

1. 前言

1.1 量化是什么？

量化是模型压缩的一种方式。量化就是把高位宽（例如 32float）表示的权值或者激活值用较低位宽来近似表示（int8），在数值上的体现就是将连续的值离散化。

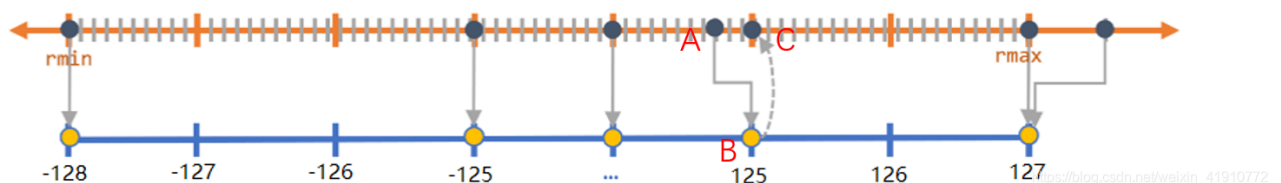
1.2 量化会带来什么？

量化主要用在边缘计算等硬件限制较大的场景下，即工业应用上（总不能在边缘上都带着 GPU 吧）。现有很多先进的神经网络（例如 resnet, densenet）在分类、识别上都取得了较好的效果，但其普及程度远不及效果稍差但模型小、运算快的 mobilenet，而 mobilenet 就是在权衡速度、识别率下产物。当然了，mobilenet 不是量化模型，只是用来举个例子，用于说明量化的潜力。

以下为量化所带来的一些影响：

优点 1：加快运算速度。当把 32float 转变为 int8 表示时，在不考虑系统有浮点加速模块时，**定点运算要比浮点运算快**，感兴趣的可查阅定点数和浮点数运算的区别 优点 2：减少存储空间。若将 32 浮点数转变为 8 位表示时，存储空间减小到了 1/4 大小。缺点 1：在用低带宽数值近似表示时，会造成一些精度损失。值得高兴的是，神经网络的参数大多是冗余的（或者说是对噪声的容忍度），所以当在近似变换时对精度的影响不是特别大。

下面用图来说明量化是怎么带来损失的，** A 为实际的浮点值，量化后近似为 B，但其表示的值为 C 点，缩放因子越大，A 和 C 的距离就越远，误差就越大，所以在量化时引入的近似会带来一些精度上的损失。（后面会具体讲解如何设置最值，来找到合适的缩放因子）



2. 量化具体介绍

2.1 非对称量化

1) 首先，设置浮点数的最大值 x_{\max} , 最小值 x_{\min} ：

对于权重：权重在训练后大小都是固定的，一般直接求出权重的最大值和最小值。对于激活值：会随着输入值的改变而改变，所以不能直接求其最值，google 使用了滑动均值平均的方法，TensorRT 使用 KL 散度，Easyquant 使用 cos 相似度，后面会具体介绍。

2) 其次，设置要量化的范围 x_{q_max}, x_{q_min} ，在非对称量化下为 $[0, 255]$ 。3) 之后，计算缩放因子 Scale (float32) 和平移因子 Zero_point (int8):

$$s = \frac{x_{\max} - x_{\min}}{255} \quad z = \text{round}\left(\frac{x_{\min}}{s}\right)$$

4) 最后进行量化和解量化：图中的 $N_{\text{levels}}-1$ 为 255, round 为四舍五入

Once the scale and zero-point are defined, quantization proceeds as follows:

$$x_{int} = \text{round}\left(\frac{x}{\Delta}\right) + z \quad (1)$$

$$x_Q = \text{clamp}(0, N_{levels} - 1, x_{int}) \quad (2)$$

where

$$\begin{aligned} \text{clamp}(a, b, x) &= a & x < a \\ &= x & a \leq x \leq b \\ &= b & x > b \end{aligned}$$

The de-quantization operation is:

$$x_{float} = (x_Q - z)\Delta \quad (3)$$

然而为什么是在量化时使用先放缩再移位，解量化时先移位再放缩呢？可不可以反过来呢？答案是否定的，主要从效果和运算两个方面考虑：

在运算上，主要考虑如何使用整形运算代替浮点运算，可以参考 [google 论文](#) (后面会有详细介绍)

在效果上，主要考虑关键数 0，浮点中的 0 的关键用法主要在于补零 padding 和激活 ReLU 上，所以为了保证量化效果，需要将浮点 0 无偏差的使用整形来代替。可以参考 [这里](#)，详细解释如下：

两种解量化方法如下：

缩放因子A **平移因子b**

$\text{real_value} = A * \text{quantized_value} + B$ (1)

$\text{real_value} = C * (\text{quantized_value} + D)$ (2)

训练完权重的最大最小值

缩放因子: $(X_{\max} - X_{\min}) / 255$

平移因子: $(\text{round}(x_{\min} / S))$

(讨论的解量化)

(本文中的解量化，先移位再放缩，上下文移位的正负号一致性请忽略)

对于第一种，让 real_value 为 0，得到的 zero_point 如下，很难保证 zero_padding 为整形，因为缩放因子为浮点数。

In equation (1), plugging $\text{real_value} = 0$ and $\text{quantized_value} = \text{zero_point}$, we get:

$0 = A * \text{zero_point} + B$

equivalently:

$\text{zero_point} = -B / A$

而对于第二种，是不是就很完美了，zero_point 很自然的就整形了。

Now let us look at equation (2). Plugging $\text{real_value} = 0$ and $\text{quantized_value} = \text{zero_point}$, we get:

$0 = C * (\text{zero_point} + D)$

Conveniently, the constant C plays no role anymore, so this equation simplifies to:

$0 = \text{zero_point} + D$

In other words, $D = -\text{zero_point}$. This suggests rewriting the quantization equation (2) into the following form (3), which will be the final form that we will consistently use:

$\text{real_value} = \text{scale} * (\text{quantized_value} - \text{zero_point})$ (3)