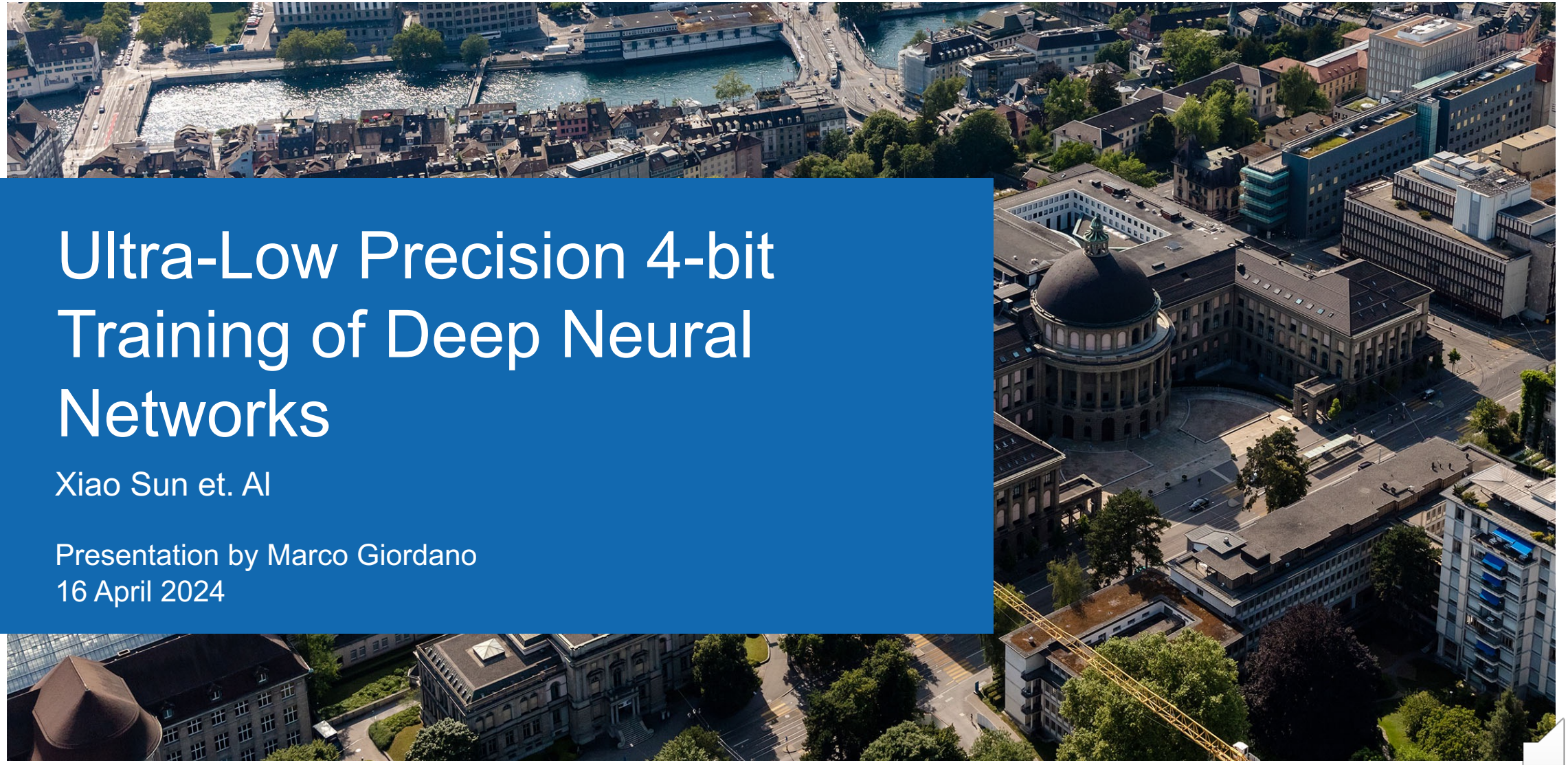
ETH zürich

PROJECT BASED LEARNING



Ultra-Low Precision 4-bit Training of Deep Neural Networks

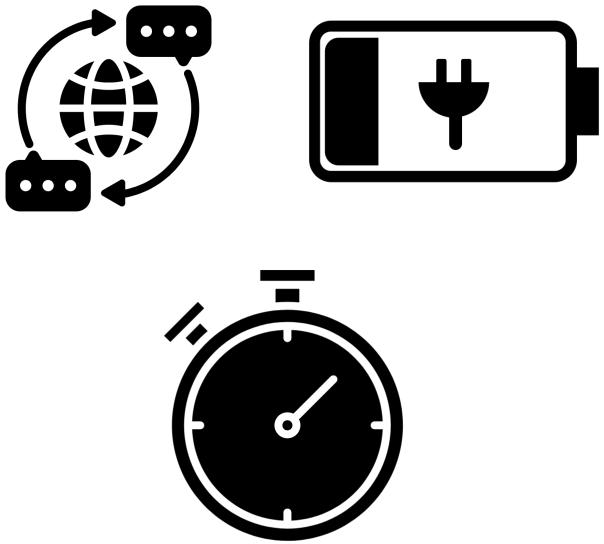
Xiao Sun et. Al

Presentation by Marco Giordano

16 April 2024

Introduction – ML on the Edge

Efficient ML:
- Mainly researched for **inference**
  - Extract information from the data where they are collected
    - Less communication
    - Lower energy requirements
    - Lower latency



Already **commonly used**:
- Face recognition
- Image enhancement
- Language translation
- Fitness algorithms
- Etc…

Introduction – ML on the Edge

Popular techniques:
- Quantization
  - low-precision representation
- Pruning
  - ”Cut” less impactful weights
- Distillation
  - Student/teacher networks

Quantization:
- Extremely widespread
- Works well on complex networks (Bert[1])
  - … On inference
- Up to binary networks (1bit weights)

Quadratic dependency throughput-precision

Can we use it **only for inferece?**
- Models growing in size and complexity
- Training ~2/3 of GEMM (General matrix multiply)
- Lower precision saves memory and training time

Does quantization work during training?

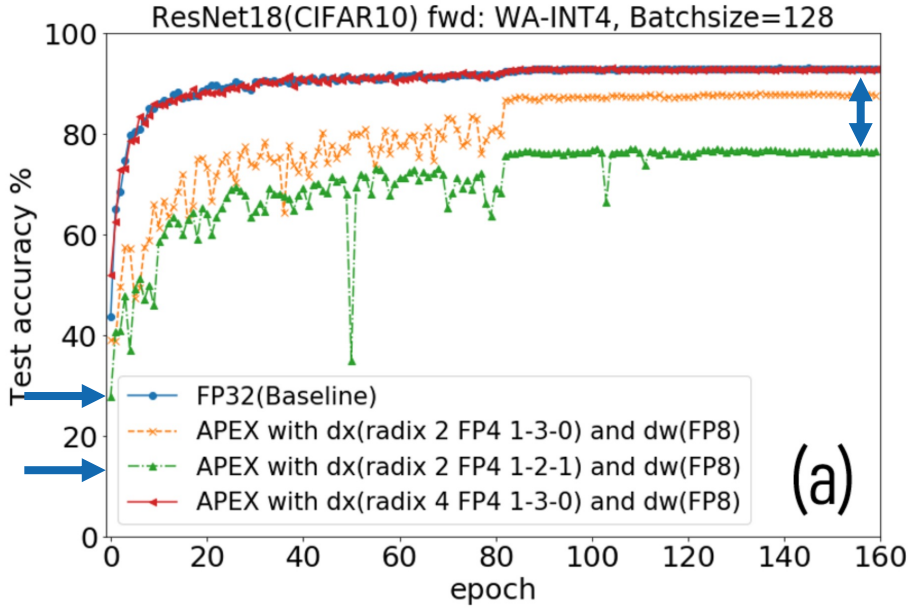
[1] Q8BERT: Quantized 8Bit BERT

Introduction – ML on the Edge

Answer:

It depends.

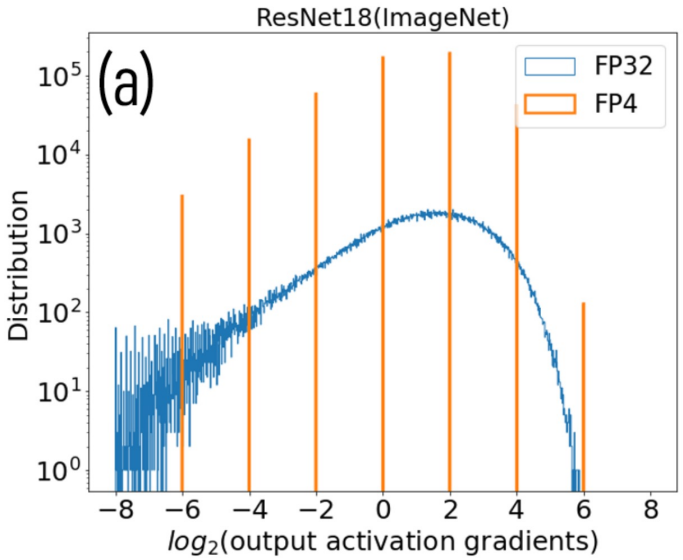
- FP16 (floating point 16bit) widely used for training
- FP8 shown in previous research
- FP4 exposes a lot of challenges



~20% acc. difference

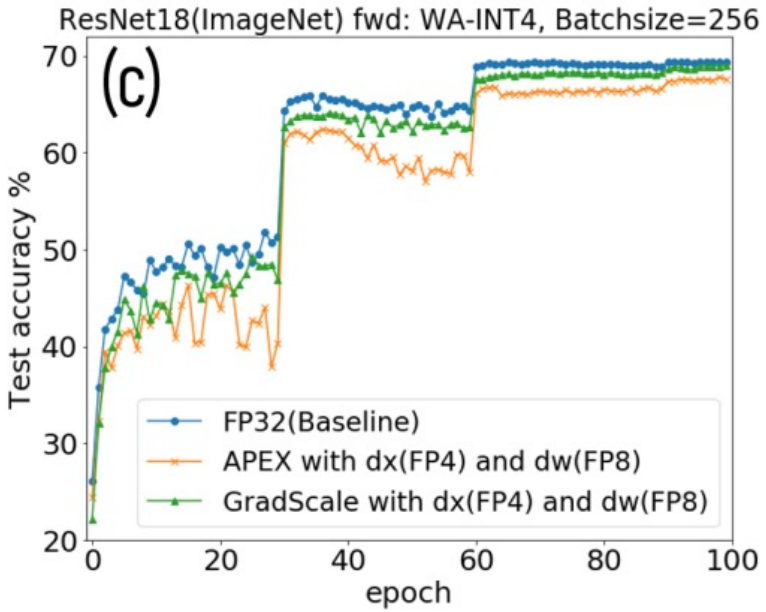
Main issues:

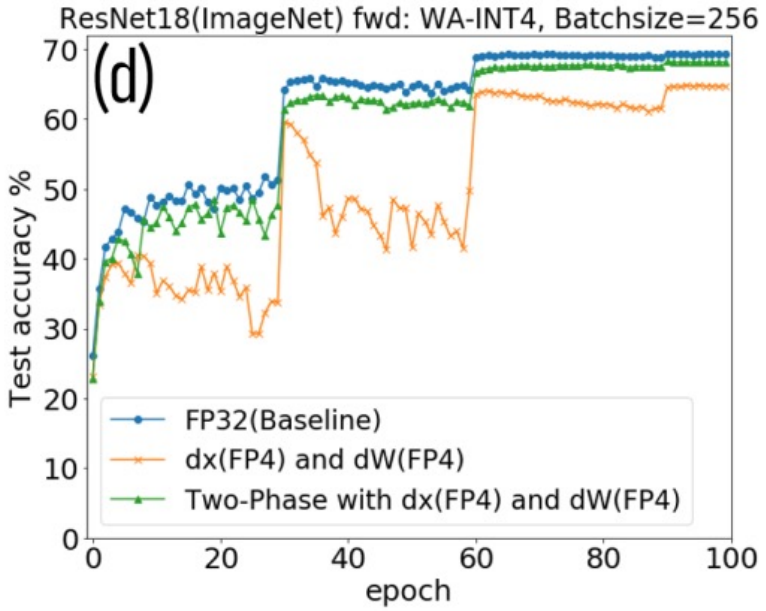
- Quantization error
- Precision
- Dynamic range



Related work

- **Banner et. Al**: 8 bit fixed point training
  - Limited dynamic range
  - Only a subset of 8bit ops in backward
- **Miyashita et. Al**: Log. 5-bit representation
  - Fixed precision, higher dynamic range
  - Limited performance on larger models
- **Wang et. Al, Sun et. Al**: float 8 bit, <1% acc
  - Hybrid fp8 for weights, acts and grads
    - Selective precision
- **Micikevicius et. Al, Esser et. Al**:
  - Automatic Loss Scaling (APEX)



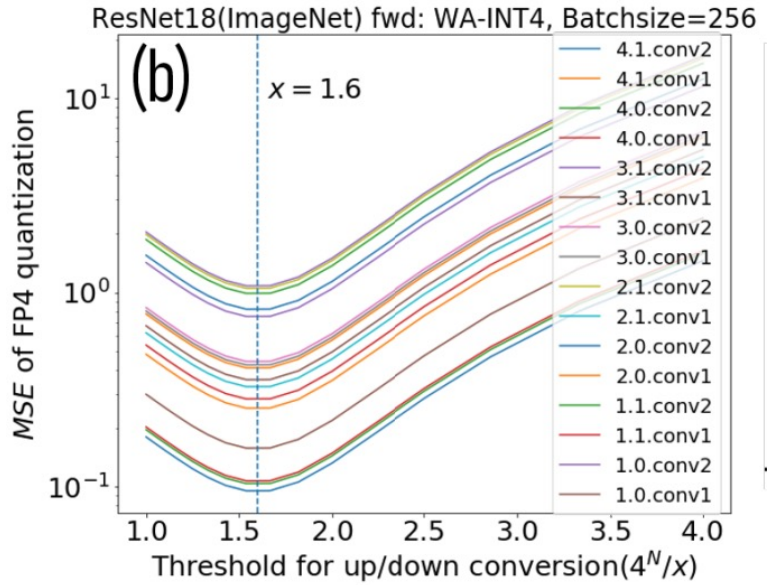


Banner et. Al: Scalable methods for 8-bit training of neural networks.
Miyashita et. Al: Convolutional neural networks using logarithmic data representation
Wang et. Al: Training deep neural networks with 8-bit floating point numbers.
Sun et. Al: Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks.
Micikevicius et. Al: Mixed precision training
Esser et. Al: Learned step size quantization.

Contributions

1. New radix-4 FP4 format: higher dynamic range for gradients
2. Per-layer trainable gradient scaling technique (GradScale)
3. Two phase quantisation technique to minimize quant. Error
4. A deeped understanding of quant. Bias and interplay with BN
5. Hybrid approach with fp4 and fp8
6. Extensive testing on a series of models and tasks

Radix-4 FP4 Format



- Limited dynamic range major issue for low-precision training
- [sign,exponent,mantissa] = [1,3,0]
- logarithmic format (4^n) that spans a range of $\pm 4^3$ ($=\pm 2^6$)
  - Higher range, lower precision
- Mid-point: (($4^n + 4^{n-1}$)/2 = $4^n/1.6$) → $round(x) = \begin{cases} 4^{n-1} & x \leq 4^n/1.6 \\ 4^n & x > 4^n/1.6 \end{cases}$
- Easy and fast algorithm for conversion: →

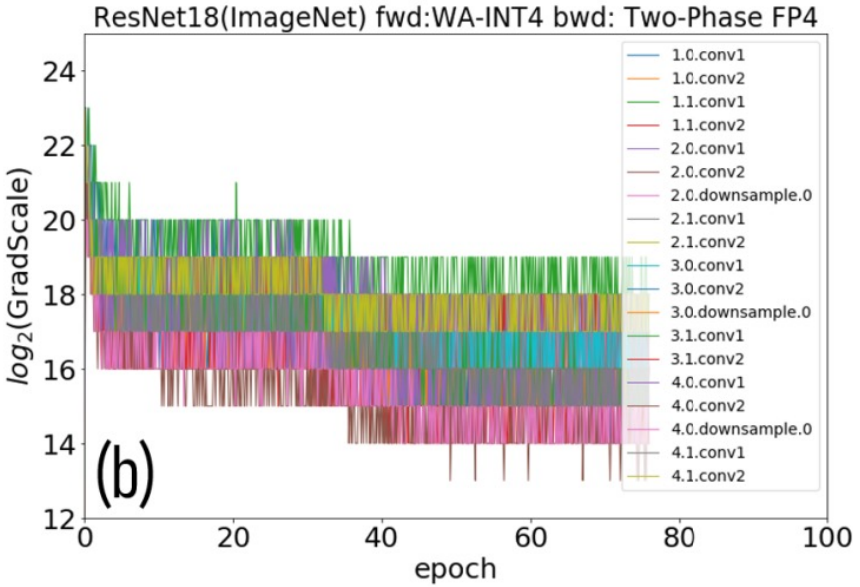
Radix-4 FP4 [1,3,0]
Nearest Rounding Method

Given an FP32 number $x \in \mathbb{R}$
$x = x \times multiplier (= 1.6)$
$ebit = \text{FLOOR}(log_4(|x|))$
if $ebit < -3$: $x = 0$
if $ebit \geq 3$: $x = sign(x) \times 64.0$
else : $x = sign(x) \times 4^{ebit}$

GradScale: Trainable Layer-wise Gradient Scaling

- FP16 and FP8 already use <u>global</u> gradient scaling
  - Loss is scaled -> Applied to all gradients

- Authors provide <u>per-layer</u> gradient scaling
  - Scaling as a learnable parameter
  - Increase or decrease scaling based on unfl/ovfl



GradScale algorithm

In GEMM units:
For the layer L:
For each worker i:

$$g_{scale_{i,L}} = \begin{cases} 0, max(|grad_i|) \in (MAX/2, MAX] \\ 1, max(|grad_i|) \in [0, MAX/2] \\ -1, max(|grad_i|) \in (MAX, \infty] \end{cases}$$

In the Optimizer:

$$g_{scale_L} = \sum_i^N g_{scale_{i,L}}$$

$$scale_L = \begin{cases} scale_L \times 1.0, & g_{scale_L} == 0 \\ scale_L \times 2.0, & g_{scale_L} > 0 \\ scale_L \times 0.5, & g_{scale_L} < 0 \end{cases}$$

(a)

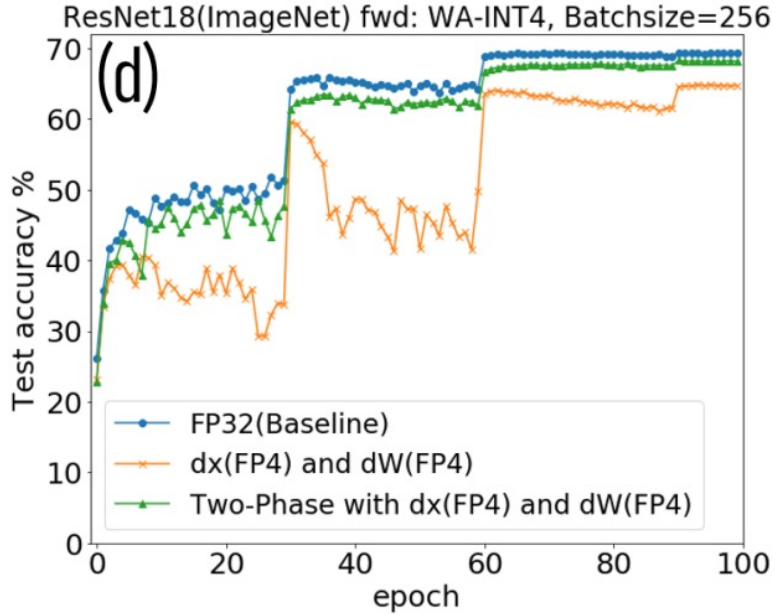
Two-Phase Rounding

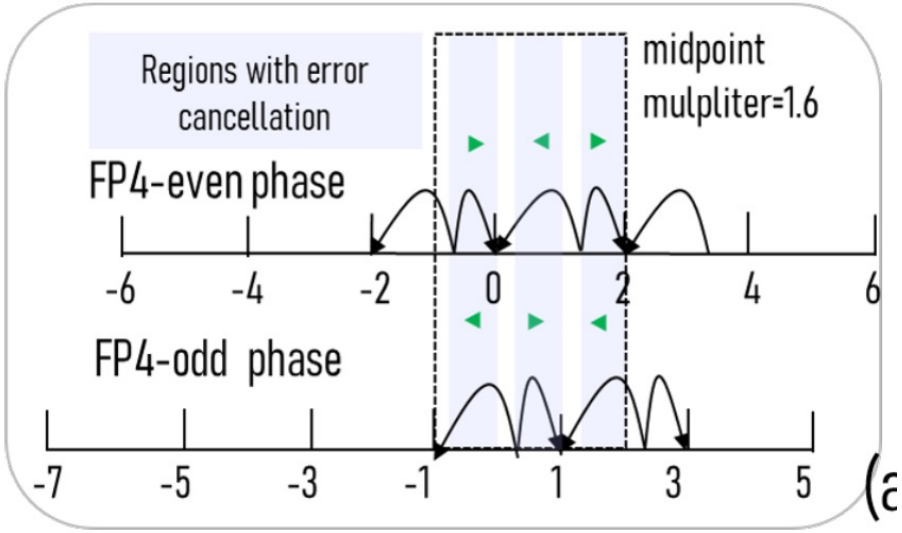
- Introduce FP4 with even and odd vaulus:
  - 2^{even} and 2^{odd}

- Intuition: same gradient used twice:
  - dL/dx = dL/dy.W^T
  - dL/dW = x.dL/dy

➢ Look at the same gradient from ”two sides”
  ➢ Different quantisation for dL/dy retain more information

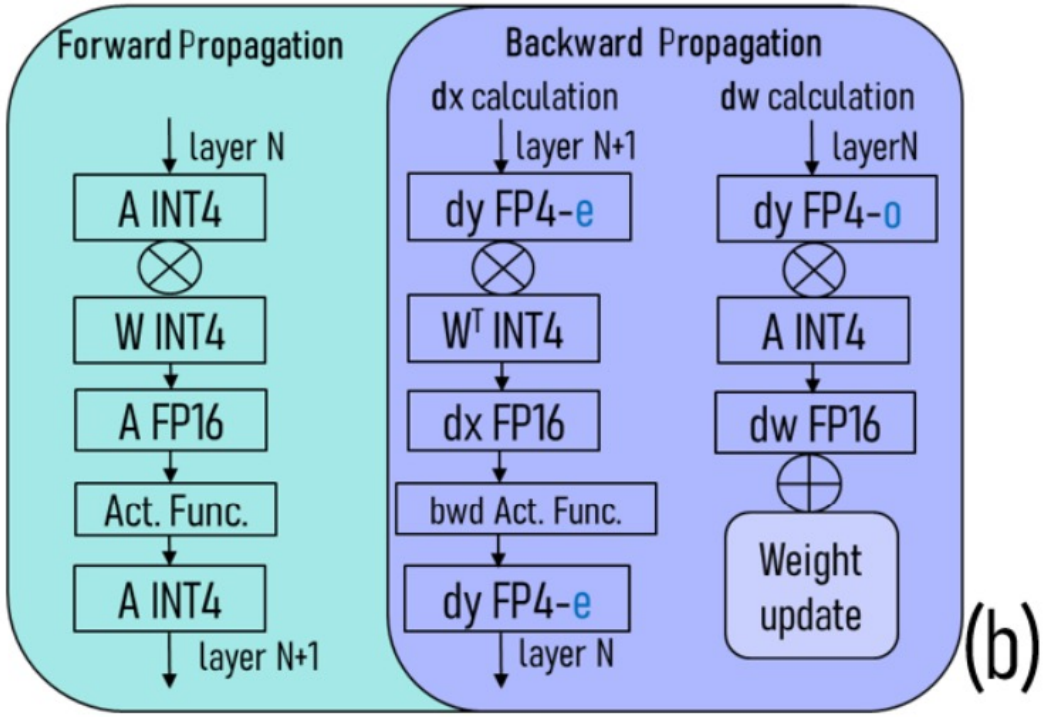
- Errors from two phases should cancel each other





Engine summary

- Requires a mixed-precision INT4-FP4 GEMM engine!
- Authors claim MAC engines with INT4×FP4 capabilities are >7× more **area** and **power efficient**



Internal covariate shift

- Looking at expected value of the gradient:

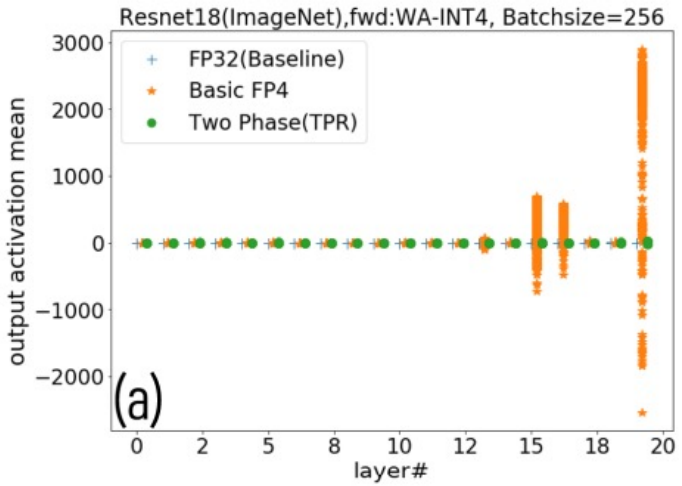
$$E[dL/dw] = E[x \cdot (dL/dy + \mathrm{q_{error}})] = E[x \cdot dL/dy] + E[x \cdot \mathrm{q_{error}}]$$

- Input activations are usually non negative (ReLU)
- q_{error} tends to be non-zero

➢ Introduction of non-zero bias

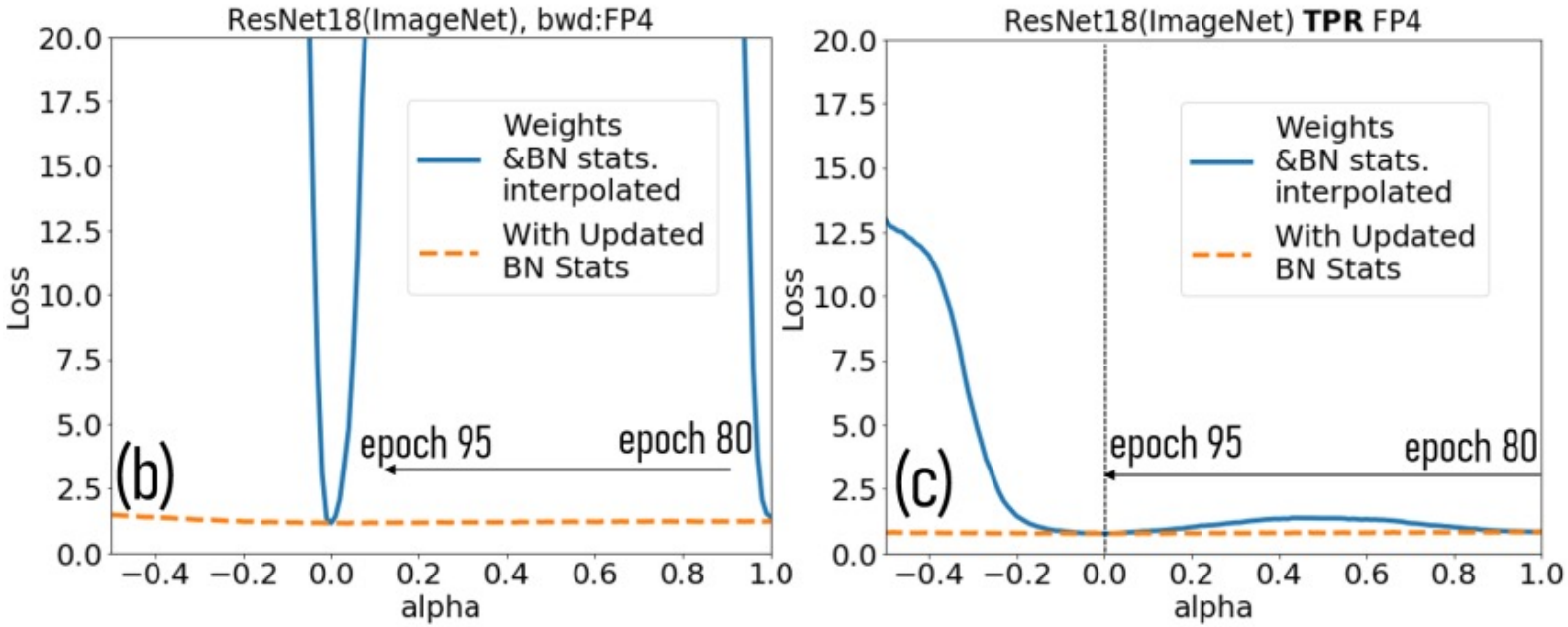
  - Propagated to weights and activation
  - Leads to Internal Covariate Shift (ICS)

- Santurkar et. al. shows that batch normalisation adjusts mean and std dev to account for ICS



Internal covariate shift

- Interpolated 1D space of all parameters (including μ_B & σ_B)
- Updating μ_B & σ_B at each iteration leads to a very smooth loss landscape
- TPR helps with smoothing out curve

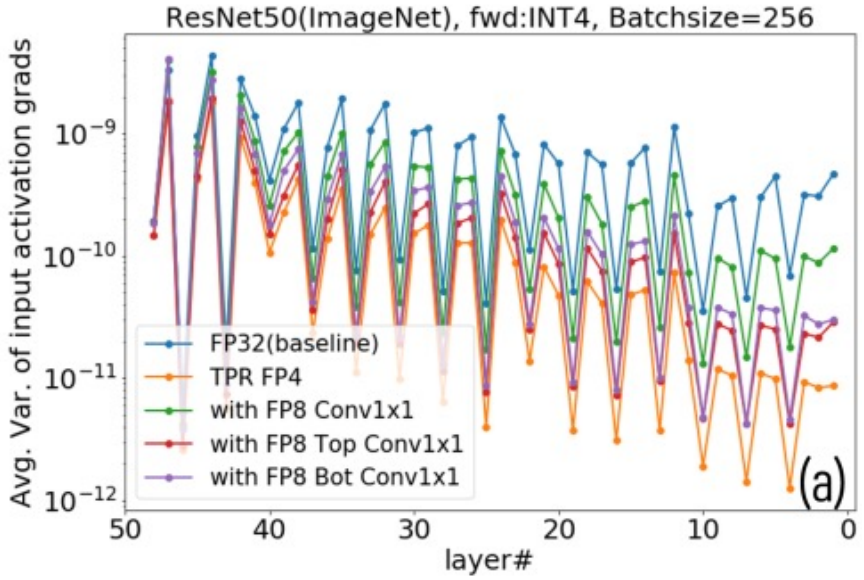


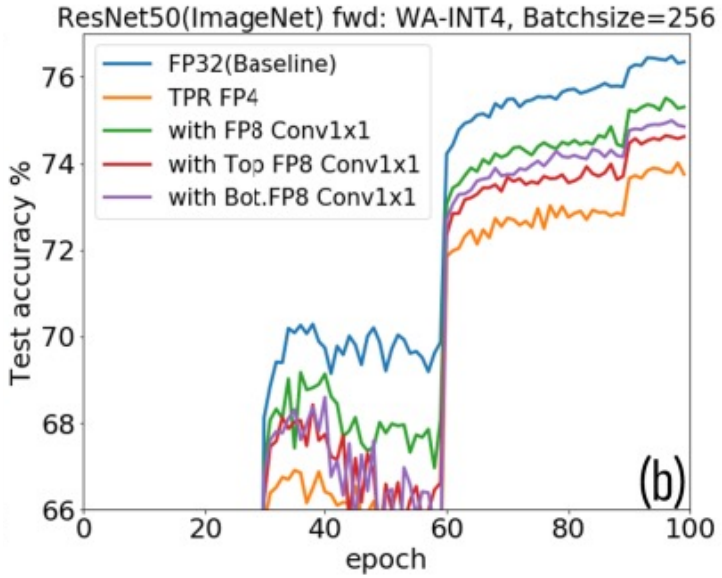
Selective FP8

- Higher grads variance is of 1x1 conv
  - When conv. removed from graph variance restored

- Selective use of FP8 for Conv. 1x1
- Accuracy increase

- Bottom (all) 1x1 layers to FP8 accuracy improves by 1.0% (1.5%)
- Still 5.2× (4.0×) training acceleration





Results – CIFAR10

- Shallower models without 1x1 convolutional layers achieve < ~0.5%, deeper models slightly > 1% accuracy losses
- With FP8 training accuracy loss ~0.5%

Table 1: CIFAR10 test accuracies using 4-bit training

FWD	FP32	INT4	INT4	INT4	INT4	INT4
BWD	FP32	FP32	TPR FP4	dx(FP4) dW(FP8)	TPR FP4 bot 1x1(FP8)	TPR FP4 1x1(FP8)
VGG16	92.07	91.81	**91.47**	91.81	-	-
GoogLeNet	94.68	94.50	**94.17**	94.12	-	-
ResNet18	93.01	92.97	**92.74**	92.76	-	-
ResNet50	94.70	94.34	93.42	93.77	94.03	**94.36**
ResNet101	95.04	94.76	93.79	94.34	94.19	**94.43**
DenseNet121	95.06	95.08	**94.71**	94.82	-	94.91
MobileNetV2	93.00	92.97	91.80	**92.39**	-	-

Results - ImageNet

- ImageNet lower performance because of less parameter redundancy
- full TPR FP4 ~1% accuracy loss AlexNet and ResNet18
- Larger models 2.5% accuracy loss on larger models (full FP4)
- Can be recovered to ~1% if FP8 used for 1x1 conv

Table 2: ImageNet test accuracies using 4-bit training

FWD	FP32	INT4	INT4	INT4	INT4	INT4
BWD	FP32	FP32	TPR FP4	dx(FP4) dW(FP8)	TPR FP4 bot1x1(FP8)	TPR FP4 1x1(FP8)
Alexnet	57.56	57.51	**56.38**	57.11	-	-
ResNet18	69.40	69.43	**68.27**	68.99	-	-
ResNet50	76.48	75.76	74.01	74.92	74.99	**75.51**
MobileNetV2	71.85	69.77	68.85	69.65	-	-

Results - NLP

- PTB dataset: negligible loss in perplexity
- ~2% loss in BLEU for transformer-based Machine Translation En-De task
- Word error rate on SWB300 < 0.5%

*FP8 for weight gradients

Table 3: NLP&speech accuracies w/4-bit training

FWD	FP32	INT4	INT4
BWD	FP32	TPR FP4	dx(FP4) dW(FP8)
2-layer-LSTM (PTB)Test ppl.	83.3	**85.3**	83.8
4-layer-biLSTM (SWB300)WER	9.9	11.3	**10.4**
Transformer-base (WMT En-De)BLEU	27.5	25.4	25.9

Performance on Hardware

- Derived from research on hardware accelerator design
- Consider 4-way INT4×FP4 MAC Unit
  - Consumes 55% area wrt FP16
  - Provides 4x the throughput
  - ➢ Compute density improvement of 7.3x
- Argue that FP8 computation would be small number
  - But still would have to be accelerated…

Estimated training performance vs. FP16

FWD	BWD	Performance
FP16	FP16	1.0
FP8	FP8	2.0
INT4	FP16	1.4
INT4	FP8	2.7
INT4	FP4	7.3
INT4	dx(FP4)dw(FP8)	3.9
INT4	FP4 all 1x1(FP8)	4.0
INT4	FP4 bot1x1(FP8)	5.2

(c)

Conclusions

- Successful 4-bit training on various deep learning benchmarks with minimal accuracy losses.
- Introduced Radix-4 FP4 format for quantized training
- Introduced two-Phase rounding (TPR) to cope with ICS
- Introduced Gradient Scaling (GradScale) techniques maximizing range and representation for gradients per layer
- Analysis of key factors impacting accuracy in 4-bit training systems
- Provided figures of efficiency gains on specialised hardware

Future work

- Bridge the accuracy gap even further (especially on big deep models)
- Evaluate the system on even more tasks and dataset
- Evaluate the system on a real FP4 platform

Limitations

- Methodology still has to resort to FP8 training for non-negligible accuracies losses on more complex models
- Optimistic performance gains (often memory bottlenecks/data movement)
- Lack of hardware implementation and real world figures

Marco Giordano
PhD student
mgiordano@pbl.ee.ethz.ch

ETH Zurich
Project-Based Learning
D97.5
Gloriastrasse 35
8092 Zurich, Switzerland

www.pbl.ee.ethz.ch

Thank you for your attention ☺