LLM.int8()
+
OPTQ

Martin Jaggi

4. April 2023, EPFL CS-723 Course

LLM.int8() - Elevator Pitch v1

- int8():
  Run a transformer with **8bit** ops instead of 32

- some models can now be used on *Google Colab* that previously couldn't

LLM.int8() - Elevator Pitch v2

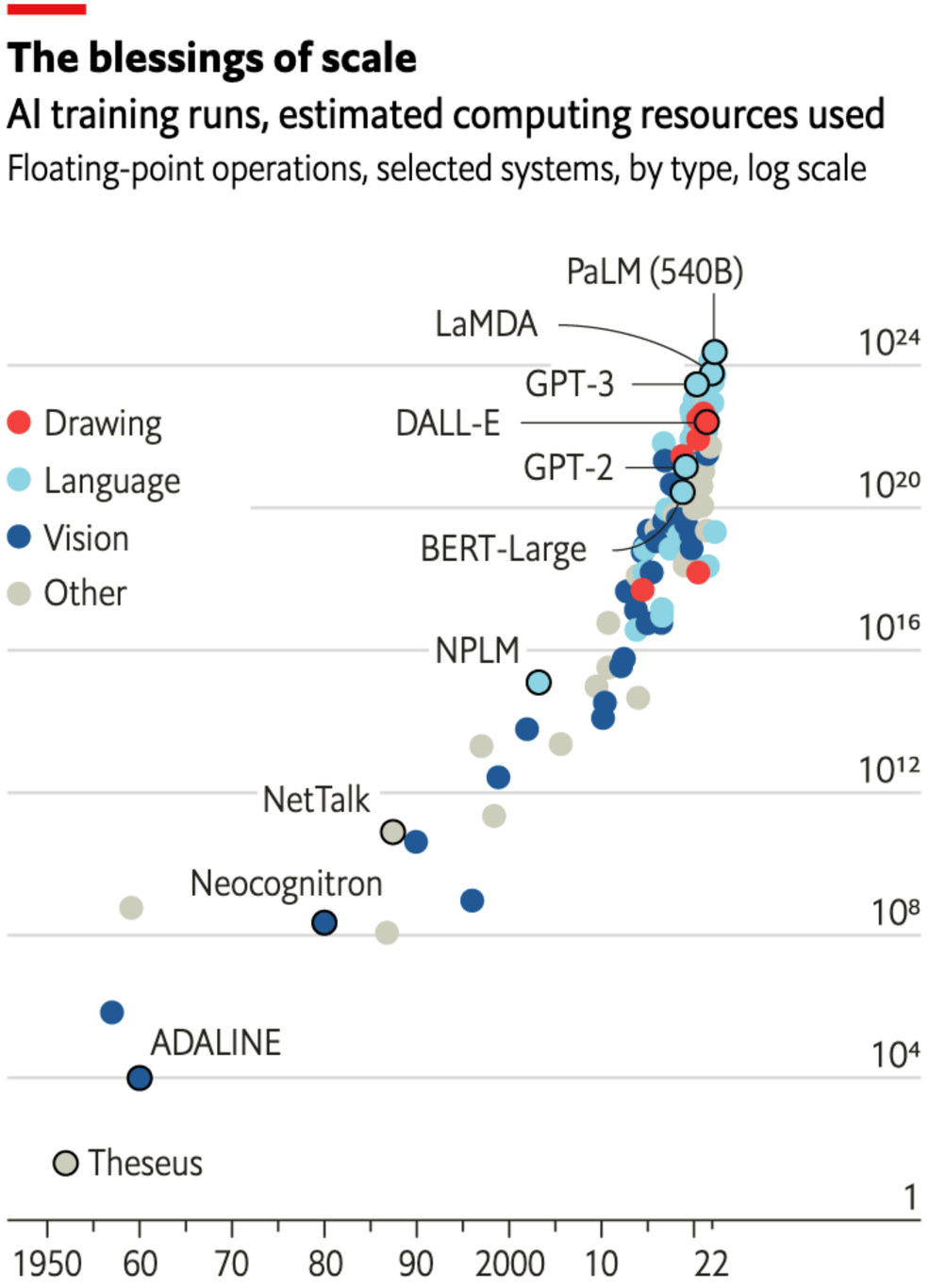
- int8():
  Run a transformer with **8bit** ops instead of 16

- reduce the **memory footprint** of a large model by 2x (compared to FP16)

- some models can now be used on *Google Colab* that previously couldn't

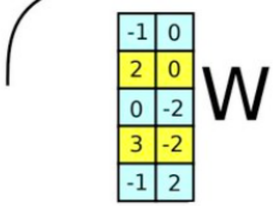
1) Problem importance

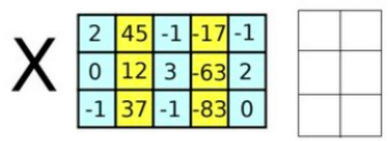
- **L**LMs are large
- Inference cost is crucial (memory & compute)

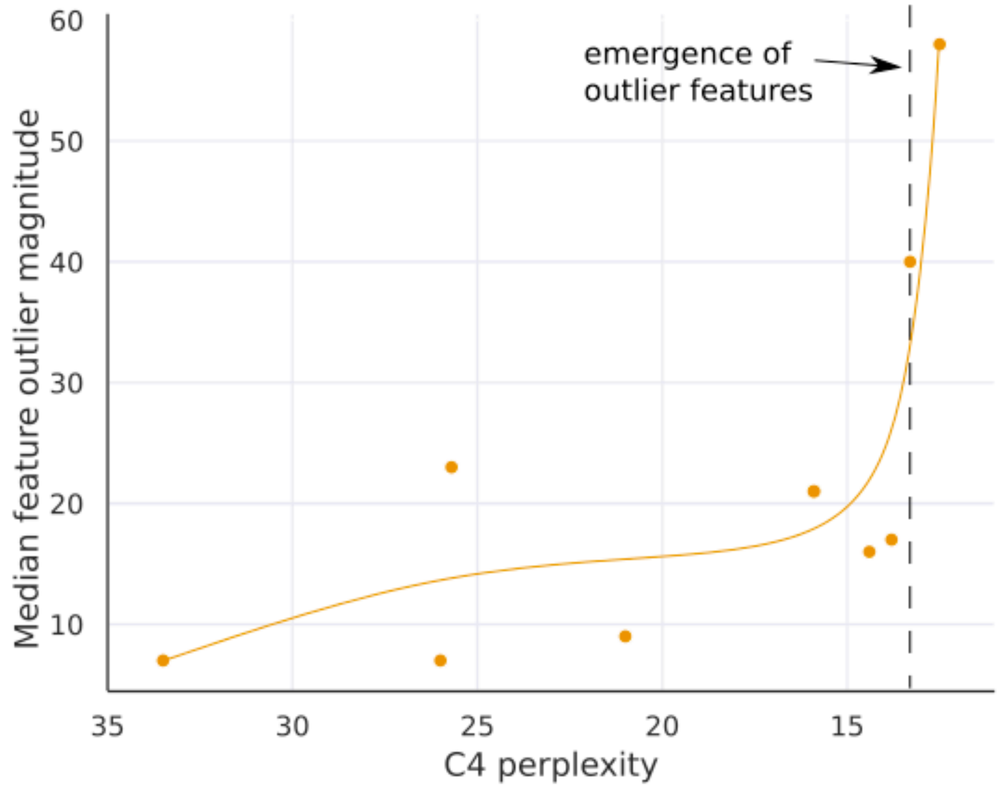


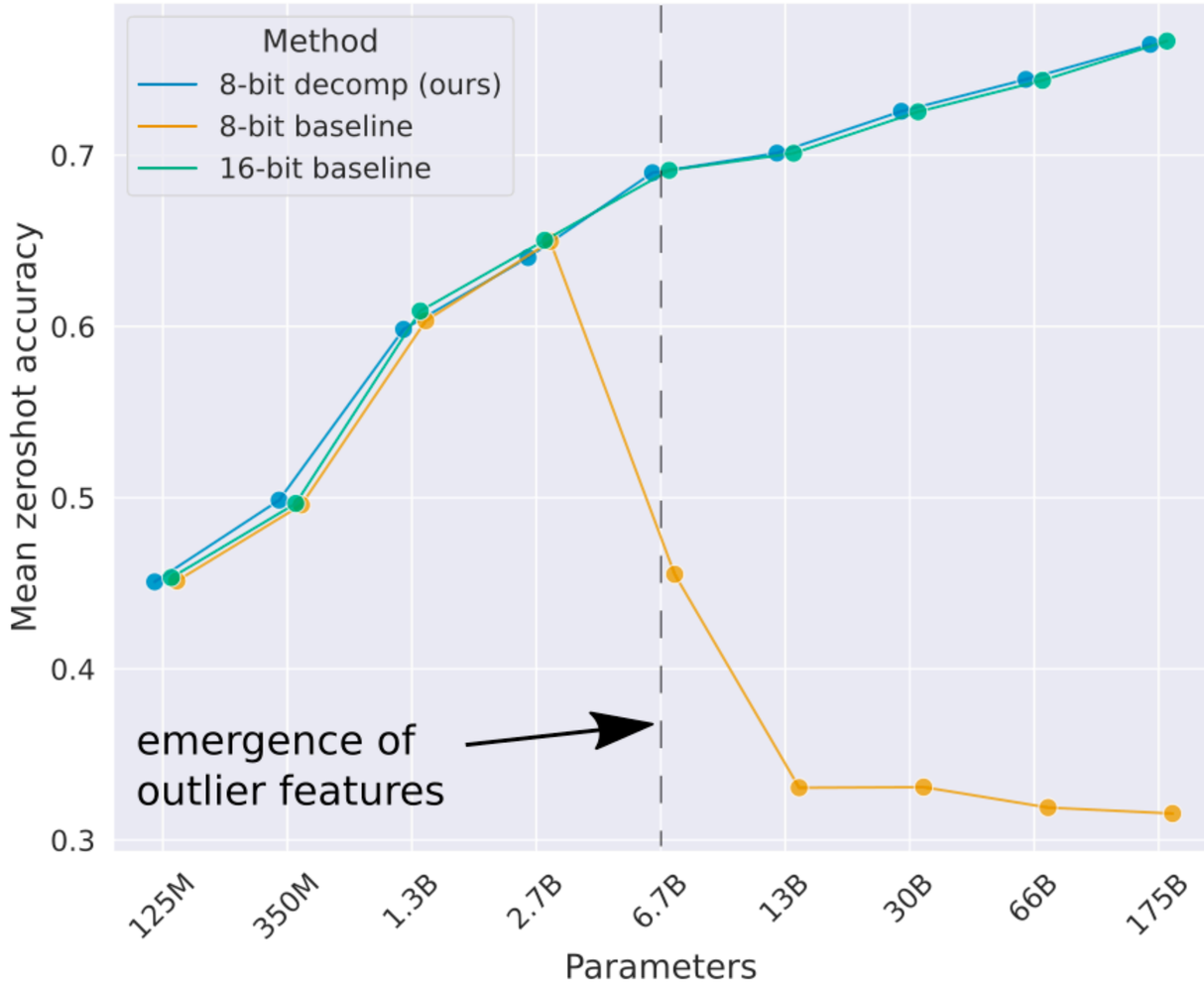
2) Insights

- Emergence of outlier features



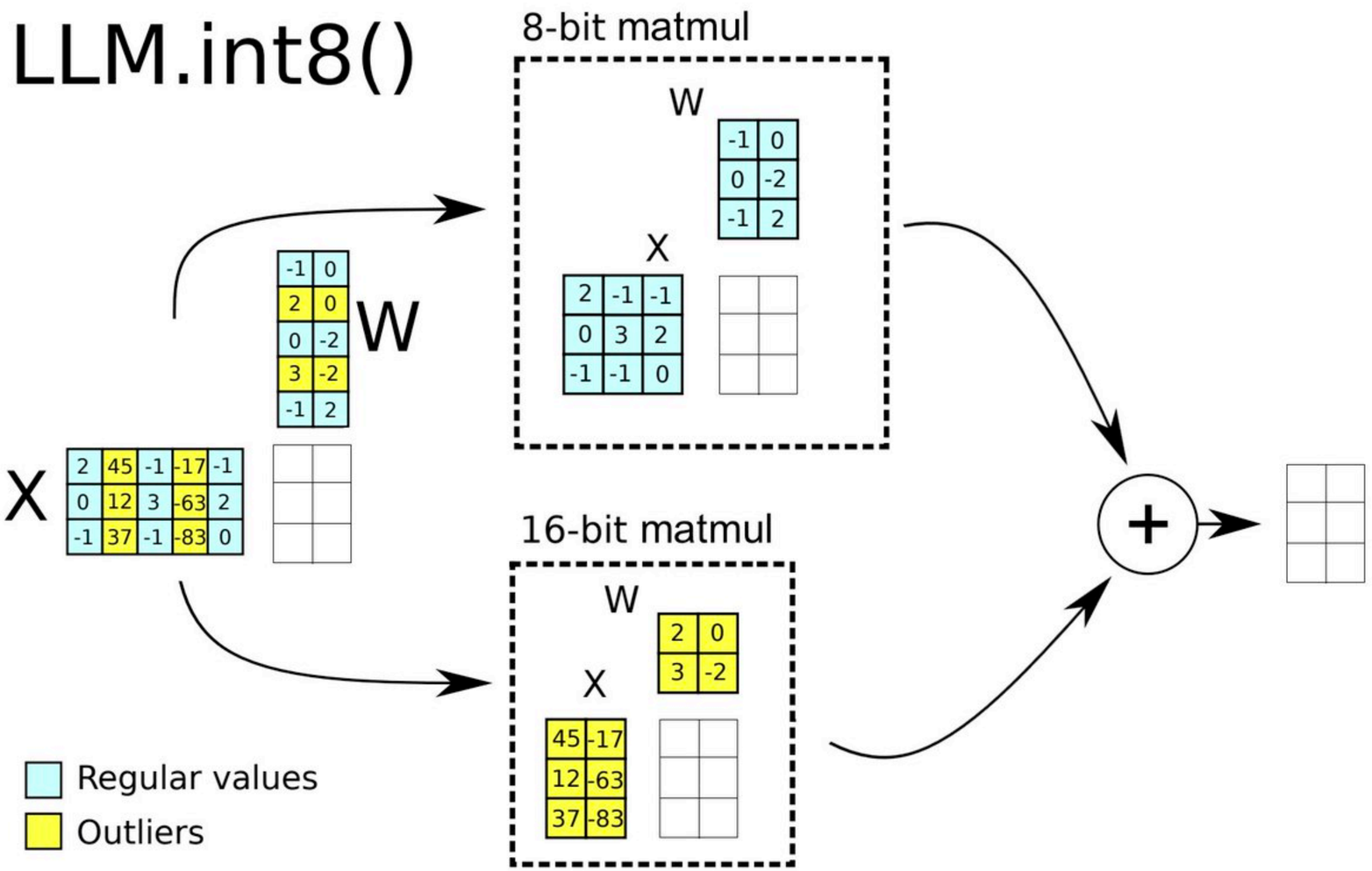






3) Solution

- Treat regular & outlier values separately
- Integration into hugging face



4) Takeaway

- reduce the **memory footprint** of a large model by *(only)* 2x.
- some models can now be used on *(insert particular hardware)* that previously couldn't
- any huggingface model can directly be converted
- no obvious speedup (only memory saving)

Extra

- Two papers addressing same problem:
  - LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale
    NeurIPS 2022 *(arXiv Aug 2022)*
  - OPTQ: Accurate Quantization for Generative Pre-trained Transformers
    ICLR 2023 (arXiv Oct 2022)

FAST: DNN training under variable precision Block Floating Point with Stochastic Rounding

CS-723 Presentation 4th April, 2023

By Ayan Chakraborty

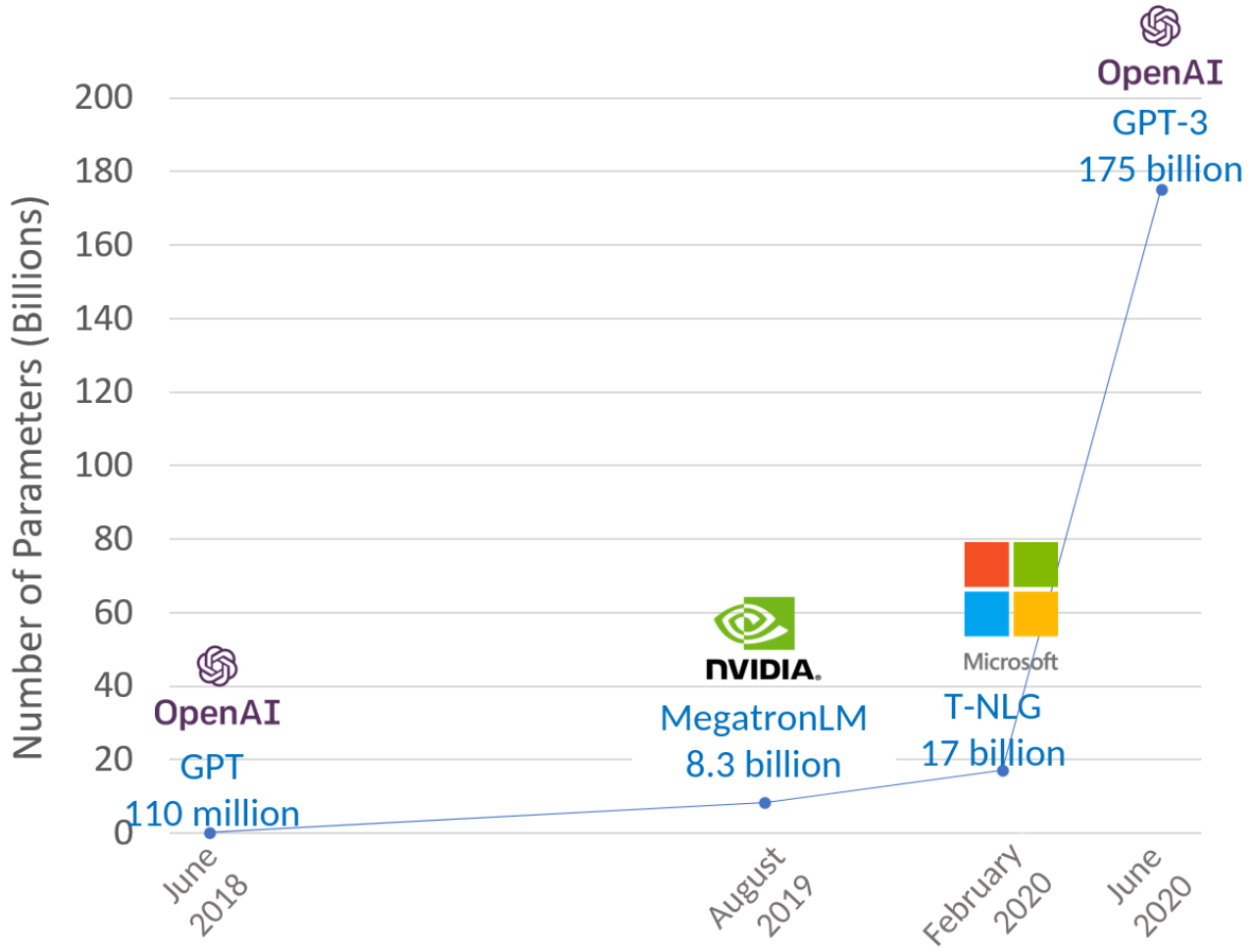
PARALLEL SYSTEMS ARCHITECTURE LAB

EPFL

PARALLEL SYSTEMS ARCHITECTURE LAB

Elevator Pitch

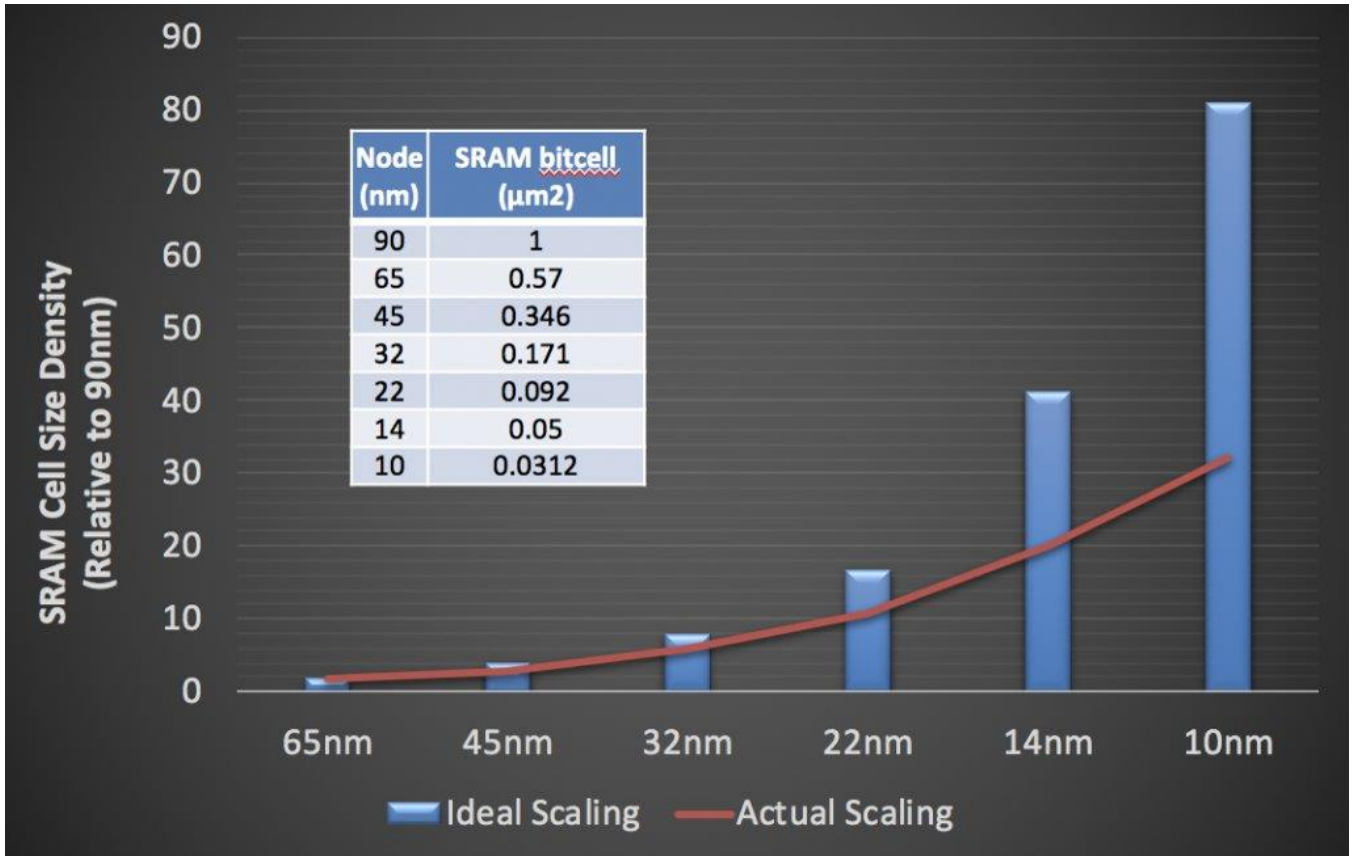
Exponential growth of DNNs



Commensurate increase in computational cost and carbon emissions!

Slowdown of Moore's Law

PARALLEL SYSTEMS ARCHITECTURE LAB



Node (nm)	SRAM bitcell (µm2)
90	1
65	0.57
45	0.346
32	0.171
22	0.092
14	0.05
10	0.0312

Need for alternate ways to increase Arithmetic Density

Use new numerical encodings for DNNs

PARALLEL SYSTEMS ARCHITECTURE LAB

- Block Floating Point promising candidate for DNNs
- Combines the advantages of Fixed point and Floating point
- Previous work does not manage to reduce precision below 8 bits for training

- The paper shows how you can use variable precision to use even lower precision for most operations

- The paper designs a custom HW accelerator to get the maximum benefits

Increase hardware efficiency while maintaining accuracy

PARALLEL SYSTEMS ARCHITECTURE LAB

What is the Problem?

Unexplored design space of BFP

PARALLEL SYSTEMS ARCHITECTURE LAB

- Prior work does not manage to use BFP with precision below 8 bits to get high accuracy.
- Variable precision and Mixed precision training used for other formats before but not studied for BFP.
- Big unexplored design space of BFP configurations below 8 bits.
- HW support for BFP in general, and more specifically for variable precision BFP not studied well in prior work.

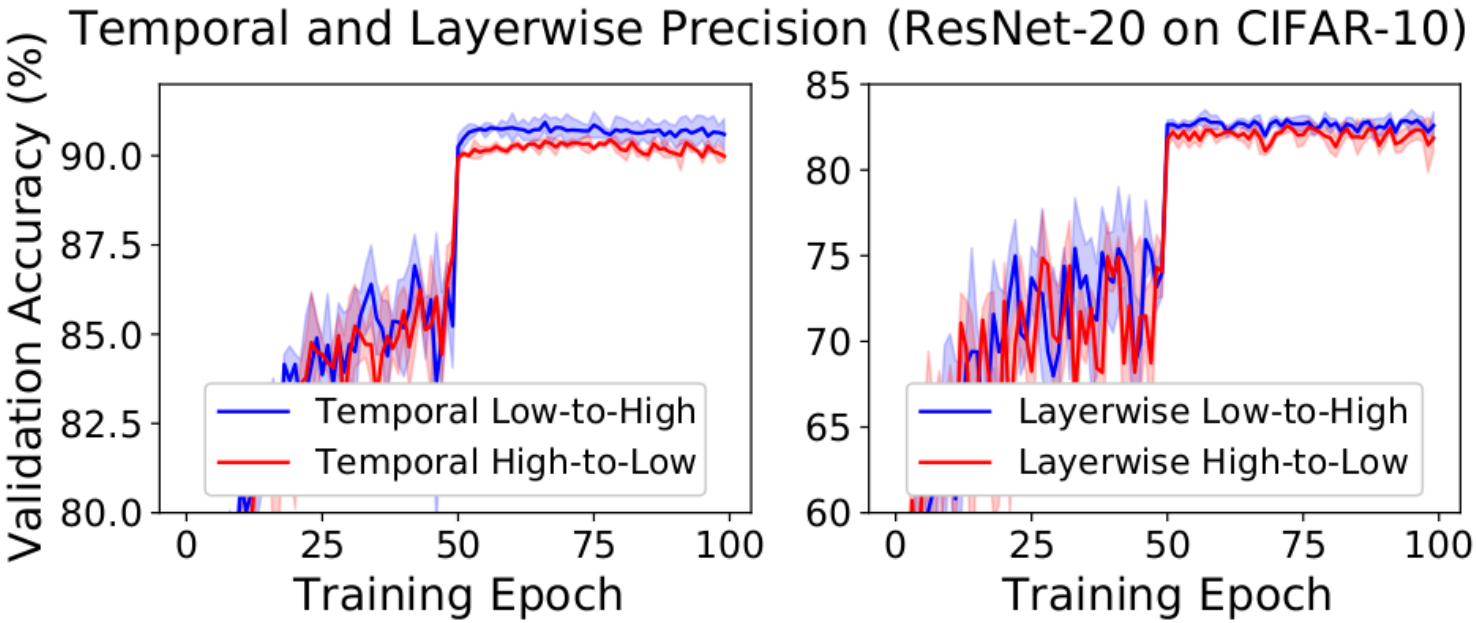
PARALLEL SYSTEMS ARCHITECTURE LAB

What are the insights?

Variable precision

- The precision of the model parameters does not need to stay fixed during training and even across model layers.

- Early layers of a model and the initial epochs of a training session are more tolerant to lower precision.

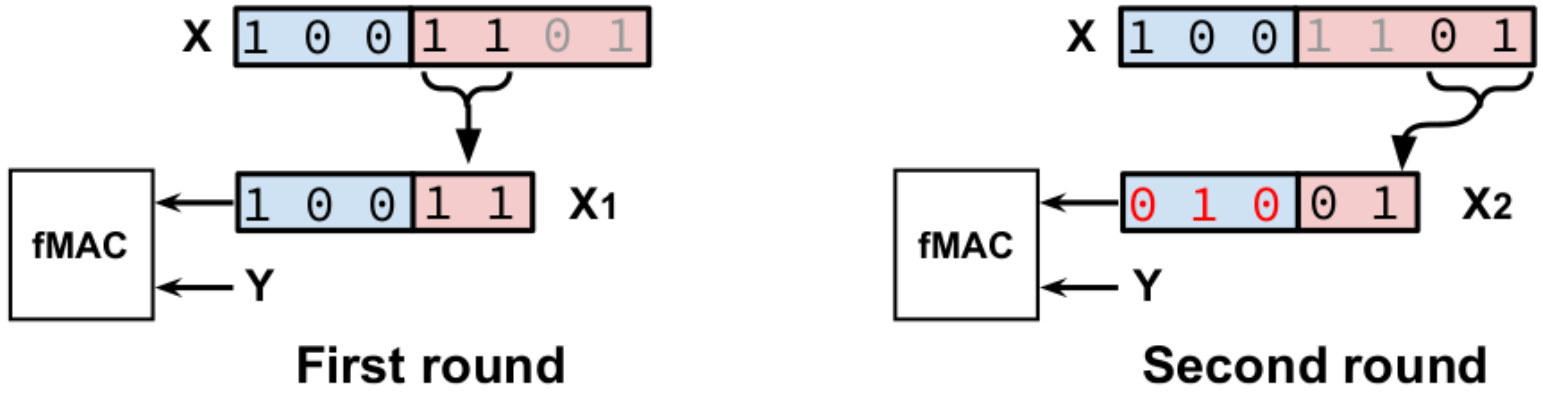


HW support for variable precision

PARALLEL SYSTEMS ARCHITECTURE LAB

- Sub-divide the computation into 2-bit chunks

- Allows the same custom HW unit to implement arithmetic operations involving higher precision mantissa by simply running multiple passes of the unit.

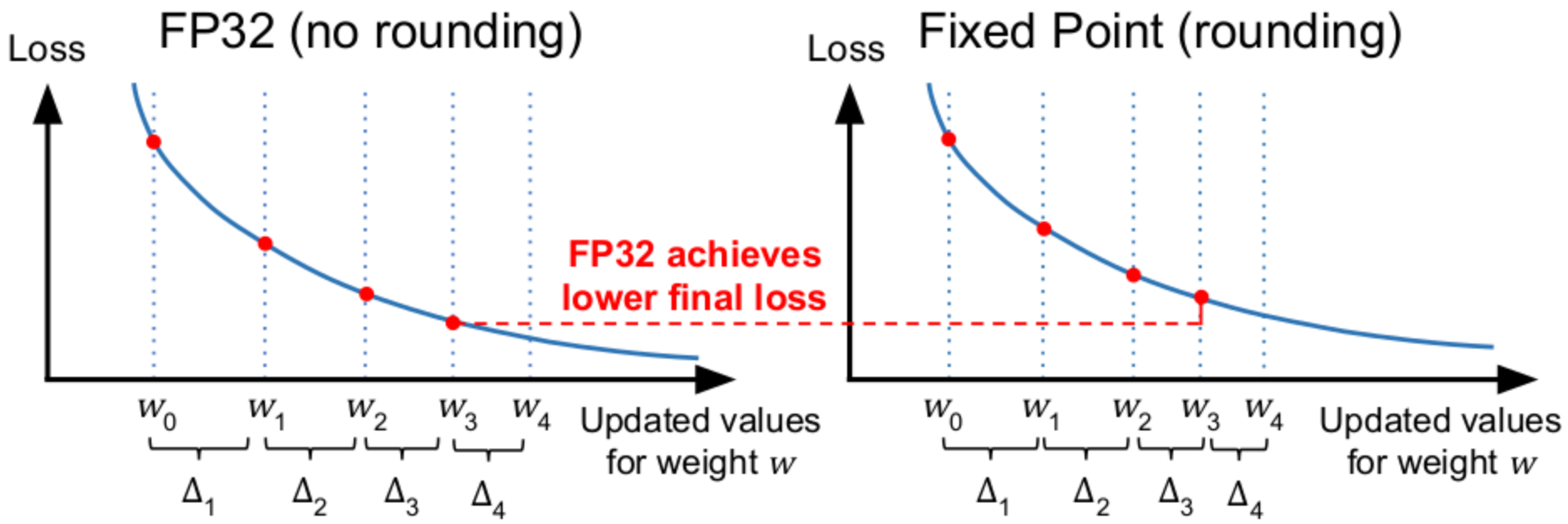


Stochastic Rounding

PARALLEL SYSTEMS ARCHITECTURE LAB

- Randomly round up or round down when converting tensors from FP32 to BFP.

- Critical to maintaining training stability when using very low precision.



PARALLEL SYSTEMS ARCHITECTURE LAB

What is the solution?

Schedule for variable precision

PARALLEL SYSTEMS ARCHITECTURE LAB

- Use 4-bit mantissas in combination with 2-bit mantissas

- Check if relative improvement of using the higher precision is smaller than a threshold for a given layer and iteration

$$r(X) = \frac{\sum_n |BFP(X_n, 4) - BFP(X_n, 2)|}{\sum_n |BFP(X_n, 2)|}$$

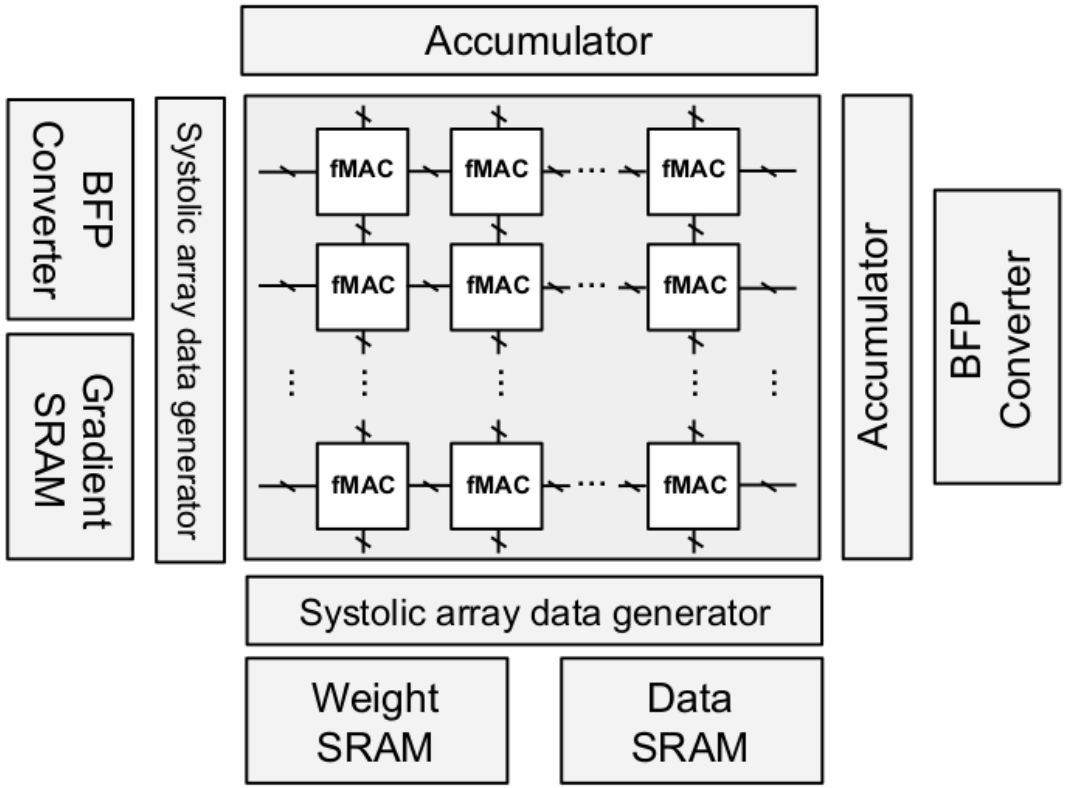
- Scale the threshold linearly across the layers and iterations

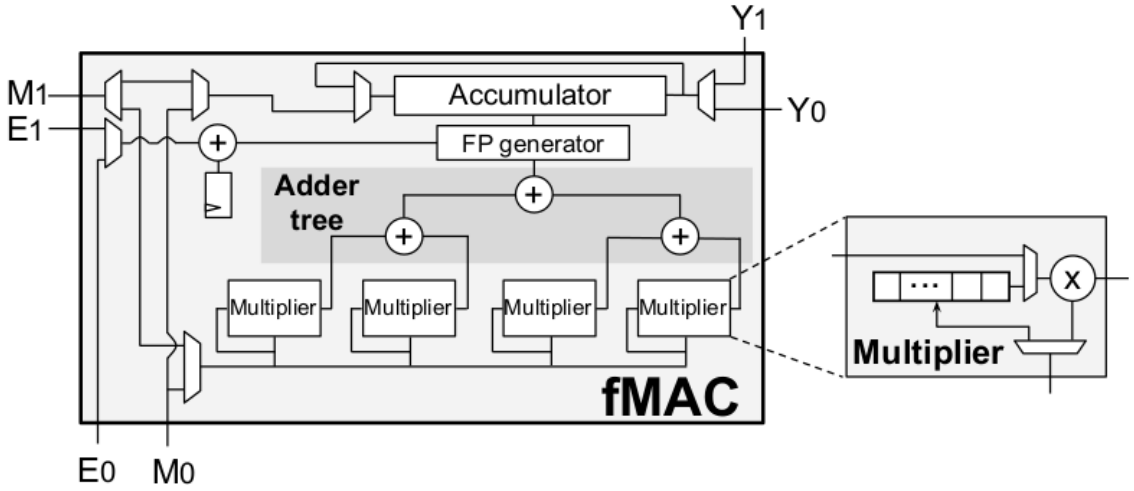
$$\varepsilon(l, i) = \alpha - \beta \frac{i}{I} - \beta \frac{l}{L}$$

HW accelerator for variable precision BFP

PARALLEL SYSTEMS ARCHITECTURE LAB

- Systolic array based accelerator with custom MAC unit.





- Novel memory layout to store variable precision BFP values.

PARALLEL SYSTEMS ARCHITECTURE LAB

What is the take-away message?

Dynamic precision is the way to go

PARALLEL SYSTEMS ARCHITECTURE LAB

- Dynamic precision enables the usage of BFP precisions below 8 bits to get comparable high performance.
- Dynamic precision allows most operations to be done at a much reduced precision.
- Building a custom HW accelerator allows you to extract the maximum benefits of dynamic precision BFP operations.

PARALLEL SYSTEMS ARCHITECTURE LAB

Will this paper win the Test of Time award?

No

PARALLEL SYSTEMS ARCHITECTURE LAB

- ML highly volatile field.
- Unlikely only using BFP by itself will turn out to be a good choice.
- Does not have any new insights which can be applied to any number format.
- Unclear how HW would generalize well for any mantissa width.
- Unlikely results shown in the paper will generalize to current LLM models.

PARALLEL SYSTEMS ARCHITECTURE LAB

Why should this paper not have appeared at HPCA?

Highly empirical with no new insights

PARALLEL SYSTEMS ARCHITECTURE LAB

- The idea of mixed precision training is not new, and the paper simply adapts it for BFP without providing any insight as to why it works.
- The design choices in the paper are mostly based on empirical results with no theoretical backing.
- The training accuracies are not too high, and it is hard to believe SOTA performance for current models can be gained using only 4-bits and 2-bits
- Design not open sourced, makes it hard to reproduce results

Thank you