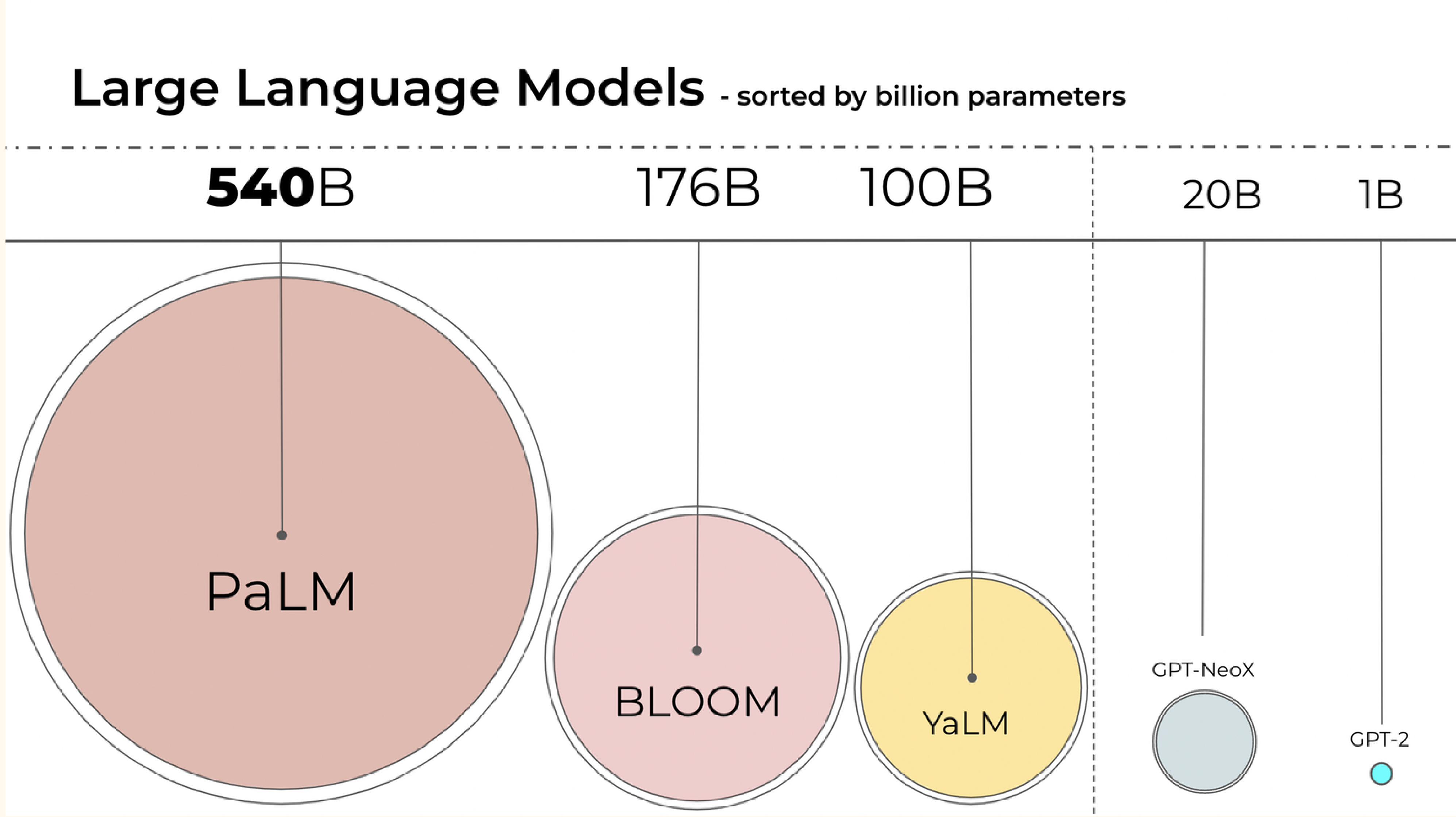
4-BIT QUANTISATION DEMISTYFIED WITH TRANSFORMERS

A SOLUTION FOR TRAINING AND REDUCING MODEL SIZES

HTTPS://GITHUB.COM/INSIGHTBUILDER

CHALLENGE SOLVED: WHY COMPRESSED MODEL SIZE

- MODELS ARE STORAGE OF FUNCTION IN NUMERICAL FORMAT
- MORE INSTRUCTION/FUNCTION A MODEL EXECUTES, BIGGER THE SIZE
- BIGGER THE SIZE BEEFIER THE COMPUTE RESOURCES LIKE GPU ARE REQUIRED
- COMPACT MODELS MEANS LOWER RUNNING COSTS



IMG:HTTPS://HUGGINGFACE.CO/BLOG/HF-BITSANDBYTES-INTEGRATION

BLOOM INFERENCE : 8 X 80 A100

BLOOM FINETUNING : 72 X 80 A100

BLOOM : 176B VS PALM : 540B

HTTPS://GITHUB.COM/INSIGHTBUILDER

DO THE MATH ON THE RESOURCES

WHY THE MODELS ARE HUGE?

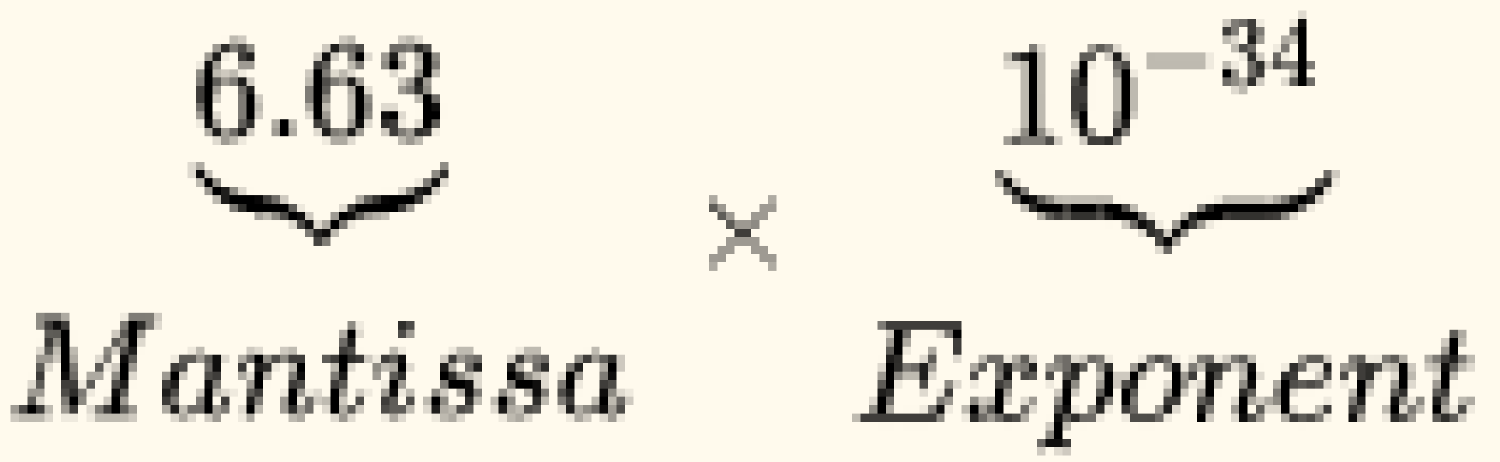
HIGHER FLOATING POINT PRECISION + MORE PARAMETER – BIGGER THE SIZE

WHY UNDERFLOWING AND OVERFLOWING OF THE DATA TYPE

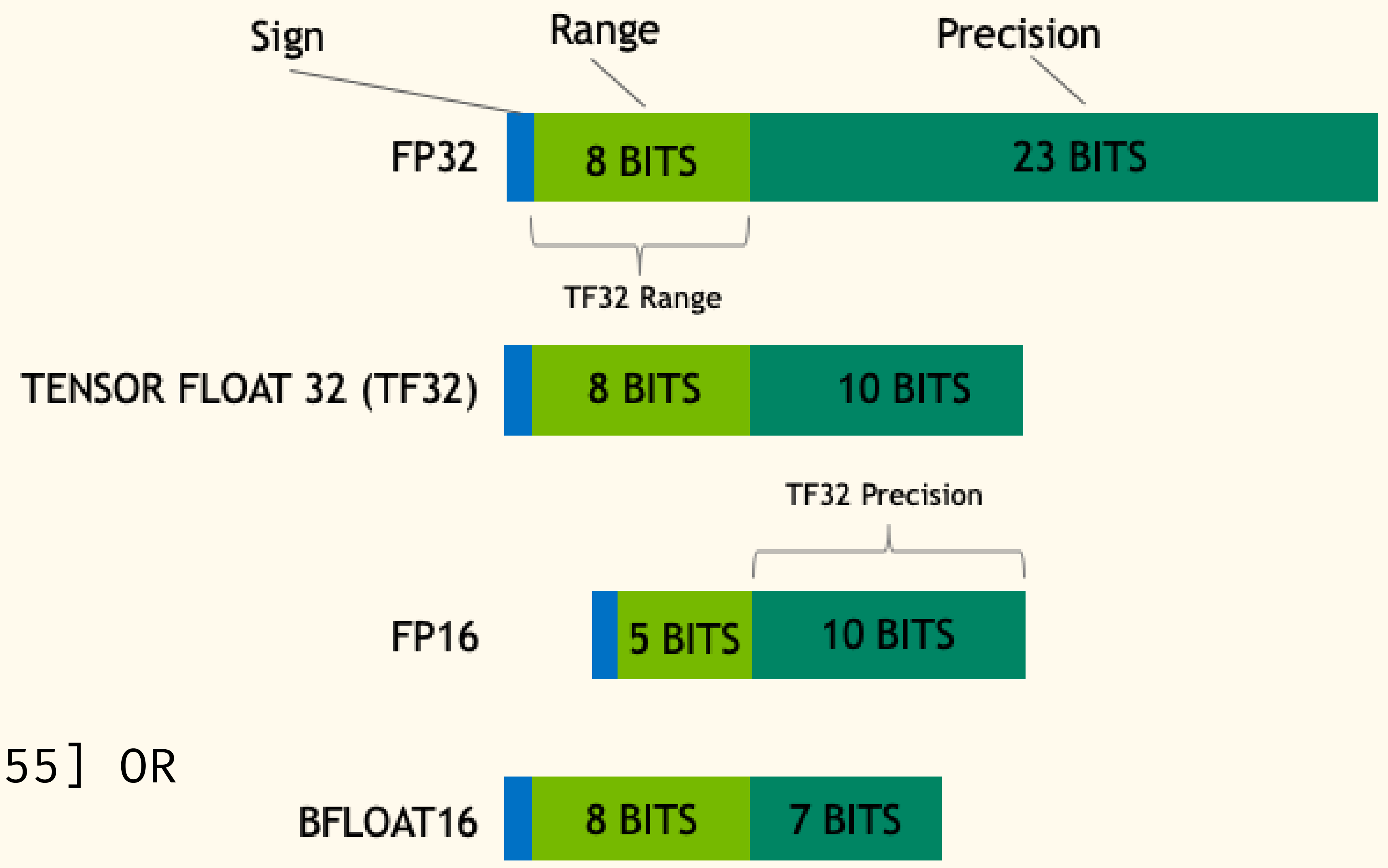
- FP-32:~1.18E-38 … ~3.40E38 WITH 6–9 SIGNIFICANT DECIMAL DIGITS PRECISION.
- FP-16: UPTO 64K 4 SIGNIFICANT DECIMAL DIGITS PRECISION.
- BF-16: ~1.18E-38 … ~3.40E38WITH 3 SIGNIFICANT DECIMAL DIGITS
- TF-32: ~1.18E-38 … ~3.40E38 WITH 4 SIGNIFICANT DECIMAL DIGITS PRECISION.

MACHINE LEARNING JARGON:

1. FP32 : FULL PRECISION (4 BYTES),
2. BF16 AND FP16 : HALF-PRECISION
3. INT8 (INT8): AN 8-BIT REPRESENTATION (BETWEEN [0, 255] OR [-128, 127])



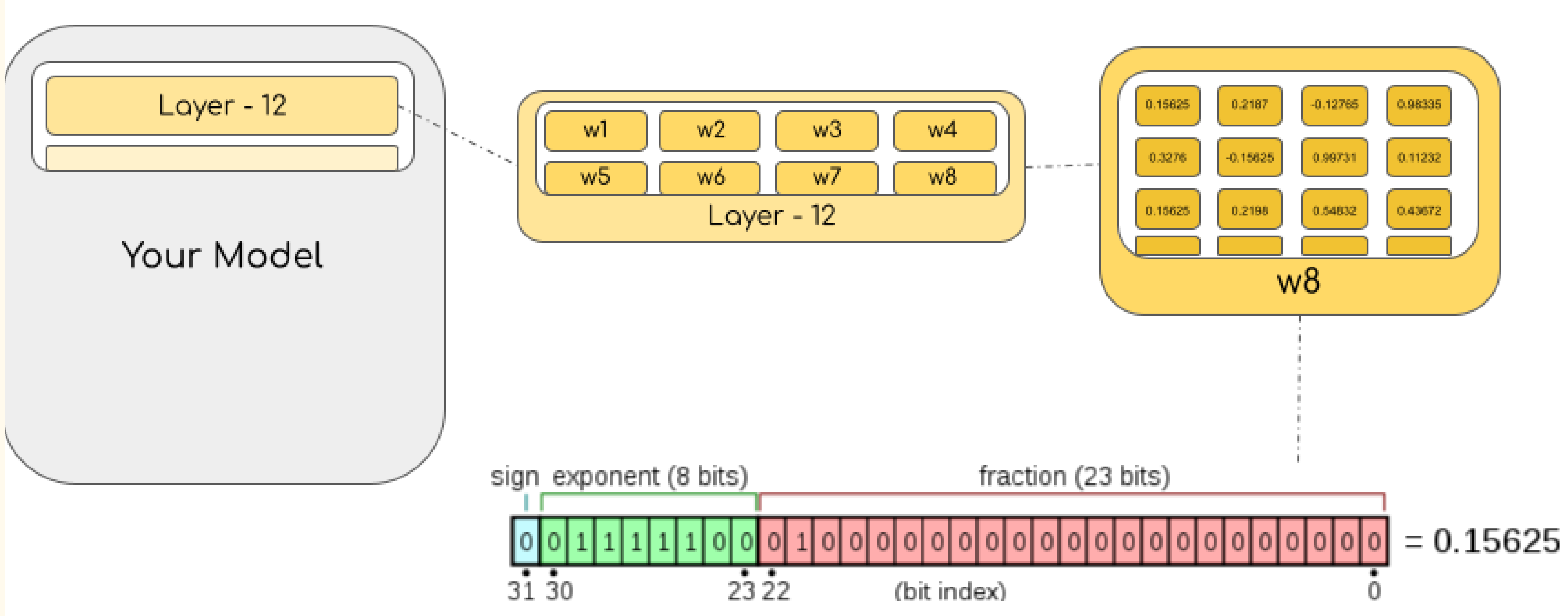
FUNDAMENTALS OF DATA REPRESENTATION: FLOATING POINT NUMBERS



HTTPS://GITHUB.COM/INSIGHTBUILDER

KEY POINT

MAIN WEIGHTS ARE ALWAYS STORED IN FP32, BUT IN PRACTICE, THE HALF-PRECISION WEIGHTS OFTEN PROVIDE SIMILAR QUALITY DURING INFERENCE AS THEIR FP32 COUNTERPART. WE CAN USE THE HALF-PRECISION WEIGHTS AND USE HALF THE GPUS TO ACCOMPLISH THE SAME OUTCOME.



CALCULATE MODEL SIZE

DATA TYPE: BFLOAT16
MODEL : BLOOM-176B,
SIZE = 176*10**9 X 2 BYTES = 352GB!
AS DISCUSSED EARLIER, THIS IS QUITE A CHALLENGE TO FIT INTO A FEW GPUS.

WHAT IS QUANTISATION?

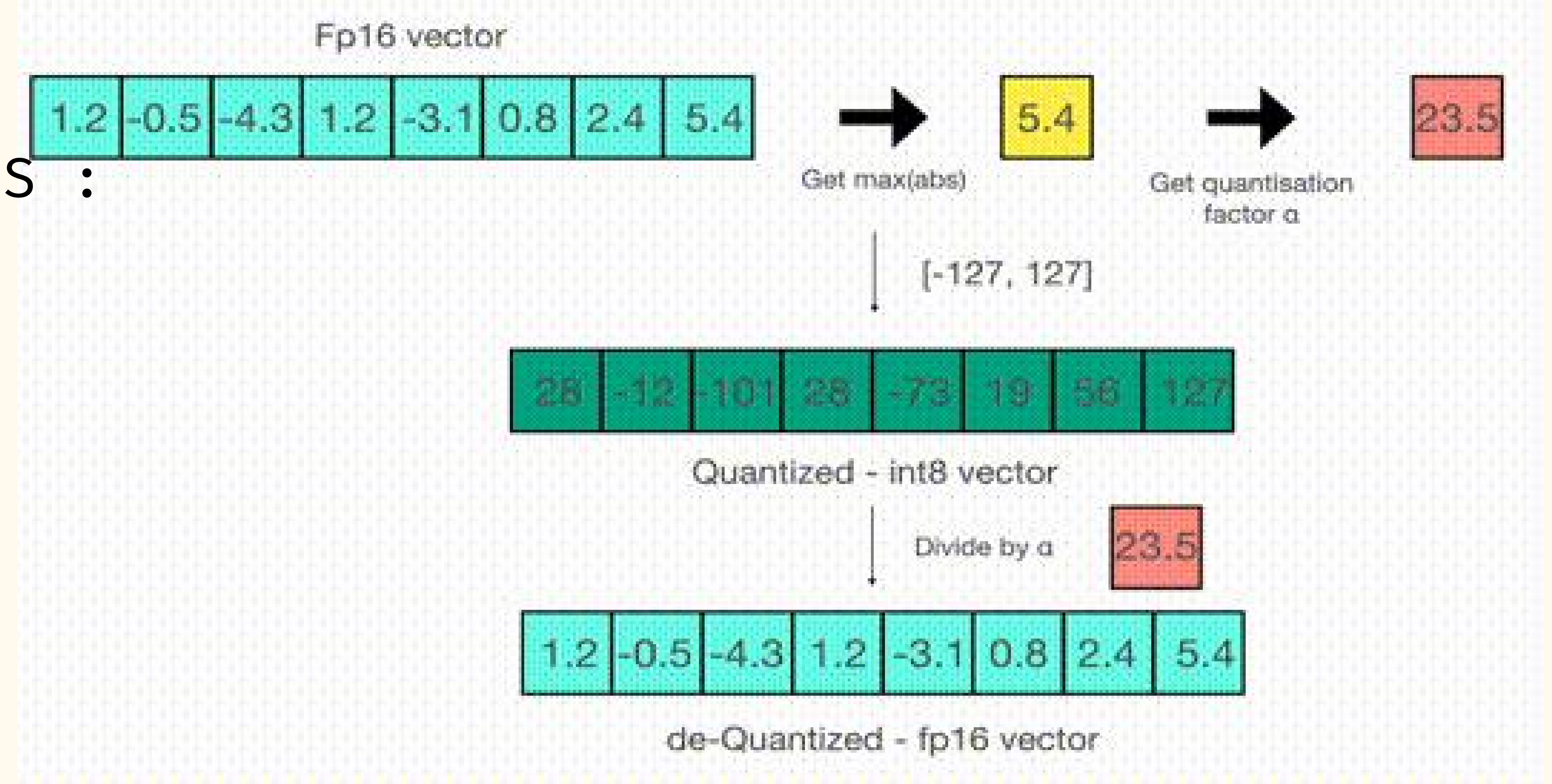
MODEL WEIGHTS ARE ALWAYS STORED IN FP32. IN PRACTICE, THE HALF-PRECISION WEIGHTS PROVIDE SIMILAR QUALITY DURING INFERENCE AS THEIR FP32 COUNTERPART. WE CAN USE THE HALF-PRECISION WEIGHTS AND USE HALF THE GPUS TO ACCOMPLISH THE SAME OUTCOME.

QUANTISATION IS BASICALLY ROUNDING. THE PROCESS OF ROUNDING THE BIGGER SIZED PARAMETERS TO SMALLER SIZE, WHICH REDUCES THE SIZE OF THE MODEL

8-BIT QUANTISATION REDUCES THE MODEL SIZE TO 1/4. 4-BYTE FP32 --> 1-BYTE

TWO 8-BIT QUANTIZATION TECHNIQUES :
1) ZERO-POINT
2) ABSOLUTE MAXIMUM (ABSMAX)

THE LLM.INT8() IMPLEMENTATION THAT WE INTEGRATED INTO HUGGING FACE TRANSFORMERS AND ACCELERATE LIBRARIES IS THE FIRST TECHNIQUE THAT DOES NOT DEGRADE PERFORMANCE EVEN FOR LARGE MODELS WITH 176B PARAMETERS, SUCH AS BLOOM.



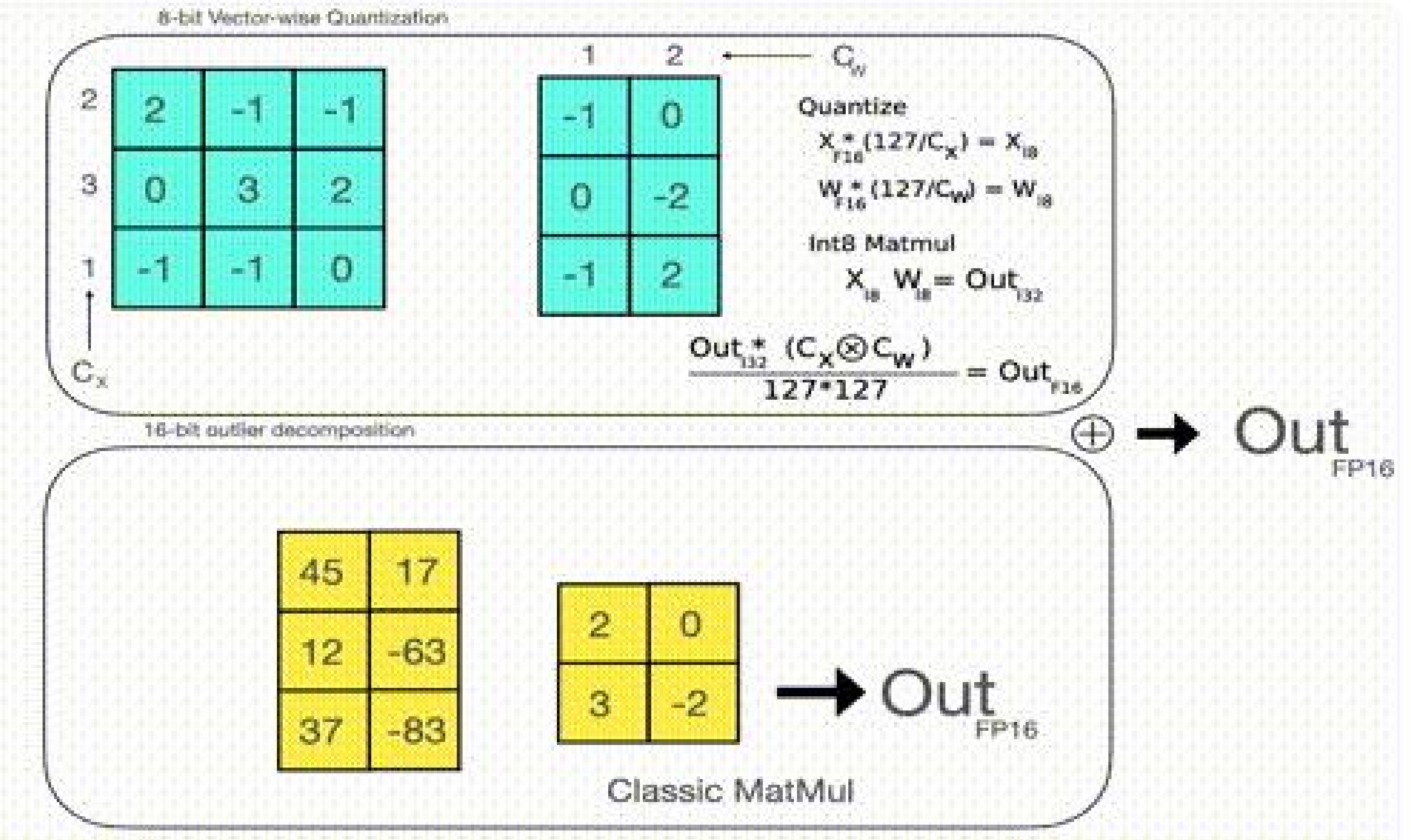
HTTPS://GITHUB.COM/INSIGHTBUILDER

LLM.INT8() ALGORITHM

CHALLENGE SOLVED : MAINTAIN THE ACCURACY & PERFORMANCE OF THE QUANTISED MODELS

LLM.INT8() SEEKS TO COMPLETE THE MATRIX MULTIPLICATION COMPUTATION IN THREE STEPS:

- FROM THE INPUT HIDDEN STATES, EXTRACT THE OUTLIERS (I.E. VALUES THAT ARE LARGER THAN A CERTAIN THRESHOLD) BY COLUMN.
- PERFORM THE MATRIX MULTIPLICATION OF THE OUTLIERS IN FP16 AND THE NON-OUTLIERS IN INT8.
- DEQUANTIZE THE NON-OUTLIER RESULTS AND ADD BOTH OUTLIER AND NON-OUTLIER RESULTS TOGETHER TO RECEIVE THE FULL RESULT IN FP16.



QUANTISATION RESULTS

For OPT-175B:

benchmarks	-	-	-	-	difference - value
name	metric	value - int8	value - fp16	std err - fp16	-
hellaswag	acc_norm	0.7849	0.7849	0.0041	0
hellaswag	acc	0.5921	0.5931	0.0049	0.001
piqa	acc	0.7965	0.7959	0.0094	0.0006
piqa	acc_norm	0.8101	0.8107	0.0091	0.0006
lambada	ppl	3.0142	3.0152	0.0552	0.001
lambada	acc	0.7464	0.7466	0.0061	0.0002
winogrande	acc	0.7174	0.7245	0.0125	0.0071

8-BIT TENSOR CORES ARE NOT SUPPORTED ON THE CPU. BITSANDBYTES CAN BE RUN ON 8-BIT TENSOR CORE-SUPPORTED HARDWARE, WHICH ARE TURING AND AMPERE GPUS (RTX 20S, RTX 30S, A40-A100, T4+). FOR EXAMPLE, GOOGLE COLAB GPUS ARE USUALLY NVIDIA T4 GPUS, AND THEIR LATEST GENERATION OF GPUS DOES SUPPORT 8-BIT TENSOR CORES.

- QUANTISED MODELS CAN BE SLOWER THAN THEIR REGULAR MODELS, ESPECIALLY IN SLOWER MODELS
- THERE IS 0-DEGRADATION IN MODEL ACCURACY

Precision	Number of parameters	Hardware	Time per token in milliseconds for Batch Size 1	Time per token in milliseconds for Batch Size 8	Time per token in milliseconds for Batch Size 32
bf16	176B	8xA100 80GB	239	32	9.9
int8	176B	4xA100 80GB	282	37.5	10.2
bf16	176B	14xA100 40GB	285	36.5	10.4
int8	176B	5xA100 40GB	367	46.4	oom
fp16	11B	2xT4 15GB	11.7	1.7	0.5
int8	11B	1xT4 15GB	43.5	5.3	1.3
fp32	3B	2xT4 15GB	45	7.2	3.1
int8	3B	1xT4 15GB	312	39.1	10.2

FURTHER IMPROVEMENT & REFERENCE

- FASTER INFERENCE FOR SMALLER MODELS
- SUPPORT FOR INT8 VECTOR CORES INSTEAD OF TENSOR CORES
- 8-BIT MODELS CANNOT BE DIRECTLY PUSHED TO THE HUB
- CPU DON'T SUPPORT THE 8-BIT CORES
- SCALING THIS FOR VISION/ AUDIO & RELATED MODALITIES WILL BE A PLUS

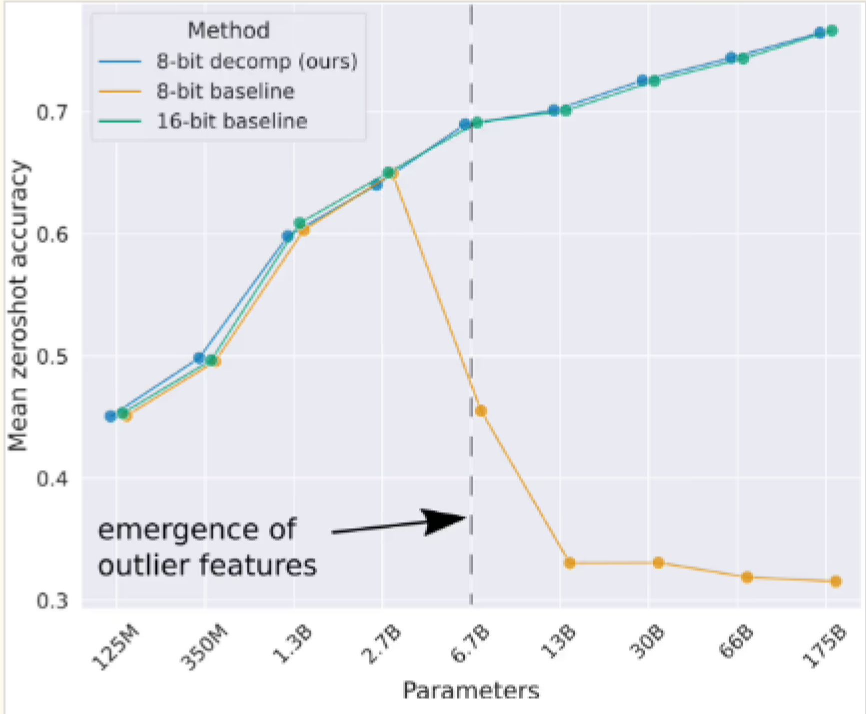
FP64, FP32, FP16, BFLOAT16, TF32, and other members of the ZOO

There are many floating point formats you can hear about in the context of deep learning. Here is a summary of what are they about and...

A Gentle Introduction to 8-bit Matrix Multiplication for transformers at scale using transformers, accelerate and...

We're on a journey to advance and democratize artificial intelligence through open source and open science.

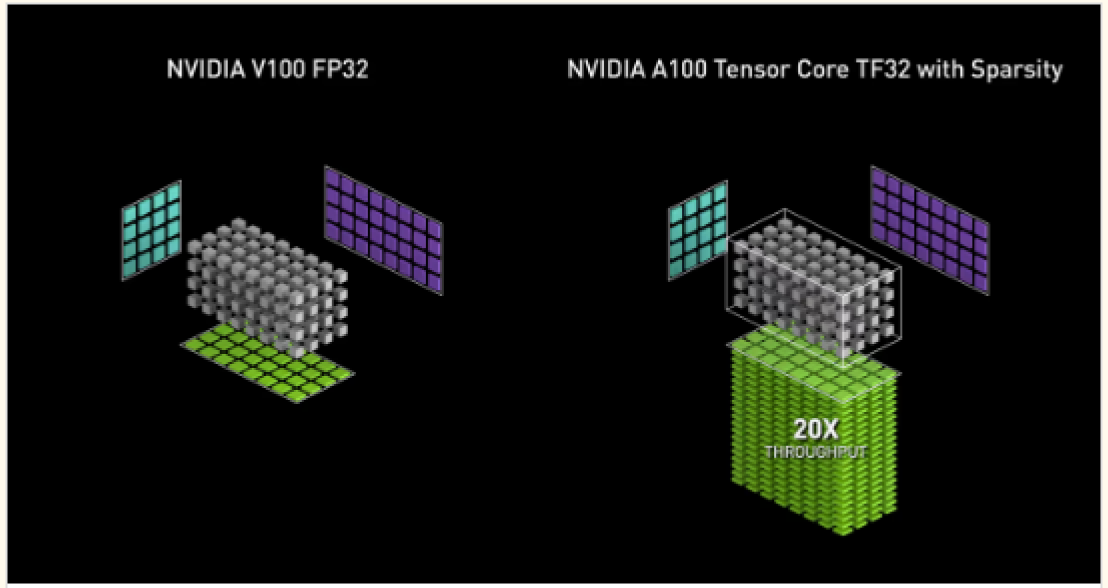
huggingface



LLM.int8() and Emergent Features

When I attended NAACL, I wanted to do a little test. I had two pitches for my LLM.int8() paper. One pitch is about how I use advanced quantization methods to achieve no...

Tim Dettmers / Aug 17, 2022



NVIDIA Blogs: TensorFloat-32 Enables Performance Gains

TensorFloat-32 provides a huge out-of-the-box performance increase for AI applications for training & inference.

NVIDIA Blog / May 14, 2020

HTTPS://GITHUB.COM/INSIGHTBUILDER

THANKS FOR WATCHING

REMEMBER TO PRACTICE WITH EXAMPLES

LIKE

SHARE

SUBSCRIBE

HTTPS://GITHUB.COM/INSIGHTBUILDER