FAST: Fixed-point Arithmetic Simulation Toolbox for CNN

使用说明

作者：赵明心

当前版本： Pre-release **v 2.1.2**

No guaranty for stability and correctness.

2019 年 1 月 11 日

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 5
§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
§2.1.1 MATLAB 数据的存储与计算特点 9
§2.1.2 CPU 加速 9
§2.1.3 多核加速的问题 9
§2.1.4 GPU 加速 11
第 3 部分 MobileNet 定点数仿真与验证 12
第 4 部分 OOP 特性 14

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，设计了这个函数库。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 的所有操作。

操作名称	说明	备注
Conv2d	二维卷积	
PointwiseConv2d	卷积核大小为 1 的卷积	通过调用 Conv2d 调用
DepthwiseConv2d	深度可分离卷积	
Pooling	用来实现 Pooling 操作	'MAX'、'AVG'
ReLU	激活函数，ReLU	
AddBias	加偏置	

§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。

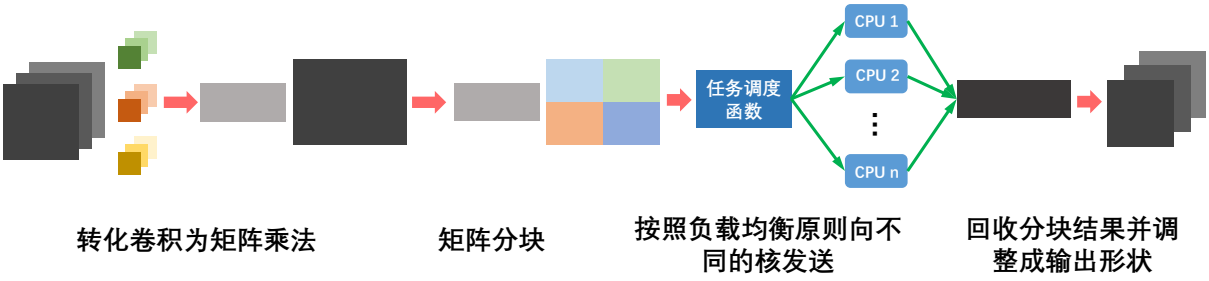

图 1.1-1 函数库的加速原理 (CPU)

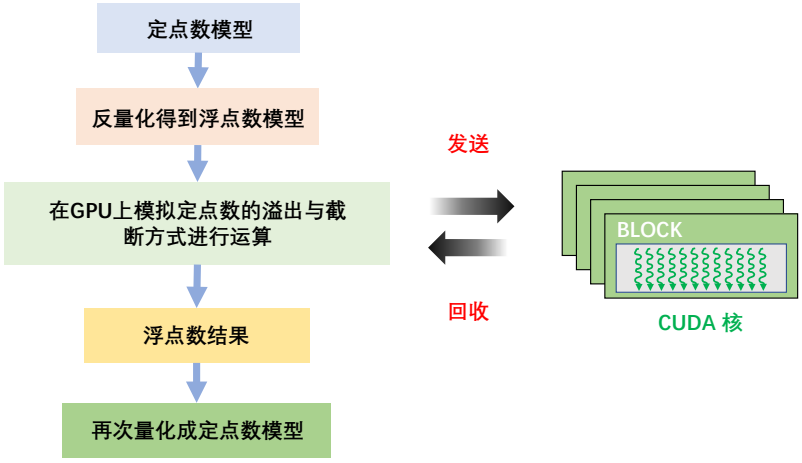

图 1.1-2 函数库的加速原理 (GPU)

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。

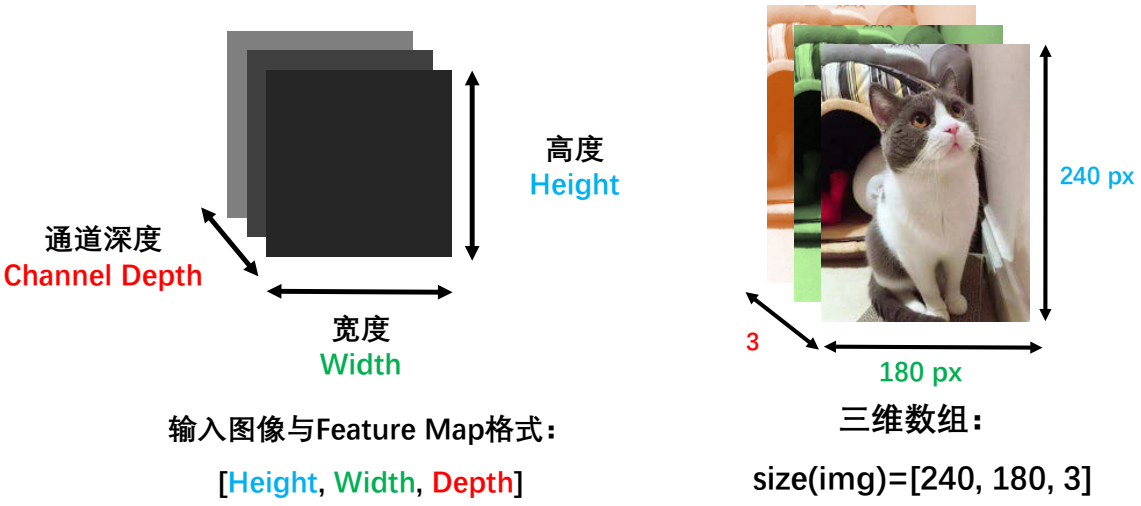

图 1.2-3 输入格式与示例

§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。

§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
img	输入图像或 Feature Map
kernel	卷积核
numeric_type	定点数格式 (位宽、小数位宽度)
fimath	定点数运算格式
stride	步长
padding_type	'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)
- nn.Pooling(img, numeric_type, fimath, window_shape, pool_type, stride, padding_type)

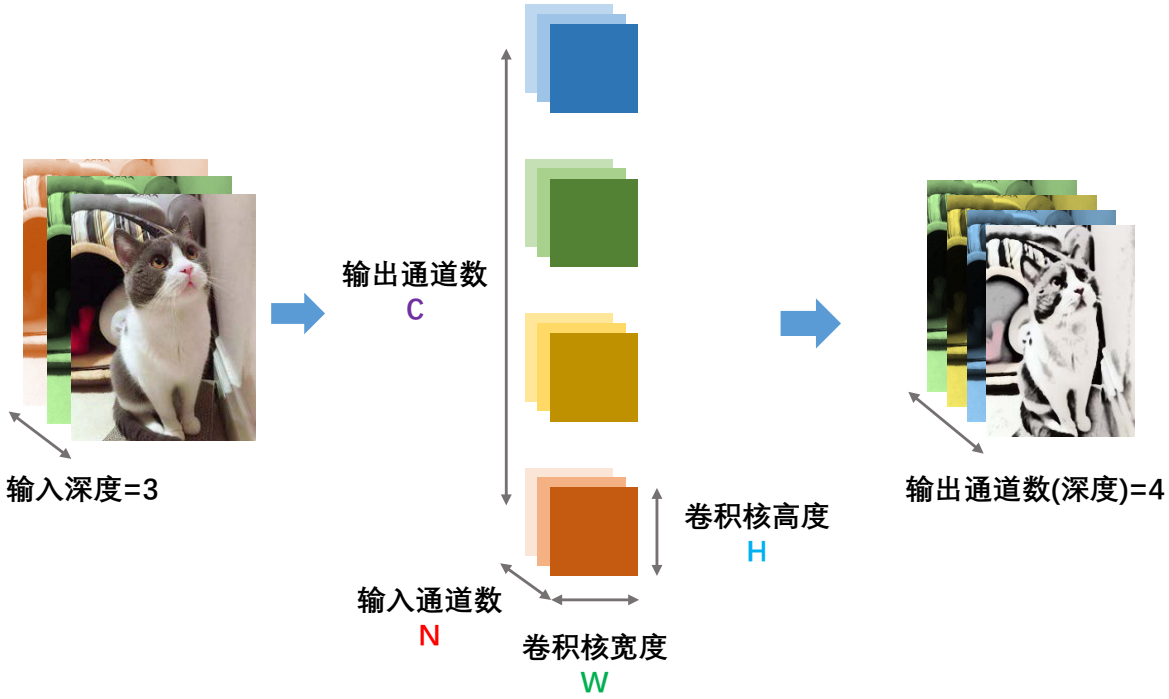

图 1.2-4 输入参数格式

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

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：

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。

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

```
% MobileNet 例子

% 定义定点数计算格式，t为定点数数据格式，f为计算格式
wordlen =16;
fraclen =8;


t = numerictype('WordLength', wordlen, 'FractionLength',fraclen);
roundm = 'floor';
f = fimath('CastBeforeSum',0, 'OverflowMode', 'Saturate', 'RoundMode', 'floor', ...
'ProductMode', 'SpecifyPrecision', 'SumMode', 'SpecifyPrecision', 'ProductWordLength',2*
wordlen, ...
'ProductFractionLength',2*fraclen, 'SumWordLength', wordlen, 'SumFractionLength', fraclen);

nn.TurnOnMultiCore();   % Turn on multicore support

% MobileNet 网络定义
% 第一层
conv1 = nn.Conv2d(input_img,kernel_1,t,f,[2,2],'SAME');
conv_bias_1 = nn.AddBias(conv1,bias_1,t,f);
layer_1 = nn.ReLU(conv_bias_1);

% 第二层
conv2 = nn.DepthwiseConv2d(layer_1,kernel_2,t,f,[1,1],'SAME');
conv_bias_2 = nn.AddBias(conv2,bias_2,t,f);
layer_2 = nn.ReLU(conv_bias_2);
```

图 1.3-5 MobileNet V1 代码示例

§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

单核、多核、GPU 加速都遵循了 im2col-GEMM-reshape 的步骤。GEMM 是 GEneral Matrix-to-matrix Multiplication(即通用矩阵-矩阵乘法) 的缩写。目前 GEMM 已经是非常成熟的领域，不同平台都有针对 GEMM 进行加速的库，例如基于 Intel CPU 的 MKL，基于 NVIDIA GPU 的 cublas 等等。

但这些库都高度封装，自带溢出处理机制，可定制空间很小。为了更好地模拟定点数的行为，需要设计一个可以实现定点数 GEMM 的程序。

§2.1.1 MATLAB 数据的存储与计算特点

MATLAB 中的多维数组都是以列主序存储的，所谓列主序，即按**列**的顺序存储数据。例如一个维度为 $[10, 5, 4]$ 的三维数组，MATLAB 先遍历第一列的 10 个元素，再遍历第二列的 10 个元素... 直至遍历完 5 列，再开始遍历第三个维度。实际就是哪个维度在前，就先从哪个维度开始存。

MATLAB 的运算顺序也是列主序的，所以在计算向量矩阵乘法 $C = xA$ 的时候，行向量 x 会按照 A 的列元素排列顺序依次与之相乘。这样就保证了乘累加的运算过程一直是有序的，只要硬件实现的时候按照 MATLAB 的遍历顺序进行计算，MATLAB 仿真结果和实际硬件实现的定点数运算就是等价的。

§2.1.2 CPU 加速

§2.1.3 多核加速的问题

矩阵分块乘法的数值特性

定点数的运算因为溢出和截位的存在而不满足加法结合律与交换律，所以在分块过程中保持运算顺序是很重要的。

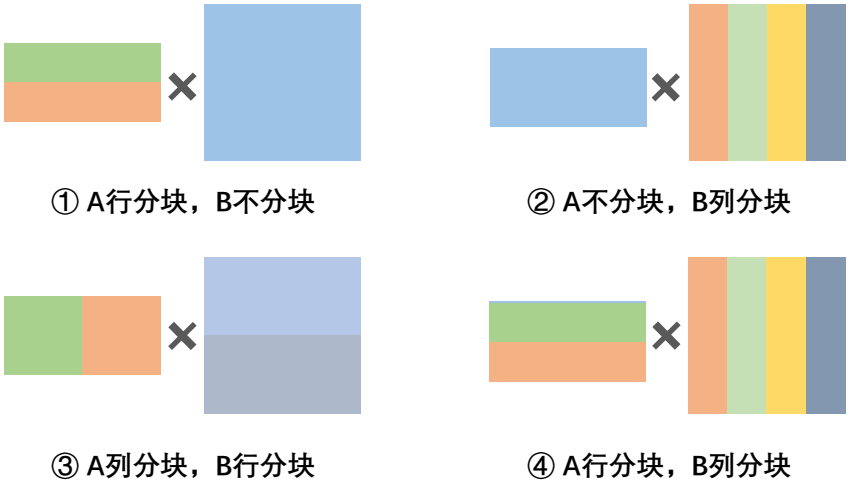

图 2.1-1 不同的矩阵分块方式

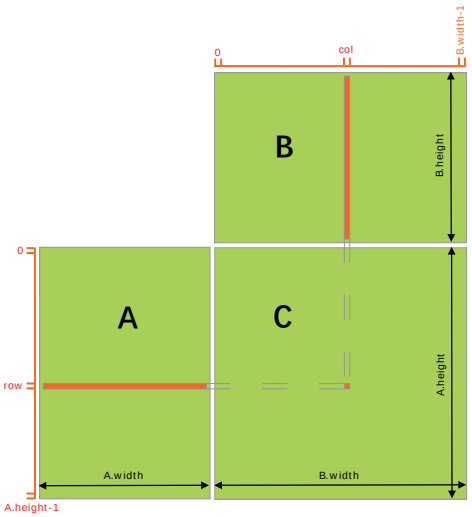

图 2.1-2 每个线程计算 A 的一行与 B 的一列的乘加

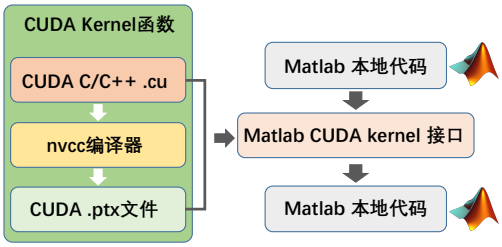

图 2.1-3 MATLAB 调用 CUDA 过程

§2.1.4 GPU 加速

Algorithm 1: GPU 定点加速伪代码

Require: 定点数矩阵 A、B
Require: 量化格式：总位宽 L_w 与小数位宽 L_f
计算 :
$A_f = A \times 2^{L_f}$
$B_f = B \times 2^{L_f}$
将 A、B 传送至 GPU
计算 : 对每个线程:$c_{ij} = 0, k = 1$
while $k < A\_width$ **do**
  if $Cvalue = a_{ik}b_{kj} > 2^{L_w} - 1$ **then**
    $Cvalue = 2^{L_w} - 1$
    $c_{ij} = c_{ij} + Cvalue$
  end
  else if $Cvalue = a_{ik}b_{kj} < -2^{L_w}$ **then**
    $Cvalue = -2^{L_w}$
    $c_{ij} = c_{ij} + Cvalue$
  end
  $k = k + 1$
end
回收计算完成的浮点格式结果矩阵 C_f
$C = \mathrm{fi}(C_f/2^{2L_f}, t, f)$

第 3 部分

MobileNet 定点数仿真与验证

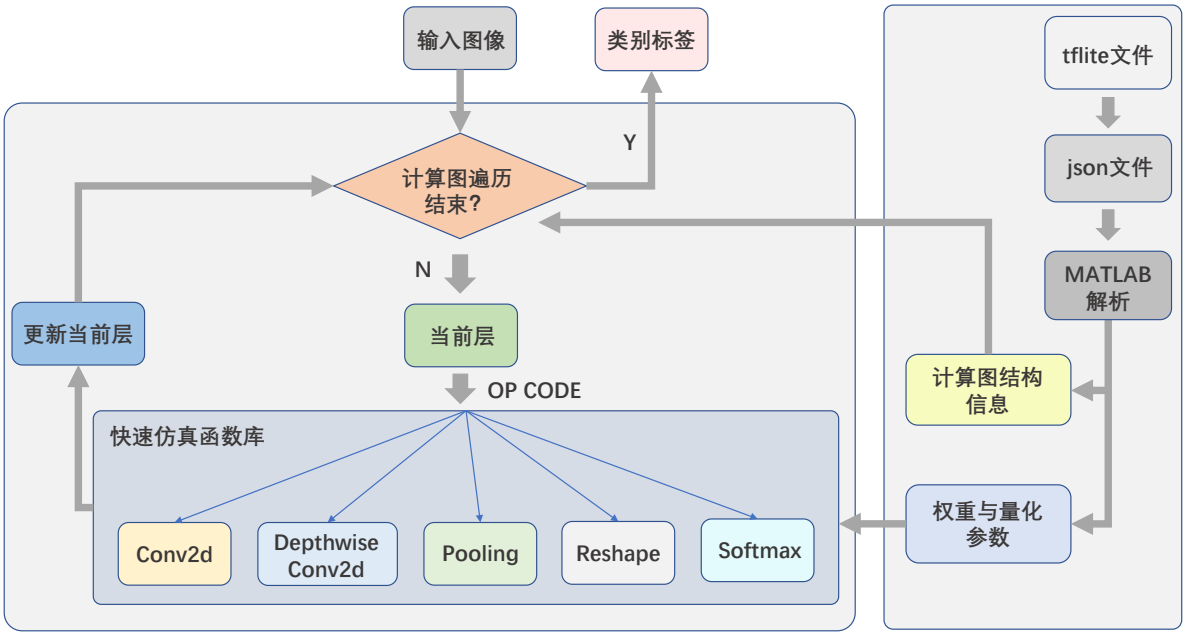

图 3.0-1 MobileNet 8-bit 量化验证

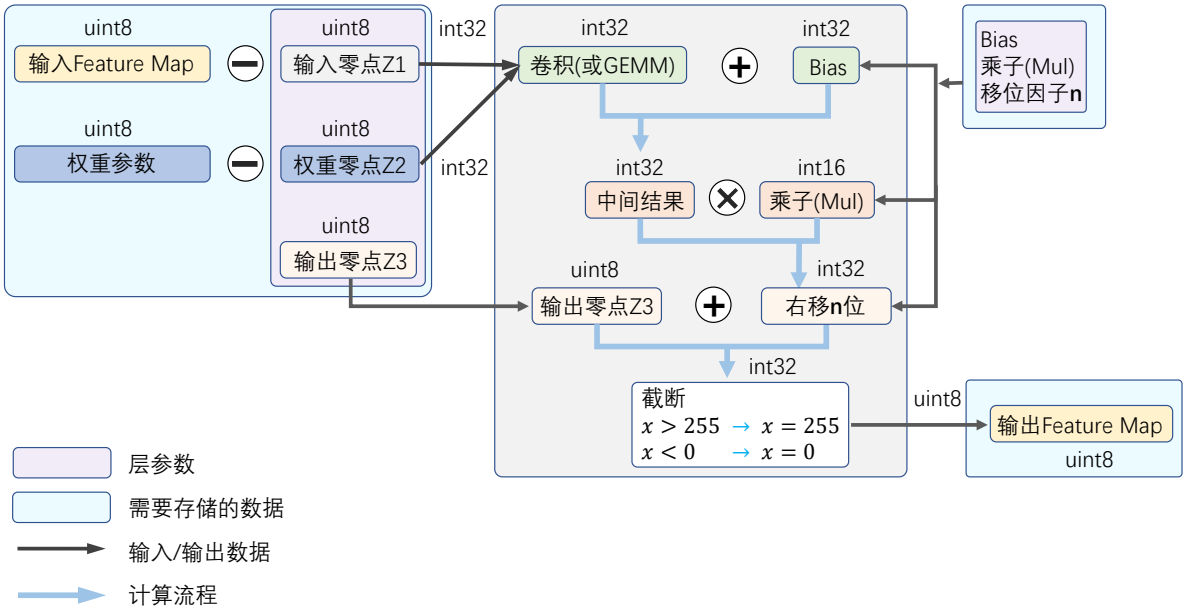

图 3.0-2 MobileNet 8-bit 量化验证

第 4 部分

OOP 特性

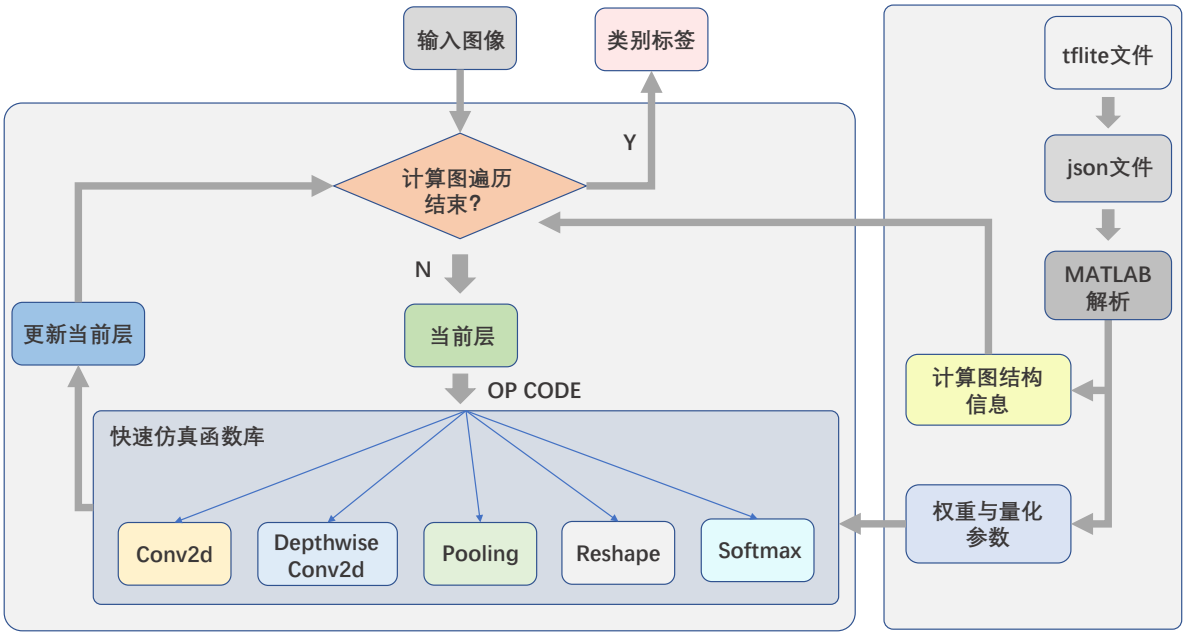

图 4.0-1 MobileNet 8-bit 量化验证

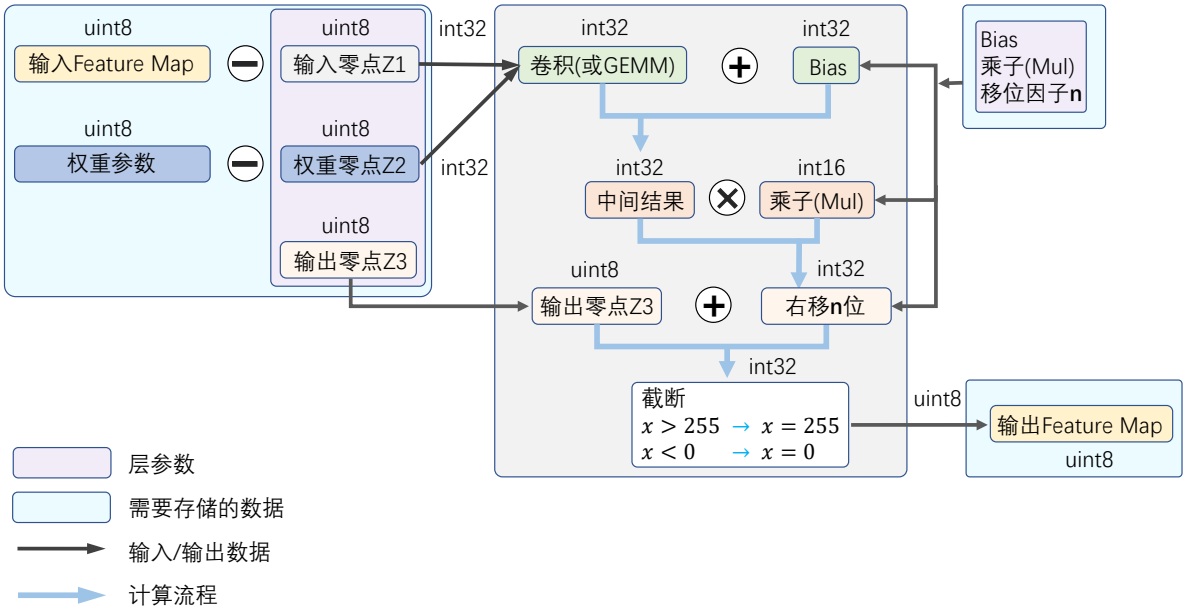

图 4.0-2 MobileNet 8-bit 量化验证