Mixed Precision Training

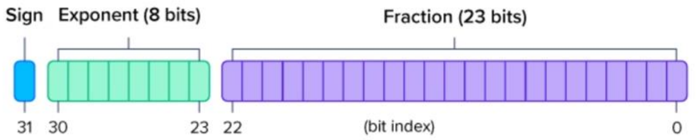
Sharan Narang, Gregory Diamos, Erich Elsen, Paulius Micikevicius, Jonah Alben, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, Hao Wu

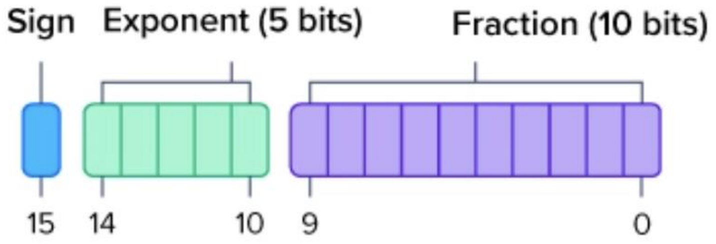
Background and Motivation

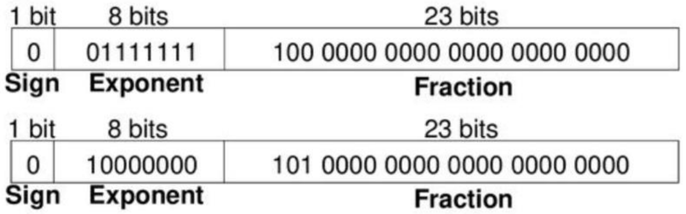
- Training with reduced precision
  - Reduces memory bandwidth pressure
  - Faster arithmetic
  - Reduces memory required for training
- But FP16 has a narrower dynamic range than FP32
  - May cause underflow/overflow and other arithmetic issues

VV Fast Refresher on IEEE FP numbers

- Representation FP16/32
- Denormalized numbers
  - The zero exponent is reserved for denormalized numbers
- FP Addition
  - Loss of precision while adding
  - For FP16, if operand exponents differ by more than 10 we lose all mantissa bits

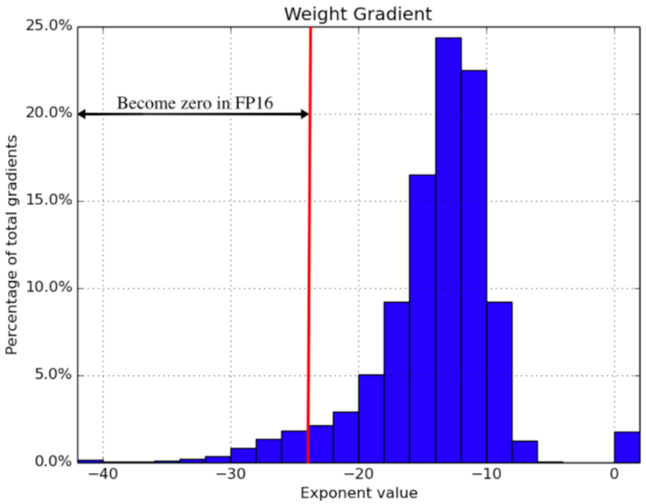






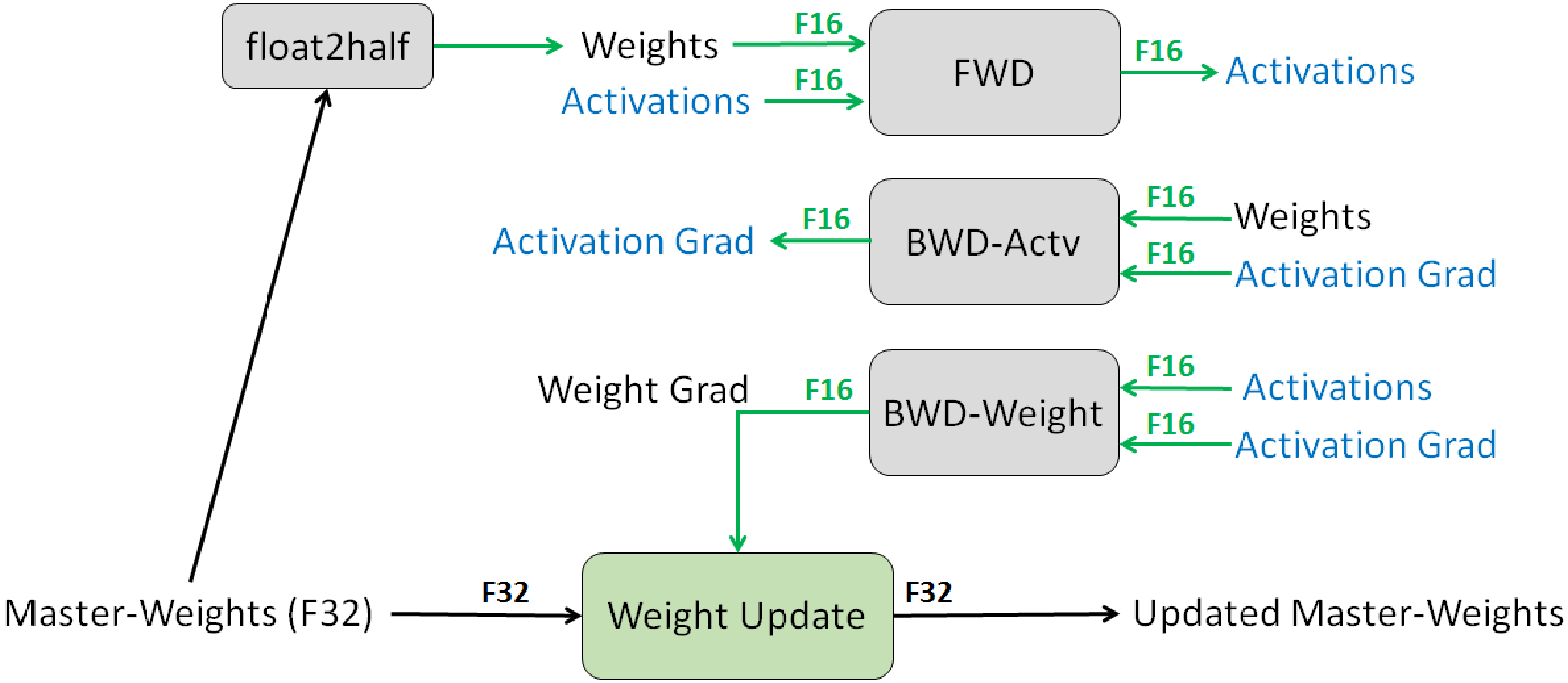
Idea 1: FP32 Master Copy Of Weights

- If model weights and gradients are in FP16, weight gradients may underflow



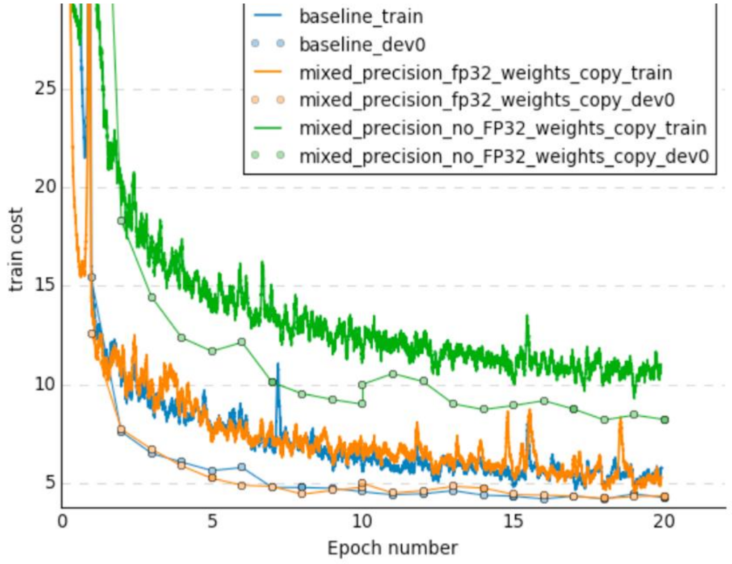
- Also, the ratio of weight value and weight update might be very large
  - Loss of precision while adding

Idea 1: FP32 Master Copy Of Weights



Idea 1: FP32 Master Copy Of Weights

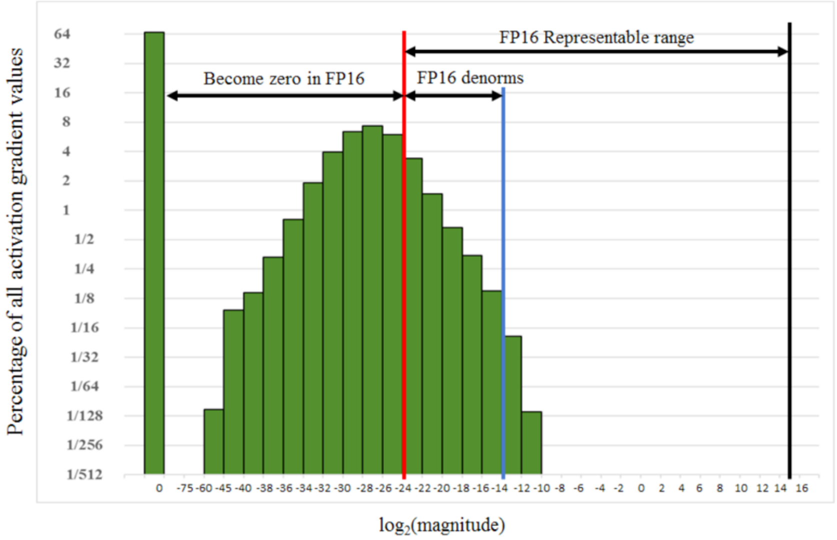
- Impact on
  - Performance
    - Using a FP32 master copy fixes training
  - Memory
    - Keeping a master copy of the weights requires more memory
    - For the models they tested the activation memory is the major bottleneck
    - May not be true if using techniques like activation checkpointing
    - May not be true for LLM training.



(a) Training and validation (dev0) curves for Mandarin speech recognition model

Idea 2: Loss Scaling

- Histogram of activation gradient values



- If cast to FP16, most gradient values will become 0!
- Scaling gradients during backpropagation prevents underflow
- The gradient is scaled before backpropagation begins and rescaled before updating weights

Idea 3: Arithmetic Precision

- Neural Net math
  - Vector dot-products
  - Reductions (BatchNorm, Softmax)
  - Point-wise operations (Non-linearities)
- Accumulating FP16 math into an FP16 value doesn’t work
- The paper proposes accumulating outputs in FP32 and saving them in FP16 format

Results

- Configuration
  - Baseline: Weights, activations, gradients, and arithmetic in FP32
  - Mixed Precision Training (MPT)
- Tasks
  - Vision: Classification, Detection
  - Language: Machine Translation, Language modeling
  - Speech recognition
  - Generative Modeling

Vision

Table 1: ILSVRC12 classification top-1 accuracy.

Model	Baseline	Mixed Precision	Reference
AlexNet	56.77%	56.93%	(Krizhevsky et al., 2012)
VGG-D	65.40%	65.43%	(Simonyan and Zisserman, 2014)
GoogLeNet (Inception v1)	68.33%	68.43%	(Szegedy et al., 2015)
Inception v2	70.03%	70.02%	(Ioffe and Szegedy, 2015)
Inception v3	73.85%	74.13%	(Szegedy et al., 2016)
Resnet50	75.92%	76.04%	(He et al., 2016b)

Model	Baseline	MP without loss-scale	MP with loss-scale
Faster R-CNN	69.1%	68.6%	69.7%
Multibox SSD	76.9%	diverges	77.1%

Language

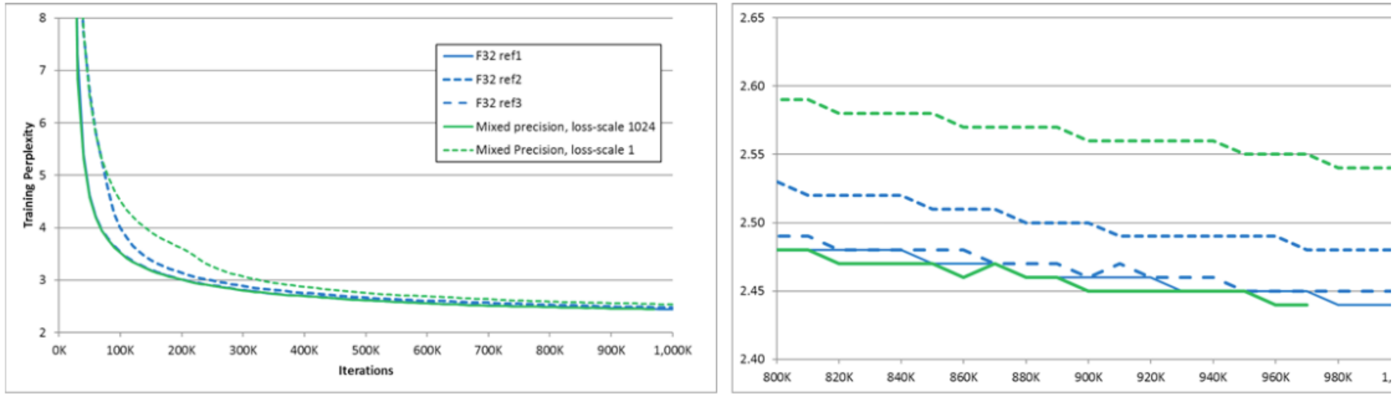
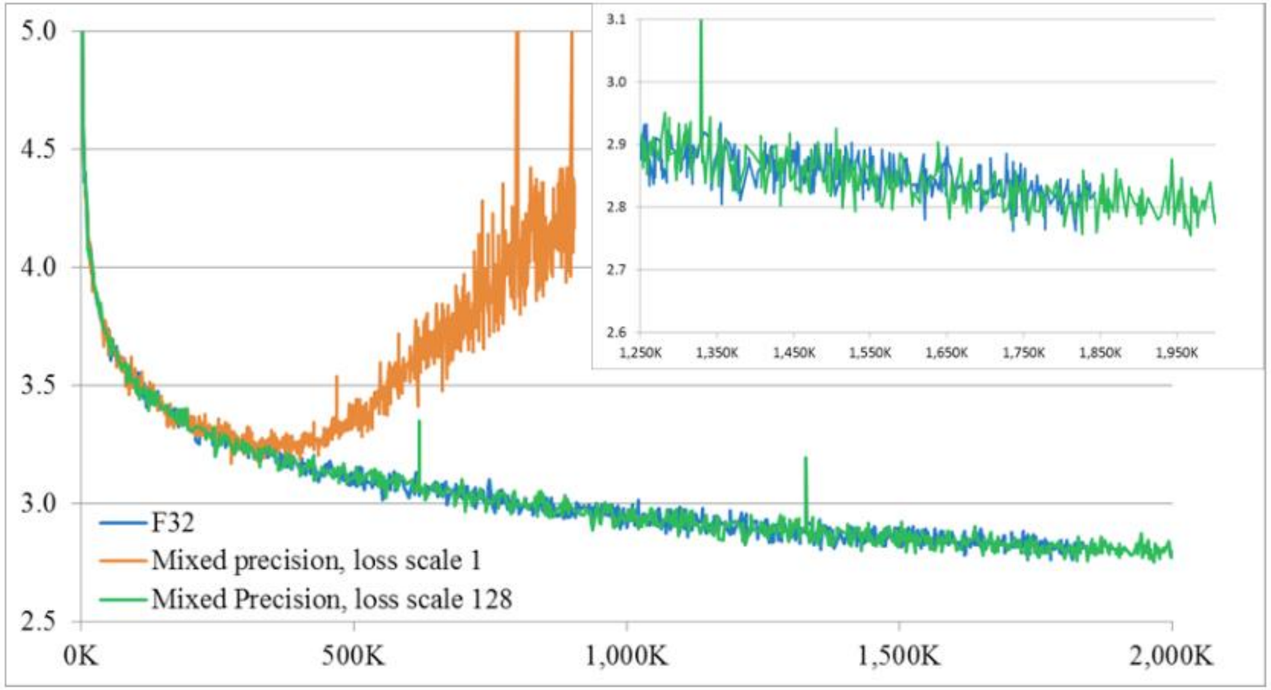


Figure 4: English to French translation network training perplexity, 3x1024 LSTM model v
attention. Ref1, ref2 and ref3 represent three different FP32 training runs.



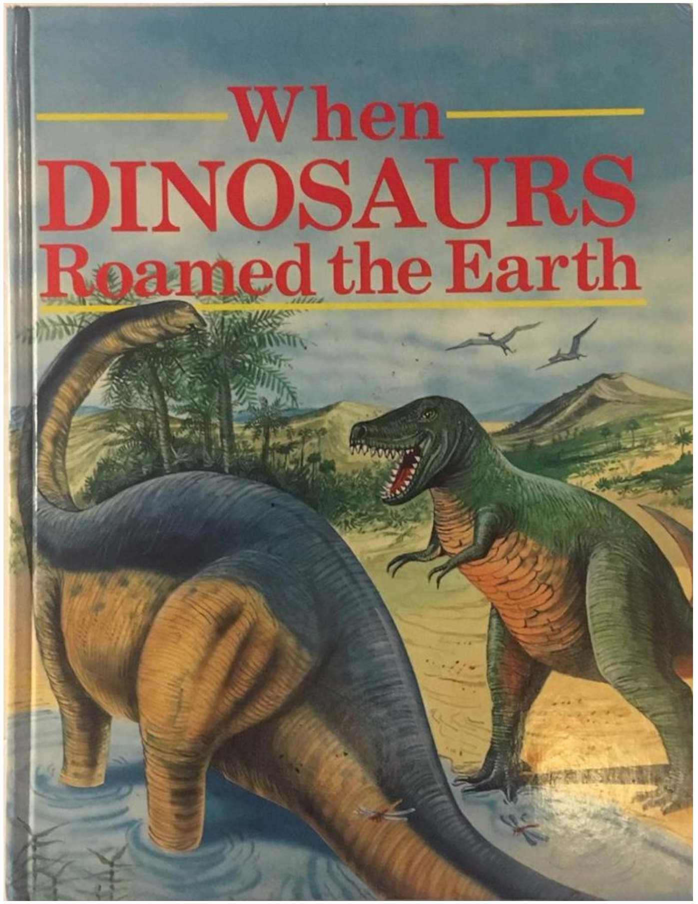
Speech + Generative Modeling

Table 3: Character Error Rate (CER) using mixed precision training for speech recognition. English results are reported on the WSJ '92 test set. Mandarin results are reported on our internal test set.

Model/Dataset	Baseline	Mixed Precision
English	2.20	1.99
Mandarin	15.82	15.01

Closing Comments

- This paper is from 2018.
- For people working in ML ...



1. https://www.amazon.com/Dinosaurs-Roamed-Earth-Stephen-Attmore/dp/0824984099

Recent work on MPT

- Automatic mixed precision package: torch.amp
  - Automatic casting to FP16/bfloat16
  - Loss scaling
  - Using underlying tensor-core units
- MPT for LLMs
  - FP8 parameter training
  - Adaptive loss scaling to prevent overflow/underflows
  - Lowering precision of some optimizer states

$$\underbrace{4}_{\text{master weights}} + \underbrace{4}_{\text{gradients}} + \underbrace{4 + 4}_{\text{Adam states}} = 16 \text{ bytes.}$$