UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Training and inference of large language models using 8-bit floating point

Sergio P. Perez,* Yan Zhang,* James Briggs,* Charlie Blake, Josh Levy-Kramer, Paul Balanca, Carlo Luschi, Stephen Barlow, Andrew Fitzgibbon

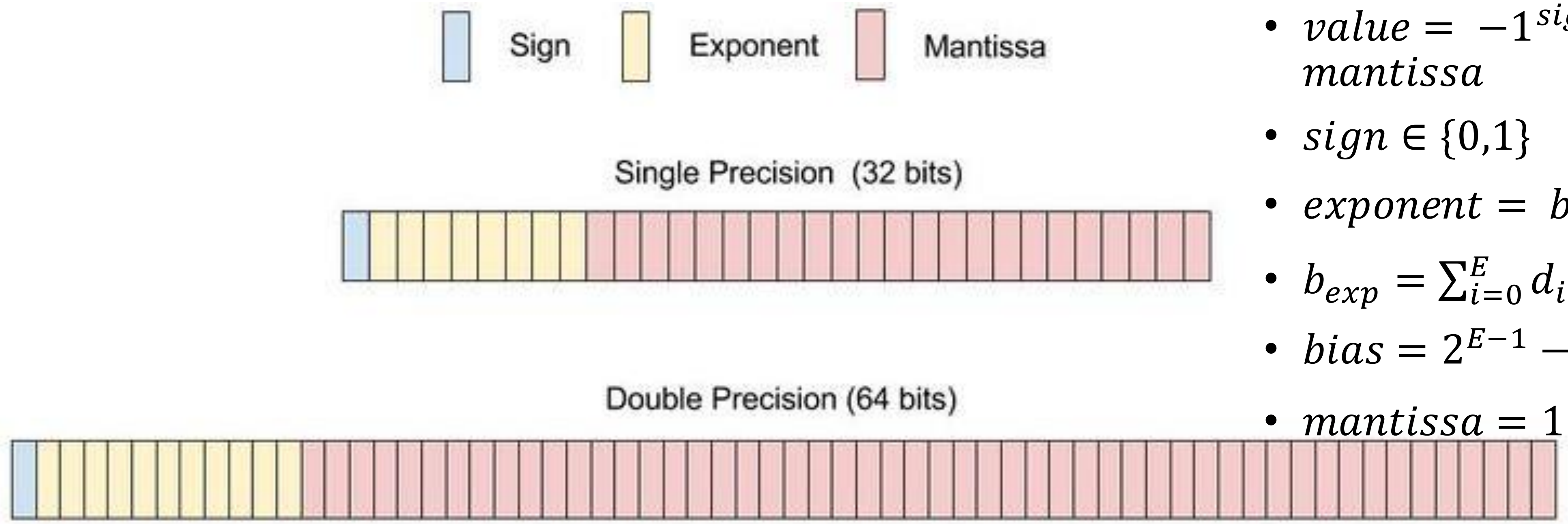
Presented by: Bakshree Mishra

April 2 2024

Agenda

- Overview of Floating-Point formats
- Impact of Quantization in Floating Point
- Quantization in Mixed Precision Training
- Scaling in Reduced Precision Training
- Proposed Methodology
- Conclusion and Discussion

Overview of Floating Point Formats



- $value = -1^{sign} \times 2^{exponent} \times mantissa$
- $sign \in \{0,1\}$
- $exponent = b_{exp} - bias$
- $b_{exp} = \sum_{i=0}^{E} d_i 2^i , d_i \in \{0,1\}$
- $bias = 2^{E-1} - 1$
- $mantissa = 1 + \sum_{i=0}^{M} d_i 2^{-i} , d_i \in \{0,1\}$

Format	E	M	Max Exp	Min Exp	Max Normal	Min Subnormal	Bias
FP32	8	23	127	-126	3.4×10^{38}	1.4×10^{-45}	127
FP16	5	10	15	-14	65504	6.0×10^{-8}	15
BF16	8	7	127	-126	3.4×10^{38}	9.2×10^{-41}	127

8 – bit Floating Point Formats vs 8 – bit Int

- **FP8 Formats:**

Format	E	M	Max Exp	Min Exp	Max Normal	Min Subnormal	Bias
FP8 E5 (a)	5	2	15	-15	57344	$7.6 \times 10{-}6$	16
FP8 E5 (b)	5	2	15	-14	57344	$1.5 \times 10{-}5$	15
FP8 E4 (c)	4	3	7	-7	240	$9.8 \times 10{-}4$	8
FP8 E4 (d)	4	3	8	-6	448	$2.0 \times 10{-}3$	7

  - Still in the process of standardization

- **Int8:**

Format	Min	Max
INT8	-128	127

  - Popularly used in inference to reduce memory overhead, and speedup computation

8 – bit Floating Point Formats vs 8 – bit Int

- **FP8 Formats:**

Format	E	M	Max Exp	Min Exp	Max Normal	Min Subnormal	Bias
FP8 E5 (a)	5	2	15	-15	57344	$7.6 \times 10{-6}$	16
FP8 E5 (b)	5	2	15	-14	57344	$1.5 \times 10{-5}$	15
FP8 E4 (c)	4	3	7	-7	240	$9.8 \times 10{-4}$	8
FP8 E4 (d)	4	3	8	-6	448	$2.0 \times 10{-3}$	7

  - Exponentially spaced values

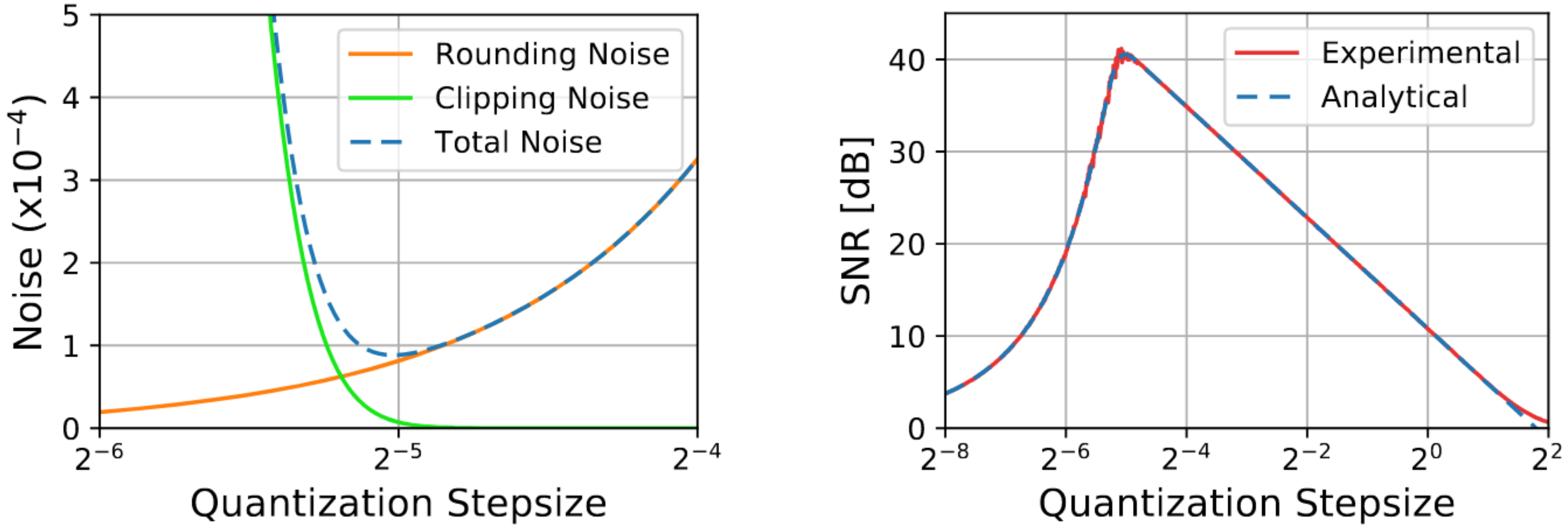
- **Int8:**

Format	Min	Max
INT8	-128	127

  - Uniformly distributed values over the representable range

Impact of FP-32 Quantization (to fixed-point)

- Quantizing from FP32 representation:
  - Noise due to rounding (reduction in precision)
  - Noise due to clipping (reduction in range)



(Conversion to 8-bit fixed point representation)

[1] B. Noune, P. Jones, D. Justus, D. Masters, and C. Luschi, "8-bit Numerical Formats for Deep Neural Networks." arXiv, Jun. 06, 2022. doi: [10.48550/arXiv.2206.02915](https://doi.org/10.48550/arXiv.2206.02915).

Impact of Quantization in Floating Point

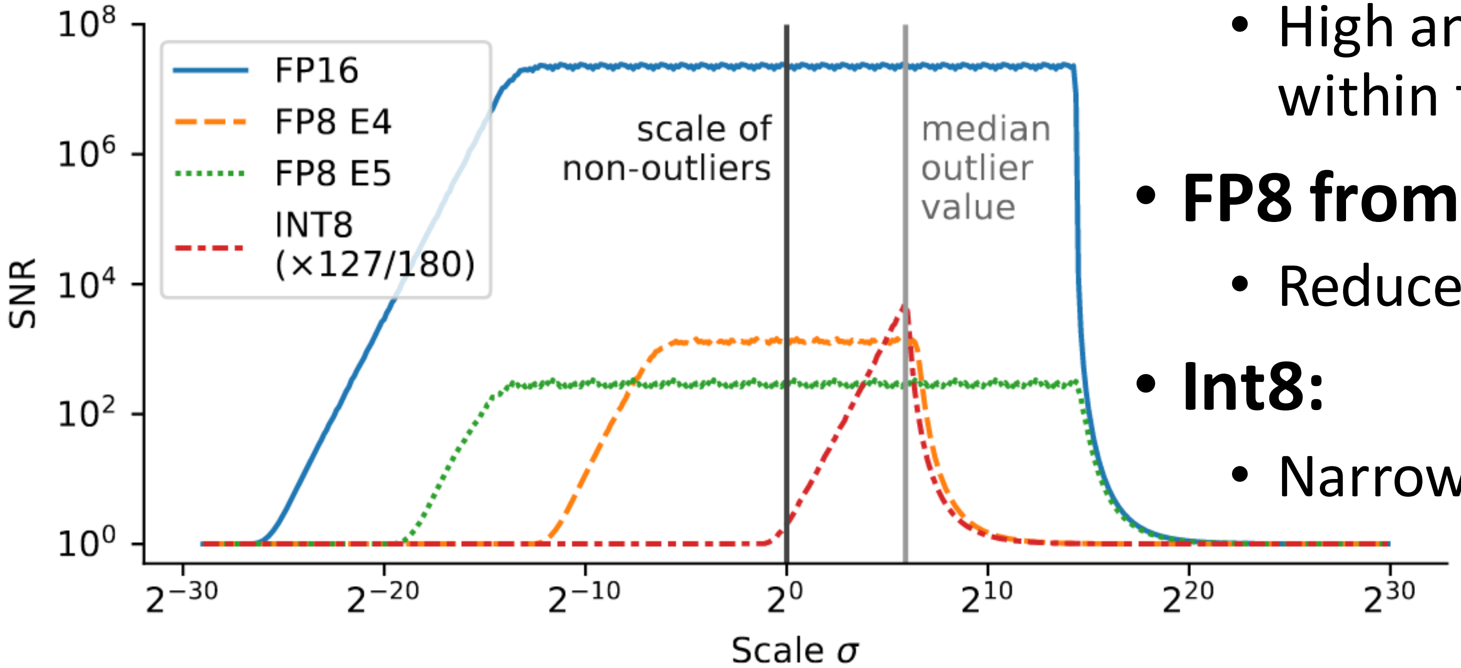


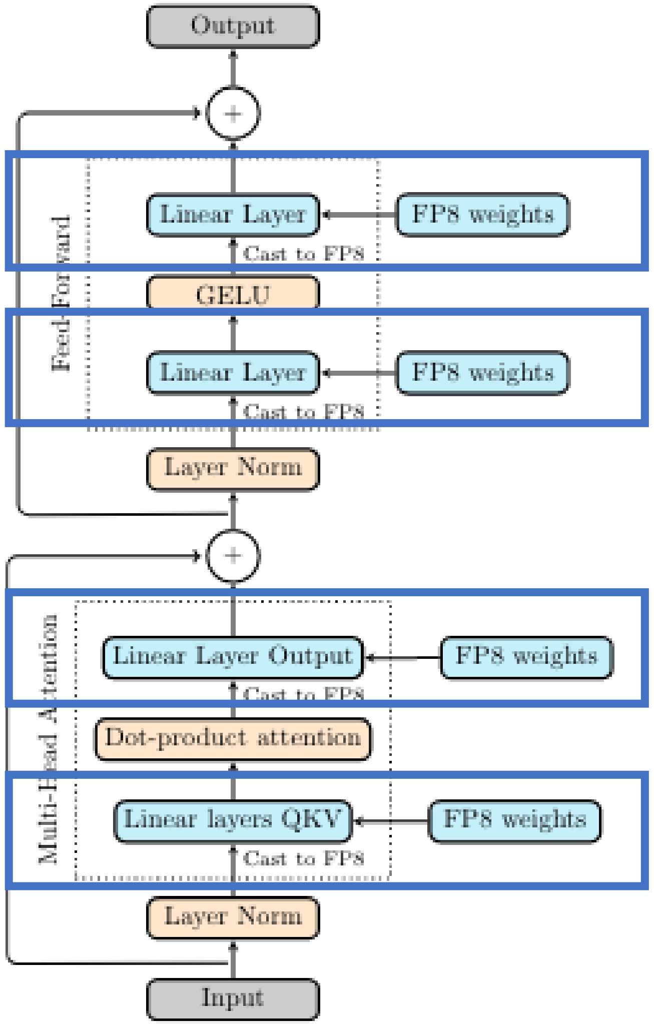
Figure A.1. The ... distribution, as a ...

- **FP16:**
  - High and approximately constant SNR within the dynamic range
- **FP8 from FP16**:
  - Reduced dynamic range, constant SNR
- **Int8:**
  - Narrow region with sufficiently high SNR

- Int 8 cannot capture distribution during training
- Need FP format

[1] C. Blake, D. Orr, and C. Luschi, "Unit Scaling: Out-of-the-Box Low-Precision Training." arXiv, May 30, 2023. doi: [10.48550/arXiv.2303.11257](https://doi.org/10.48550/arXiv.2303.11257).

Quantization in Mixed Precision Training



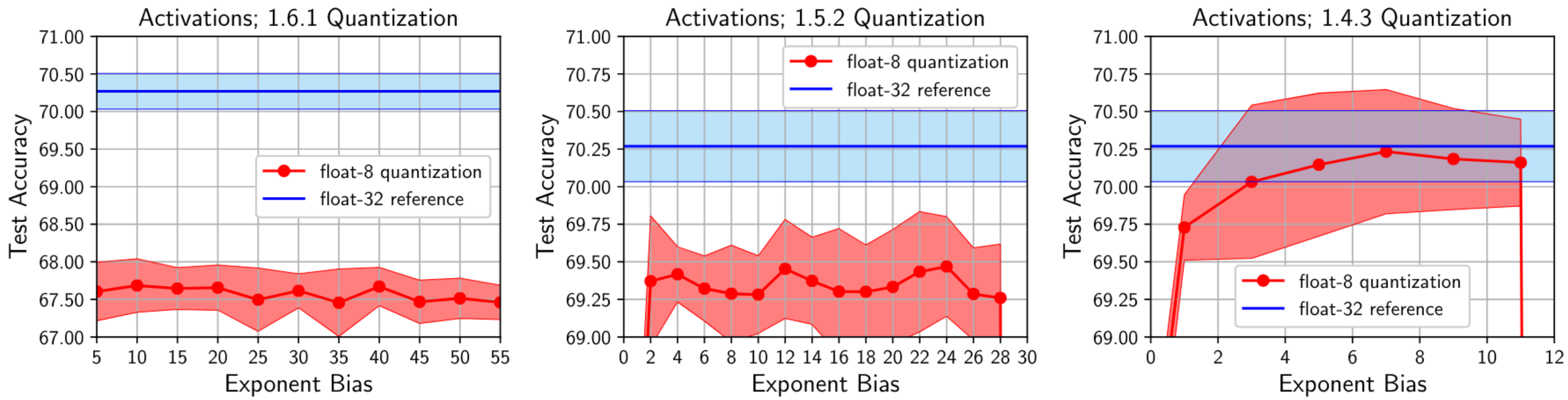
(a) GPT decoder.

- Attention linear layers to project Q,K V matrices
- Attention linear layer after outputs of heads are concatenated
- First feed-forward layer
- Second feedforward layer

- The optimizations in this paper focus on shorter seq lengths
- Linear layers constitute 99.9% of total compute in decoder layer
  - Linear layers: complexity quadratic with hidden dimension of model
  - Dot products: quadratic with sequence length

- Mixed precision training with FP8 with two different FP8 formats

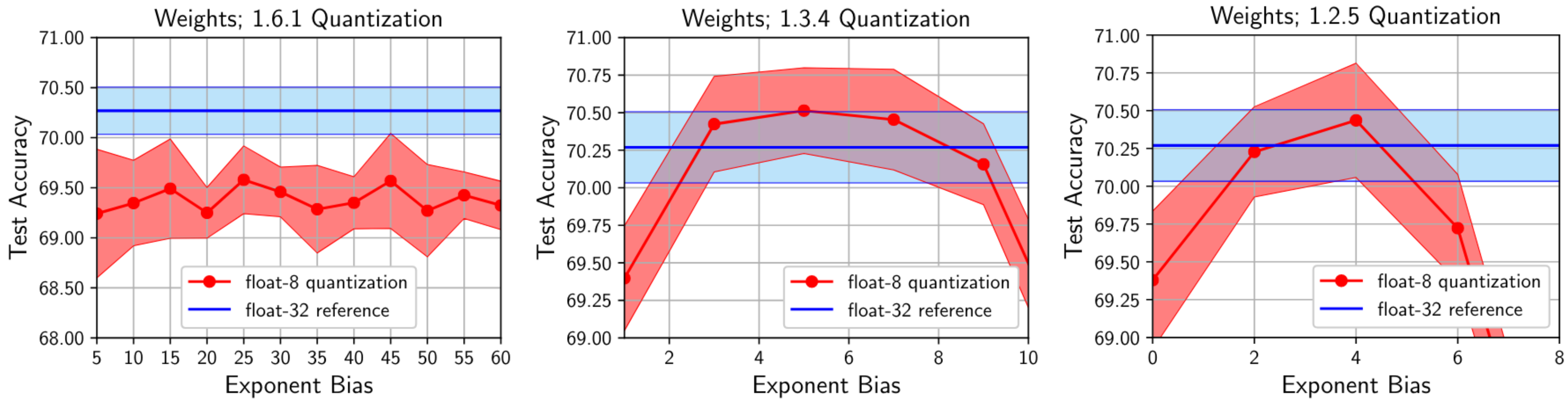
- Gradients: require more range
  - Need E5 format (5 exponent bits, 2 mantissa bits)



[1] *B. Noune, P. Jones, D. Justus, D. Masters, and C. Luschi, "8-bit Numerical Formats for Deep Neural Networks." arXiv, Jun. 06, 2022. doi: 10.48550/arXiv.2206.02915.*

FP8 Formats for Training and Inference

- Mixed precision training with FP8 with two different FP8 formats
- Gradients: require more range
  - Need E5 format (5 exponent bits, 2 mantissa bits)
- Activations and weights need at least 3 mantissa bits for numerical accuracy
  - Need(monolithic) E4 format (4 exponent bits, 3 mantissa bits)



[1] *B. Noune, P. Jones, D. Justus, D. Masters, and C. Luschi, "8-bit Numerical Formats for Deep Neural Networks." arXiv, Jun. 06, 2022. doi: [10.48550/arXiv.2206.02915](https://doi.org/10.48550/arXiv.2206.02915).*

FP8 Formats for Training and Inference

- Mixed precision training with FP8 with two different FP8 formats
- Gradients: require more range
  - Need E5 format (5 exponent bits, 2 mantissa bits)
- Activations and weights need at least 3 mantissa bits for numerical accuracy
  - Need(monolithic) E4 format (4 exponent bits, 3 mantissa bits)
- Mixed precision training:
  - FP8 used only for matrix multiplications
  - Values accumulated and stored in higher precision (FP32)

Scaling for Reduced-Precision Training

- Methods to retain higher-precision range:
  - Loss scaling
  - Automatic Loss Scaling
  - Automatic per-tensor Scaling
  - Unit Scaling

Scaling for Reduced-Precision Training

- Loss scaling
  - Tackles underflow in gradients by multiplying loss with a scalar
  - Weight gradients are then divided by the same scalar in the optimizer

  - Requires hyperparameter sweep must be conducted to find the loss scale value
  - Single scaling factor, no mechanism to combat differences in scale in gradient tensors

[1] C. Blake, D. Orr, and C. Luschi, "Unit Scaling: Out-of-the-Box Low-Precision Training." arXiv, May 30, 2023. doi: [10.48550/arXiv.2303.11257](https://doi.org/10.48550/arXiv.2303.11257).

Scaling for Reduced-Precision Training

- Loss scaling
  - Tackles underflow in gradients by multiplying loss with a scalar
  - Weight gradients are then divided by the same scalar in the optimizer

  - Requires hyperparameter sweep must be conducted to find the loss scale value
  - Single scaling factor, no mechanism to combat differences in scale in gradient tensors
- Automatic Loss scaling
  - Dynamic adjustment of the loss scale during training
  - Remove the need to sweep the initial loss scale
  - Combats shifts in tensor distributions during training

[1] C. Blake, D. Orr, and C. Luschi, "Unit Scaling: Out-of-the-Box Low-Precision Training." arXiv, May 30, 2023. doi: [10.48550/arXiv.2303.11257](https://doi.org/10.48550/arXiv.2303.11257).

Scaling for Reduced-Precision Training

- Per-tensor scaling
  - Addresses scaling difficulties in FP8 training
  - Rescale locally based on runtime statistics
  - Additional compute, memory, bandwidth and cross-device communication costs

[1] C. Blake, D. Orr, and C. Luschi, "Unit Scaling: Out-of-the-Box Low-Precision Training." arXiv, May 30, 2023. doi: [10.48550/arXiv.2303.11257](https://doi.org/10.48550/arXiv.2303.11257).

Scaling for Reduced-Precision Training

- Per-tensor scaling
  - Addresses scaling difficulties in FP8 training
  - Rescale locally based on runtime statistics
  - Additional compute, memory, bandwidth and cross-device communication costs
- Unit Scaling
  - Activations, weight and gradients have approximately unit variance at initialization.
  - Insert scaling factors into the forward and backward passes
  - Unit scaling determines these scales based on a set of rules for each operation
  - Does not address the issue of adapting scales during training

[1] C. Blake, D. Orr, and C. Luschi, "Unit Scaling: Out-of-the-Box Low-Precision Training." arXiv, May 30, 2023. doi: 10.48550/arXiv.2303.11257.

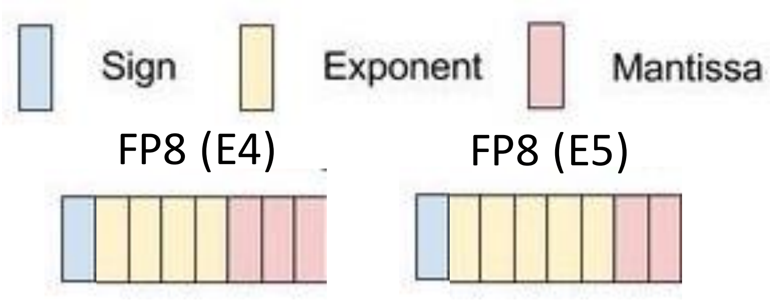
- Summary of scaling techniques:

Method	Fine-grained scaling	No tuning required	Adapts during training
Loss scaling	×	×	×
Automatic loss scaling	×	✓	✓
Automatic per-tensor scaling	✓	∼	✓
Unit scaling	✓	✓	×

- This paper: per-tensor scaling

[1] *C. Blake, D. Orr, and C. Luschi, "Unit Scaling: Out-of-the-Box Low-Precision Training." arXiv, May 30, 2023. doi: 10.48550/arXiv.2303.11257.*

Per-Tensor Scaling Bias



- $exponent = b_{exp} - bias$
- $scaled\ exponent = b_{exp} - bias + b_{scale}$

- Scaling biases for weights and activations: $b_{w,scale}$ and $b_{x,scale}$
- Use scaled FP8 values for FP8 computation
- For computation in FP16, unscaling the activations:
  - $unscaled\ exponent = b_{exp} - bias - (b_{w,scale} + b_{x,scale})$

Choosing Per-Tensor Scaling Bias

- Two ways of selecting scaling bias:
  - Just-in-time scaling
  - Constant scaling
- Proposed methodology: Just-in-time scaling (AMAX)
  - Choose scaling bias based on maximum value in tensor
  - Depends on maximum representable value in the FP8 format (max_{num})

$$amax = max(|tensor|)$$
$$scaling\ bias = floor\left(log2\left({}^{max_{num}}/_{amax}\right)\right)$$

- Tradeoff methodology: Constant scaling bias (CSCALE)
  - Sweeps of scaling bias values to identify ones that don't degrade accuracy
  - Constant for weights, activations and gradients
  - Remains constant throughout the training and inference

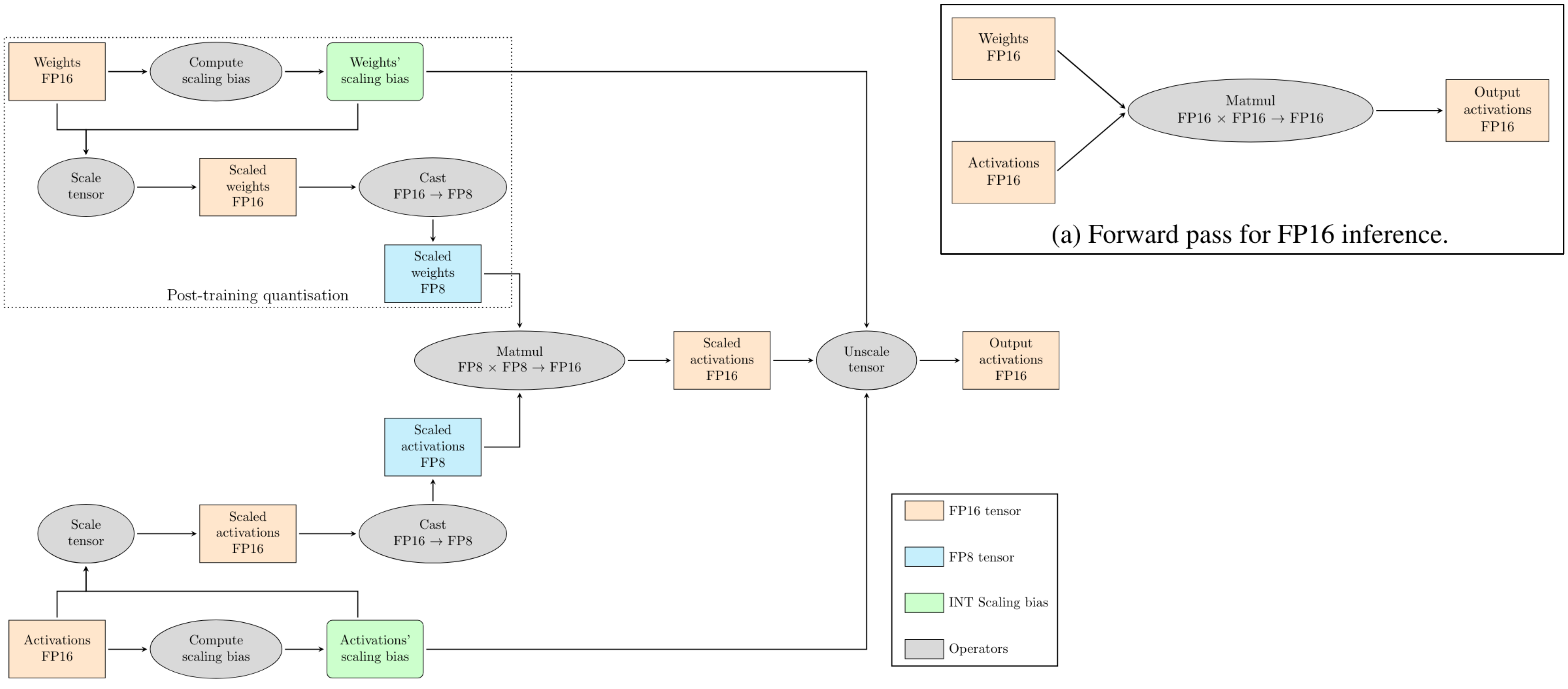
AMAX vs CSCALE

- AMAT:
  - Pros:
    - Higher SNR (sets dynamic range based on tensor values)
    - Better model accuracy and faster convergence*
    - *With hardware support (and sufficient SRAM), can be computed just-in-time
  - Cons:
    - When SRAM is limited, FP16 tensors reside in L2-cache
    - Can result in additional round-trip to memory that can cancel FP8 speedups
- CSCALE:
  - Pros:
    - Less memory overhead
    - Can enable speedup due to FP8 operations in hardware with less SRAM
  - Cons: ??

Loss Scaling for FP16

- Loss scaling necessary in FP16 due to narrower dynamic range than FP32
  - Gradients can underflow
- Relevance to FP8 quantization: accumulation in FP16
  - FP8 accumulation does not work due to
    - limited dynamic range (E4)
    - limited precision (E5)
  - FP8 operations actually mixed FP8-FP16 operations
- This paper: uses constant loss scaling

Applying Per-Tensor Scaling Bias in FP8 Inference



(a) Forward pass for FP16 inference.

(b) Forward pass for FP8 inference.

Evaluation

- Hardware:
  - Graphcore IPUs for training
  - Does not have native FP8 support – enabled in Software
- Models evaluated:
  - GPT-3-like architecture, using dense attention
  - Llama 2 model
  - Linear layers quantized to FP8, with FP8-AMAX and FP8-CSCALE
  - Dot-product and other computation still in FP16

Parameters	d_{model}	n_{layers}	n_{heads}	d_{head}	d_{ffn}
GPT 111M	768	10	12	64	3072
GPT 590M	1536	18	12	128	6144
GPT 1.3B	2048	24	16	128	8192
GPT 6.7B	4096	32	32	128	16384
GPT 13B	5120	40	40	128	20480
Llama 2 7B	4096	32	32	128	11008
Llama 2 70B	8192	80	64	128	28672

Inference Results with GPT model

Table 2: Inference results: validation accuracy comparing FP16 with FP8-AMAX and FP8-CSCALE, for the different GPT model sizes.

Model	Quantisation	MNLI	QQP	SST-2
111M	FP16	72.61	85.76	84.26
	FP8-AMAX	72.39	85.78	84.38
	FP8-CSCALE	72.49	85.73	84.59
590M	FP16	78.59	88.40	90.63
	FP8-AMAX	78.44	88.37	90.63
	FP8-CSCALE	78.56	88.40	90.54
1.3B	FP16	82.82	89.43	91.55
	FP8-AMAX	82.68	89.42	91.44
	FP8-CSCALE	82.72	89.36	91.42
6.7B	FP16	87.17	91.19	94.50
	FP8-AMAX	87.15	91.22	94.38
	FP8-CSCALE	87.18	91.18	94.48
13B	FP16	88.26	91.22	94.61
	FP8-AMAX	88.27	91.21	94.61
	FP8-CSCALE	88.26	91.20	94.50

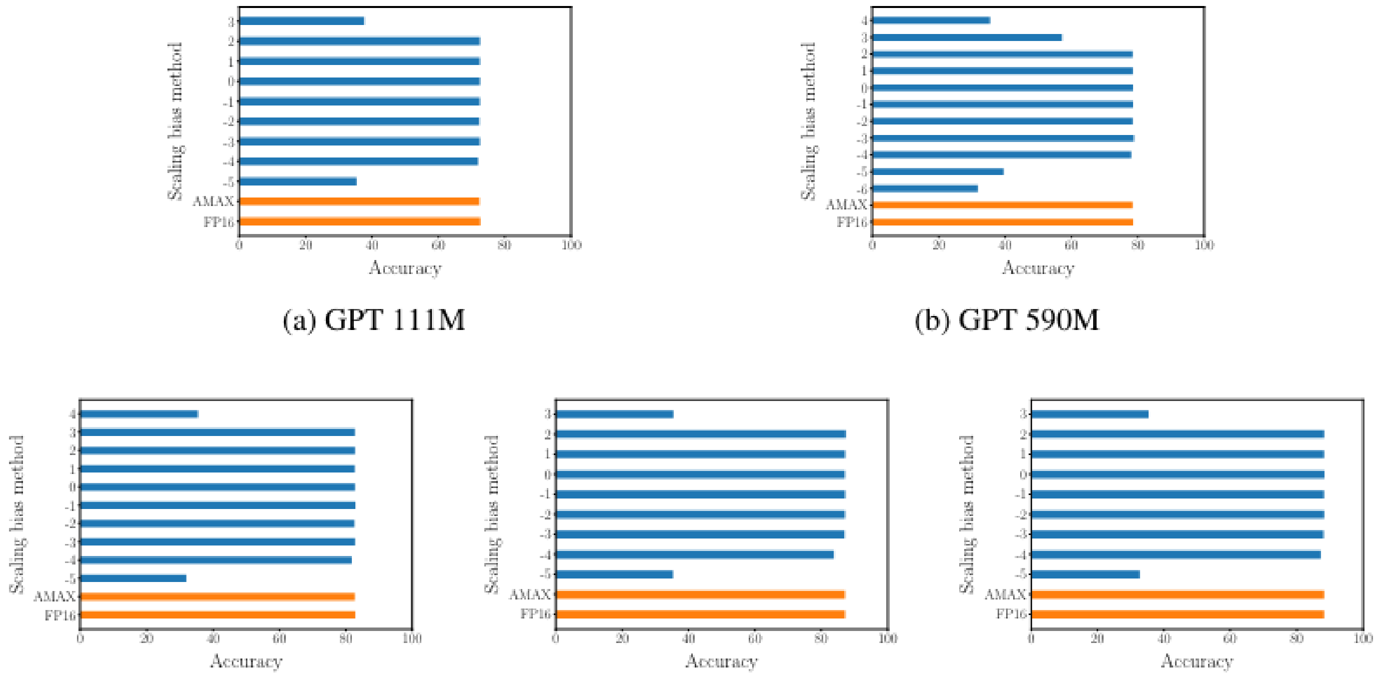
- FP16 validation accuracy matched for AMAX
  - Different scaling biases for weight, activation tensors
  - Scaling biases calculated just in time per tensor
- FP16 validation accuracy matched for CSCALE
  - Same scaling bias for weights and activations
  - Sweep of scaling biases, not all meet accuracy
  - Results reported with scaling biases within range

Model	MNLI	QQP	SST-2
111M	[-3, 2]	[-4, 2]	[-4, 2]
590M	[-3, 2]	[-4, 2]	[-1, 2]
1.3B	[-3, 3]	[-4, 2]	[-3, 2]
6.7B	[-3, 2]	[-3, 2]	[-3, 2]
13B	[-3, 2]	[-4, 2]	[-4, 2]

Inference Results with GPT model

Table 2: Inference results: validation accuracy comparing FP16 with FP8-AMAX and FP8-CSCALE, for the different GPT model sizes.

Model	Quantisation	MNLI	QQP	SST-2
111M	FP16	72.61	85.76	84.26
	FP8-AMAX	72.39	85.78	84.38
	FP8-CSCALE	72.49	85.73	84.59
590M	FP16	78.59	88.40	90.63
	FP8-AMAX	78.44	88.37	90.63
	FP8-CSCALE	78.56	88.40	90.54
1.3B	FP16	82.82	89.43	91.55
	FP8-AMAX	82.68	89.42	91.44
	FP8-CSCALE	82.72	89.36	91.42
6.7B	FP16	87.17	91.19	94.50
	FP8-AMAX	87.15	91.22	94.38
	FP8-CSCALE	87.18	91.18	94.48
13B	FP16	88.26	91.22	94.61
	FP8-AMAX	88.27	91.21	94.61
	FP8-CSCALE	88.26	91.20	94.50



(a) GPT 111M (b) GPT 590M

(c) GPT 1.3B (d) GPT 6.7B (e) GPT 13B

Figure 5: Comparison of scaling bias methods for the MNLI validation, for the different GPT sizes. Whereas the FP8-AMAX method always matches the FP16 accuracy, the FP8-CSCALE method only converges in an interval of scaling values. The specific interval that reaches at least 99.5% of the FP16 value is displayed in Table 3.

Inference Results with Llama

Model	Quantisation	MMLU	HellaSwag	ARC-e	ARC-c	PIQA	WinoGrande
7B	Llama 2 paper	45.3	77.2	75.2	45.9	78.8	69.2
	FP16	46.6	76.0	74.6	46.3	79.1	69.1
	FP8-AMAX	46.3	75.8	74.5	45.7	78.7	69.1
70B	Llama 2 paper	68.9	85.3	80.2	57.4	82.8	80.2
	FP16	69.6	83.8	81.1	57.3	82.8	78.0
	FP8-AMAX	69.3	83.8	80.9	57.7	82.6	78.5

- FP8-AMAX gives comparable results as FP16
- Interestingly, do not provide FP8-CSCALE results

Training results with GPT model

Table 5: Fine-tuning results: validation accuracy after fine-tuning in FP16 and FP8-AMAX for 3 epochs.

Model	Quantisation	MNLI	QQP	SST-2
111M	FP16	72.61	85.32	85.07
	FP8-AMAX	72.50	85.84	85.57
590M	FP16	78.59	88.25	89.27
	FP8-AMAX	79.12	88.31	89.00
1.3B	FP16	82.82	89.32	91.36
	FP8-AMAX	82.58	89.32	91.28
6.7B	FP16	87.17	91.19	94.53
	FP8-AMAX	87.26	91.06	94.84
13B	FP16	88.26	91.22	94.61
	FP8-AMAX	88.28	91.53	94.50

- Convergence achieved with FP-AMAX
  - Different scaling biases for weight, activation tensors
  - Tested for 3 tasks
- Convergence with FP-CSCALE:
  - Tested for 1 task
  - Convergence achieved with range

Model	MNLI
111M	[-3, 2]
590M	[-2, 2]
1.3B	[-2, 1]
6.7B	[-1, 1]
13B	[-1, 0]

  - Larger the model, range of scaling biases reduces
  - For 7.7B and 13B model, convergence not guaranteed
    - A different seed can cause divergence

Observation: per-tensor scaling bias with FP8-AMAX

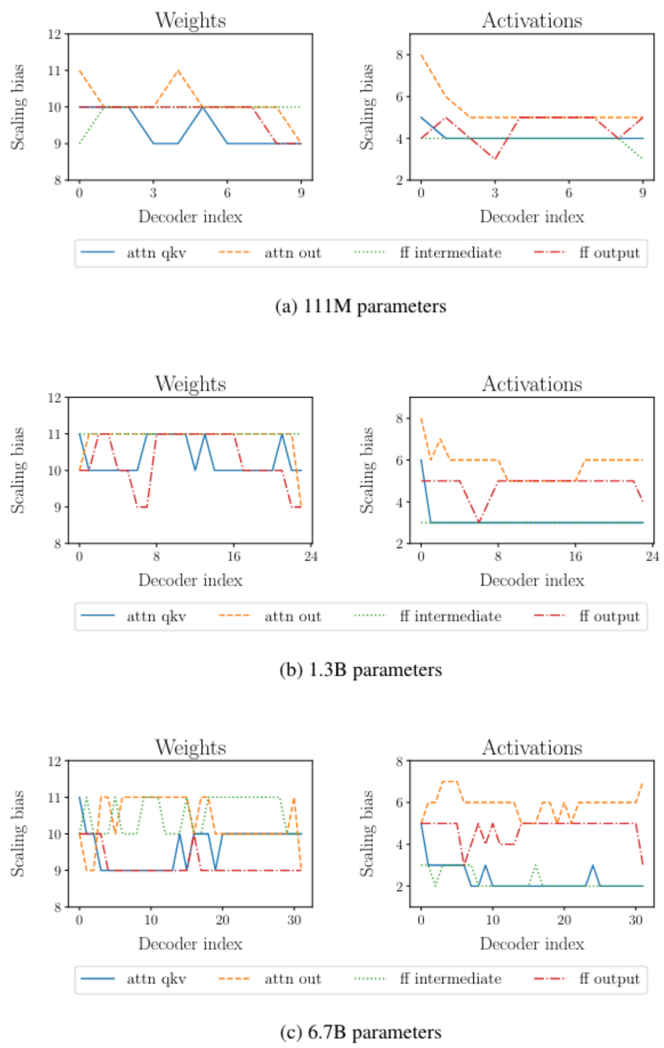


Figure 6: Scaling bias distribution per decoder and type of linear layer for the MNLI validation, comparing different sizes of the GPT model. The scaling bias is computed with the FP8-AMAX method in Subsection 2.3.

- Weight versus activation scaling bias:
  - Weight: larger values and narrower distribution
  - Activation: larger distribution
- Size of models: determines range of scaling bias
- Type of linear layer:
  - the scaling biases of the attention linear layer after the outputs take greater values than the other linear layers

Conclusions

- Scaling necessary in FP8 quantization to reduce loss, improve SNR
- Scaling bias can be constant or per-tensor, with trade-offs
- This paper proposes just-in-time scaling bias per tensor
- Matches FP16 accuracy for training and inference tasks

Discussion

- Evaluation platform unclear: evaluated on HW without native FP8 support
  - Are the overheads of the quantization properly captured?
  - Why not compare against NVIDIA H100 with native FP8 support?
- What’s next: FP4 representation
  - NVIDIA Blackwell support
  - Either E2 or E1: further compression in model weights and activations
    - Implications on results from this paper? (For example, requiring E4 or E5)