|  |  |
| --- | --- |
| Quantization调研 | 文件类型：技术文档 |
| 文件编号： |
| 面向的部门： 研发中心 |
| 保密等级：高 |
| 作者： 刘凯 |
| 日期：2019-05-21 |
| 版本：1.0V |

Quantization调研



目 录

[目 录 ii](#_Toc615510541)

[1 深度神经网络模型压缩 3](#_Toc1869803024)

[2 量化-TensorRT方案 3](#_Toc1615557293)

[2.1 量化 3](#_Toc1720023455)

[2.2 校准 5](#_Toc480047749)

[2.3 结果 6](#_Toc1180188194)

[3 量化-Google方案 6](#_Toc1129165203)

[3.1 量化方案 6](#_Toc271641466)

[3.2 量化乘法 7](#_Toc567253777)

[3.3 零点的简化 7](#_Toc1439988696)

[3.4 加bias 8](#_Toc1700812250)

[3.5 训练 9](#_Toc224514808)

[参考文献 1](#_Toc1703358224)

# 深度神经网络模型压缩

随着神经网络算法的发展，网络性能虽然越来越强大，但是也耗费了太多的计算资源和内存，为了得到更有效率的网络以及能部署在移动端，近几年神经网络的压缩算法成了一个研究热点。主要的网络压缩有五个方向：低秩分解(Low-rank factorization)，剪枝(Pruning)，量化(Quantization)，知识蒸馏(Knowledge distilling)和轻量级网络设计。

低秩分解。卷积神经网络中的主要计算量在于卷积层的计算，而卷积核是一个4维张量，且这个张量一般是含有很多冗余信息的，所以可以通过张量分解来减少矩阵运算的计算量。对于权重为2维矩阵的全连接层也适用。

剪枝就是将深度神经网络中一些不重要的连接减掉，只保留重要的部分，从而减小网络模型。

知识蒸馏。最早由文献[1]提出，核心思想是通过迁移知识，从而通过训练好的大模型得到更加适合推理的小模型，其本质上是一种迁移学习(Transfer Learning)。

轻量级网络设计。设计更小更紧致的轻量级网络，典型的轻量级网络包括MobileNet[2-3]，SqueezeNet[4]，ShuffleNet[5]等。

量化。深度学习框架一般都是采用全精度FP32(Full Precision)，而量化是采用更低的bit数(如INT8)来表示权重和激活值，可以在很小的精度损失下减少模型大小和计算量。量化的代表算法包括BNN[6]，TWN[7]，XNOR-net[8]等。

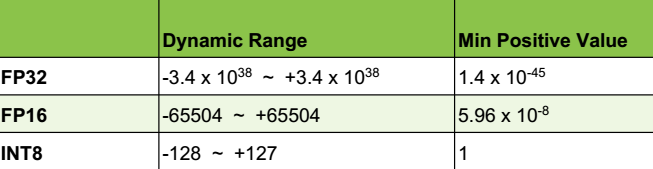
虽然以上算法可以大幅度压缩模型参数，但也有很大的精度损失，在工业界主要还是应用INT8量化。其中主要有两个方案，其一是英伟达的TensorRT方案[9]，是直接量化，没有重新训练，实现较简单；其二是谷歌的方案[10]，需要重新训练，前向用量化，反向用浮点，实现较复杂。本文主要介绍这两种方案。

在量化网络中，需要量化的对象通常包括两类：权重(Weights)和激活值(Activations)。其中权重在与输入计算之前量化，激活则在其前向推断时量化，如在卷积后激化函数前的激活值。

# 量化-TensorRT方案

* 1. 量化

TensorRT方案的目标是在性能损失很小的情况下，将FP32的CNN网络转换成INT8，因为INT8的吞吐量(Troughtput)更高且存储需求更低，但难点是INT8的精度低和动态范围小，如图所示。



图：FP32，FP16和INT8的动态范围和精度

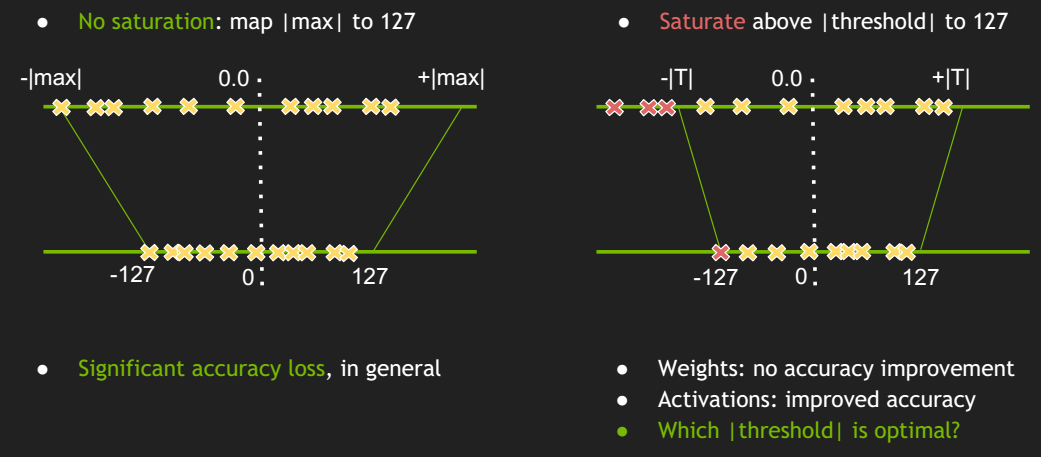
INT8的量化公式如下：



其中Tensor Values为FP32的数据，FP32 scale factor为32位的扩展系数，int8 array为量化值，FP32 bias为量化偏置。但从实验得知，量化偏置舍弃对结果影响不大，所以量化方案为：

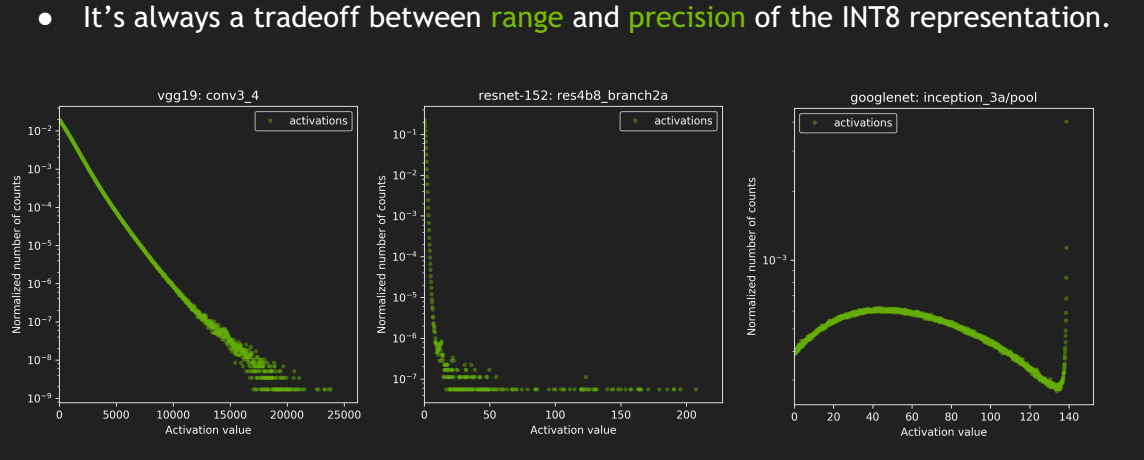


现在的问题是如何确定scale factor？图显示了两种方法。左边的是不饱和方法，即将FP32的正负最大值映射到INT8的动态范围[-127,127]，很明显只要数据不是均匀分别，精度损失很大。因此TensorRT采用的是右边的饱和方法，即选取一个截断值T，将[-T,T]映射到[-127,127]上。

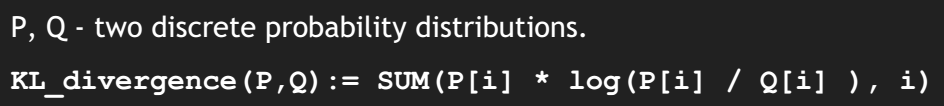


图：两种量化方法

现在的问题是如何确定截断阈值T。因为神经网络的激活值分布不同，如图所示，因此截断阈值的选取要让量化后信息损失越小越好。TensorRT方案选择KL散度(Kullback-Leibler Divergence)作为信息损失的度量，如图所示。具体实现参考2.2小结。



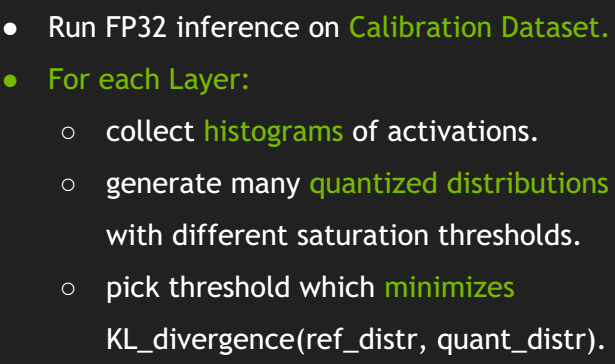
图：不同网络的激活值分布



图：KL散度

* 1. 校准

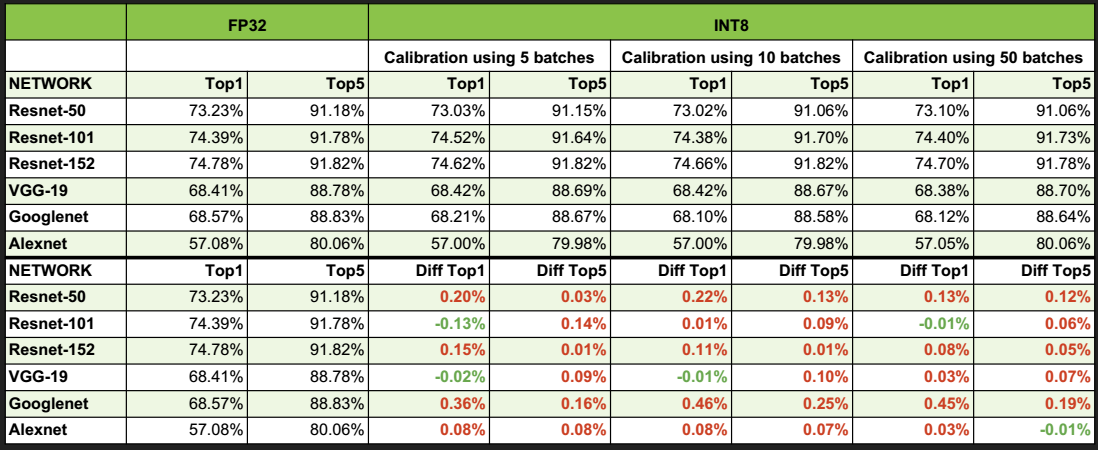
上一小结说明了TensorRT的量化方案，其关键在于确定截断值T使得量化后损失的信息最小，因此需要校准(calibration)。校准方案如图所示。具体而言，首先准备一个校准数据集，在数据集上前向推理，保存FP32的激活值，TensorRT的校准数据集有500张图片。然后对每层：1)统计激活值的直方图，用2048个bins；2)根据不同的截断阈值产生量化分布；3)选择能最小化量化前后分布的KL散度的截断阈值。



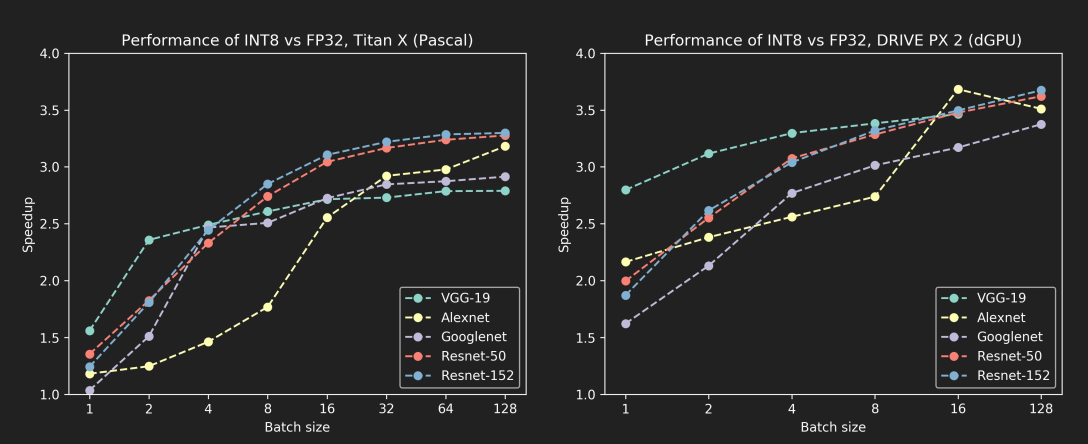
图：校准方案

* 1. 结果

图为FP32和INT8的精度对比，可以看到精度损失很小。图2为INT8和FP32运行速度的对比。



图：FP32和INT8的精度对比



图：INT8和FP32运行速度的对比

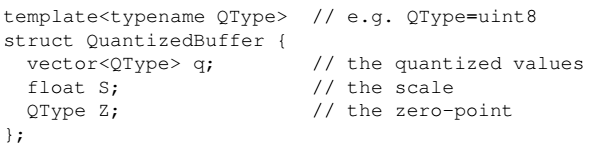
# 量化-Google方案

* 1. 量化方案

量化将整数q仿射映射(affine mapping)到实数r，即：



其中r为待量化的实数，通常为FP32。q为量化后的整数，如INT8。常数S(scale)是正实数，表示缩放比例，常数Z(zero-point)和q类型一样，表示q在零点的量化值。一个通用的量化缓存数据结构如下所示：

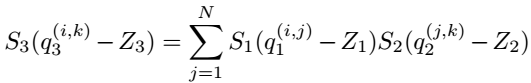


* 1. 量化乘法

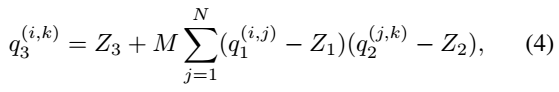
设有两个N×N的实数矩阵*r1*和*r2*，且*r3=r1r2*根据前面的量化方案有：



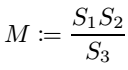
则矩阵乘法为：



上式可以转换为：



其中：



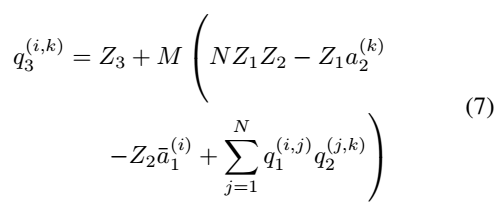
量化乘法中只有常数M不是整数。事实上M通常都落在(0, 1)区间，因此可以写成如下形式



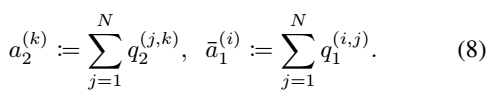
其中M0落在[0.5, 1)区间，且n为一个非负整数。这其实相当于将32位浮点数M量化为int32，M0就是离231M最近的int32值，且精度在2-30以上。

* 1. 零点的简化

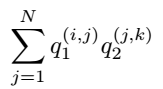
上面(4)式要执行2N3次减法操作，为了优化零点的计算，可以将(4)式展开：



其中：



因此与零点相关的计算需要2N2次加法，(7)式的主要计算集中在



上式需要2N3次算数操作。其计算量与其他zero-point-free的量化方案相当了。

* 1. 加bias

量化网络的乘累加运算需要一个32bit的累加器，即：

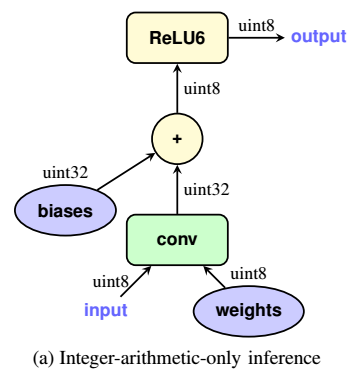


为了方便与bias相加，量化方案将bias量化为32bit，即：



可以看到，bias的缩放因子是input和weight缩放因子的乘积，零点为0。可以用32bit量化bias的原因如下：1) 在神经网络中，bias只占很小一部分，不会影响模型压缩；2)每个bias都是加到所有输出的激活值上，因此需要更高的精度。

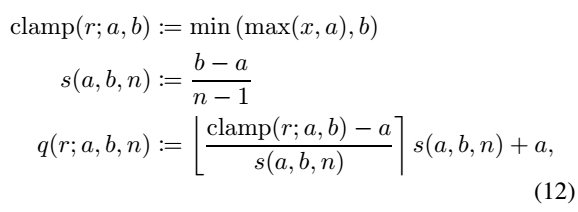
得到int32的累加结果后，需要缩放回8bit，这里是缩放到unit8，再经过激活函数，得到最终的8bit输出。图是TensorFlow里一个卷积层的量化流程图，其中ReLU6=min(max(x, 0), 6)，可以保证量化精度。



图：TF里量化卷积层图

* 1. 训练

训练策略是前向推断采用量化后的定点计算，反向传播则采用浮点计算，并保留浮点权重以进行微调。量化公式如下：



其中r为待量化的实数，[a, b]为量化范围，n为量化阶数，如8bit量化时，n=256。

对权重和激活值，量化范围的选取策略不同。权重的量化范围很简单，,。对8bit，量化范围选取[-127, 127]。激活值的量化范围选取跟输入有关，策略是在训练中用EMA(Exponential Moving Average)平滑增强。无论哪种情况，[a, b]都会进行微调，使得0.0能正好由z(a,b,n)表示。

参考文献

1. G. E. Hinton, O. Vinyals, and et al. Distilling the knowledge in a neural network. CoRR, vol. abs/1503.02531, 2015.
2. Ristani, E., Solera, F., Zou, R., Cucchiara, R., Tomasi, C.: Performance measures and a data set for multi-target, multi-camera tracking. In: European Conference on Computer VisiMobileNet. A. G. Howard, M. Zhu, B. Chen, and et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. https://arxiv.org/abs/1704.04861, 2017.
3. MobileNet2. Mark Sandler, Andrew Howard, and et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks. https://arxiv.org/abs/1801.04381, 2018.
4. Squeezenet. F. N. Iandola, M. W. Moskewicz, and et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5mb model size. https://arxiv.org/abs/1602.07360, 2016.
5. Shufflenet. X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. https://arxiv.org/abs/1707.01083, 2017.
6. BNN. I. Hubara, M. Courbariaux, and et al. Binarized neural networks. In Advances in neural information processing systems, pages 4107–4115, 2016.
7. TWN. F. Li, B. Zhang, and B. Liu. Ternary weight networks. arXiv preprint arXiv:1605.04711, 2016.
8. Xnornet. M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnornet: Imagenet classification using binary convolutional neural networks. arXiv preprint arXiv:1603.05279, 2016.
9. NVIDIA. 8-bit Inference with TensorRT. http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf
10. Benoit Jacob, Skirmantas Kligys, and et al. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. https://arxiv.org/abs/1712.05877, 2018.on workshop on Benchmarking Multi-Target Tracking (2016).