照 TensorFlow 函数的格式, 下面从 MobileNet V1 的 28 层中选取前两层作为例子:

```

1 % MobileNet 例子
2
3 % 定义定点数计算格式, t为定点数数据格式, f为计算格式
4 wordlen =16;
5 fraclen =8;
6
7
8 t = numerictype('WordLength', wordlen, 'FractionLength',fraclen);
9 roundm = 'floor';
10 f = fimath('CastBeforeSum',0, 'OverflowMode', 'Saturate', 'RoundMode', 'floor', ...
11 'ProductMode', 'SpecifyPrecision', 'SumMode', 'SpecifyPrecision', 'ProductWordLength',2*
wordlen, ...
12 'ProductFractionLength',2*fraclen, 'SumWordLength', wordlen, 'SumFractionLength', fraclen);
13
14 nn.TurnOnMultiCore(); % Turn on multicore support
15
16 % MobileNet 网络定义
17 % 第一层
18 conv1 = nn.Conv2d(input_img,kernel_1,t,f,[2,2],'SAME');
19 conv_bias_1 = nn.AddBias(conv1,bias_1,t,f);
20 layer_1 = nn.ReLU(conv_bias_1);
21
22 % 第二层
23 conv2 = nn.DepthwiseConv2d(layer_1,kernel_2,t,f,[1,1],'SAME');
24 conv_bias_2 = nn.AddBias(conv2,bias_2,t,f);
25 layer_2 = nn.ReLU(conv_bias_2);

```

图 1.3-5 MobileNet V1 代码示例

其中 wordlen 和 fraclen 分别定义了定点数的位宽和小数位占据的位宽, t 和 f 分别为定点数的数据格式和计算格式, 这和 MATLAB 的 fi 定点数的初始化方式是一样的。当然在网络的层与层之间也可以动态改变数据格式。

nn.TurnOnMultiCore() 为开启 MATLAB 的并行计算工具箱 (PCT) 来使用多核加速。所有函数在不使用多核加速的情况下仍然可以运行, 但是仿真时间会增加 3 到 5 倍。

中间结果都是 MATLAB 的定点数对象 fi 格式, 所以可以直接从结果中获得对应的二进制表示, 用来和硬件仿真结果进行对比。

§ 1.4 获取仿真结果

利用之前小节搭建好的网络模型, 在 MATLAB 中运行脚本, 即可得到网络的输出结果, 如果想要保留中间结果进行验证, 只需要给想要保留的结果取特定的变量名 (这样可以保证不会被其他后来生成的新变量覆盖掉结果) 就可以。

得到的网络输出结果和中间变量都是 fi 格式的, 可以直接获得对应的二进制、十六进制以及实数值。

§ 1.5 完整流程



图 1.5-6 完整的 CNN 定点数 MATLAB 仿真流程示意

第 2 部分

细节与常见问题

这部分主要是库函数的细节和一些常见问题。

§ 2.1 函数细节

To be completed ...