# 深度学习模型量化调研

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **完成时间** | **完成人** | **完成内容** |
| 1 | 2018.11.11 | Shen Jia | 完成第一版  第一章、第二章、第三章（3.1、3.2.1、3.2.2、3.2.4、3.2.7） |
| 2 | 2018.11.13 | Shen Jia | 补充第三章（3.2.3、3.2.5、3.2.6、3.2.7）和第四章 |

## 前言

按照我们公司的整体技术规划，下一个阶段要开展深度学习模型量化的工作。量化工作分几个阶段，最初是要实现float16数据类型量化，接着再实现int16、int8等类型量化。所以，我们最初的工作重点是如何实现float16数据类型量化。

因为，在现阶段我们缺少做深度学习模型量化的技术积累和验证平台，所以，我们要先来做量化方面的调研。

本文档为深度学习模型量化的调研文档，主要介绍量化基本内容、数据格式、关键技术点、一些量化方法以及与NVDLA相关的量化内容等，侧重数据类型为float16，考虑的量化过程是float32转float16。其它数据类型在后续的开发过程中再不断补充和完善。

本文档目的是，通过此次调研，我们能够熟悉量化原理、实现过程，了解已有的解决方案，再结合NVDLA的量化内容，使我们能够提出适合我们自己框架和硬件平台的、高效的量化方法。

本文档提到的量化都是指深度学习中模型的量化。研究推理模型的量化，暂时不考虑训练模型的量化。

## 量化基本介绍

### 量化概念

量化（Quantization）又称定点，深度学习模型量化是指用更少的数据位宽进行网络存储和计算。比如，32位的浮点数用16位的浮点数近似表达。

### 量化意义

深度学习网络应用在移动设备中逐渐成为一种趋势，但是目前移动设备一般只有比较低的计算能力，而且还会受限于内存和电量消耗。因此，采用一定方法使模型的尺寸更小、推理速度更快、耗电更低成为十分迫切的需求。

半精度浮点格式（float16）占用存储空间为单精度的一半，根据资料显示，在一些实验中显示使用半精度浮点格式对于训练神经网络几乎没有影响，因此，越来越多的硬件（Nvidia Tesla V100/P100，Nvidia TX2，ARMv8.2等等）开始原生支持使用 float16（又称 fp16、FP16）做计算。所以，研究模型的量化是十分有意义的。

### 1.3 量化研究基本思路

量化工作的本质是对数据的转换，内容包含数据格式和转换过程。

首先，我们要明确数据格式具体是什么样的，以及表达范围分别是多少。其次，这些数据格式之间是如何转换的，也就是量化的过程。我一方面调研相关论文和资料，另一方面看主流框架、硬件平台是如何实现量化过程的。这些主流公司或者是项目推出的量化方法还是十分耐人寻味的。再次，了解一下NVDLA有关量化是如何介绍的，因为，我们的Convolution计算是在DLA硬件做的，并且DLA是支持float16的。那么，我们就一定要遵循DLA关于量化的设计原理。最终，我们的工作量主要是如何高效地进行数据转换。

怎么理解高效呢？是模型准确率还是推理速度？我想，量化后的数据处理速度是受DLA影响控制的。而且，数据格式确定下来后，模型存储大小基本就确定下来了。我们做的是把一个数转换成多少的数值，所以，应该更关心的是转换后的准确率。大家都知道一般做量化是有损失的，而我们的目标是，保持模型精度损失很小的前提下，实现HNN推理框架的量化。

## 数据格式

IEEE二进制浮点数算术标准（IEEE 754）是20世纪80年代以来最广泛使用的浮点数运算标准，为许多CPU与浮点运算器所采用。这个标准定义了表示浮点数的格式（包括负零-0）与反常值（denormal number）），一些特殊数值（无穷（Inf）与非数值（NaN）），以及这些数值的“浮点数运算符”。 IEEE 754规定了浮点数值的方式：单精确度（32位）、双精确度（64位）、延伸单精确度（43比特以上，很少使用）与延伸双精确度（79比特以上，通常以80位实现）。

但是，也有的项目采用自己定义的数据格式，为了区分，该文档把IEEE 754定义的数据格式称为标准格式，自己定义的数据格式统称为非标准格式。

### 2.1 标准格式

IEEE 754标准格式如下：



**图 IEEE 754数据格式示意图**

一个浮点数由三部分组成：符号位S、指数部分E（阶码）以及尾数部分M。

符号位用1位表示，0表示正数，1表示负数；指数部分即使用所谓的偏正值形式表示(也就是移码)，实际值为表示值。与一个固定值(32位单精度的情况是127)的和。采用这种方式表示的目的是简化比较。因为，指数的值可能为正也可能为负，如果采用补码表示的话,全体符号位S和E自身的符号位将导致不能简单的进行大小比较。正因为如此，指数部分通常采用一个无符号的正数值存储。尾数部分采用原码表示，根据二进制的规格化方法,最高数字位总是1,该标准将这个1缺省存储,使得尾数表示范围比实际存储的多一位。

#### 2.1.1 单精度浮点Float32

单精度浮点数(float)总共用32位来表示浮点数，在IEEE 754-2008 标准中，32位base-2格式官方称为binary32，我们称之为float32，或fp32。

IEEE 754标准规定binary32具有：

|  |  |
| --- | --- |
| 位名称 | 位数 |
| 符号位 | 1位 |
| 指数宽度 | 8位 |
| 尾数位（精度） | 23位（24位） |

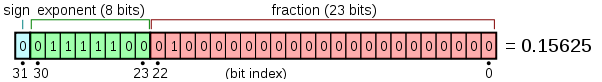
**表 IEEE 754标准binary32格式表**

符号位确定数字的符号，这也是有效数字的符号。

指数是从-128到127的8位有符号整数或从0到255的8位无符号整数，这是IEEE 754 binary32定义中可接受的偏置形式。

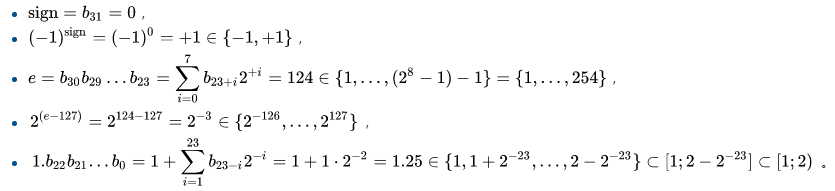
尾数有效数包括二进制点右侧的23个小数位和值为1的隐式前导位（位于二进制点的左侧）。因此，只有23个小数位的有效位出现在存储器格式中，但总精度是24位。

例如：



**图 IEEE 754标准binary32格式图**







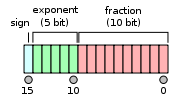
#### 2.1.2 半精度浮点Float16

在计算中，半精度是二进制浮点计算机数字格式，占用计算机存储器中的16位（现代计算机中的两个字节）。在IEEE 754-2008标准中，16位base-2格式称为binary16，我们称之为float16，或fp16。

IEEE 754标准将binary16指定为具有以下格式：

|  |  |
| --- | --- |
| 位名称 | 位数 |
| 符号位 | 1位 |
| 指数宽度 | 5位 |
| 尾数位（精度） | 10位（11位） |

**表 IEEE 754标准binary16格式表**



**图 IEEE 754标准binary16格式图**

float16格式与float32格式类似，符号位确定数字的符号，这也是有效数字的符号。指数是从-16到15的8位有符号整数或从0到31的8位无符号整数。只有10位有效数字出现在存储器格式中，但总精度为11位。计算方法与float32一样。

### 2.2 非标准格式

#### 2.2.1 半精度浮点Bfloat16

由于 Exponent 位宽不同，float16 与 float32 转换时需要做 re-bias 运算，开销较大。在 TensorFlow 中，除了标准 float16 之外还创造了一种新的数据类型 bfloat16，叫做截断浮点数，它是由一个float32截断前16位而成。同样只占用一半的存储空间，但与 float32 转换更方便，动态范围与float32相当。

|  |  |
| --- | --- |
| 位名称 | 位数 |
| 符号位 | 1位 |
| 指数宽度 | 8位 |
| 尾数位（精度） | 7位（8位） |

**表Tensorflow采用的bfloat16格式表**

它和IEEE 754定义的float16不同，主要用于取代float32来加速训练网络，同时降低梯度消失的风险，也可以防止出现NaN这样的异常值。深度学习网络每次梯度相乘的系数如果小于1，那就是浮点数，如果层数越来越多，那这个系数就会越来越大，传播到最底层可以学习到的参数就很小了，所以，需要截断来防止（或降低）梯度消失。  
 Amazon也证明Deep Speech模型使用bfloat16的训练和推理的效果都足够好。uint8在大部分情况下不能用于训练，只能用于推理，大多数的uint8的模型都从float32转换而来。所以，bfloat16可能是未来包括移动端的主流格式，尤其是需要语言相关的模型。

### 2.3 数据范围

|  |  |  |
| --- | --- | --- |
| 数据类型 | Dynamic Range | Min Positive Value |
| FP32 | -3.4 x 10 -38 ~ +3.4 x 10 -38 | 1.4 x 10 -45 |
| FP16 | -65504 ~ + 65504 | 5.96 x 10 -8 |
| INT8 | -128 ~ +127 | 1 |

**表 数据表示范围表**

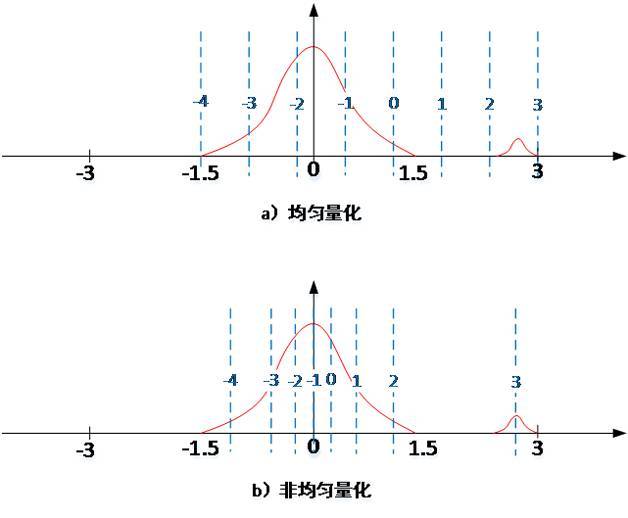
## 量化方法

### 3.1 基本介绍

模型量化目前主要有两个研究方向，一个研究方向是权值共享，基本思想是多个网络连接的权重共用一个权值，还有一个是权值精简，既权值的低比特表示。

对于权值精简，量化策略是多种多样的。出发的角度不同，解决的问题也不同。

从量化步长角度，量化方法主要有两种类型，均匀量化和非均匀量化。如果不论权值的疏密，直接对应，我们称之为均匀量化。如果权值密的量化后的范围也较密，权值稀疏的量化后的范围也较稀疏，我们称之为非均匀量化。如下图所示：



**图 均匀量化与非均匀量化图**

根据查看到的资料和论文，总结下面的量化方法分类。

从量化范围对称性角度，量化方法分为对称量化和非对称量化。

从量化与训练关系角度，量化方法分为训练时量化和完整训练后量化。

从量化与网络关系角度，量化方法分为静态量化有动态量化。

### 3.2 具体方法介绍

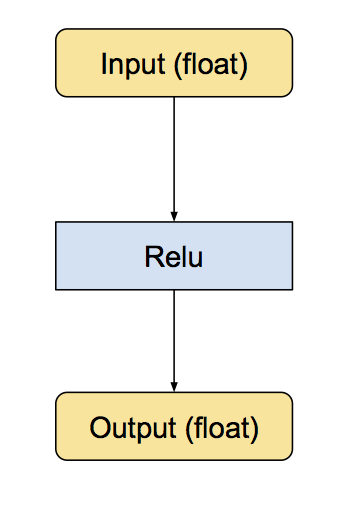
#### 3.2.1 Tensorflow

#### （1）简介

Tensorflow有一套量化感知训练方法。因为量化感知训练方法需要对模型进行重新训练，所以，我们暂时先不介绍这套方法。Tensorflow 也提供了一套支持用来推理的量化int8的工具，该工具不需要重新训练模型。我们这里面重点介绍该量化工具。

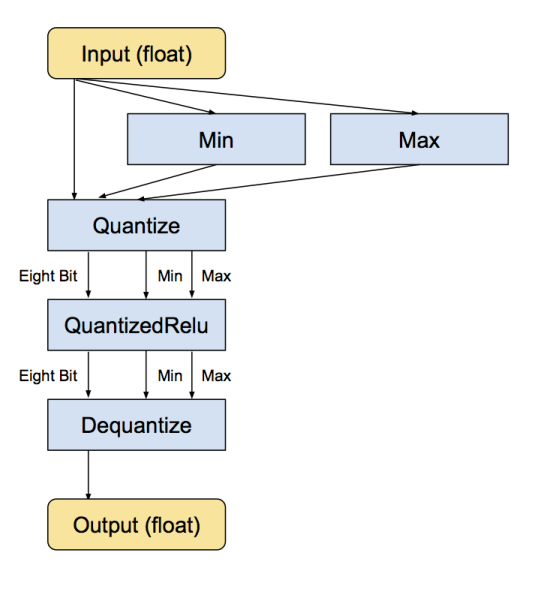
#### （2）原理

它对量化的实现是通过把常见操作转换为等价的八位版本达到的。涉及的操作包括卷积，矩阵乘法，激活函数，池化操作，以及拼接。转换脚本先把每个已知的操作替换为等价的量化版本。然后在操作的前后加上含有转换函数的子图，将input从浮点数转换成8 bit，再把output从8 bit转回浮点数。下面是 ReLu 的例子：



**图 tensorflow relu图**

经过转换后，如下图所示：



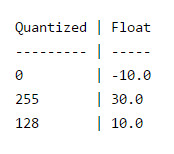
**图 tensorflow relu量化过程图**

经过转换后，输入输出依旧是float，只不过中间的计算是用过8 bit计算的。

量化后的张量是如何表示的呢？

用两个浮点表示，它存储对应于最小值和最大值的浮点数。量化阵列中的每个条目表示该范围内的浮点值，在最小值和最大值之间**线性分布**。

这里介绍下quantize和dequantize函数的逻辑。quantize取输入中的min和max，分别对应被量化的输入中的最小值（0）和最大值（255），把[min, max]这个区间**均匀**分成255个小区间，把输入中的值对应到对应的区间中。反量化操作则是把上述操作反向执行。例如input的最大值是30.0，最小值是-10.0，则量化后的值为：



**图 tensorflow量化值对应示意图**

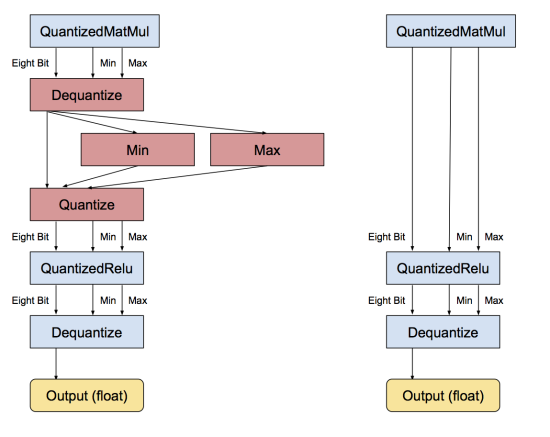
之所以这么做，Tensorflow的论述是：

1、权重、活化张量的数值通常分布在一个相对较小的范围中（weight：-15 ~ 15，activatios：-500 ~ 1000）；

2、神经网络对噪音的适应性强，将数量化到一个更小的数集中并不会对整体的结果带来很大的影响；

3、通过量化操作，可以有效提高点乘的计算效率。

有一个优化是当连续出现多个被量化了的操作时，没有必要在每个操作前做反序列化/序列化，因为上一个操作的反序列化和下一个操作的序列化是会被互相抵消的。例如下图：



**图 tensorflow连续被量化示意图**

如何决定范围？

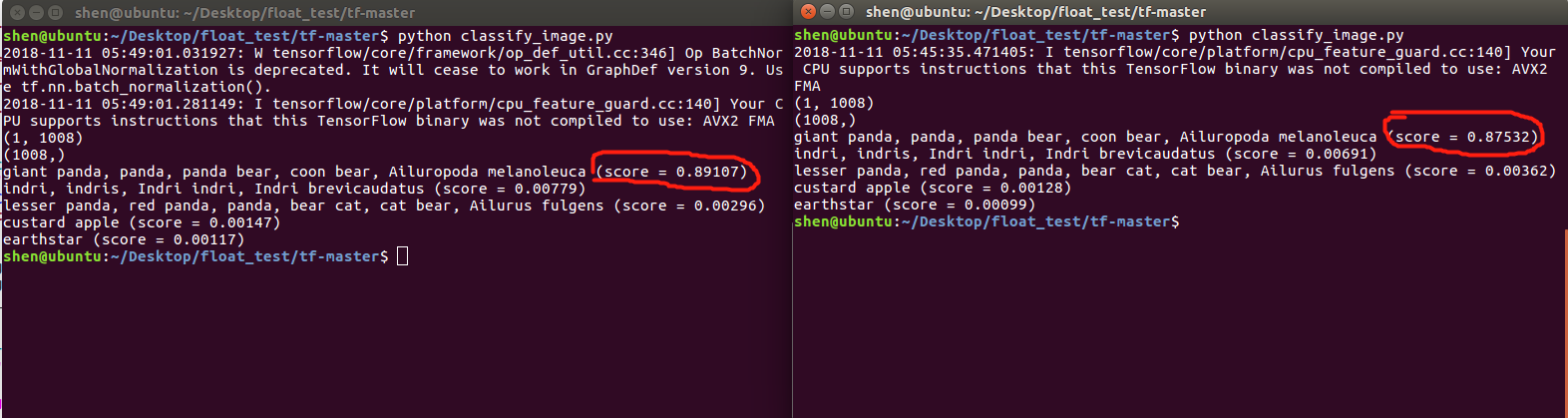
最小值和最大值经常可以被预先算出。权重参数在载入阶段就知道了，所以它们的范围可以作为常数保存。输入（比如图片）和激活函数也常常预先知道范围。这避免了对一个操作的输出进行分析来决定范围。对于conv和matmul来说还是得分析，因为8位的输入可能会生成32位的结果。如果对8位输入进行任何算术计算，会自然地累积得到超过8位精度能表达的数。比如，8位数相加结果需要9位，8位数相乘需要16位。如果8位数乘法的结果求和，结果会超过16位，通常需要至少20~25位。

这带来一个问题：我们需要把超过8位的输出缩减传递给一下个操作。对matmul来说，一个办法是根据可能的极端输入计算其输出的范围。Tensorflow有通过分析证明其正确性，是安全的。但实际上多数权重和激励值都分布得更均匀，意味着实际的范围更窄。所以，Tensorflow使用了QuantizeDownAndShrinkRange操作符来分析实际的范围。这个操作符实质上就是在求一个张量各元素的最大值和最小值。也可以通过大量的训练数据来获取参数范围。然后把这些参数范围编码进去，但是现在还没有这样做。

量化过程中，我们会遇到最难且最微妙的一个问题就是偏差的累计。虽然神经网络对噪声不敏感，但如果对舍入操作不小心，会导致偏差往某个方向累积，最终影响精度。**Tensorflow有提供出来公式**，重要的一点是，我们需要让舍入后的输入值减去摄入后的最小值，而不是让输入值减去最小值然后才做舍入。

#### （3）性能

用image class进行实际测试，从测试结果来看，量化前模型是97M，用int8量化后模型是24M，压缩了75%左右；量化前精度top1是89%左右，量化后精度top1是87%左右。实际速度没有测试，用虚拟机，速度测试并不准确。另外，从资料上看，该量化工具并没有介绍float16，从代码上看，也没有找到提供float16的代码，所以，可以认为，该工具主要是支持int8，没有支持float16。



速度测试方面的测试，官网上给出的描述是：“Post-training quantization is a general technique to reduce the model size while also providing up to **3x lower latency** with little degradation in model accuracy. Post-training quantization quantizes weights to 8-bits of precision from floating-point”。速度应该提升3倍。

#### （4）小结

针对Tensorflow提供的量化工具，总结一下：

1、做了权值量化，没有做权值共享。

2、针对完整训练后的、对称的、动态的、均匀量化。

3、不需要先做float32推理。

4、官网称支持float16、int8类型量化，开源代码里面只有int8量化代码。工具只支持int8量化。

#### 3.2.2 TensorRT

#### （1）简介

NVIDIA TensorRT™是一个高性能深度学习推理平台。 它包括深度学习推理优化器和运行时，可为深度学习推理应用程序提供低延迟和高吞吐量。在推理期间，基于TensorRT的应用程序比仅CPU平台的执行速度快40倍。使用TensorRT，开发者可以优化在所有主要框架中训练的神经网络模型，以高精度校准低精度，最后部署到超大规模数据中心，嵌入式或汽车产品平台。

TensorRT为深度学习推理应用的生产部署提供INT8和FP16优化，例如视频流，语音识别，推荐和自然语言处理。降低精度推断可显着减少应用程序延迟，这是许多实时服务，自动和嵌入式应用程序的要求。在部署阶段，latency是非常重要的点，而TensorRT是专门针对部署端进行优化的，目前TensorRT支持大部分主流的深度学习应用，当然最擅长的是CNN（卷积神经网络）领域。

#### （2）原理

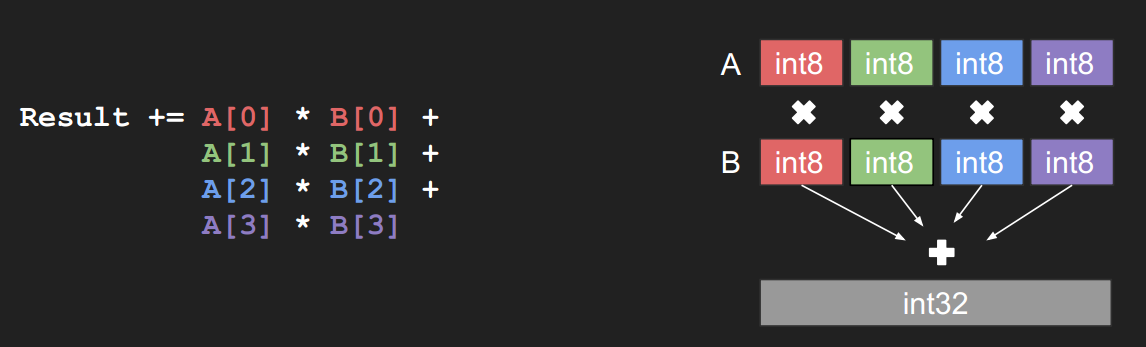
TensorRT本质上是个推理框架，而不是一个转换工具。输入的格式支持Tensorflow、Caffe、ONNX，经过TensorRT的量化等优化后，转成TensorRT网络格式，在TensorRT框架中进行推理。



**图 TensorRT在推理过程中的关系图**

从目前可以看到的资料，有介绍TensorRT量化int8推理过程，没有找到量化float16推理过程的。这里面简单介绍一下这种方法。

1、int8计算high-throughput



**图 TensorRT int8 high-throughput示意图**

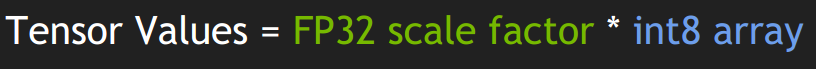
2、均匀量化方式

均匀量化公式：



**图 TensorRT tensor value示意图1**

通过实验得知，bias值去掉对精度的影响不是很大，因此我们直接去掉：

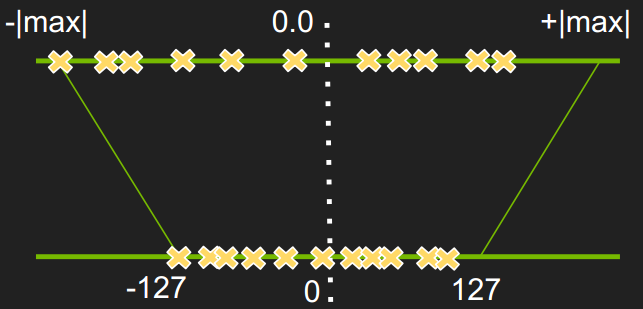


**图 TensorRT tensor value示意图2**

引出问题：如何设置缩放因子。

3、量化采用的是**对称量化**，考虑到有两类情况，

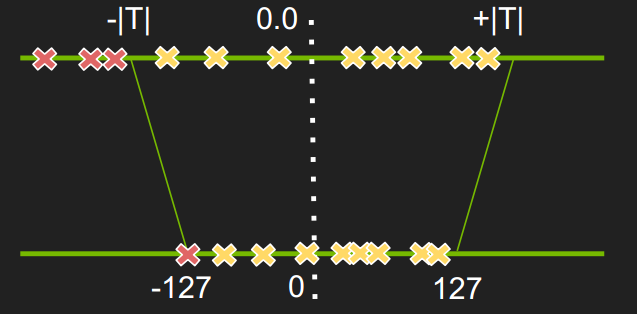
（1）map映射到127，呈饱和。相当于所有的float数都可以映射到-127~+127的区间里。



**图 TensorRT量化饱和图**

上面是简单的max-max 映射，这是针对均匀分布的，很明显的可以知道，只要数据分布的不是很均匀，那么精度损失是很大明显的。很多情况下是采用下面（2）的方法。

（2）阈值T映射到127，没有饱和。相当于有一部分float数没有映射到-127~+127的区间里。



**图 TensorRT量化不饱和图**

解释一下该图。

首先，要知道量化是可以保存源信息的，但是会有损失。如同高低分辨率图的区别，只要大体能识别图中的关键信息、细节可以忽略，低分辨率图也是可以接受的。

由于值正负分布不均匀，值的密度也不一样，如果按照（1）最大值映射，+max区间就有一块区域浪费了，也就是缩放到int8后，int8的动态范围就更小了。

4、如何选择最佳的阈值

如何选择最佳的阈值是int8表达范围与精度的权衡。从float32转到int8相当于重新编码相同的信息，会产生信息的损失，所以，这就变成了如何最小化信息损失的问题。信息损失可以用引入KL散度来处理。KL散度衡量相同事件空间里的两个概率分布的差异情况，所以，它可以用来衡量相关联的两个编码（float32和int8）之间信息损失的数量。

具体方案是利用Calibration Dataset。

（1）先在Calibration Dataset上进行float32的模型推理。

（2）对于每一层：

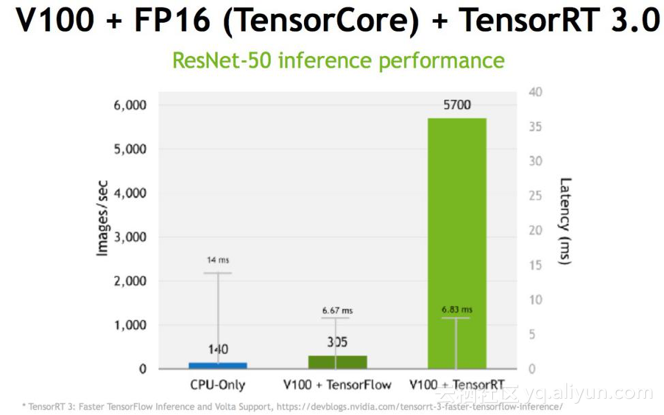
a）搜集激励函数的直方图。

b）根据不同的饱和阈值产生很多量化分布。

c）选择KL最小的阈值。

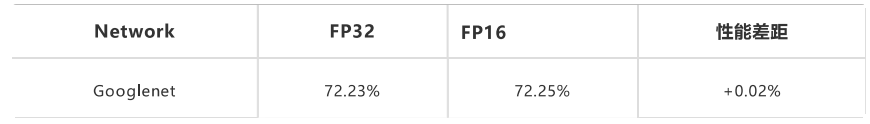
#### （3）性能

从网上找到的资料可以看出，TensorRT3.0版本加速效果还是很不错的。



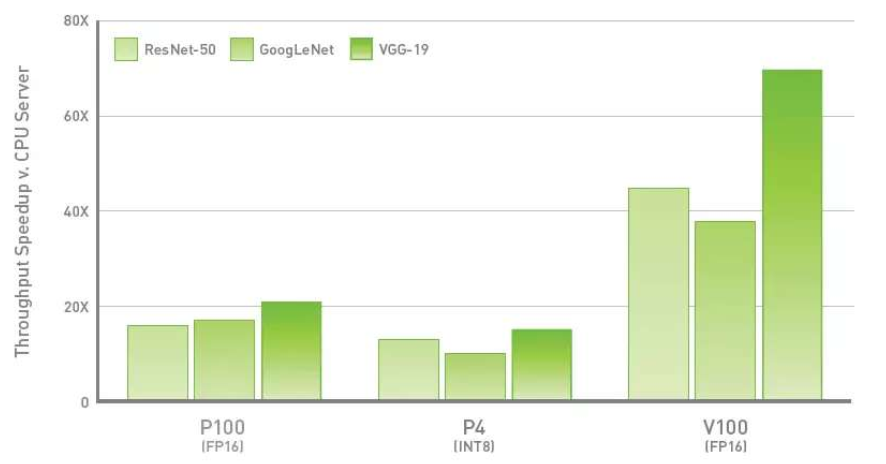
**图 TensorRT在V100上测试float16推理速度**

但是，TensorRT实际上并没有做真正的float16量化，也没有做校准。由于，float16依然具有较大的数值表示范围，float32多出来的较小的数值，并不会对准确率造成大的影响。以GoogleNet为例：



**图 GoogleNet fp32与fp16准确度对比**

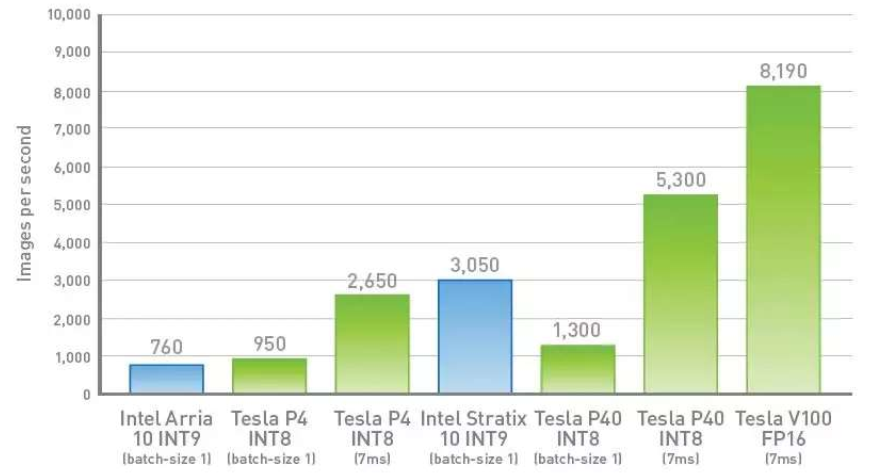
但是吞吐量的提升却是显著的，性能如下图所示：



**图 P100 fp16模式、P4 int8模式、V100 fp16模式的与CPU相比的加速比柱状图**

CPU Servers：Xeon E5-2690 v4。

很明显可以看到，P100的fp16模式相比于P4同样也有吞吐率性能提升，而且省去了量化和校准的流程。V100由于Tensor Core的引入，其相较于P4 int8模式有着很大的性能提升。



**图 batch size 1或7ms延迟情况下，GoogleNet网络吞吐率对比图**

#### （4）小结

针对TensorRT，总结一下：

1、TensorRT不仅仅是优化工具，它是一个针对GPU的带有优化工具的推理框架。

2、针对完整训练后的、对称的、动态的、均匀量化。

3、需要先做float32推理，转换时间比较长。

4、官网称支持float16、int8类型量化，没有开源代码，只有int8量化的资料介绍。

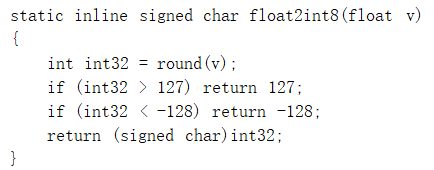
#### 3.2.3 Pytorch

Caffe2也集成到了Pytorch中，调研Pytorch也包含了Caffe2。很遗憾的是，网上并没有相关资料。从github上看，Pytorch没有量化相关代码。Caffe2是支持int8部分算子推理，Caffe2支持float16，但是没有量化过程，输入的是已经量化过的数据。

#### 3.2.4 NCNN

#### （1）简述

NCNN官网上声称支持int8和float16，做法和理论较为简单。从源代码上看，只能看到int8的代码，如下：

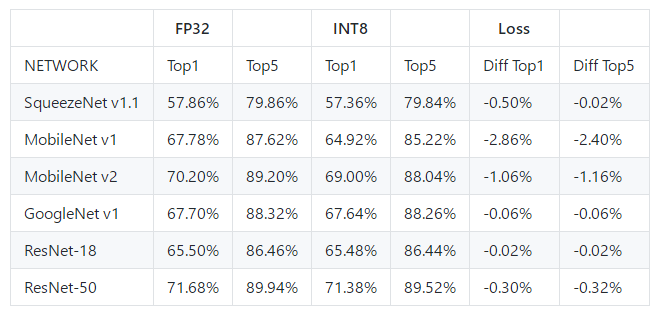


**图 NCNN int8量化过程**

同时，NCNN主推的是caffe-int8-convert-tools，基于TensorRT 2.0 Int8校准工具，使用KL散度找到合适的阈值。

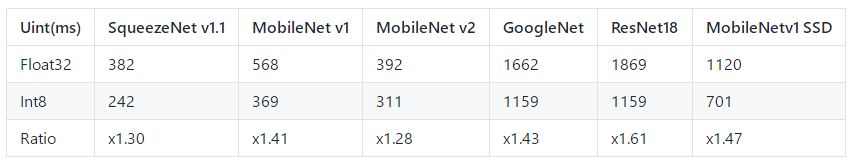
#### （2）性能

从Github上caffe-int8-convert-tools的网站（https://github.com/BUG1989/caffe-int8-convert-tools）获取到数据，float32和int8推理top1和top5精度测试结果：



**图 caffe-int8-convert-tools float32和int8推理结果对比**

在Hisi3519([Cortex-A17@1.2GHz](mailto:Cortex-A17@1.2GHz))平台上，float32和int8推理速度测试结果：



**图 caffe-int8-convert-tools float32和int8推理结果对比**

#### 3.2.5 Pocketflow

#### （1）简述

今年11月1日，腾讯AI Lab在南京举办的腾讯全球合作伙伴论坛上宣布正式开源“PocketFlow”项目， 该项目是一个自动化深度学习模型压缩与加速框架，整合多种模型压缩与加速算法并利用强化学习自动搜索合适压缩参数，解决传统深度学习模型由于模型体积太大，计算资源消耗高而难以在移动设备上部署的痛点，同时极大程度的降低了模型压缩的技术门槛，赋能移动端AI应用开发。

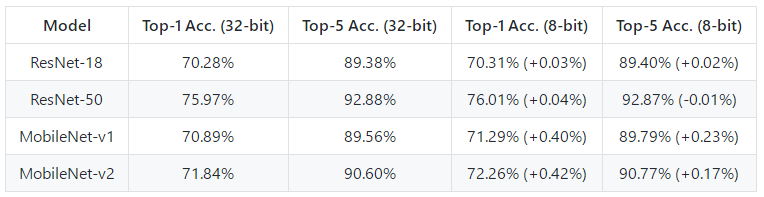
这是一款适用于各个专业能力层面开发者的模型压缩框架，基于Tensorflow开发，集成了当前主流与AI Lab自研的多个模型压缩与训练算法，并采用超参数优化组件实现了全程自动化托管式的模型压缩。开发者无需了解具体模型压缩算法细节，即可快速地将AI技术部署到移动端产品上，实现用户数据的本地高效处理。

目前该框架在腾讯内部已对多个移动端AI应用模型进行压缩和加速，并取得了令人满意的效果， 对应用整体的上线效果起到了非常重要的作用。

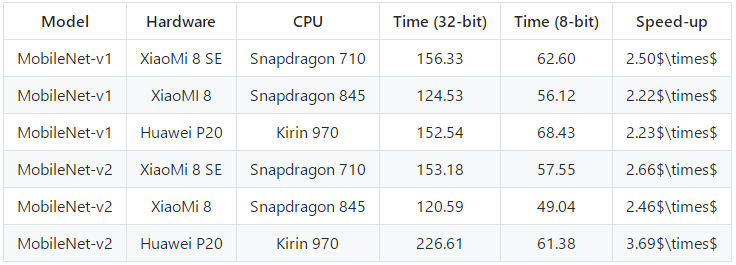
关于量化，PocketFlow提供了自己的量化感知训练来产生量化模型，这需要更长的时间，但准确性会更高。PocketFlow没有做训练后量化的开发，推荐使用的是Tensorflow的量化工具。

#### （2）性能

在下面两个表中，PocketFlow展示了8位**量化模型**可以和原来的32位模型一样精确(甚至更好)，量化后推理时间可以明显缩短。



**图 PocketFlow float32和int8推理结果精度对比**



**图 PocketFlow float32和int8推理结果速度对比**

以上时间是以毫秒为单位。

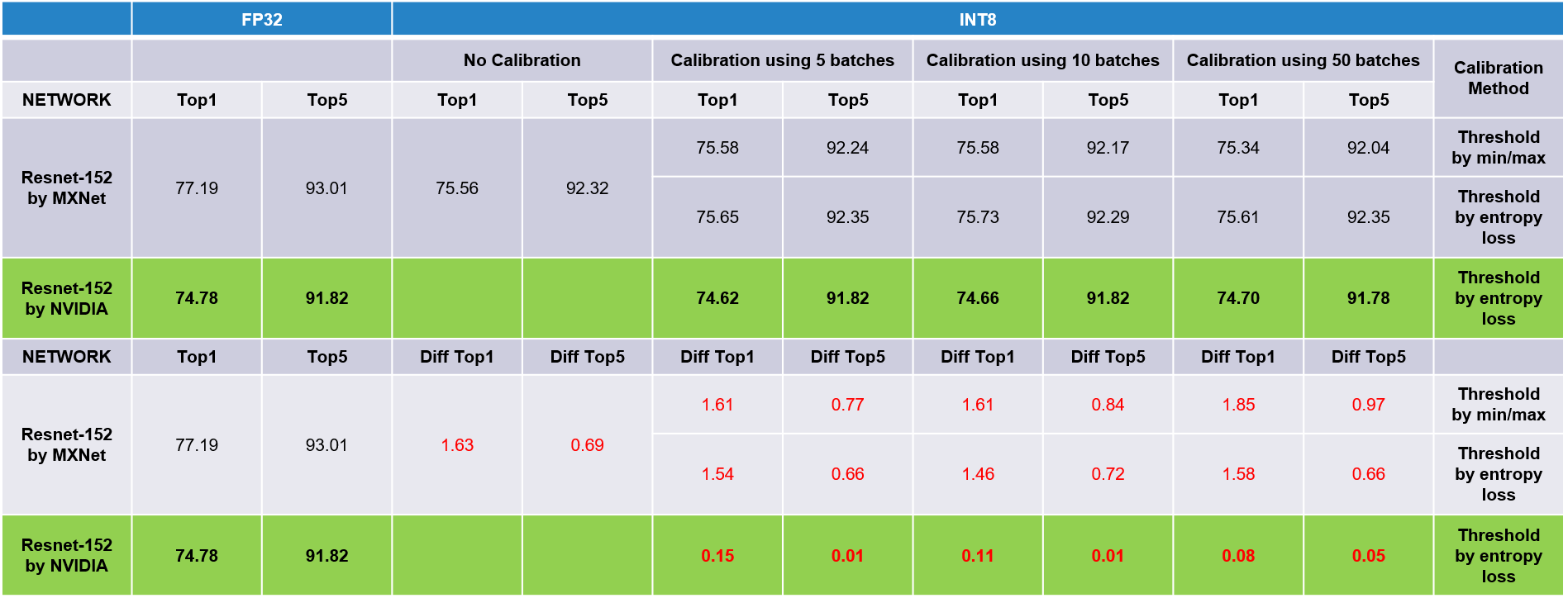
#### 3.2.6 MXNet

#### （1）简述

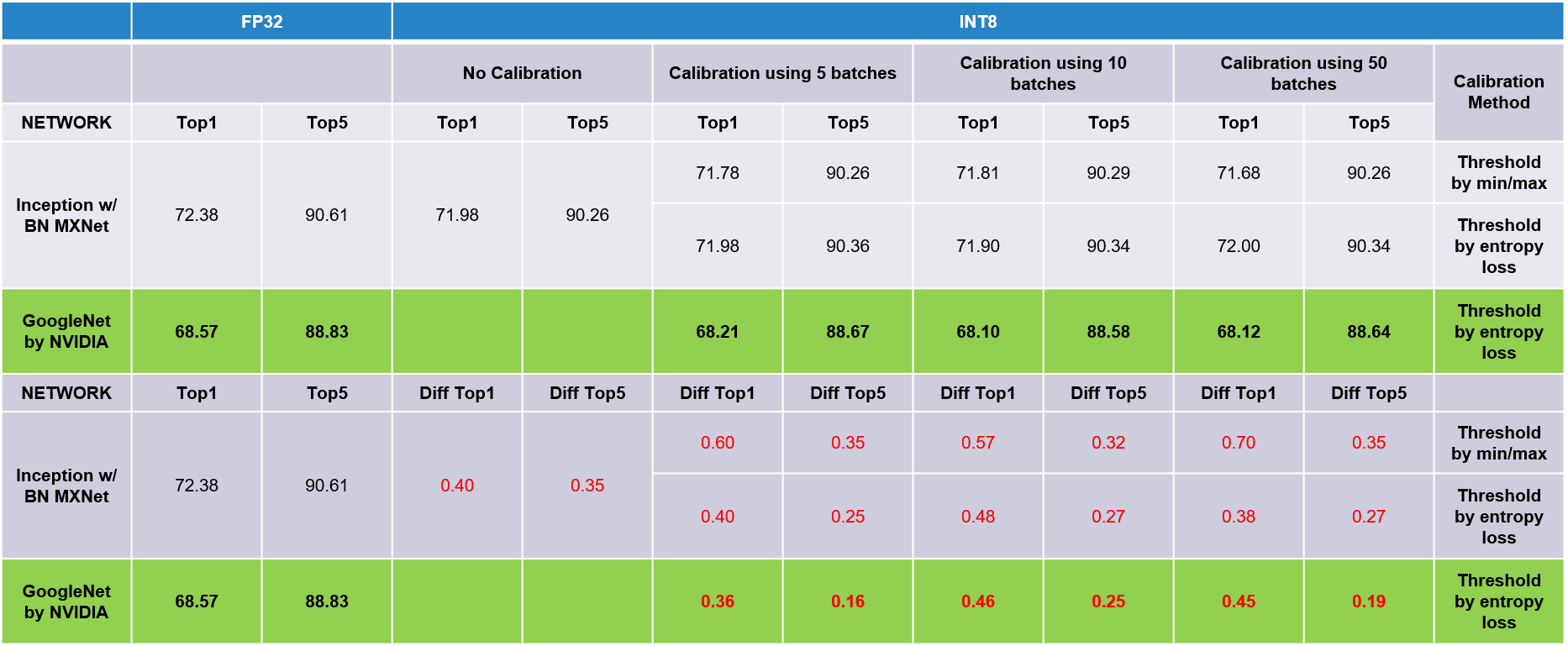
MXNet量化整体上是tensorflow量化工具的流程，但是加入了tensorRT做KL散度校验的思想。

#### （2）性能

来自Github的测试资料：https://github.com/apache/incubator-mxnet/pull/9552



**图 MXNet float32和int8推理结果精度对比1**



**图 MXNet float32和int8推理结果精度对比2**

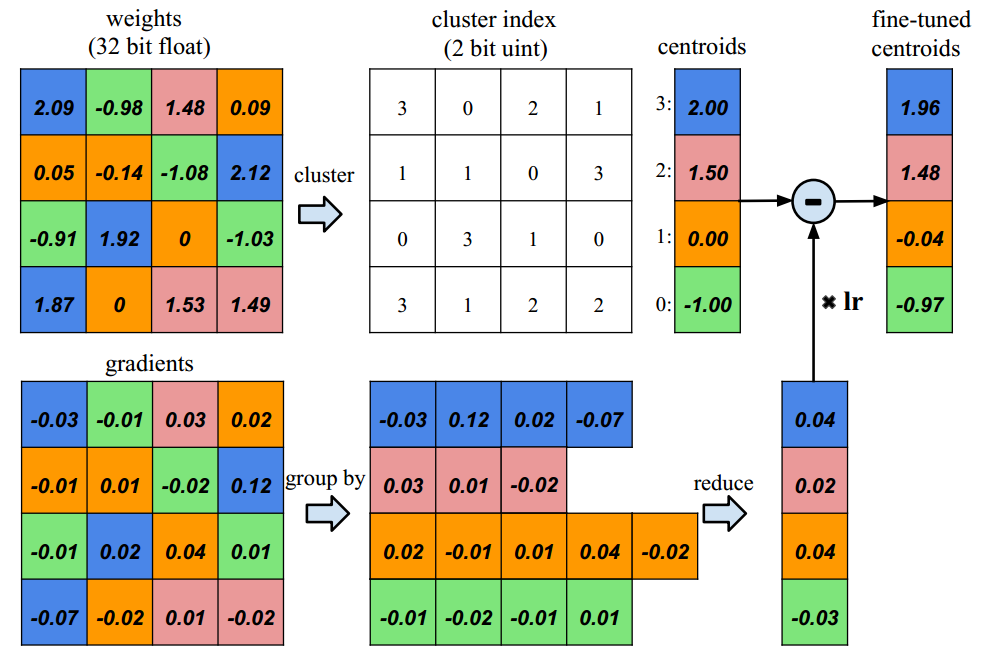
#### 3.2.7 深鉴科技

#### （1）简介

韩松博士在其毕业论文里提出了量化训练压缩的方法。通过权值共享来减少模型所需表达的权重个数，再通过整数量化进一步减少权重所需要的位数。

#### （2）原理

论文对每一层weight矩阵利用K-means聚类算法聚类成若干个cluster，用每个cluster的聚类中心值代表该cluster的权重，由于同一cluster的weight共享一个权重大小，因此我们只需要存储权值的cluster的index即可，通过查表获取该index对应的value，为了减少精度损失，再通过训练微调的方式对权重进行补偿，所有的梯度信息按照权重矩阵之前的分组进行。另外，权值的聚类分类，只在同一层layer中有效，不能跨layer共享权值。下图很形象的说明了聚类量化与训练过程。

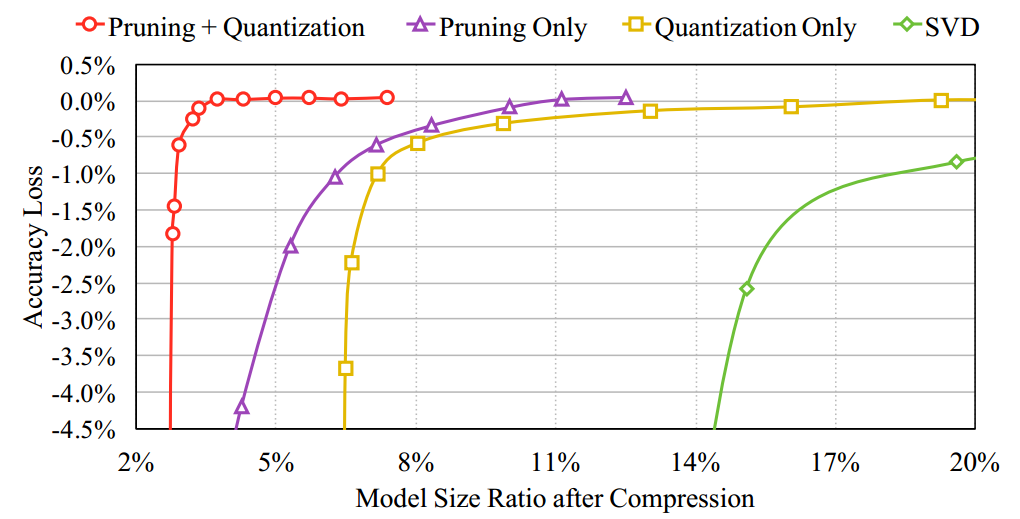


**图 Trained quantization by weight sharing (top) and centroids fine-tuning (bottom)**

该量化是非均匀量化。

#### （3）性能

剪枝和量化是可以独立进行的，从韩松的实验结果来看，单独量化和单独剪枝的压缩效果十分好，如果两个一起，压缩率做会达到惊人的3%左右。

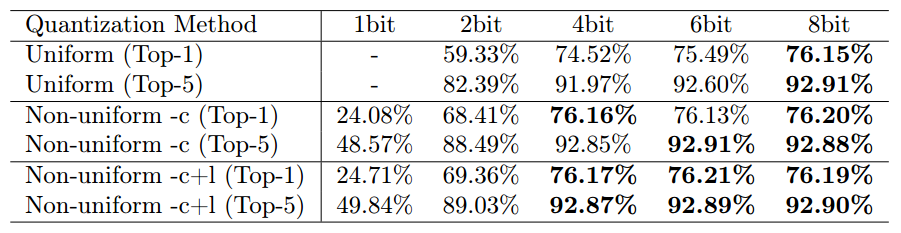


**图 Accuracy vs. compression rates under different compression methods. Pruning and quantization works best when combined**

韩松博士对均匀量化与非均匀量化选择和是否重训练这个两个问题进行了分析。

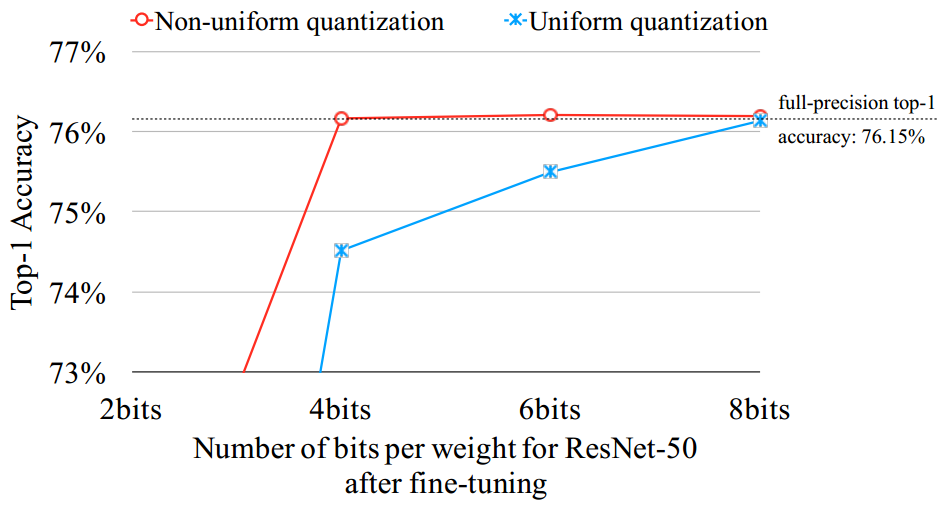
他用ResNet-50进行了重训练和没有重训练的测试。比如，量化4bit，没有重训练，top-1准确率是**68.30**%；而经过重训练，top-1准确率是76.17%。量化2bit，没有重训练，top-1准确率几乎是0，而经过重训练，top-1准确率是69.36%。下图是经过重训练的测试数据。

ResNet-50基准准确率是76.15%和92.87%。



**图 Non-uniform quantization performs better than uniform quantization 1**

对于非均匀量化，基线 ResNet-50 所有层级的参数可以压缩为4bit而没有准确率损失。然而对于均匀量化，基线 ResNet 所有层的参数只能压缩到8bit而没有准确率损失（压缩到 4bit会产生 1.6% 的 Top-1 准确度损失）。非均匀量化可以很好的捕捉到权重的不均匀分布，而均匀量化不能很好的实现这一点。



**图 Non-uniform quantization performs better than uniform quantization 2**

## NVDLA量化相关介绍

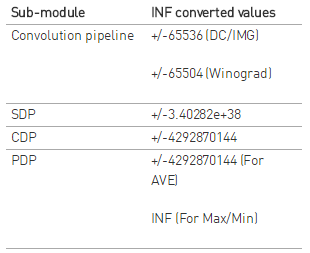
#### 4.1、开源资料

在编译器资料中提到，编译器负责讲训练后的float32模型量化为精度更低的模型。This phase is also responsible for quantizing higher precision trained models (like 32-bit floating point) into lower precisions, such as 8-bit or 16-bit integer, or 16-bit floating point, and for determining memory allocations for the quantized weights during inferencing jobs。

在硬件资料中有一些关于fp16的说明。简单说一下NVDLA如何在数据通路支持fp16的。

（1）Infinity

NVDLA将无穷大值视为根据模块不同的标准化值。



**图 NVDLA sub module支持fp16 infinity值**

任何NVDLA子模块都不会有任何INF输出，如果发生饱和，NVDLA将输出最大可表示的值（FP16为+/- 65504，INT16为32767 / -32768，INT8为127 / -128）。

（2）NaN

NVDLA不会产生NaN，因为任何操作都不涉及无穷大值。但它支持NaN传送，如果输入数据具有NaN，则与NaN操作数相关的任何结果将是NaN（尾数传送行为未定义？？）。NVDLA提供了一个寄存器字段，用于将NaN刷新为零。 如果设置了寄存器，则所有在浮点数据通路上的输入的NaN都被视为零值，输出data cube没有任何NaN。否则输入NaN传送到输出。

NVDLA还提供输入/输出NaN计数寄存器，用于汇总输入/输出data cube中的总NaN数。 完成图层时更新计数寄存器。 当完成中断到达时，FW可以轮询NaN计数寄存器以确定输入/输出data cube是否具有任何NaN值。

（3）非标准化值

NVDLA支持输入和输出的非标准化值。非标准化值的处理完全符合IEEE754标准的要求。实际上，NVDLA内部浮点数据通路通常提供fp17 / fp32值以获得更好的精度。即使这些格式具有比具有非标准化值的fp16更好的精度，这些fp17和fp32格式在计算期间也不支持非标准化值。在写回内存之前，fp17 / fp32将转换为具有非标准化值的fp16。

（4）Rounding

除溢出情况外，NVDLA在计算中支持舍入到最近（或RN）。 如果结果超出最大正常值，则将其剪裁为最大标准化值。

#### 4.2、half转换库

这是一个C++头文件库，提供符合IEEE 754标准的16位半精度浮点类型以及相应的算术运算符，类型转换和常用数学函数。它旨在提高效率和易用性，尝试以尽可能最佳的性能准确模拟内置浮点类型的行为。

从float32转到float16默认的舍入模式截断，向0取整，映射溢出取无穷大。这是最快的舍入模式，通常来说是足够了。也可以使用其它舍入模式。

参考网站：<http://half.sourceforge.net/index.html>

#### 4.3、对比平台

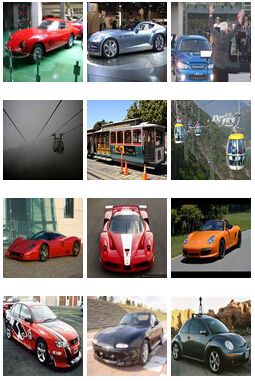
一般在做模型量化研究时，大家会对在各自的平台上对常用的标准模型利用标准的数据集进行测试对比，测试包括准确度和推理速度。

作为测试的常用的标准模型，主要分为两类：图像分类和物体检测。图像分类模型包括VGG、ResNet、Inceptionv3、DenseNet121、Squeezenet、MobileNet等系列模型；物体检测模型包括Faster-RCNN、SSD、YOLO等系列模型。

图像分类最常用的数据集是ImageNet。

【**ImageNet**】

ImageNet是一个计算机视觉系统识别项目，是目前世界上图像识别最大的数据库。是美国斯坦福的计算机科学家李飞飞模拟人类的识别系统建立的。能够从图片识别物体。目前已经包含14197122张图像，是已知的最大的图像数据库。每年的ImageNet大赛更是魂萦梦牵着国内外各个名校和大型IT公司以及网络巨头的心。图像如下图所示，需要注册ImageNet帐号才可以下载，下载链接为<http://www.image-net.org/>



**图 ImageNet数据集示意图**

物体检测最常用的数据集是PASCAL VOC和COCO数据集。

【**PASCAL VOC**】

PASCAL VOC 数据集是视觉对象的分类识别和检测的一个基准测试，提供

了检测算法和学习性能的标准图像注释数据集和标准的评估系统。图像如下图所示，包含VOC2007（430M），VOC2012（1.9G）两个下载版本。下载链接为<http://pjreddie.com/projects/pascal-voc-dataset-mirror/>



**图 PASCAL VOC数据集示意图**

【**COCO**】

COCO是一种新的图像识别，分割和加字幕标注的数据集。主要由Tsung-Yi

Lin（Cornell Tech），Genevieve Patterson （Brown），MatteoRuggero Ronchi （Caltech），Yin Cui （Cornell Tech），Michael Maire （TTI Chicago），Serge Belongie （Cornell Tech），Lubomir Bourdev （UC Berkeley），Ross Girshick （Facebook AI), James Hays (Georgia Tech),PietroPerona (Caltech)，Deva Ramanan (CMU），Larry Zitnick （Facebook AI）， Piotr Dollár （Facebook AI）等人收集而成。其主要特征如下：

（1）目标分割

（2）通过上下文进行识别

（3）每个图像包含多个目标对象

（4）超过300000个图像

（5）超过2000000个实例

（6）80种对象

（7）每个图像包含5个字幕

（8）包含100000个人的关键点

图像如下图所示，支持Matlab和Python两种下载方式，下载链接为

<http://mscoco.org/>



**图 COCO数据集示意图**

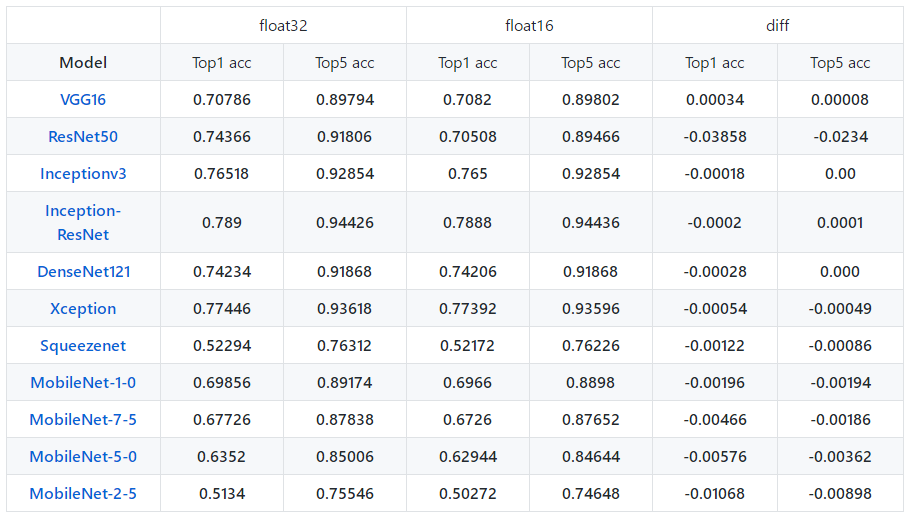
我们希望能够搭建一个通用的测试平台，可以采用keras+tensorflow的平台，测试常用的标准模型，数据集为ImageNet、VOC2007和COCO2017。

在网上找到一个比较好的开源测试代码，见：

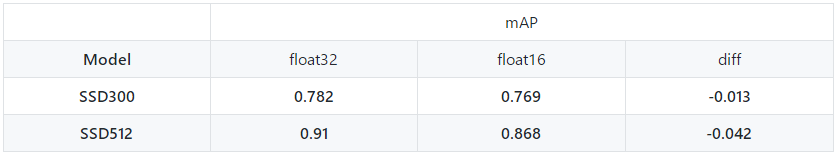
<https://github.com/TianzhongSong/keras-FP16-test>

通过分别测试ImageNet 、VOC2007和COCO2017 , 5万张图片，统计float32模型和通过float32模型转到float16模型的准确率。

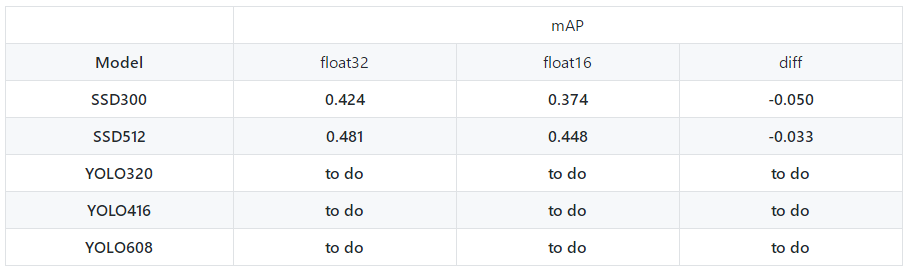
测试效果如下：



**图 keras-FP16-test图像分类在ImageNet上测试结果**



**图 keras-FP16-test物体检测在VOC2007上测试结果**



**图 keras-FP16-test物体检测在COCO2017上测试结果**

## 总结

### 5.1 简要总结

目前，深度学习量化工是作深度学习研究领域的热门课题之一。它的意义不仅仅是追求学术上的创新，更重要的是能够将复杂的模型实际用在有限的硬件资源上，使人工智能产品快速走进生活。

深度学习量化可调研的资源主要是具有代表性的、主流的框架和硬件平台，再结合一些比较好的经典论文。本文调研侧重的是框架和平台，因为虽然有的论文比较精彩，但是我们从实际应用的角度来考虑，主流的框架和平台更具有代表性和说服力。主流的框架和平台往往能体现了深度学习应用的发展过程和发展导向。

从上面的调研来看，可以得出几个基本结论。

1、基本上没有人特别来做float16的量化，所有的量化工作都集中在了较比float16更低比特位的整数型数据的量化上面，研究最多的就是int8量化。所谓的float16量化就是float32数据类型转到float16数据类型简单的过程，只要遵循IEEE 754的标准，就可以实现，并具有通用性。

2、主流框架基本上采用的是均匀量化。好处是，量化速度快，不需要复杂的处理过程。

3、主流框架基本上提供两种量化方式，训练时量化和训练后量化。训练时量化要重新训练，训练后量化不需要重新训练。前者的准确率更高，但是转换时间较长，后者的准确率略低于前者，但是转换时间短而且方便。主流框架更侧重、更投入做前者的量化工作，推出量化感知训练。

4、主流框架基本上时对于每层或者每个算子单独做量化，也就是动态量化。

5、大部分框架采用的是IEEE 754标准数据格式。

### 5.2 需要考虑的问题

我们即将开案的是float16，int8暂时没有开案，下一步，我们要考虑如下的问题：

**1、16bit浮点数采用的数据格式。**

IEEE 754标准float 16格式。

**2、训练时量化还是训练后量化。**

训练后量化。

**3、转换手段。**

采用标准C++ half库，对float32数据类型转换到float16数据类型。

**4、float16转换与HNN关系。**

float16转换可以作为独立的转换工具，在模型送给HNN框架前执行该工具。输入的是float 32的ONNX模型，经过float16转换工具，输出的是float 16的ONNX模型，再送给HNN框架执行。



**图 FP16 convertor与HNN关系**