香港中文大學
The Chinese University of Hong Kong

# CMSC5743
# L05: Quantization

**Bei Yu**

(Latest update: October 12, 2020)

Fall 2020

# Overview

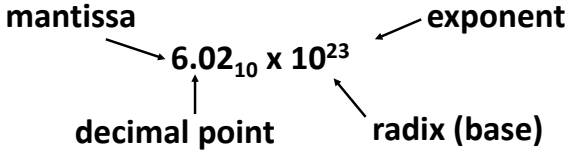
# Overview

## These slides contain/adapt materials developed by

- Hardware for Machine Learning, Shao Spring 2020 @ UCB
- 8-bit Inference with TensorRT
- Junru Wu et al. (2018). “Deep $k$-Means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions”. In: *Proc. ICML*
- Shijin Zhang et al. (2016). “Cambricon-x: An accelerator for sparse neural networks”. In: *Proc. MICRO*. IEEE, pp. 1–12
- Jorge Albericio et al. (2016). “Cnvlutin: Ineffectual-neuron-free deep neural network computing”. In: *ACM SIGARCH Computer Architecture News* 44.3, pp. 1–13

# Scientific Notation

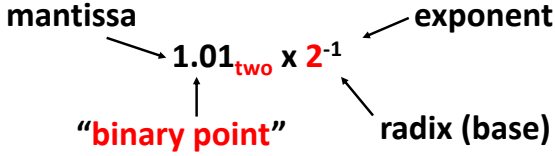
## Decimal representation




- Normalized form: no leadings 0s (exactly one digit to left of decimal point)
- Alternatives to representing 1/1,000,000,000
  - Normalized: $1.0 \times 10^{-9}$
  - Not normalized: $0.1 \times 10^{-8}$,$10.0 \times 10^{-10}$

# Scientific Notation

## Binary representation




- Computer arithmetic that supports it called floating point, because it represents numbers where the binary point is not fixed, as it is for integers

# Normalized Form

- Floating Point Numbers can have multiple forms, e.g.

$$
\begin{aligned}
0.232 \times 10^4 &= 2.32 \times 10^3 \\
&= 23.2 \times 10^2 \\
&= 2320. \times 10^0 \\
&= 232000. \times 10^{-2}
\end{aligned}
$$

- It is desirable for each number to have a unique representation => Normalized Form
- We normalize Mantissa's in the Range $[1..R)$, where R is the Base, e.g.:
  - $[1..2)$ for BINARY
  - $[1..10)$ for DECIMAL

# Floating-Point Representation

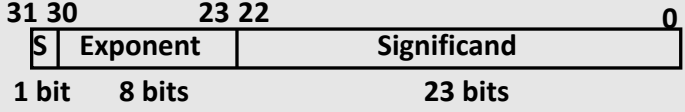
- Normal format: $+1.xxx\ldots x_{two} * 2^{yyy\ldots y_{two}}$


- **S represents Sign**
- **Exponent represents y's**
- **Significand represents x's**
- **Represent numbers as small as $2.0 \times 10^{-38}$ to as large as $2.0 \times 10^{38}$**

# Floating-Point Representation (FP32)

- IEEE 754 Floating Point Standard
  - Called **Biased Notation**, where bias is number subtracted to get real number
  - IEEE 754 uses bias of 127 for single prec.
  - Subtract 127 from Exponent field to get actual value for exponent
  - 1023 is bias for double precision

- Summary (single precision, or fp32):

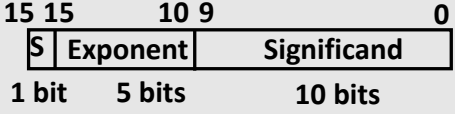



- $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$

# Floating-Point Representation (FP16)

- IEEE 754 Floating Point Standard
  - Called **Biased Notation**, where bias is number subtracted to get real number
  - IEEE 754 uses bias of 15 for half prec.
  - Subtract 15 from Exponent field to get actual value for exponent

- Summary (half precision, or fp15):




- $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-15)}$

**Question:**

What is the IEEE single precision number 40C0 $0000_{16}$ in decimal?

**Question:**

What is -0.5$_{10}$ in IEEE single precision binary floating point format?

# Fixed-Point Arithmetic

- Integers with a binary point and a bias
  - “slope and bias”: y = s*x + z
  - Qm.n: m (# of integer bits) n (# of fractional bits)

s = 1, z = 0

| 2^2 | 2^1 | 2^0 | Val |
|---|---|---|---|
| 0 | 0 | 0 | **0** |
| 0 | 0 | 1 | **1** |
| 0 | 1 | 0 | **2** |
| 0 | 1 | 1 | **3** |
| 1 | 0 | 0 | **4** |
| 1 | 0 | 1 | **5** |
| 1 | 1 | 0 | **6** |
| 1 | 1 | 1 | **7** |

s = 1/4, z = 0

| 2^0 | 2^-1 | 2^-2 | Val |
|---|---|---|---|
| 0 | 0 | 0 | **0** |
| 0 | 0 | 1 | **1/4** |
| 0 | 1 | 0 | **2/4** |
| 0 | 1 | 1 | **3/4** |
| 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | **5/4** |
| 1 | 1 | 0 | **6/4** |
| 1 | 1 | 1 | **7/4** |

s = 4, z = 0

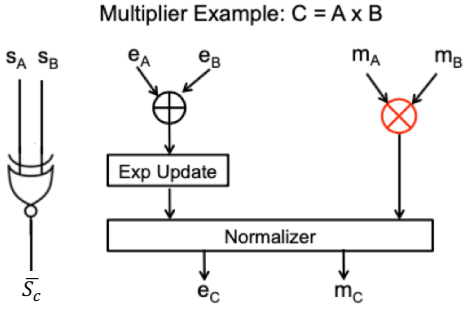
| 2^4 | 2^3 | 2^2 | Val |
|---|---|---|---|
| 0 | 0 | 0 | **0** |
| 0 | 0 | 1 | **4** |
| 0 | 1 | 0 | **8** |
| 0 | 1 | 1 | **12** |
| 1 | 0 | 0 | **16** |
| 1 | 0 | 1 | **20** |
| 1 | 1 | 0 | **24** |
| 1 | 1 | 1 | **28** |

s = 1.5, z =10

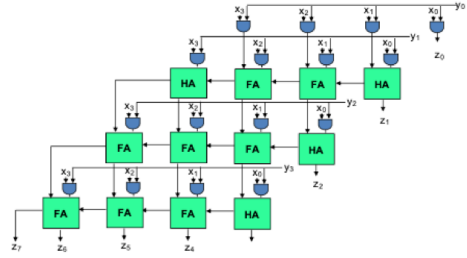
| 2^2 | 2^1 | 2^0 | Val |
|---|---|---|---|
| 0 | 0 | 0 | **1.5*0 +10** |
| 0 | 0 | 1 | **1.5*1 +10** |
| 0 | 1 | 0 | **1.5*2 +10** |
| 0 | 1 | 1 | **1.5*3 +10** |
| 1 | 0 | 0 | **1.5*4 +10** |
| 1 | 0 | 1 | **1.5*5 +10** |
| 1 | 1 | 0 | **1.5*6 +10** |
| 1 | 1 | 1 | **1.5*7 +10** |

# Hardware Implications

## Multipliers




Floating-point multiplier




Fixed-point multiplier

# Overview

# Greedy Layer-wise Quantization[1]

## Quantization flow

- For a fixed-point number, it representation is:

$$n = \sum_{i=0}^{bw-1} B_i \cdot 2^{-f_l} \cdot 2^i,$$

where $bw$ is the bit width and $f_l$ is the fractional length which is dynamic for different layers and feature map sets while static in one layer.

- Weight quantization: find the optimal $f_l$ for weights:

$$f_l = \arg\min_{f_l} \sum |W_{float} - W(bw, f_l)|,$$

where $W$ is a weight and $W(bw, f_l)$ represents the fixed-point format of $W$ under the given $bw$ and $f_l$.

[1] Jiantao Qiu et al. (2016). "Going deeper with embedded fpga platform for convolutional neural network". In: *Proc. FPGA*, pp. 26–35.

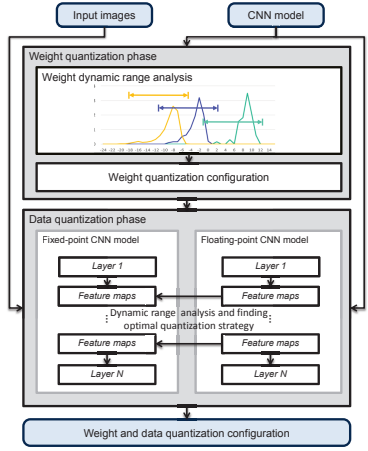
# Greedy Layer-wise Quantization

## Quantization flow

- Feature quantization: find the optimal $f_l$ for features:

$$f_l = \arg\min_{f_l} \sum |x^+_{float} - x^+(bw, f_l)|,$$

where $x^+$ represents the result of a layer when we denote the computation of a layer as $x^+ = A \cdot x$.



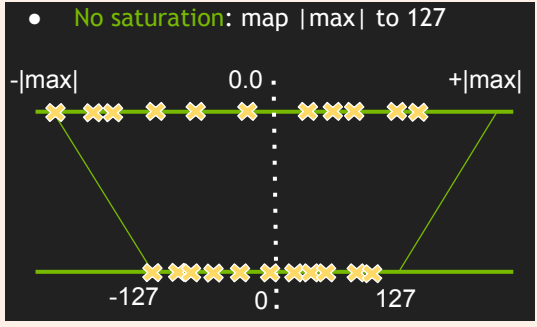
# Dynamic-Precision Data Quantization Results

| Network | VGG16 | | | | | | |
|---|---|---|---|---|---|---|---|
| Data Bits | Single-float | 16 | 16 | 8 | 8 | 8 | 8 |
| Weight Bits | Single-float | 16 | 8 | 8 | 8 | 8 | 8 or 4 |
| Data Precision | N/A | $2^{-2}$ | $2^{-2}$ | Impossible | $2^{-5}/2^{-1}$ | Dynamic | Dynamic |
| Weight Precision | N/A | $2^{-15}$ | $2^{-7}$ | Impossible | $2^{-7}$ | Dynamic | Dynamic |
| Top-1 Accuracy | 68.1% | 68.0% | 53.0% | Impossible | 28.2% | 66.6% | 67.0% |
| Top-5 Accuracy | 88.0% | 87.9% | 76.6% | Impossible | 49.7% | 87.4% | 87.6% |

| Network | CaffeNet | | | VGG16-SVD | | |
|---|---|---|---|---|---|---|
| Data Bits | Single-float | 16 | 8 | Single-float | 16 | 8 |
| Weight Bits | Single-float | 16 | 8 | Single-float | 16 | 8 or 4 |
| Data Precision | N/A | Dynamic | Dynamic | N/A | Dynamic | Dynamic |
| Weight Precision | N/A | Dynamic | Dynamic | N/A | Dynamic | Dynamic |
| Top-1 Accuracy | 53.9% | 53.9% | 53.0% | 68.0% | 64.6% | 64.1% |
| Top-5 Accuracy | 77.7% | 77.1% | 76.6% | 88.0% | 86.7% | 86.3% |

# Industrial Implementations – Nvidia TensorRT

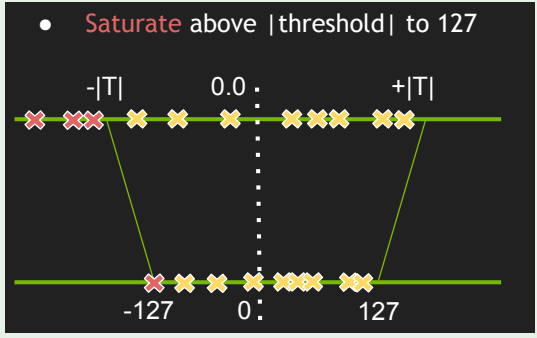
## No Saturation Quantization – INT8 Inference




- Map the maximum value to 127, with unifrom step length.
- Suffer from outliers.

# Industrial Implementations – Nvidia TensorRT

## Saturation Quantization – INT8 Inference




- Set a threshold as the maxiumum value.
- Divide the value domain into 2048 groups.
- Traverse all the possible thresholds to find the best one with minimum KL divergence.

# Industrial Implementations – Nvidia TensorRT

## Relative Entropy of two encodings

- INT8 model encodes the same information as the original FP32 model.
- Minimize the loss of information.
- Loss of information is measured by Kullback-Leibler divergence (*a.k.a.*, relative entropy or information divergence).
  - $P$, $Q$ - two discrete probability distributions:

$$D_{KL}(P\|Q) = \sum_{i=1}^{N} P(x_i) \log \frac{P(x_i)}{Q(x_i)}$$

- Intuition: KL divergence measures the amount of information lost when approximating a given encoding.

# Overview

# Straight-Through Estimator (STE)[2]

- A straight-through estimator is a way of estimating gradients for a threshold operation in a neural network.
- The threshold could be as simple as the following function:

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & \text{else} \end{cases}$$

- The derivate of this threshold function will be 0 and during back-propagation, the network will learn anything since it gets 0 gradients and the weights won't get updated.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville (2013). "Estimating or propagating gradients through stochastic neurons for conditional computation". In: *arXiv preprint arXiv:1308.3432*.

# PArameterized Clipping acTivation Function (PACT)[3]

- A new activation quantization scheme in which the activation function has a parameterized clipping level $\alpha$.
- The clipping level is dynamically adjusted vias stochastic gradient descent (SGD)-based training with the goal of minimizing the quantization error.
- In PACT, the convolutional ReLU activation function in CNN is replaced with:

$$f(x) = 0.5\left(|x| - |x-\alpha| + \alpha\right) = \begin{cases} 0, & x \in (\infty, 0) \\ x, & x \in [0, \alpha) \\ \alpha, & x \in [\alpha, +\infty) \end{cases}$$

where $\alpha$ limits the dynamic range of activation to $[0, \alpha]$.

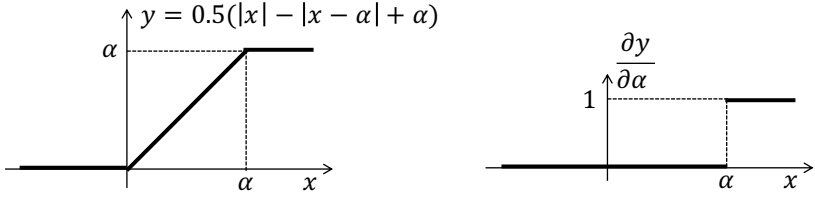
[3] Jungwook Choi et al. (2019). "Accurate and efficient 2-bit quantized neural networks". In: *Proceedings of Machine Learning and Systems* 1.

# PArameterized Clipping acTivation Function (PACT)

- The truncated activation output is the linearly quantized to $k$-bits for the dot-product computations:

$$y_q = \text{round}\,(y \cdot \frac{2^k - 1}{\alpha}) \cdot \frac{\alpha}{2^k - 1}$$

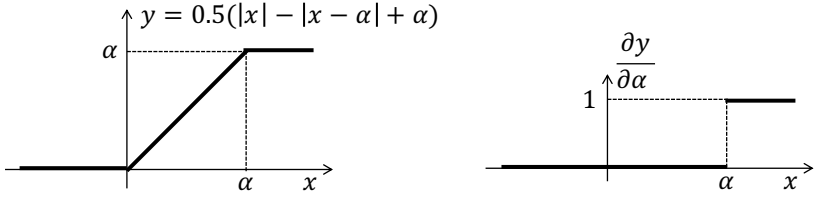
- With this new activation function, $\alpha$ is a variable in the loss function, whose value can be optimized during training.
- For back-propagation, gradient $\frac{\partial y_q}{\partial \alpha}$ can be computed using STE to estimate $\frac{\partial y_q}{\partial y}$ as 1.




PACT activation function and its gradient.

# Better Gradients

## Is Straight-Through Estimator (STE) the best?




PACT activation function and its gradient.

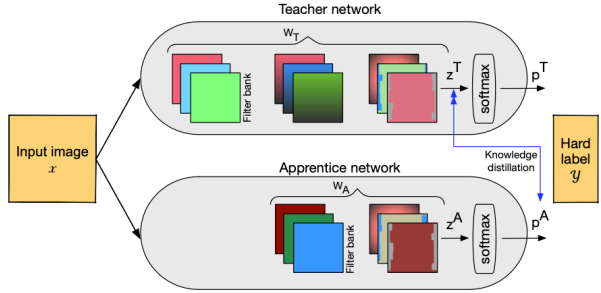
- Gradient mismatch: the gradients of the weights are not generated using the value of weights, but rather its quantized value.
- Poor gradient: STE fails at investigating better gradients for quantization training.

# Knowledge Distillation-Based Quantization[4]

- Knowledge distillation trains a student model under the supervision of a well trained teacher model.
- Regard the pre-trained FP32 model as the teacher model and the quantized models as the student models.

$$\mathcal{L}(x; W_T, W_A) = \alpha\mathcal{H}(y, p^T) + \beta\mathcal{H}(y, p^A) + \gamma\mathcal{H}(z^T, p^A) \quad (1)$$

where, $W_T$ and $W_A$ are the parameters of the teacher and the student (apprentice) network, respectively, $y$ is the ground truth, $\mathcal{H}(\cdot)$ denotes a loss function and, $\alpha$, $\beta$ and $\gamma$ are weighting factors to prioritize the output of a certain loss function over the other.




[4] Asit Mishra and Debbie Marr (2017). "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy". In: *arXiv preprint arXiv:1711.05852*.

# Overview

# Further Reading List


- Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy (2016). “Fixed point quantization of deep convolutional networks”. In: *Proc. ICML*, pp. 2849–2858
- Soroosh Khoram and Jing Li (2018). “Adaptive quantization of neural networks”. In: *Proc. ICLR*
- Jan Achterhold et al. (2018). “Variational network quantization”. In: *Proc. ICLR*
- Antonio Polino, Razvan Pascanu, and Dan Alistarh (2018). “Model compression via distillation and quantization”. In: *arXiv preprint arXiv:1802.05668*
- Yue Yu, Jiaxiang Wu, and Longbo Huang (2019). “Double quantization for communication-efficient distributed optimization”. In: *Proc. NIPS*, pp. 4438–4449
- Markus Nagel et al. (2019). “Data-free quantization through weight equalization and bias correction”. In: *Proc. ICCV*, pp. 1325–1334