TensorRT的INT8量化原理

- Goal: Convert FP32 CNNs into INT8 without significant accuracy loss.
- Why: INT8 math has higher throughput, and lower memory requirements.
- Challenge: INT8 has significantly lower precision and dynamic range than FP32.
- Solution: Minimize loss of information when quantizing trained model weights to INT8 and during INT8 computation of activations.
- Result: Method was implemented in TensorRT. It does not require any additional fine tuning or retraining.

INT8 Inference

Challenge

- INT8 has significantly lower precision and dynamic range compared to FP32.

	Dynamic Range	Min Positive Value
FP32	-3.4×10^{38} ~ $+3.4 \times 10^{38}$	1.4×10^{-45}
FP16	-65504 ~ +65504	5.96×10^{-8}
INT8	-128 ~ +127	1

- Requires more than a simple type conversion from FP32 to INT8.

Linear quantization

Representation:

Tensor Values = FP32 scale factor * int8 array + FP32 bias

Symmetric linear quantization

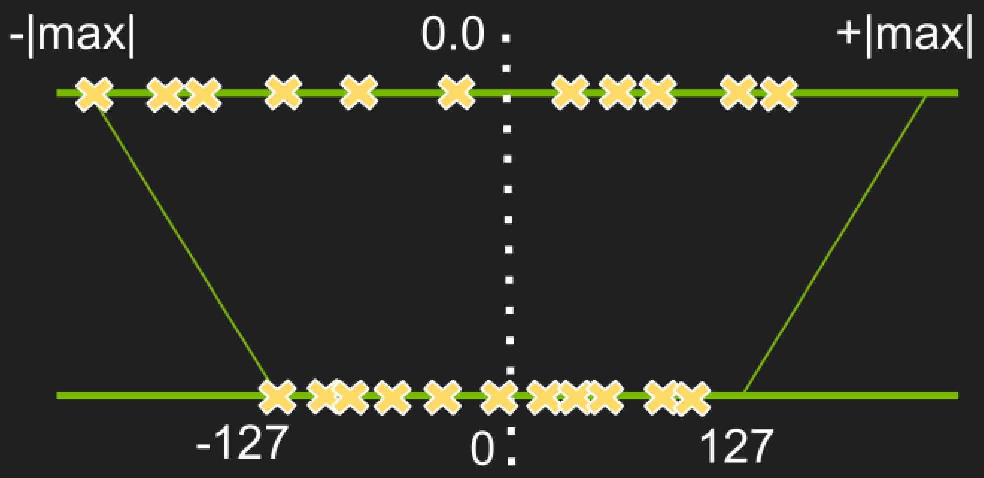
Representation:

Tensor Values = FP32 scale factor * int8 array

One FP32 scale factor for the entire int8 tensor

Quantization

- No saturation: map |max| to 127



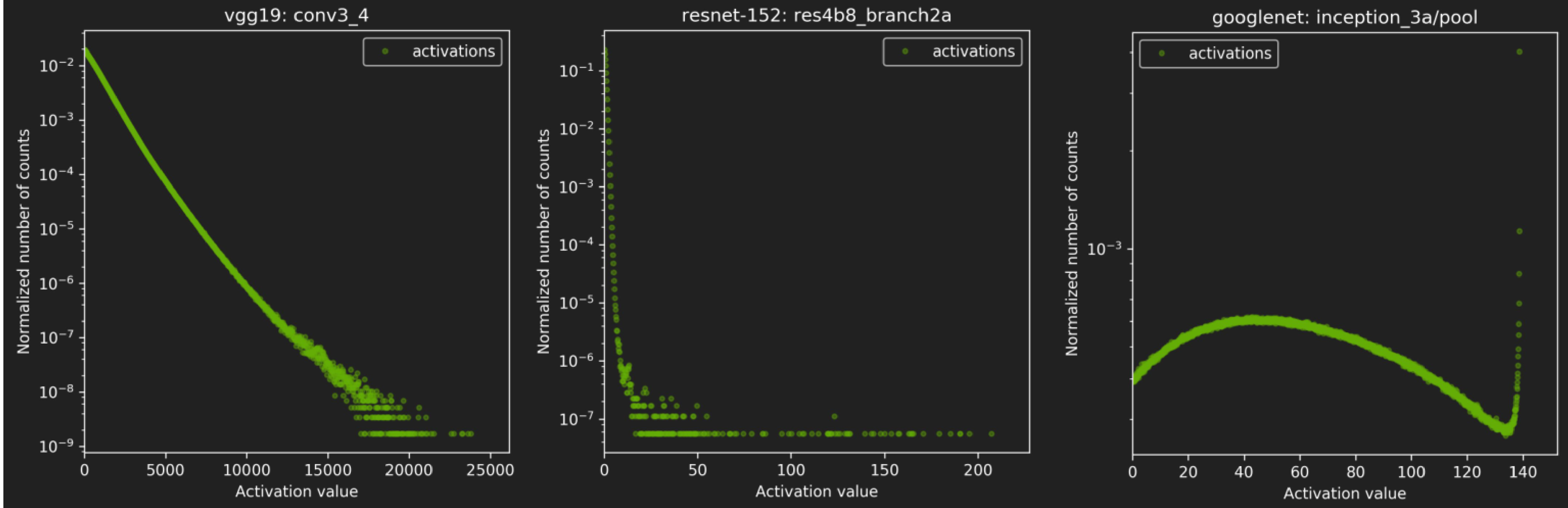
- Saturate above |threshold| to 127

- Significant accuracy loss, in general

- Weights: no accuracy improvement
- Activations: improved accuracy
- Which |threshold| is optimal?

Q: How to optimize threshold selection?

- It's always a tradeoff between range and precision of the INT8 representation.

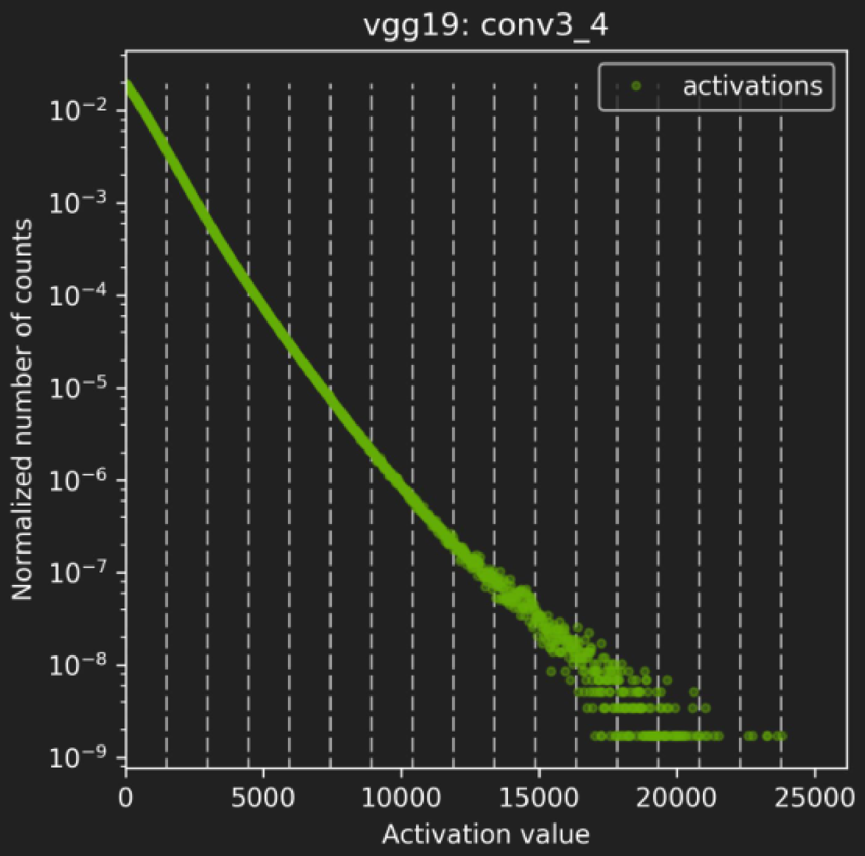


A: Minimize information loss, since FP32 → INT8 is just re-encoding information.

“Relative Entropy” of two encodings

- INT8 model encodes the same information as the original FP32 model.
- We want to minimize loss of information.
- Loss of information is measured by Kullback-Leibler divergence (AKA *relative entropy* or *information divergence*).
    - P, Q - two discrete probability distributions.
    - `KL_divergence(P,Q):= SUM(P[i] * log(P[i] / Q[i] ), i)`
- Intuition: KL divergence measures the amount of information lost when approximating a given encoding.

Solution: Calibration



- Run FP32 inference on Calibration Dataset.
- For each Layer:
  - collect histograms of activations.
  - generate many quantized distributions with different saturation thresholds.
  - pick threshold which minimizes KL_divergence(ref_distr, quant_distr).
- Entire process takes a few minutes on a typical desktop workstation.

如何寻找最优的阈值T使得精度的损失最小?

NVIDIA选择的是KL-divergence，其实就是相对熵。相对熵表述的就是两个分布的差异程度，这里就是量化前后两个分布的差异程度。差异最小就是最好的了，因此问题转换为求相对熵的最小值。

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot (\log p(x_i) - \log q(x_i))$$

KL散度来精确测量这种最优和次优之间的差异。

F32就是原来的最优编码，INT8就是次优的编码，用KL散度来描述这两种编码之间的差异。

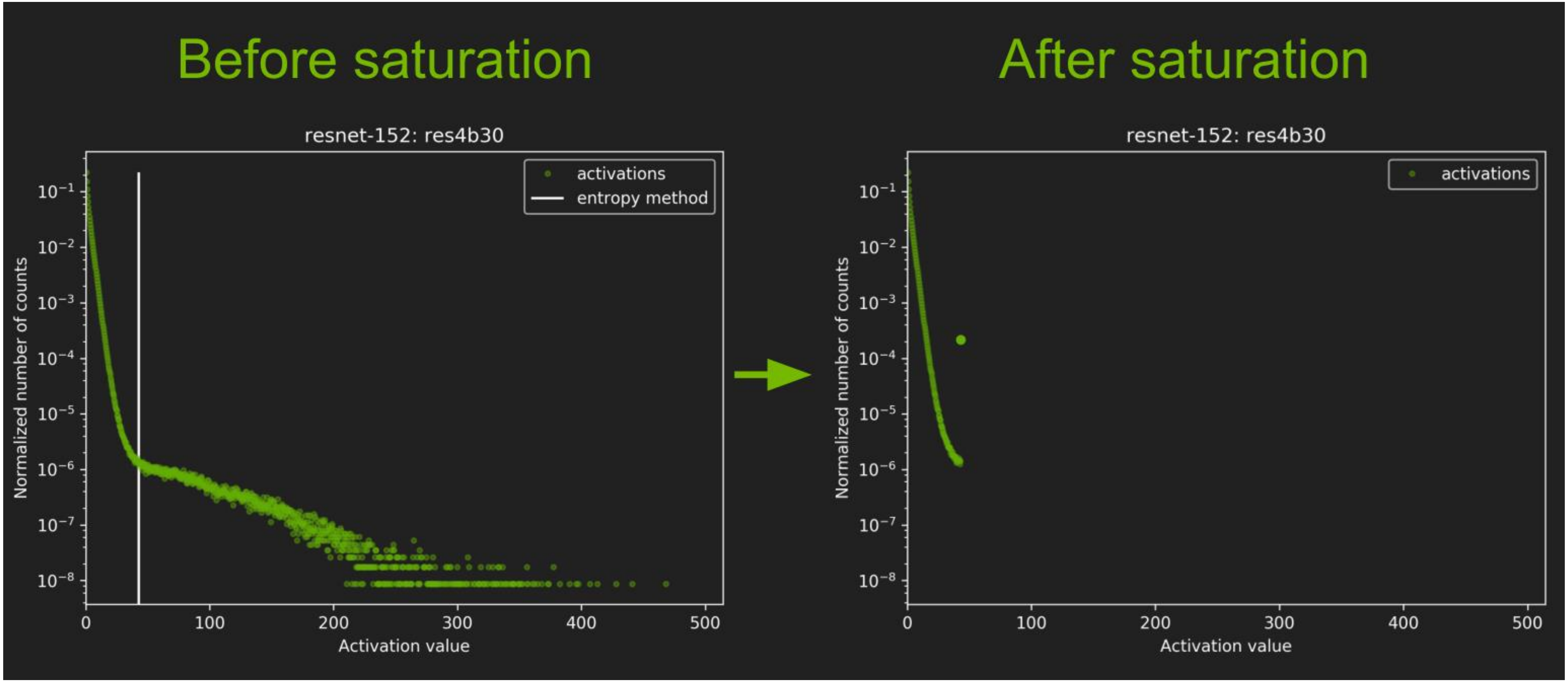
相对熵表示的是采用次优编码时会多需要多少个bits来编码，也就是与最优编码之间的bit差;

而交叉熵表示的是用次优编码方式时确切需要多少个bits来表示;

因此，最优编码所需要的bits=交叉熵-相对熵。

Typical workflow in TensorRT

- You will need:
  - Model trained in FP32.
  - Calibration dataset.
- TensorRT will:
  - Run inference in FP32 on calibration dataset.
  - Collect required statistics.
  - Run calibration algorithm → optimal scaling factors.
  - Quantize FP32 weights → INT8.
  - Generate “CalibrationTable” and INT8 execution engine.


Before saturation
After saturation
resnet-152: res4b30
activations
entropy method
Normalized number of counts
Activation value
resnet-152: res4b30
activations
Normalized number of counts
Activation value

校准算法：

calibration：基于实验的迭代搜索阈值。

- 提供一个样本数据集（最好是验证集的子集），称为“校准数据集”，用来做校准。

- 在校准数据集上运行FP32推理。收集激活的直方图，并生成一组具有不同阈值的8位表示法，并选择具有最少KL散度的表示。

  KL散度是在参考分布（即FP32激活）和量化分布之间（即8位量化激活）之间。

TRT提供了IInt8EntropyCalibrator，该接口需要由客户端实现，以提供校准数据集和一些用于缓存校准结果的样板代码。

Entropy Calibration - pseudocode

```
Input: FP32 histogram H with 2048 bins: bin[ 0 ], …, bin[ 2047 ]

For i in range( 128 , 2048 ):
        reference_distribution_P = [ bin[ 0 ] , ..., bin[ i-1 ] ]                          // take first ‘ i ‘ bins from H
        outliers_count = sum( bin[ i ] , bin[ i+1 ] , … , bin[ 2047 ] )
        reference_distribution_P[ i-1 ] += outliers_count
        P /= sum(P)                                                                          // normalize distribution P
        candidate_distribution_Q = quantize [ bin[ 0 ], …, bin[ i-1 ] ] into 128 levels    // explained later
        expand candidate_distribution_Q to ‘ i ’ bins                                        // explained later
        Q /= sum(Q)                                                                          // normalize distribution Q
        divergence[ i ] = KL_divergence( reference_distribution_P,  candidate_distribution_Q)
End For

Find index ‘m’ for which divergence[ m ] is minimal

threshold = ( m + 0.5 ) * ( width of a bin )
```

上面就是一个循环，不断地构造P和Q，并计算相对熵，然后找到最小（截断长度为m）的相对熵，此时表示Q能比较好地拟合P分布了。而阈值就等于（m + 0.5）*一个bin的长度。

要做INT8量化，需要：

- **原来的未量化的模型**
- **一个校准数据集**
- **进行量化过程的校准器**

校准过程我们是不用参与的，全部都由TensorRT内部完成，但是，需要告诉校准器如何获取一个batch的数据，也就是说，需要重写校准器类中的一些方法。

- 准备一个校准集，用于在转换过程中寻找使得转换后的激活值分布与原来的FP32类型的激活值分布差异最小的阈值;
- 写一个校准器类，该类需继承trt.IInt8EntropyCalibrator2父类，并重写get_batch_size, get_batch, read_calibration_cache, write_calibration_cache这几个方法。
- 使用时，需额外指定cache_file，该参数是校准集cache文件的路径，会在校准过程中生成，方便下一次校准时快速提取。

Check if Your GPU Supports FP16/INT8

1. check your GPU Compute Capability

visit https://developer.nvidia.com/cuda-gpus#compute and check your GPU compute capability.

For example, GTX1080 is 6.1, Tesla T4 is 7.5.

2. check the hardware-precision-matrix

visit https://docs.nvidia.com/deeplearning/tensorrt/support-matrix/index.html#hardware-precision-matrix and check the matrix.

For example, compute capability 6.1 supports FP32 and INT8. 7.5 supports FP32, FP16, INT8, FP16 tensor core, etc.