Introduction to Data Types for AI: Trade-Offs and Trends

Joep Boonstra

Synopsys Scientist

Synopsys

SYNOPSYS®

Overview

Purpose: Overview of pros and cons in state-of-the-art data types for AI

Agenda:

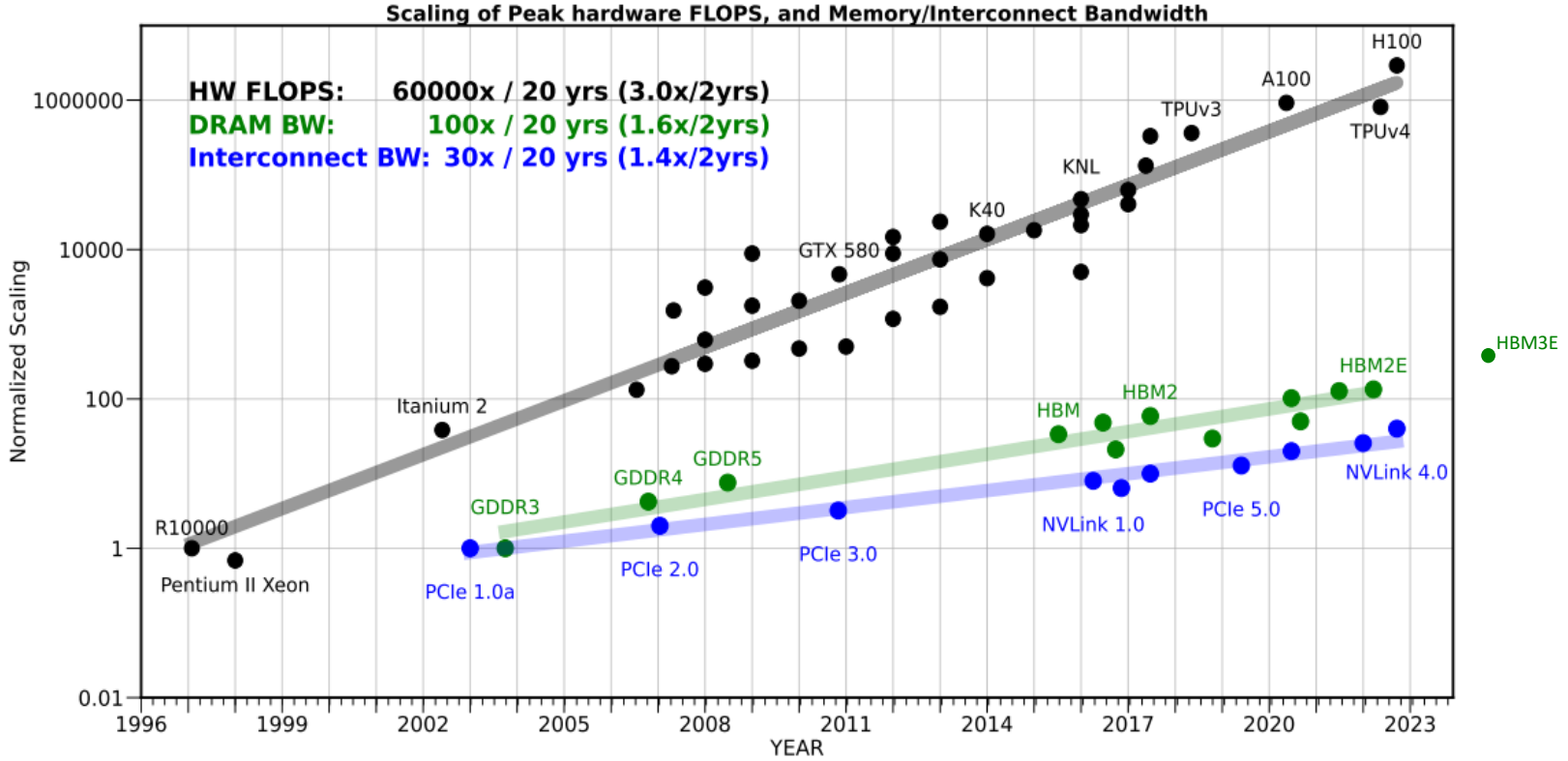
- The AI memory wall: history, trends, background
- Challenges in AI related to data types
- Floating-point data types and micro-scaling
- Quantized data types, integer, fractional, ternary
- Summary

Terminology

- Quantization: the process of mapping continuous infinite values to a smaller set of discrete finite values
  - For AI: conversion of a data type to a narrower data type
- Compression: encoding information using fewer bits than the original representation
  - Considerations: lossy vs lossless, weights vs feature maps
- Data type: unit of data for AI compute
  - May have a smaller, compressed representation in memory

The AI Memory Wall: History, Trends, Background

AI Memory Wall, Compute Bandwidth versus Memory Bandwidth [1]

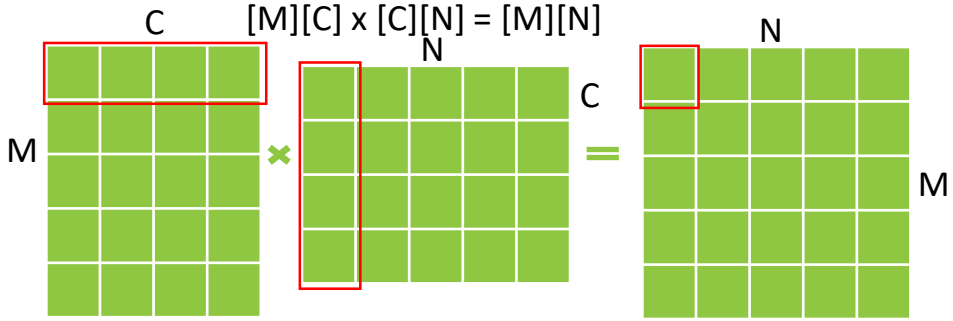


AI Memory Wall,
Matrix x Matrix versus Matrix x Vector Computations

Compute Bandwidth: $BW_c = M*N*C$

Memory Bandwidth: $BW_m = (M+N)*C$

Memory/Compute Ratio = (M+N) / (M*N)



Recent models (LLM,VLM) use more matrix x vector computations

Smaller data-types enable:

- Reduced memory bandwidth, foot-print and reduce latency to load data
- Increased tile sizes in caches → bigger, more efficient matrix multiplication
- More parallelization in hardware

	M=C=	N=	BW_m : BW_c	Example applications
matrix*vector	1024	1	1:1	FC layers, MLP
matrix*matrix	1024	32	3 : 100	LLM batched prompt processing
matrix*matrix	1024	64	3 : 200	CNN, limited batch processing
matrix*matrix	1024	1024	1 : 500	CNN, large batch processing

Challenges in Data Types Selection for AI

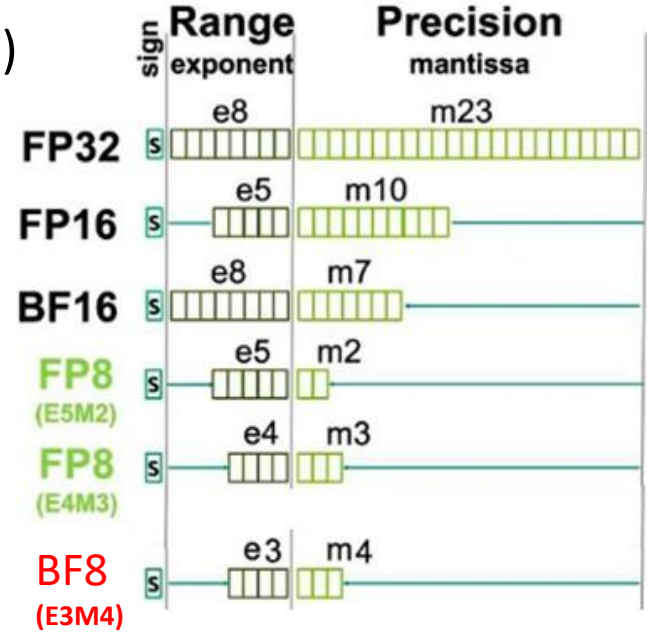
Challenges

- Quantization-Aware Training or Post-Training Quantization:
  - Mapping (continuous) values to a limited set of discrete values
  - While minimizing overall accuracy loss
- Inference:
  - In data-centers: option to batch multiple tasks
  - At the edge: limited batching opportunities, limited memory footprint
- Complexity of hardware and software implementation
  - Floating-point more complex than integer & fractional
  - Data compression and decompression

Floating-Point and Micro-Scaling Data Types

Floating-Point [2,3]

- Widely used in gradient descent AI training
- Resolution (mantissa) versus dynamic range (exponent)
  E*x*M*y* = 1 sign bit, *x* exponents bits, *y* mantissa bits
- Trend to lower bit-width:
  FP32 → FP16/BF16 → OCP-FP8 (E5M2/E4M3)
  or BF8 (E3M4)
- FP8 supported in most recent hardware platforms
- Mixed types: per weights, activations, layer, expert
- More complex hardware implementation than integer



Floating-Point [4,5]

- Example Mistral-7B, quantized to FP8 on H100:

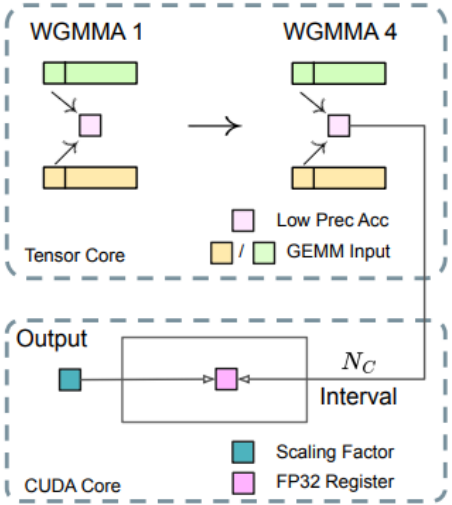
Data format	Perplexity score	Increase vs FP16
FP16	7.4701	1x
FP8	7.4918	1.0029x
INT8 SmoothQuant	13.6648	1.8293x

FP16 → FP8

- An 8.5% decrease in latency in the form of time to first token
- A 33% improvement in speed, measured as output tokens per second
- A 31% increase in throughput in terms of total output tokens
- A 24% reduction in cost per million tokens

Example DeekSeek-V3:

- Mixed precision training, mostly FP8; critical sections BF16/FP32
- Fine-grain FP8 quantization in blocks of $1xN_c$ and N_cxN_c FP8 values, common scaling per block



Micro-Scaling Types [6]

Micro-scaling types consist of three components:

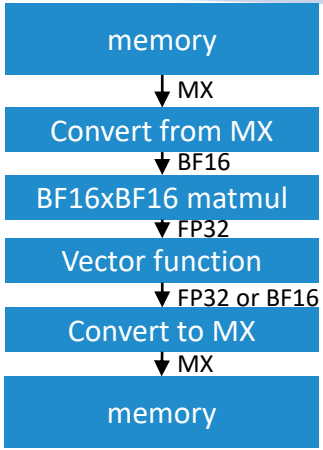
- A set of private elements P_i , each d bits wide
- The number of private elements in the set *k*, 1 < *i* <= *k*
- A common scale *X*, *w* bits wide

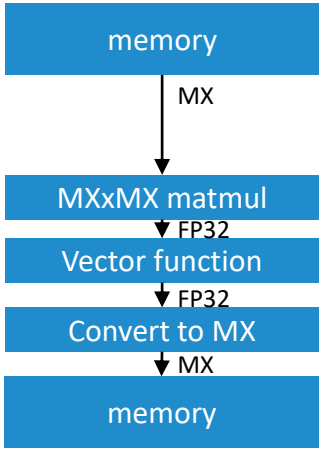
Characteristics:

- High dynamic range, many exponent bits
- Low precision, few mantissa bits
- $V_i = (-1)^{S_i} * M_i * 2^{E_i+X\text{-bias}}$
- Mix types for weights and activations
- Supported by most recent hardware platforms

Format Name	Element Data Type	Elements Bits [d]	Scaling Block Size (k)	Scale Data Type	Scale Bits (w)
MXFP8	FP8 (E5M2) FP8 (E4M3)	8	32	E8M0	8
MXFP6	FP6 (E3M2) FP6 (E2M3)	6	32	E8M0	8
MXFP4	FP4 (E2M1)	4	32	E8M0	8
MXINT8	INT8	8	32	E8M0	8

Micro-Scaling Implementation Options





MX as memory compression technique only:

- Decompress OCP-MX data to BF16 before computation and compress to OCP-MX after computations
- Saves memory bandwidth and foot-print
- Double rounding risk (accuracy)
- Does not save compute bandwidth
- Limited power saving

Compute optimized for MX :

- Fewer bits in mantissa unless MXINT8 → less logic to implement multipliers
- Limited number of bits in element exponent → less normalization logic
- No double rounding
- Saving silicon area, saving power

Micro-Scaling Accuracy: Discriminative Inference [7]

Direct-cast from FP32

Task	Family	Model	Dataset	Metric	Baseline FP32	MXINT8	MXFP8		MXFP6		MXFP4
							E4M3	E5M2	E2M3	E3M2	
Language Translation	Transformers (Enc-Dec)	Transformer-Base [9]	WMT-17	BLEU Score ↑	26.85	26.64	26.27	25.75	26.38	25.97	22.68
		Transformer-Large [9]			27.63	27.56	27.44	27.02	27.52	27.22	26.33
	LSTM	GNMT [10]	WMT-16		24.44	24.52	24.53	24.45	24.51	24.44	23.75
Language Encoding	Transformers (Enc-Only)	BERT-Base [11]	Wikipedia	F-1 Score ↑	88.63	88.58	88.47	87.04	88.38	88.05	84.94
		BERT-Large [11]			93.47	93.41	93.42	93.32	93.45	93.27	90.97
Image Classification	Vision Transformer	DeiT-Tiny [12]	ImageNet ILSVRC12	Top-1 Acc. ↑	72.16	72.20	71.37	70.11	71.56	70.16	56.72
		DeiT-Small [12]			80.54	80.56	79.83	79.00	80.11	79.04	71.35
	CNN	ResNet-18 [13]			70.79	70.80	69.08	66.16	69.71	66.10	48.77
		ResNet-50 [13]			77.40	77.27	75.94	73.78	76.42	73.75	42.39
		MobileNet v2 [14]			72.14	71.61	65.74	53.50	67.76	53.46	0.25
Speech Recognition	Transformer	Wav2Vec 2.0 [15]	LibriSpeech	WER ↓	18.90	18.83	23.71	21.99	20.63	21.98	42.62
Recommendations	MLPs	DLRM [16]	Criteo Terabyte	AUC ↑	0.803	0.803	0.802	0.801	0.802	0.801	0.7947

Conclusion:

- For direct-cast inference, MXINT8 is effective as replacement for FP32
- Formats with more mantissa bits are more accurate
- Not shown: After fine-tuning, MXFP6_E2M3 accuracy is close to FP32

Micro-Scaling Accuracy: Generative Inference, Llama7B [7]

Tasks	FP32	MXINT8	MXFP8	MXFP6	MXFP6 Wt MXFP8 Act	MXFP4 Wt MXFP8 Act	MXFP4 Wt MXFP6 Act	MXFP4
ARC easy ↑	0.729 ± 0.009	0.725 ± 0.009	0.716 ± 0.009	0.718 ± 0.009	0.726 ± 0.009	0.697 ± 0.010	0.696 ± 0.010	0.637 ± 0.010
ARC challenge ↑	0.447 ± 0.015	0.444 ± 0.015	0.430 ± 0.015	0.445 ± 0.015	0.442 ± 0.015	0.412 ± 0.014	0.406 ± 0.014	0.355 ± 0.014
Lambada ↑	0.736 ± 0.006	0.731 ± 0.006	0.720 ± 0.006	0.724 ± 0.006	0.721 ± 0.006	0.675 ± 0.006	0.678 ± 0.007	0.557 ± 0.007
College CS ↑	0.260 ± 0.044	0.220 ± 0.045	0.270 ± 0.042	0.240 ± 0.043	0.280 ± 0.045	0.240 ± 0.043	0.210 ± 0.041	0.220 ± 0.042
Int. law ↑	0.463 ± 0.046	0.430 ± 0.045	0.413 ± 0.045	0.422 ± 0.045	0.413 ± 0.045	0.398 ± 0.045	0.405 ± 0.045	0.331 ± 0.041
Jurisprudence ↑	0.361 ± 0.046	0.370 ± 0.047	0.380 ± 0.047	0.370 ± 0.046	0.352 ± 0.047	0.269 ± 0.045	0.296 ± 0.044	0.269 ± 0.043
wikitext ↓	9.488	9.504	9.768	9.628	9.683	11.476	11.147	27.201

↑ higher is better (all except wikitext)
↓ lower is better (wikitext)
All GeMM in quantized format

Conclusion

- For direct-cast inference, MXINT8 is effective as replacement for FP32

Quantized Data Types

The Quantization Challenge

- Convert continuous infinite input values from a large set to discrete finite output values in a smaller set
- Reduce the precision of calculations to enhance efficiency
- While minimizing impact on training and inference accuracy
- Considerations:
  - What is the quantization range? How to deal with outliers?
  - Uniform vs non-uniform quantization intervals (codebook)
  - Mixed quantization for layers, channels, blocks, weight, activation…
- Perplexity:
  - A measure of how well a language model predicts a sample
  - Higher perplexity → more uncertainty

Integer and Fixed-Point Quantization

- In general, in CNN models, INT4, INT8, INT16 can be used to represent a quantized floating-point value
- Scale & offset per output channel: $W_{ci,co} = Q_{ci,co} * SCALE_{co} + OFFSET_{co}$
- Quantization options:
  - Quantization-Aware Training
  - Post-training quantization: AdaRound, AdaQuant, GPTQ
  - Model fine-tuning to fit original floating-point (limited epochs)
- INT4 weights can be mixed with FP16 activations for selected layers, if required
- Complex training, simple inference hardware/software

Advanced Quantized Data Types Compression [8,9]

HuggingFace, GPT-Generated Unified Format (GGUF):

- Support for various data types, amongst others:
  - Qx: quantized data type with block-scaling (weights-only)
  - IQx: quantized data type with block-scaling and importance matrix (weights-only)
- Support for different PTQ algorithms: k-quant, i-quant, …
- Decompress weights before compute to FP8, FP16 or BF16

- Large LLMs have significant redundancy, allowing high compression rate
- Example: DeepSeekR1 671B MoE, ternary weight quantization → 1.58 bits/weight

Summary

Summary

- Data type selection, compression and quantization are important aspects of AI
- Smaller data types enable:
  - External memory bandwidth, footprint, power, latency and system cost reduction
  - Cache spilling reduction, bigger tiles for matrix multiplication
  - Hardware parallelization for higher performance
- Memory bandwidth bottleneck and footprint drive research into narrow data type quantization and compression, especially for LLM
- Mixed integer and floating-point can be used to quantize CNN and transformers
- Micro-scaling types can be used to compress models without loss of accuracy
- Future will see more hardware and software optimizations to support compressed data types

Resources

[1] **AI and Memory Wall**, https://arxiv.org/pdf/2403.14123v1

[2] **IEEE Standard for Floating-Point Arithmetic**, IEEE Std 754-2019E

[3] **OCP-FP8 Standard:** https://www.opencompute.org/documents/ocp-8-bit-floating-point-specification-ofp8-revision-1-0-2023-12-01-pdf-1

[4] **33% faster LLM inference with FP8 quantization**, https://www.baseten.co/blog/33-faster-llm-inference-with-fp8-quantization

[5] **DeepSeek-V3 Technical Report**, https://arxiv.org/html/2412.19437v1

[6] OCP Microscaling Formats (MX) Specification version 1.0, https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf

[7] Microscaling Data Formats for Deep Learning, https://arxiv.org/pdf/2310.10537

[8] **QuIP#: Even Better LLM Quantization with Hadamard Incoherence and Lattice Codebooks**, https://arxiv.org/pdf/2402.04396

[9] **Unsloth's DeepSeek-R1 1.58-bit + 2-bit Dynamic Quants**, https://huggingface.co/unsloth/DeepSeek-R1-GGU

2025 Embedded Vision Summit:

Visit Synopsys at Booth #717