EFFICIENT AI COMPUTE PLATFORM

REAL TIME CONTROL FOR ADAPTIVE OPTICS (RTC4AO)
CONFERENCE / WORKSHOP
6 - 8 NOVEMBER 2023
ESO GARCHING

GRAPHCORE

Gautier Soubrane
Business Development Director

Neural network visualization from POPLAR™

GRAPHCORE

Graphcore is a UK-based global chip design company established in 2016.

Graphcore has designed the Intelligence Processing Unit (IPU) from the ground up considering the requirements of Machine Intelligence workloads.

Find out more at www.graphcore.ai

GRAPHCORE'S MISSION

Deliver the world's best processors

and software tools, to support the next

breakthroughs in machine intelligence,

that will expand human potential.

The Hardware Lottery

Sara Hooker

Google Research, Brain Team
shooker@google.com

Abstract

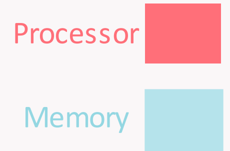
Hardware, systems and algorithms research communities have historically had different incentive structures and fluctuating motivation to engage with each other explicitly. This historical treatment is odd given that hardware and software have frequently determined which research ideas succeed (and fail). This essay introduces the term hardware lottery to describe when a research idea wins because it is suited to the available software and hardware and *not* because the idea is superior to alternative research directions. Examples from early computer science history illustrate how hardware lotteries can delay research progress by casting successful ideas as failures. These lessons are particularly salient given the advent of domain specialized hardware which make it increasingly costly to stray off of the beaten path of research ideas. This essay posits that the gains from progress in computing are likely to become even more uneven, with certain research directions moving into the fast-lane while progress on others is further obstructed.

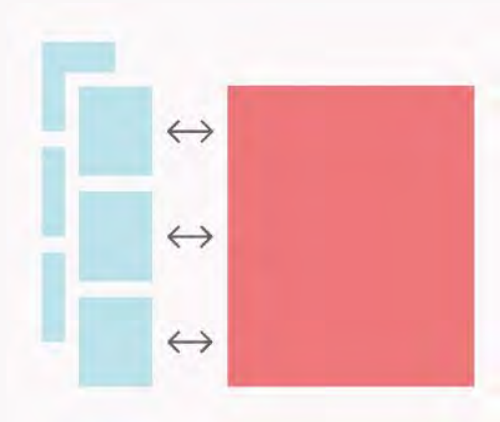
HARDWARE CAN BE AN ENABLING OR A LIMITING FACTOR IN THE SUCCESS OF IDEAS

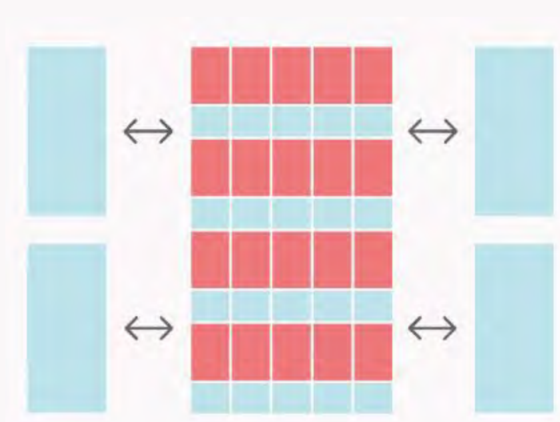
THE INTELLIGENCE PROCESSING UNIT (IPU)

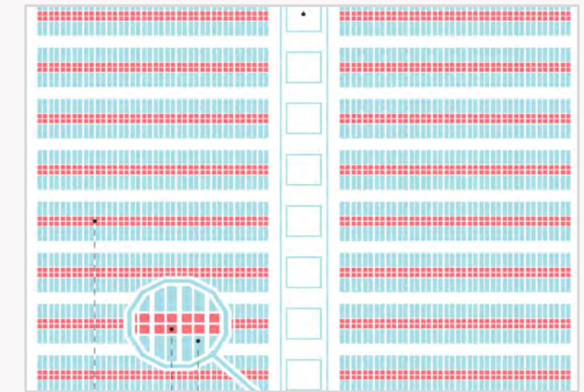
WHAT MAKES IT DIFFERENT?

	CPU	GPU	IPU
Parallelism	Designed for scalar processing	SIMD/SIMT architecture. Designed for large blocks of dense contiguous data	Massively parallel MIMD architecture. High performance/efficiency for future ML trends
Memory Bandwidth	Off-chip memory	Model and Data spread across off-chip and small on-chip cache and shared memory (2TB/s for A100 HBM)	Main Model & Data in tightly coupled large locally distributed SRAM (~65 TB/s for Bow IPU)









INTRODUCING THE BOW IPU
WORLD'S FIRST 3D WAFER-ON-WAFER PROCESSOR

3D silicon wafer stacked processor

350 TeraFLOPS AI compute

Optimized silicon power delivery

0.9 GigaByte In-Processor-Memory @ **65TB/s**

1,472 independent processor cores

8,832 independent parallel programs

10x IPU-Links™ delivering 320GB/s

IPU vs GPU MICROARCHITECTURE

Feature	Bow IPU	A100 GPU
FP16/ FP32 Flops	350 / 87 TFlops	312 / 156 TFlops
Independent Threads (IIS: Independent Instruction Stream)	1472 (tiles) x 6 (threads) = 8832 IIS	108 (SMs) x 64 (warps) = 6912 IIS
SRAM per core	624KiB (per tile)	192KiB (up to 164KiB shared) (per SM)
"Contention free" single-cycle-access memory (SRAM/registers) per IIS	624/6 **= 104 KiB**	192/64 + 256/64 **= 7 KiB**
SIMD vector width per IIS (Lower is better, easier to keep filled)	2 (float32) 4 (float16)	32 (float32) 64 (float16)
Integer	Non-vectorized (but can be dual issued with vector float ops).	Vectorized with above width (but no dual issue).
HW RNG	Yes, based on xoroshiro (enables HW stochastic rounding)	No (but maybe not a bottleneck in most applications?)

IPU unique MICROARCHITECTURE

Steering Customized AI Architectures for HPC Scientific Applications

Hatem Ltaief[1], Yuxi Hong[1], Adel Dabah[1], Rabab Alomairy[1], Sameh Abdulah[1], Chris Goreczny[2], Pawel Gepner[3], Matteo Ravasi[1], Damien Gratadour[4], and David Keyes[1]

[1]Extreme Computing Research Center
Division of Computer, Electrical, and Mathematical Sciences and Engineering
King Abdullah University of Science and Technology
Thuwal, Jeddah 23955 Saudi Arabia
Hatem.Ltaief, Yuxi.Hong, Adel.Dabah.1, Rabab.Alomairy, Sameh.Abdulah, Matteo.Ravasi, David.Keyes@kaust.edu.sa
[2]Graphcore, Poland
chrisgo@graphcore.ai
[3]Warsaw University of Technology, Poland
pawel.gepner@pw.edu.pl
[4]Paris Observatory, France
damien.gratadour@obspm.fr

Abstract. AI hardware technologies have revolutionized computational science. While they have been mostly used to accelerate deep learning training and inference models for machine learning, HPC scientific applications do not seem to directly benefit from these specific hardware features unless AI-based components are introduced into their simulation workflows, for instance, as a replacement of their numerical solvers. This paper proposes to take another direction in an attempt to democratize customized AI architectures for HPC scientific computing. The main idea consists in demonstrating how legacy applications can leverage these AI engines after a necessary algorithmic redesign. It is critical that the resulting software implementations map onto the underlying memory-austere hardware architectures to extract the expected performance. To facilitate this process, we promote the matricization technique for restructuring codes (1) by exploiting data sparsity via algebraic compression and (2) by expressing the critical computational phases in terms of tile low-rank matrix-vector multiplications (TLR-MVM) and batch matrix-matrix multiplications (batch GEMM). Algebraic compression enables to reduce memory footprint and to fit into small local cache/memory, while batch execution ensures high occupancy. We highlight how we can steer the Graphcore AI-focused Wafer-on-Wafer Intelligence Processing Units (IPUs) to deliver high performance for both operations. We conduct a performance benchmarking campaign of these two matrix operations that account for most of the elapsed times of four real applications in computational astronomy, seismic imaging, wireless communications, and climate/weather predictions. We report bandwidth and execution rates with speedup factors up to 150X/14X/25X/40X, respectively, on IPUs compared to other systems.

GRAPHCORE

ESO

Intelligence Processing Unit (IPU)

Competitor to NVIDIA A100/H100

- **Similar FLOPs, but lots more SRAM/thread**

Better perf/$ for most ML workloads

- **IPUs don't rely on expensive HBM logic and sub-7nm manufacturing processes.**

Much better for "interesting" workloads

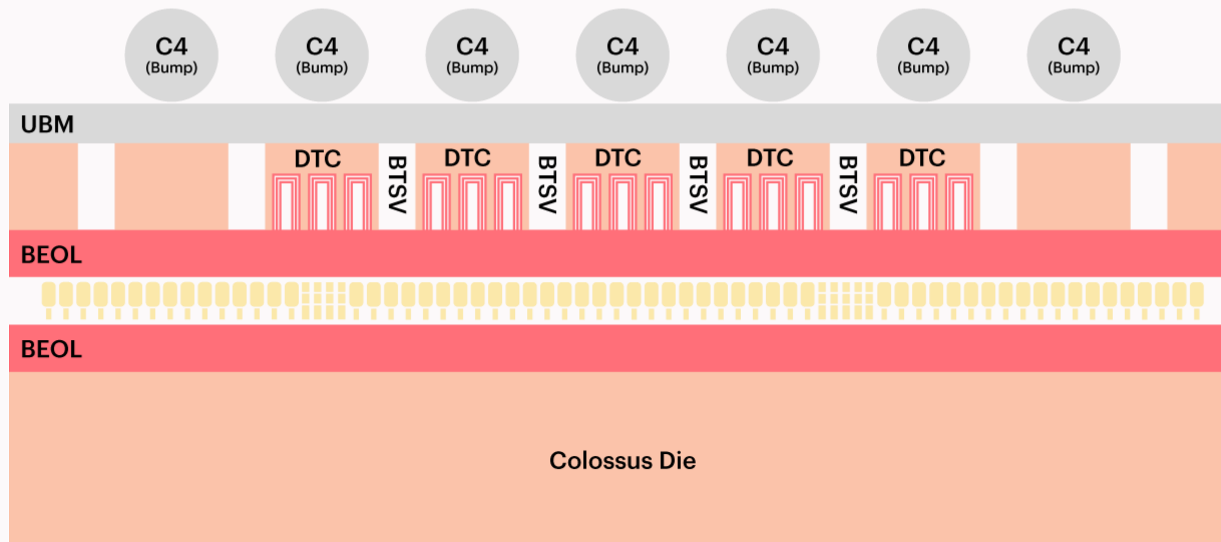
- **Graph Neural Networks**
- **Sparse models**
- **Hybrid ML/HPC workloads**

Available in cloud or on prem

- **6000 in data centers worldwide**
- **Try for free on DigitalOcean Paperspace**
- **Program in PyTorch, TF, JAX, or low-level C++**

GRAPHCORE

BOW IPU: 3D WAFER-ON-WAFER PROCESSOR

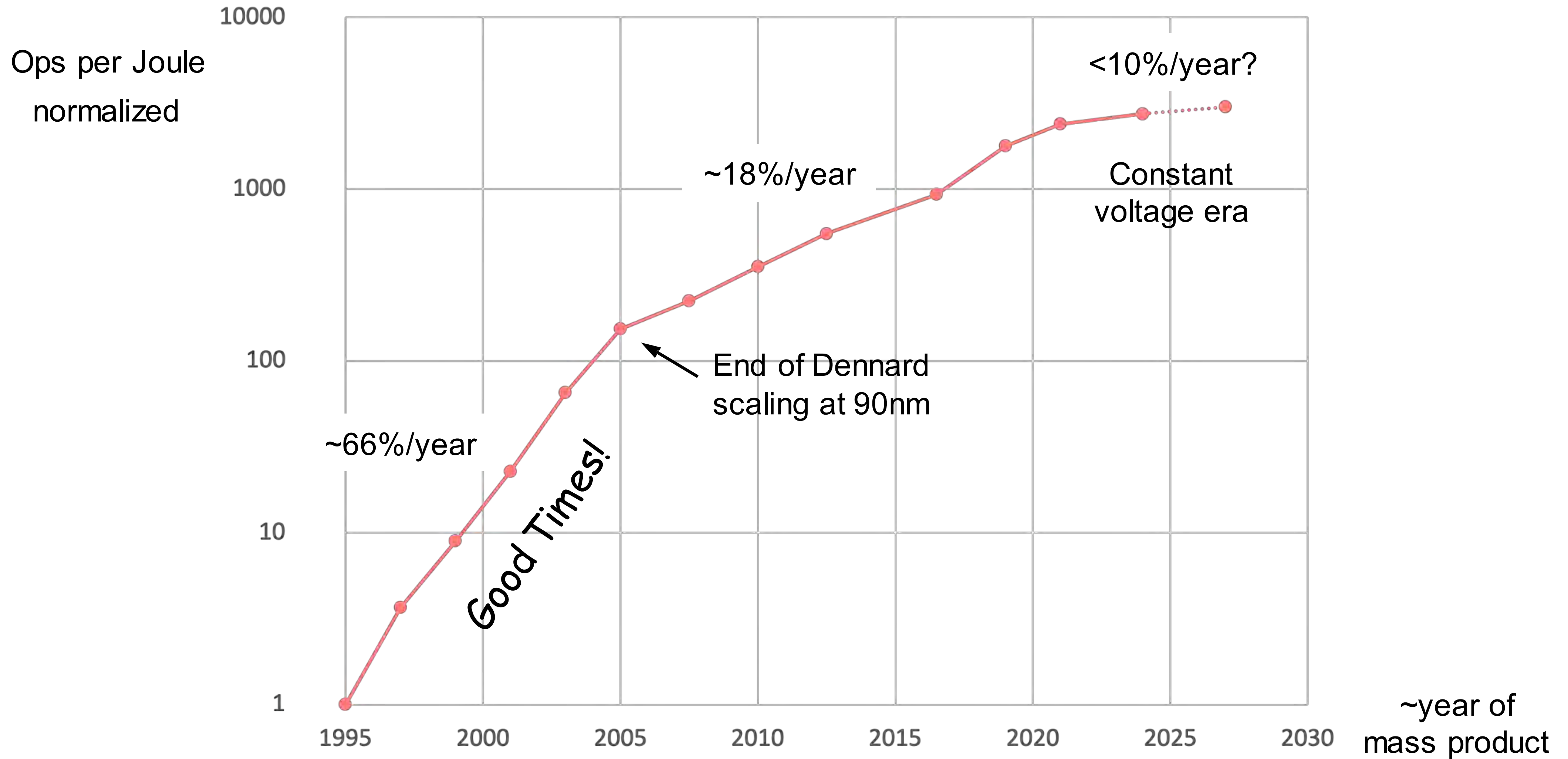


Advanced silicon wafer stacking technology co-developed between Graphcore and TSMC

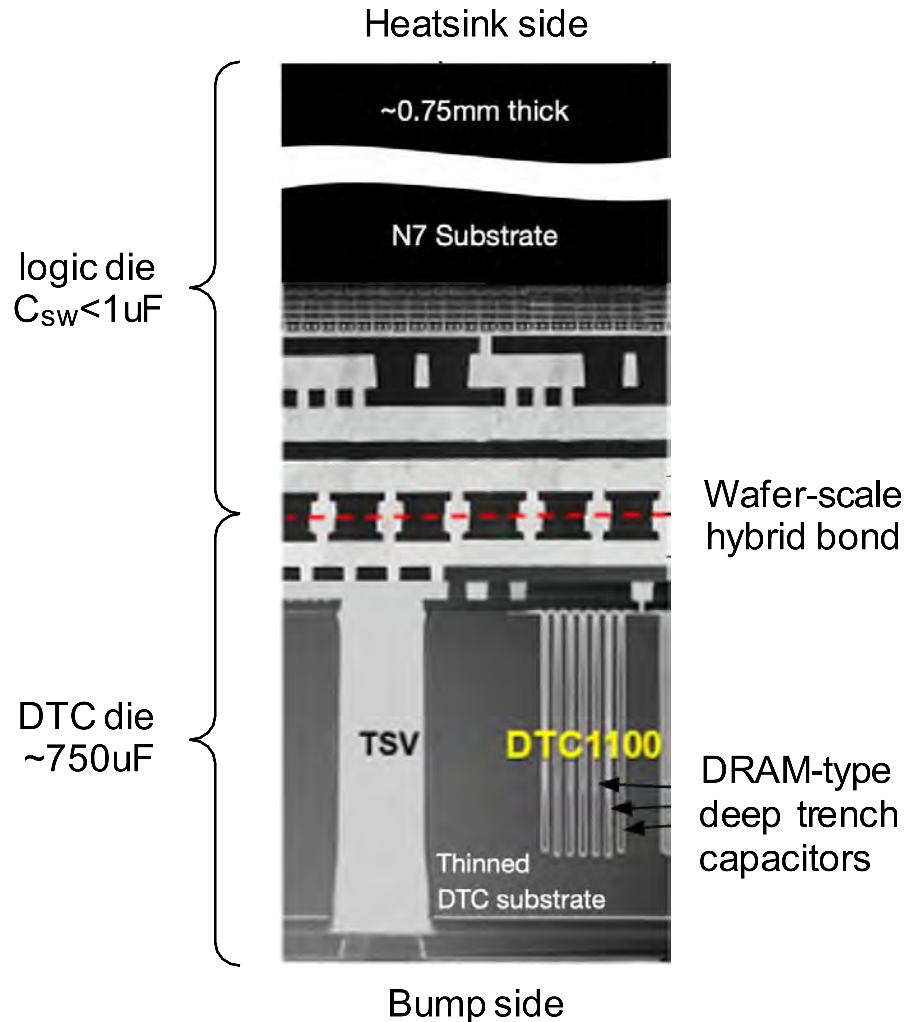
World's first commercial deployment using TSMC SoIC-WoW™ technology in Bow IPU

Enabling technology for closely coupled power delivery die to maximize application performance

Performance per Watt

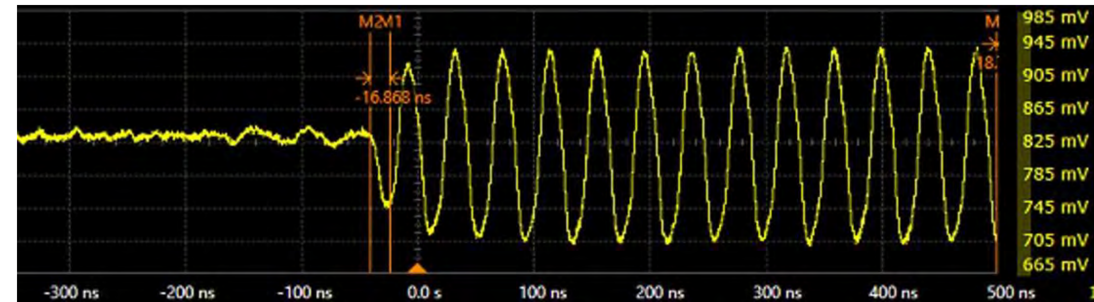


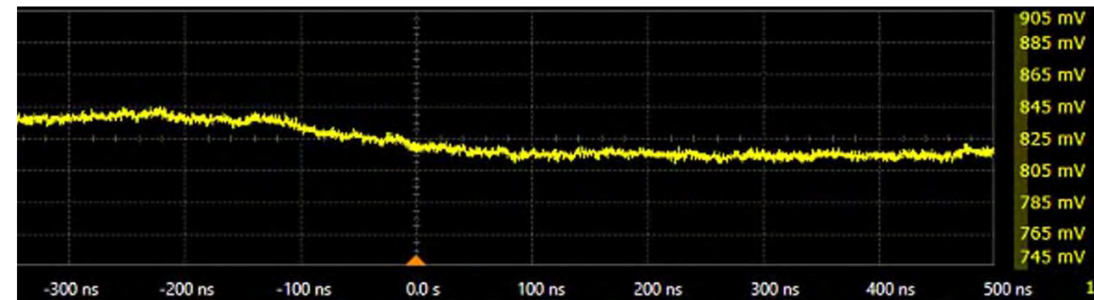
TAMING VDD



Graphcore Colossus Mk2w:

- First Wafer-on-Wafer (WoW) 3D logic chip
- 1.4x speed at same energy/op





Vdd at die without/with DTC; 25MHz 50/50 min/max activity virus

GRAPHCORE COLOSSUS MK2 IPU

- Full-reticle N7, 14 metals, 59bn transistors
- 1472 processors (MIMD)
- 350Tflop/s fp16, 87Tflop/s fp32
- 897MiB distributed SRAM @ 65TB/s
- 11TB/s crossbar inter-tile interconnect
- 10x 16GB/s inter-chip links, 2x 16GB/s PCIe

Link

Tile

Exchange

BOW IPU

Deep Trench Capacitor

Efficient power delivery
Enables increase in operational performance

Wafer-On-Wafer

Advanced silicon 3D stacking technology

Closely coupled power delivery die

Higher operating frequency and enhanced overall performance

Solder Bumps

IPU-Links™

10x IPU-Links, 320GB/s chip to chip bandwidth

IPU-Tiles™

1472 independent IPU-Tiles™ each with an IPU-Core™ and In-Processor-Memory™

IPU-Exchange™

11 TB/s all to all IPU-Exchange™
Non-blocking, any communication pattern

IPU-Core™

1472 independent IPU-Core™

8832 independent program threads executing in parallel

PCIe

PCI Gen4 x16
64 GB/s bidirectional bandwidth to host

In-Processor-Memory™

900MB In-Processor-Memory™ per IPU

65.4TB/s memory bandwidth per IPU

BOW-2000 IPU MACHINE

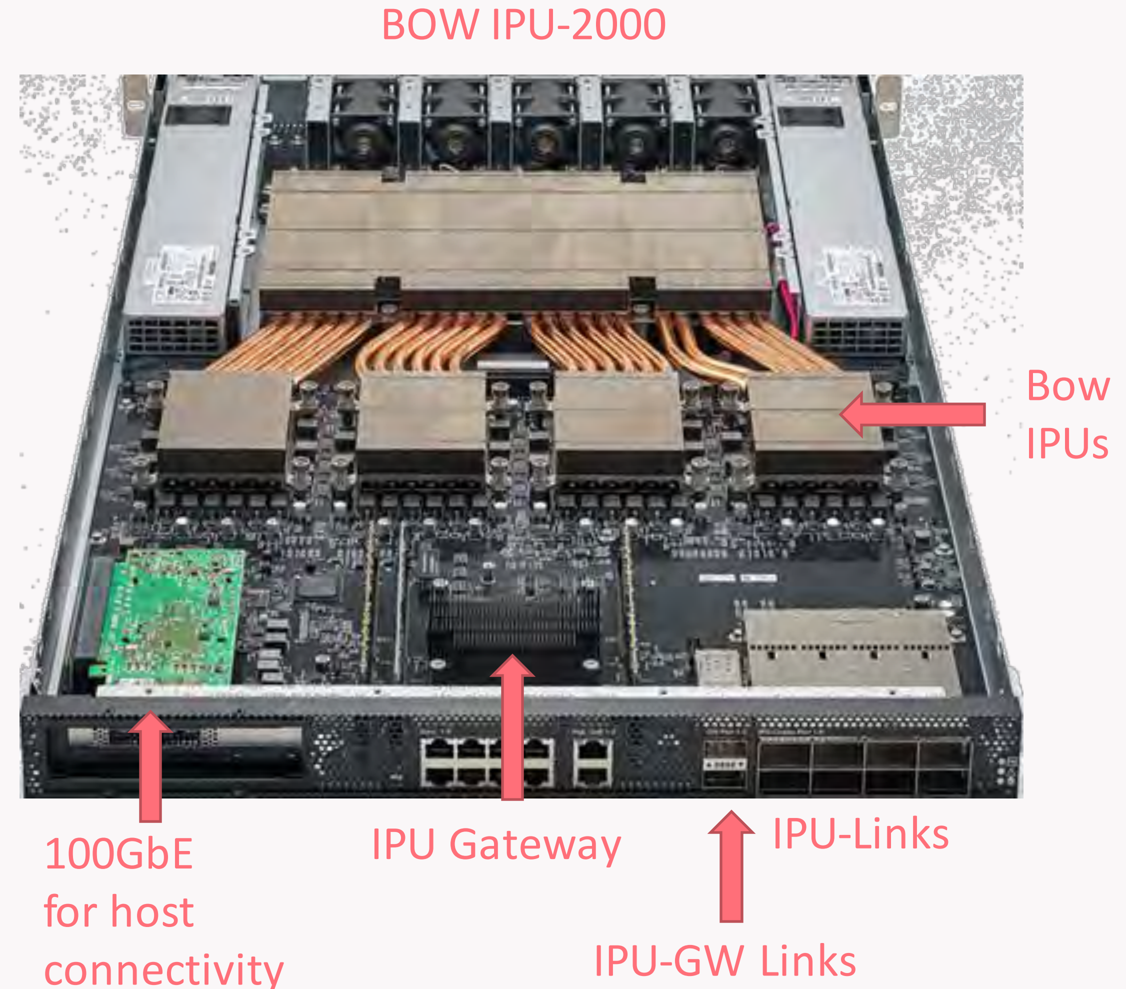
1U blade form factor delivering 1.4 PetaFLOPS AI Compute

Disaggregated AI/ML accelerator platform

Excellent performance & TCO leveraging In-Processor memory & IPU-Exchange

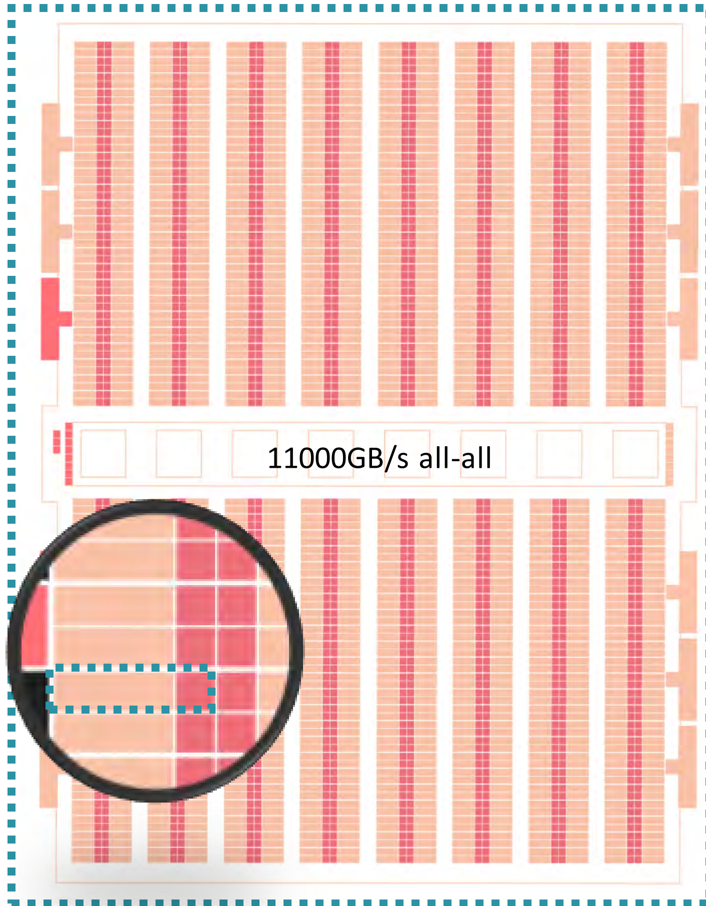
IPU-Links scale to Bow Pod64

Expansion to Bow Pod256 and beyond with IPU-GW Links



CHIP-LEVEL

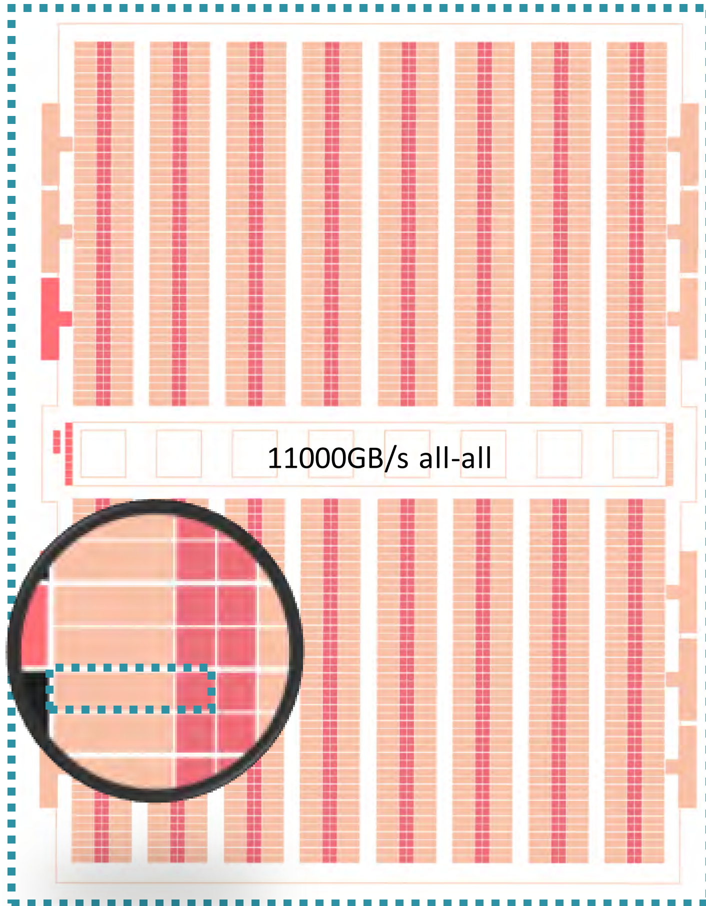
BOW IPU
350TFLOPS(F16)
900MB SRAM
1472 IPU-Tiles
8832 independent
instruction streams



IPU-Tile
236GFLOPS(F16)
640KB SRAM
6 HW Threads 1.83GHz
128 F16Ops/Cycle

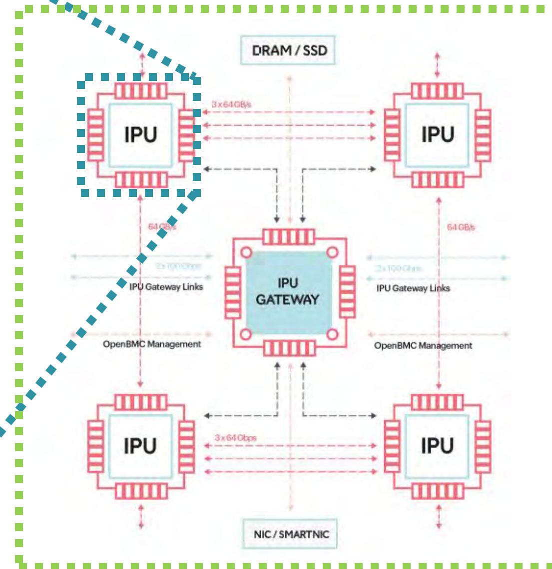
POD-LEVEL

BOW IPU
350TFLOPS(F16)
900MB SRAM
1472 IPU-Tiles
8832 independent instruction streams



IPU-Tile
236GFLOPS(F16)
640KB SRAM
6 HW Threads 1.83GHz
128 F16Ops/Cycle

BOW-2000 IPU-Machine
4x IPU
256 GB DRAM

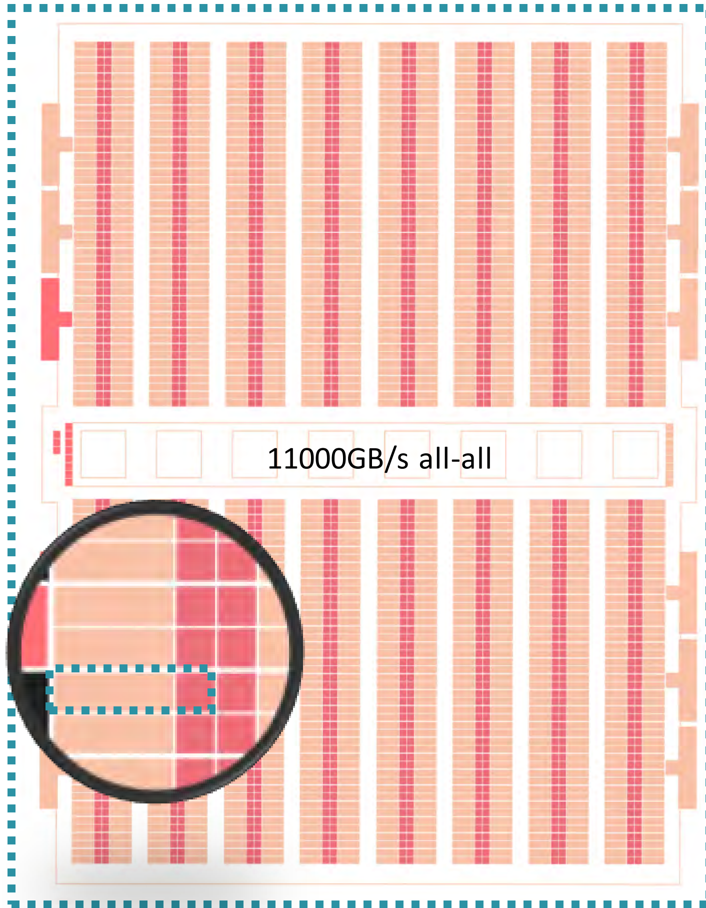


320 GB/s chip-to-chip bandwidth

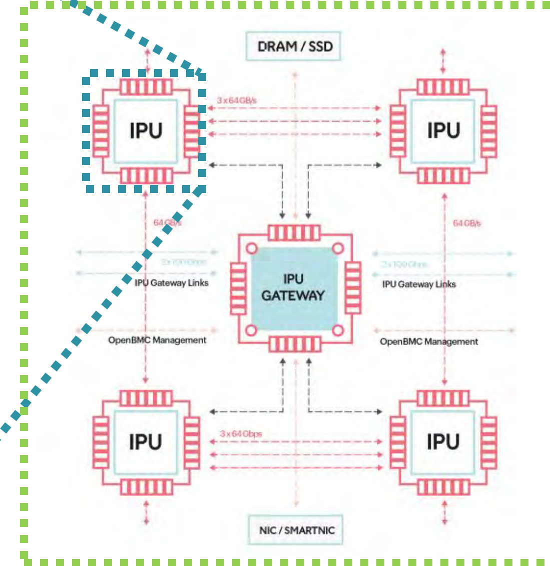
SYSTEM-LEVEL

BOW IPU
350TFLOPS(F16)
900MB SRAM
1472 IPU-Tiles
8832 independent instruction streams

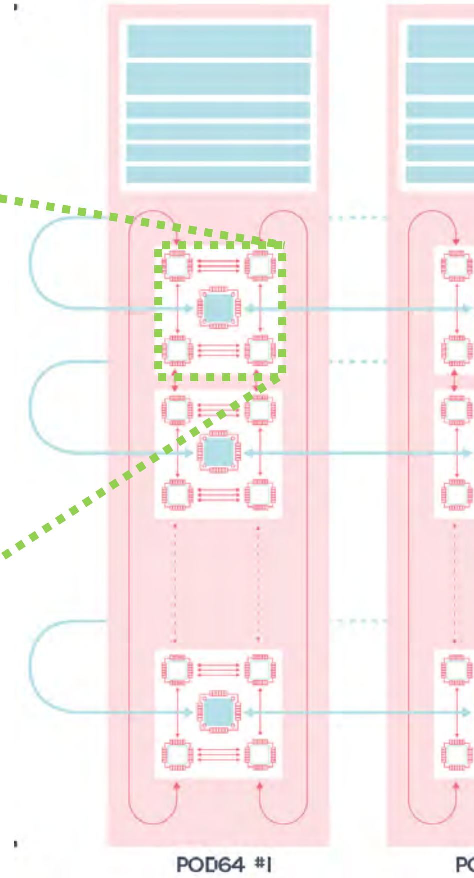
IPU-Tile
236GFLOPS(F16)
640KB SRAM
6 HW Threads 1.83GHz
128 F16Ops/Cycle



BOW-2000 IPU-Machine
4x IPU
256 GB DRAM



320 GB/s chip-to-chip bandwidth



SCALING THE SYSTEM

	POD_{16}	POD_{64}	POD_{256}
COMPUTE:	5.6 PetaFLOPS	22.4 PetaFLOPS	89.6 PetaFLOPS
ON-CHIP CACHE:	14.7 GB	58.9 GB	235.5 GB
OFF-CHIP MEMORY:	256 GB	1024 GB	4096 GB
CPU SERVERS:	1	1-4	4-16

PROGRAMMING THE IPU

PyTorch/JAX Model Code
~100 lines

```python
# Start with token embeddings
embeddings = cfg.lambda_e * params.embeddings[x, :]     # L x Dm

# Add (learned) positional encodings
embeddings += cfg.lambda_pe * params.positional_encodings[:L, :]

# Apply the transformer layers
for layer in params.layers:

    # Layer-normalize embeddings
    t1 = vmap(standardize)(embeddings)
    t1 = elementwise_linear(layer.norm_self_attn, t1)   # L x Dm

    # Multi-head self-attention
    for head in layer.heads:

        # Project into this head's query/key space
        query = linear(head.query, t1)                  # L x Dk
        key = linear(head.key, t1)                      # L x Dk

        # Compute L x L attention matrix
        score = query @ key.T + mask                    # L x L
        attn = jax.nn.softmax(cfg.tau * score, axis=1)  # L x L

        value = linear(head.value, t1)                  # L x Dm
        self_attn = attn @ value                        # L x Dm

        # Add this head's contribution into embeddings
        embeddings += self_attn                         # L x Dm

    # Layer-normalize embeddings
    t2 = vmap(standardize)(embeddings)
    t2 = elementwise_linear(layer.norm_ff, t2)          # L x Dm

    # Feedforward fully connected
    t2 = linear(layer.ffn1, t2)                         # L x Dff
    t2 = jax.nn.relu(t2)
    t2 = linear(layer.ffn2, t2)                         # L x Dm

    # Add this layer's contribution into embeddings
    embeddings += t2

# Layer-normalize embeddings
embeddings = vmap(standardize)(embeddings)
embeddings = elementwise_linear(params.pre_output_norm, embeddings)

# And linearly project to output dimension
return linear(params.output, embeddings)                # L x n_vocab
```

XLA/MLIR Compute Graph
~ 1K nodes, O(GB) tensors

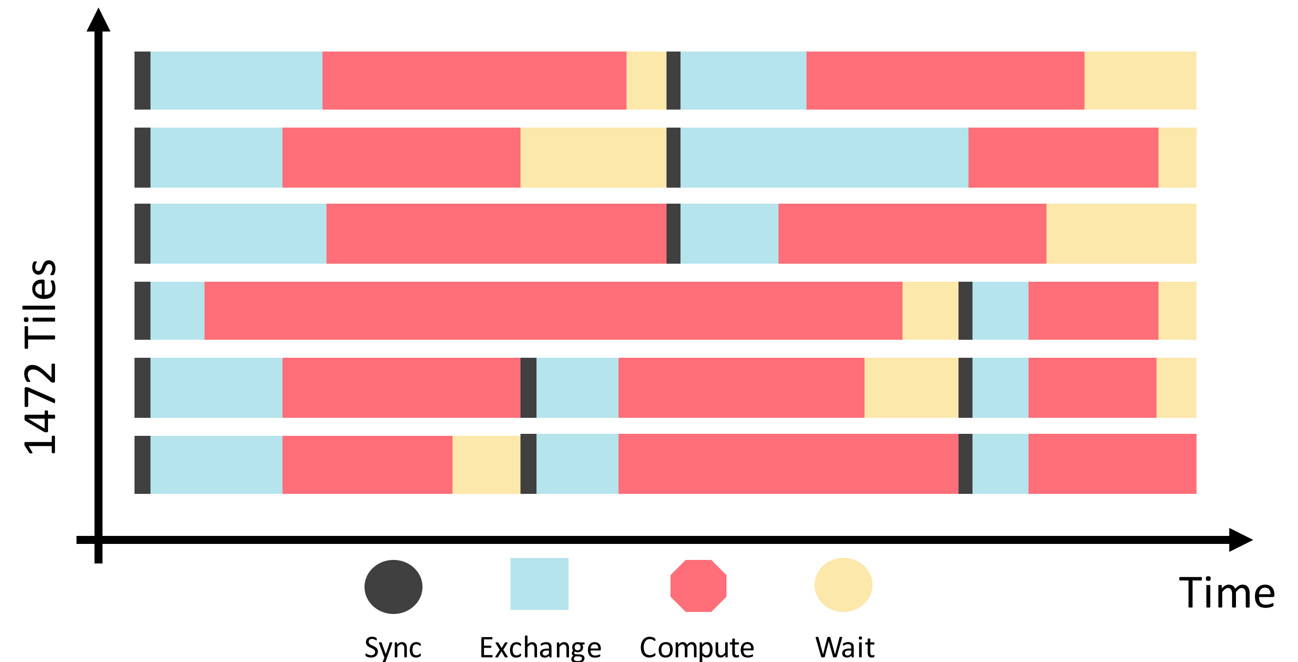
Poplar Vertex Graph
~ 1M nodes
O(KB) tensors

LLVM->Object code: 1472*D ELFs

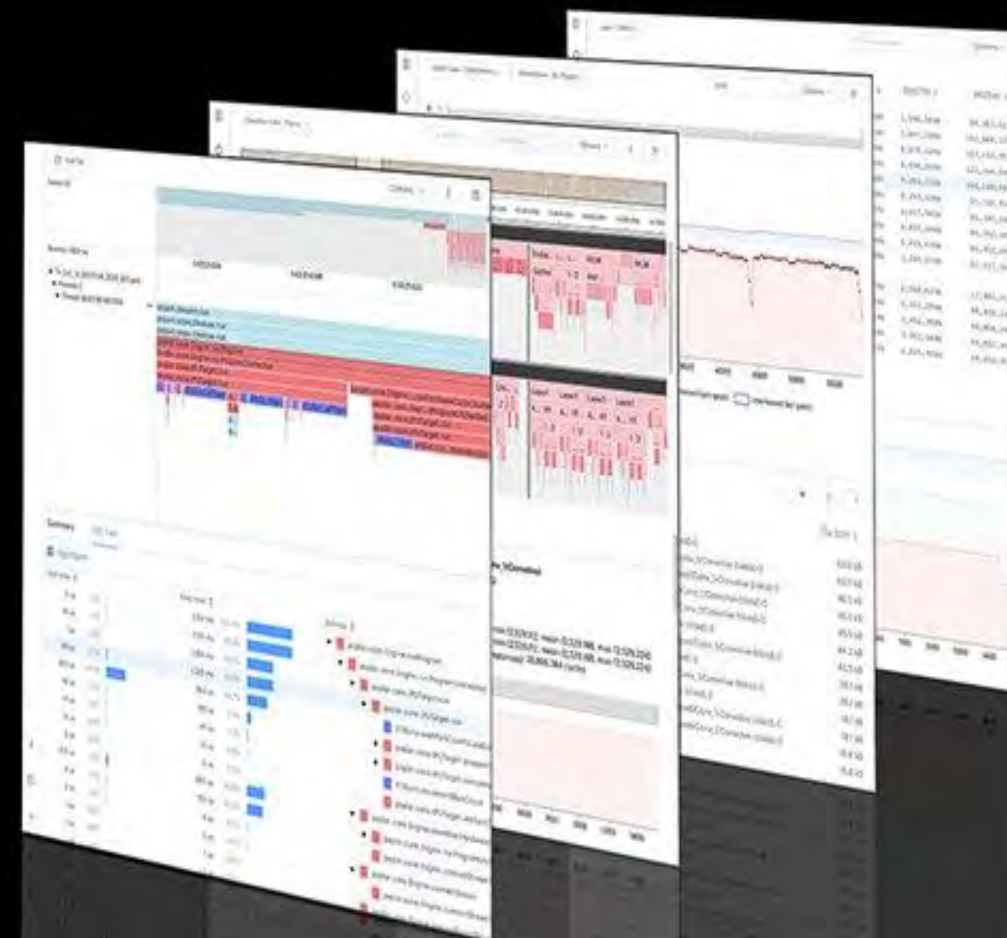
```
ar tv out.a
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0000.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0001.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0002.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0003.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0004.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0005.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0006.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0007.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0008.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0009.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0010.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0011.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0012.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0013.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0014.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0015.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0016.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0017.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0018.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0019.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0020.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0021.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0022.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0023.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0024.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0025.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0026.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0027.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0028.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0029.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0030.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0031.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0032.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0033.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0034.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0035.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0036.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0037.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0038.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0039.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0040.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0041.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0042.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0043.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0044.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0045.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0046.o
rw-rw-r-- 1000/1000  623820 May  7 10:24 tile0047.o
```

GROUP SYNCHRONOUS PARALLEL (GSP)

- Bulk synchronous parallel (BSP) algorithms proceed in a series of global supersteps, which consist of concurrent computation, communication, and barrier synchronization.
- The BSP model is well-suited for automatic memory management for distributed-memory computing:
  - decomposition of the problem into parallel jobs
  - allows every participating processor to perform local computations asynchronously
  - this strategy can lead to almost perfect load balancing, both of work and communication
- Grouped synchronous parallel means different subsets of tiles can participate in each sync



POPVISION™ TOOLS



GRAPH ANALYSER

Useful for analysing and optimising the memory use and execution performance of ML models on the IPU

SYSTEM ANALYSER

Graphical view of the timeline of host-side application execution steps

"Our team was very impressed by the care and effort Graphcore has clearly put into the PopVision graph and system analysers. It's hard to imagine getting such a helpful and comprehensive profiling of the code elsewhere, so this was really a standout feature in our IPU experience."

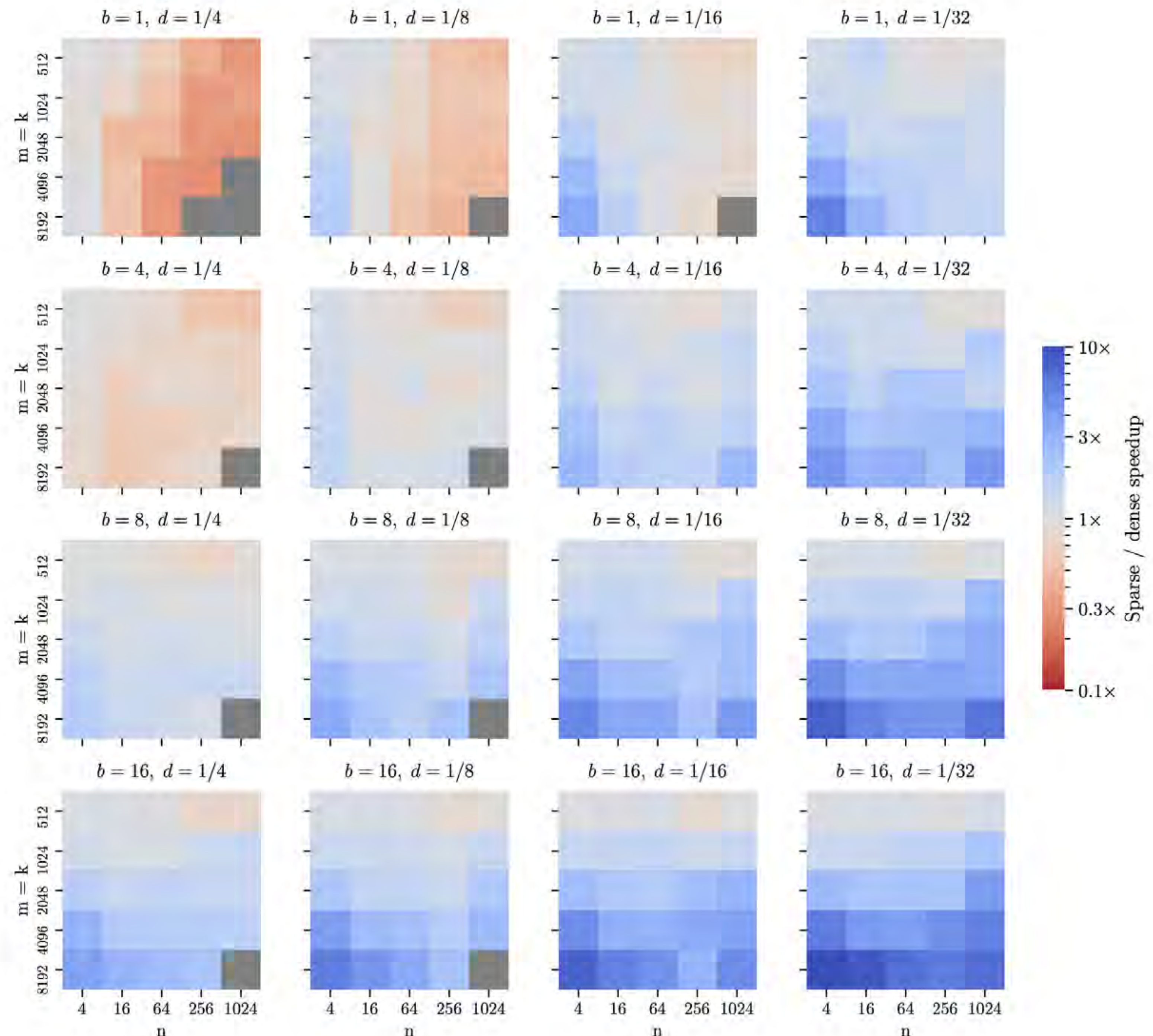
Dominique Beaini, Valence Discovery, a leader in AI-first drug design

AHEAD OF TIME COMPILATION

- Makes best use of HW
- Increasingly widely used in ML
  - XLA in TensorFlow and Jax
  - PyTorch 2.0 introduction of torch.compile()
- Knowing the whole computational graph allows performing advanced code optimisations
- On the IPUs, tile-level code can be determined at the runtime, based on the input data. However, communication between tiles needs to be compiled ahead of time.

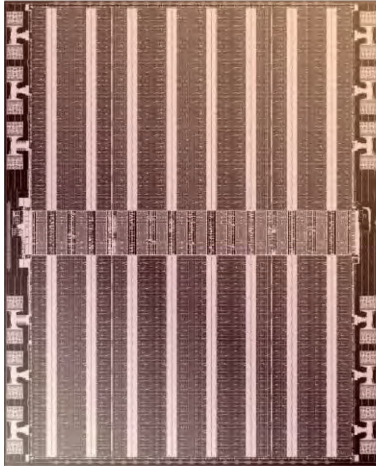
SPARSITY

- PopSparse API
  - provides support for unstructured (b=1) and structured (b>1) sparsity
  - benchmark for sparse-dense matmul either static or dynamic



[2023] Li, Zhiyi, et al. "PopSparse: Accelerated Block Sparse Matrix Multiplication on IPU." https://arxiv.org/abs/2303.16999

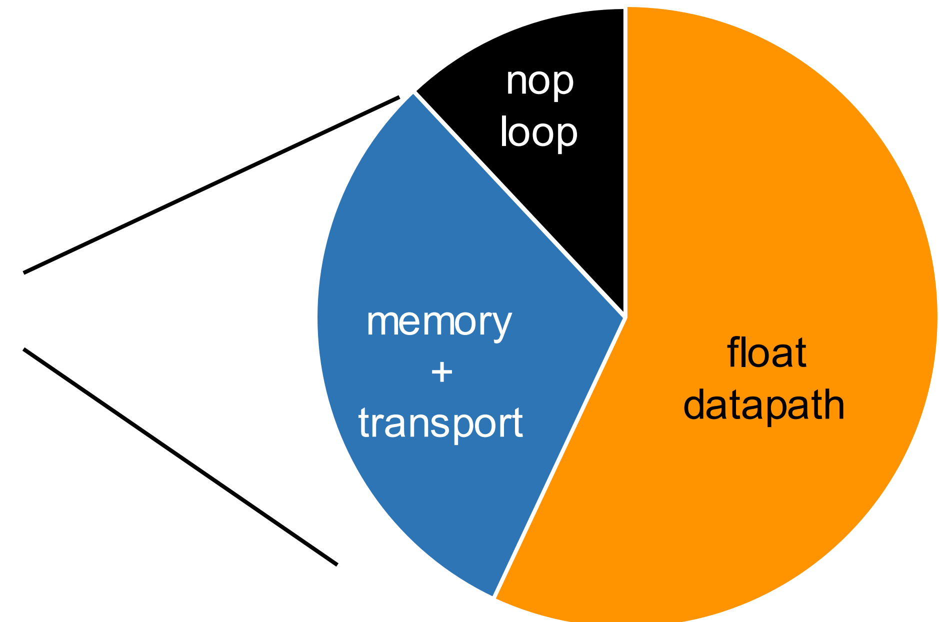
KERNEL POWER



Convolution dynamic power measured at the die level with synthetic data

- Real application data is typically 1/3~1/2 less energetic.
- Power-optimized Mk2 die with wafer-on-wafer DTC decoupler.

Multiply	Accumulate Datapath	Accumulate Memory	pJ/flop
f16	f32	f16	1.0
f16	f32	f32	1.3
f32	f32	f32	2.5



FLOAT8

IEEE Standards WG P3109

- ~50% of fp16 energy/flop
- Works for training with managed scaling
- More accurate for inference than int8

1.4.3 “af8” : 4b exponent, 4b precision

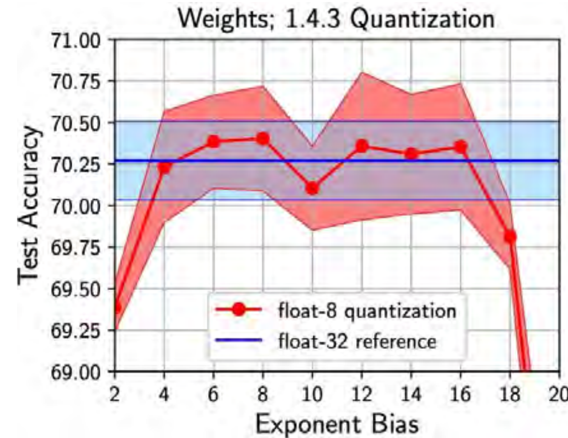
For weights and activations; best accuracy

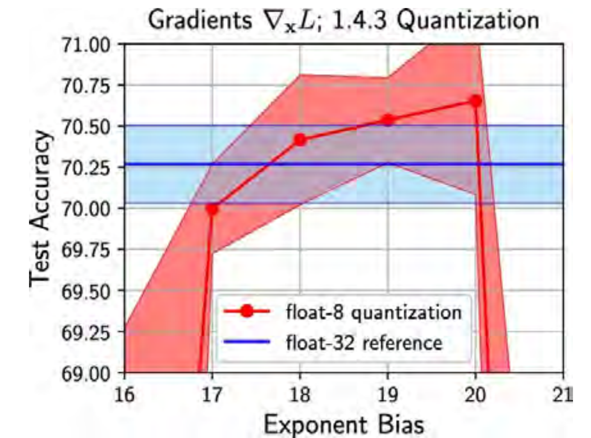
1.5.2 “bf8” : 5b exponent, 3b precision

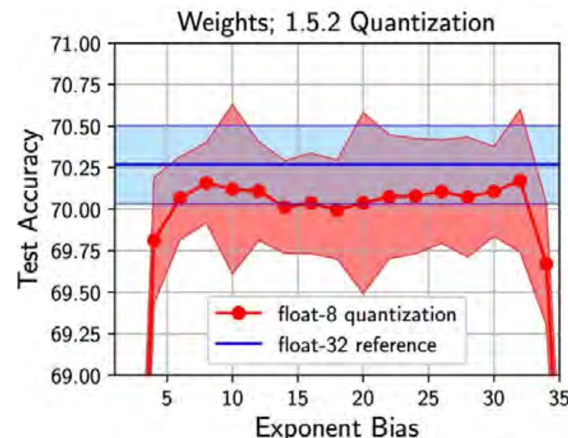
For gradients; best stability

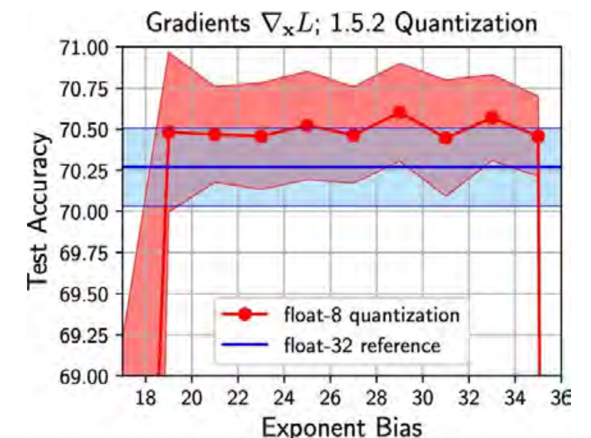
weights
(activations similar)

gradA
(gradW similar)





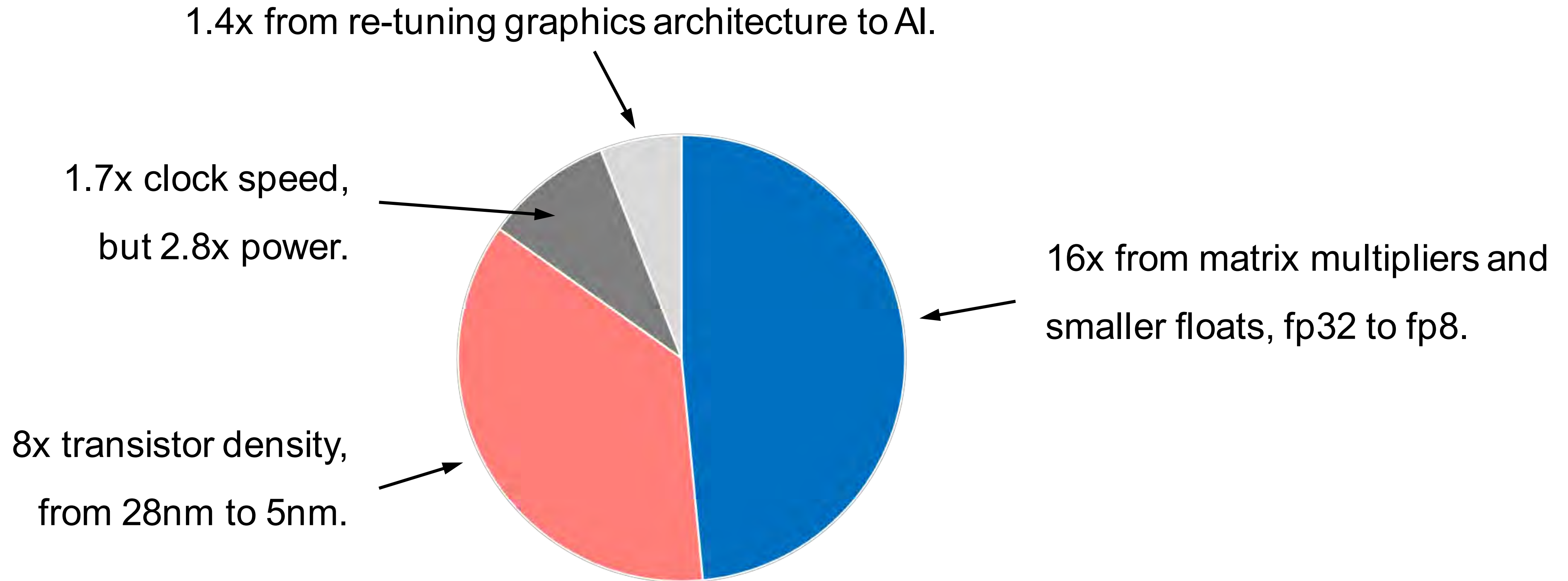




Noune et al. 2022, “8-bit Numerical Formats for Deep Neural Networks”, arXiv:2206.02915

~300X PEAK GPU ARITHMETIC IN THE FIRST AI DECADE

NVIDIA Maxwell 6.6Tflop32/s in 2014 to Hopper 2000Tflop8/s in 2023



THE NEXT DECADE?

More from AI-specific architectures

1.4x from re-tuning graphics architecture to AI.

1.7x clock speed,

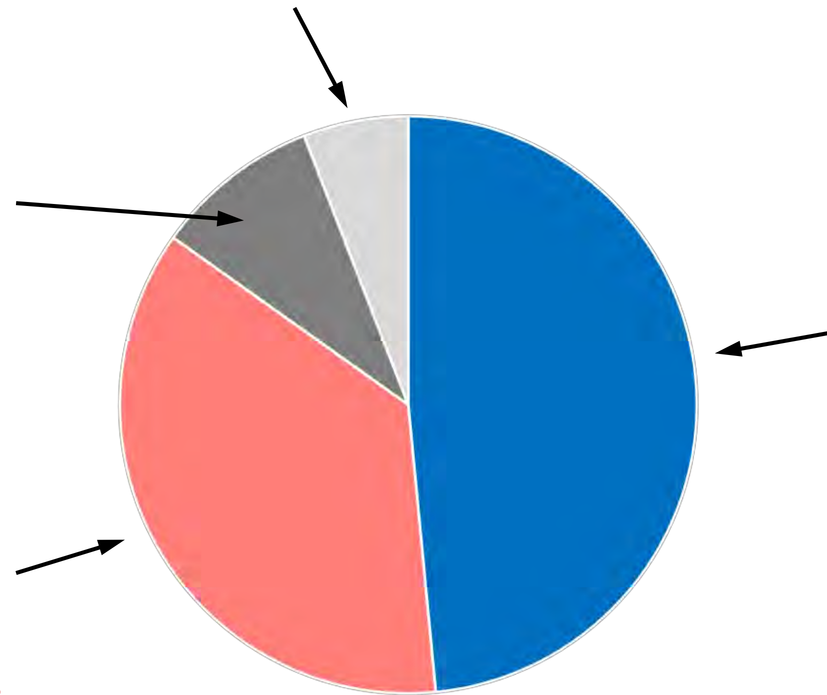
but 2.8x power.

Another 2x, at 3x power?

8x transistor density,

from 28nm to 5nm.

Another 2-3x?



16x from matrix multipliers and

smaller floats, fp32 to fp8.

Done?

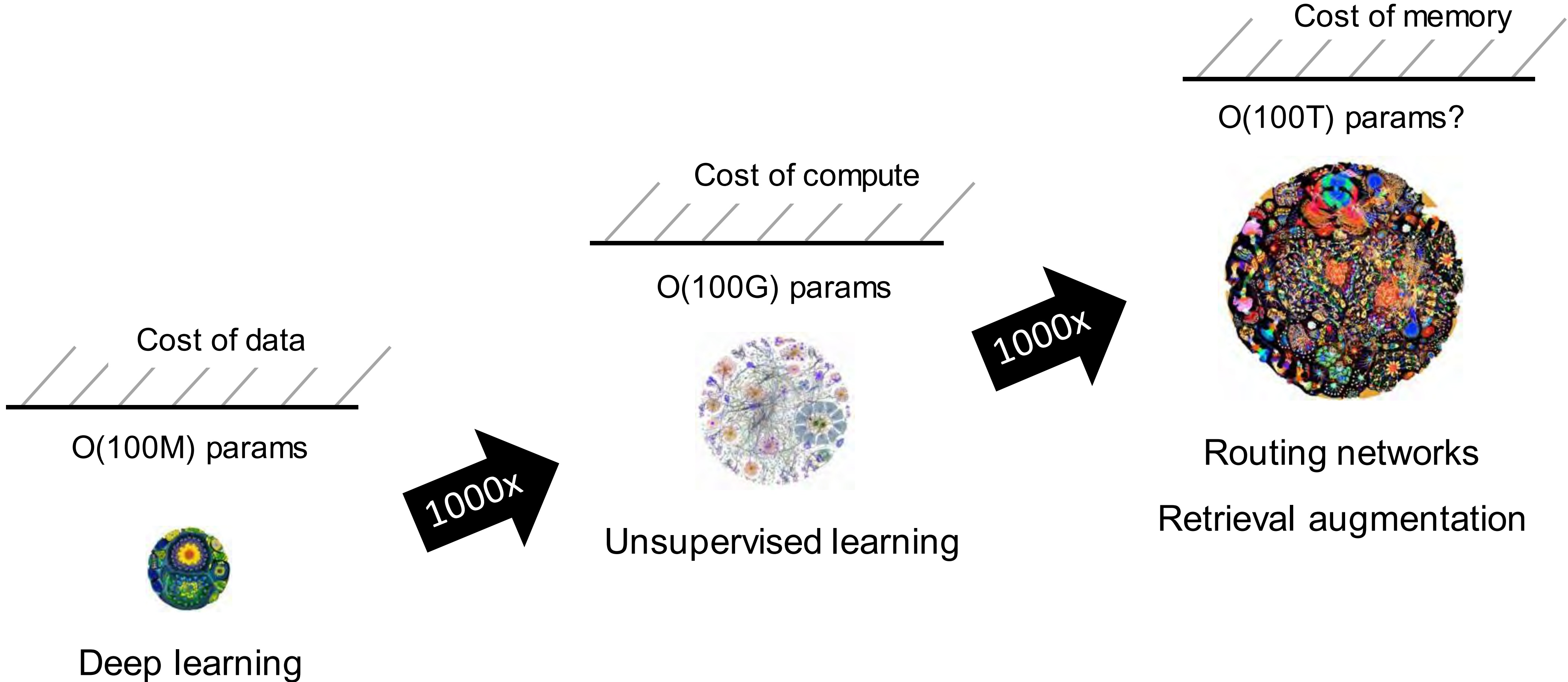
AI ALGORITHMS - STATE OF PLAY

- Dense neural networks are pervasive.
- Useful, efficient models of O(100m - 100bn) weights.
- Training compute ~ O(100 • weights^2)… Eflop - Yflop
- Inference compute ~ O(100 • weights) … Gflops - Tflops
- All signs point to bigger models being more capable

ALGORITHM IMPERATIVE: USE FEWER FLOPS, MORE INFORMATION

- Routing *aka* Conditional Sparsity … like a brain
- Retrieval Augmentation … like a human + www

MODEL SCALE BREAKTHROUGHS



AI ACCELERATORS WILL NEED MORE (CHEAP, FAST) MEMORY

SoTA per reticle-sized logic die:

	GB	GB/s	pJ/B	normalized $cost/B
SRAM over 50% die	1	>> 50,000	<< 1	1
6x HBM3-4800 on silicon substrate	96	3000	40	4
16x LPDDR5-6400 on organic substrate	512	800	50	1
12x DDR5-5600 on 128GB DIMMs	1536	500	300	1.25

…all +0.5pJ/B/mm on die (max 30pJ/B for half-perimeter)

AI ACCELERATORS WILL NEED MORE (CHEAP, FAST) MEMORY

SoTA per reticle-sized logic die:

	GB	GB/s	pJ/B	normalized $cost/B
SRAM over 50% die	1	>> 50,000	<< 1	1
6x HBM3-4800 on silicon substrate	96	3000	40	4
16x LPDDR5-6400 on organic substrate	512	800	50	1
12x DDR5-5600 on 128GB DIMMs	1536	500	300	1.25

…all +0.5pJ/B/mm on die (max 30pJ/B for half-perimeter)

GRAPHCORE

ESO

ROADMAP TO ULTRA-INTELLIGENCE AI

Human brain has around 100 billion neurons

With 100Tn+ synapses, equivalent to parameters in an AI model

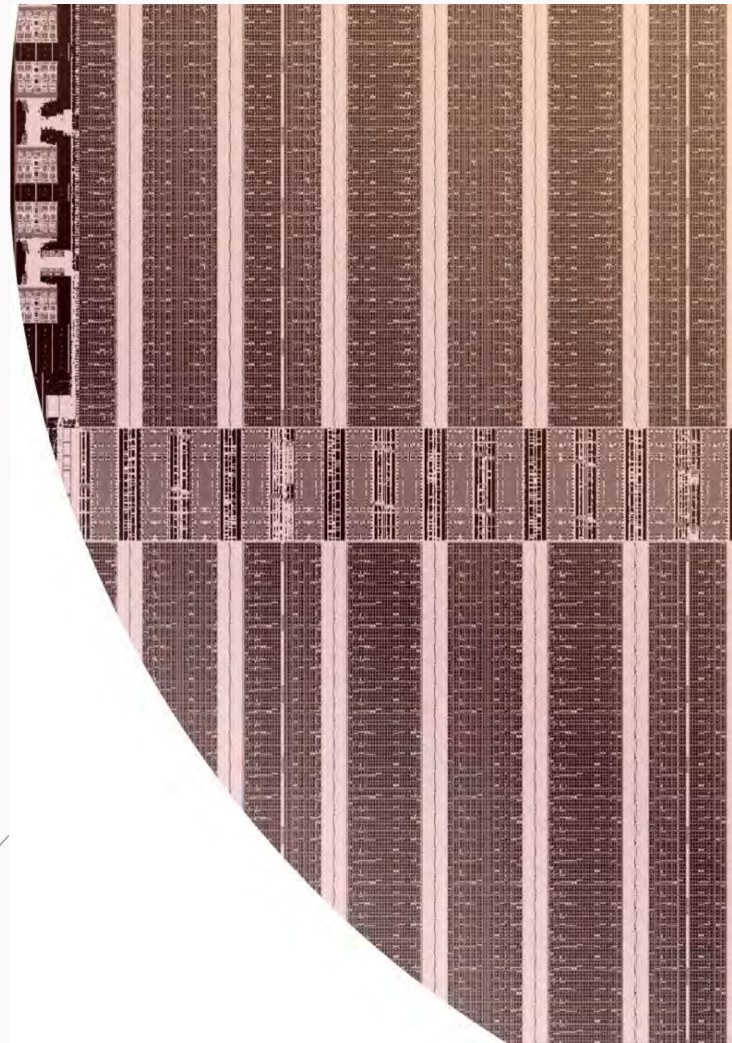
Current largest AI models are around 1Tn parameters

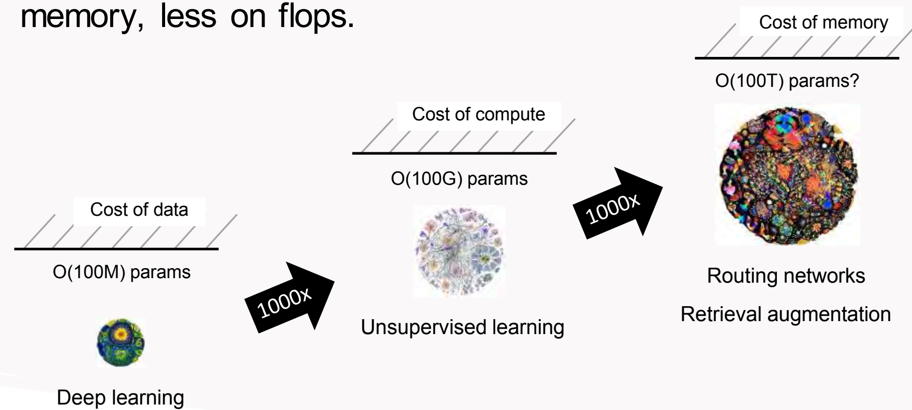
Graphcore is developing an Ultra-Intelligence Machine
that will surpass the parametric capacity of the brain

GRAPHCORE

REMEMBER THIS:

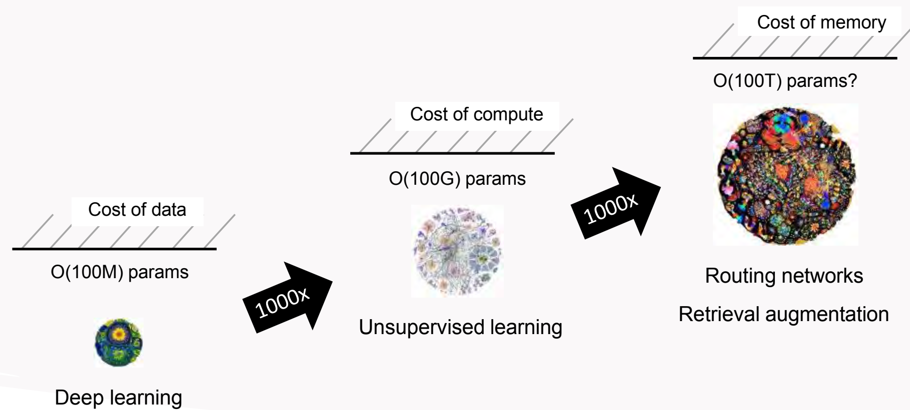
- Silicon is approaching “constant energy per op”.
- Information capacity ultimately determines the potency of an AI, given sufficient training.
- AI computing will be limited by power and memory.
- “Human scale” AI is feasible.
- AI algorithm innovation needs to focus more on memory, less on flops.

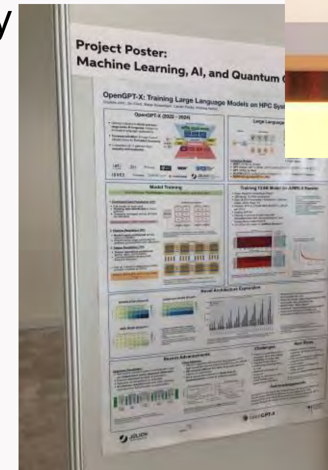


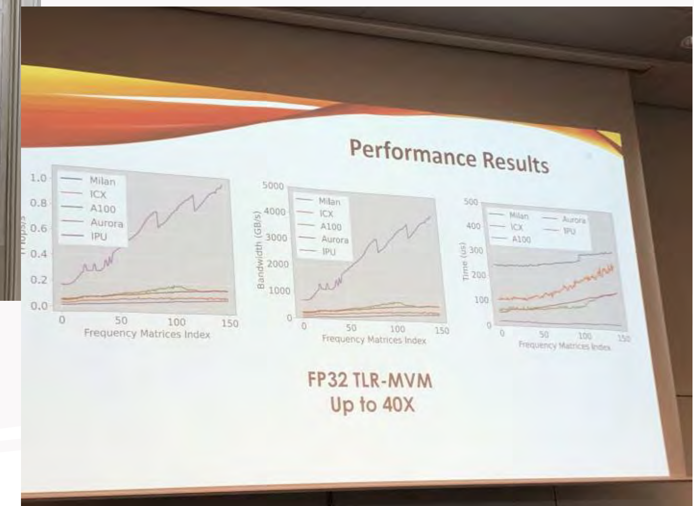


FEW TAKEAWAYS:

- The AI and HPC research community, OBSPM and KAUST obtained between x20 and x150 perf improvement
- Memory needs have to be harnessed in order to regain acquisition and operating cost control is key
- IPU could well be the new golden age of compute and this is designed in Europe !







GRAPHCORE

https://cemse.kaust.edu.sa/hicma/news/steering-customized-ai-architectures-hpc-scientific-workloads-bird-feather-sc22

APPLICATIONS

Very AI Large-Models:

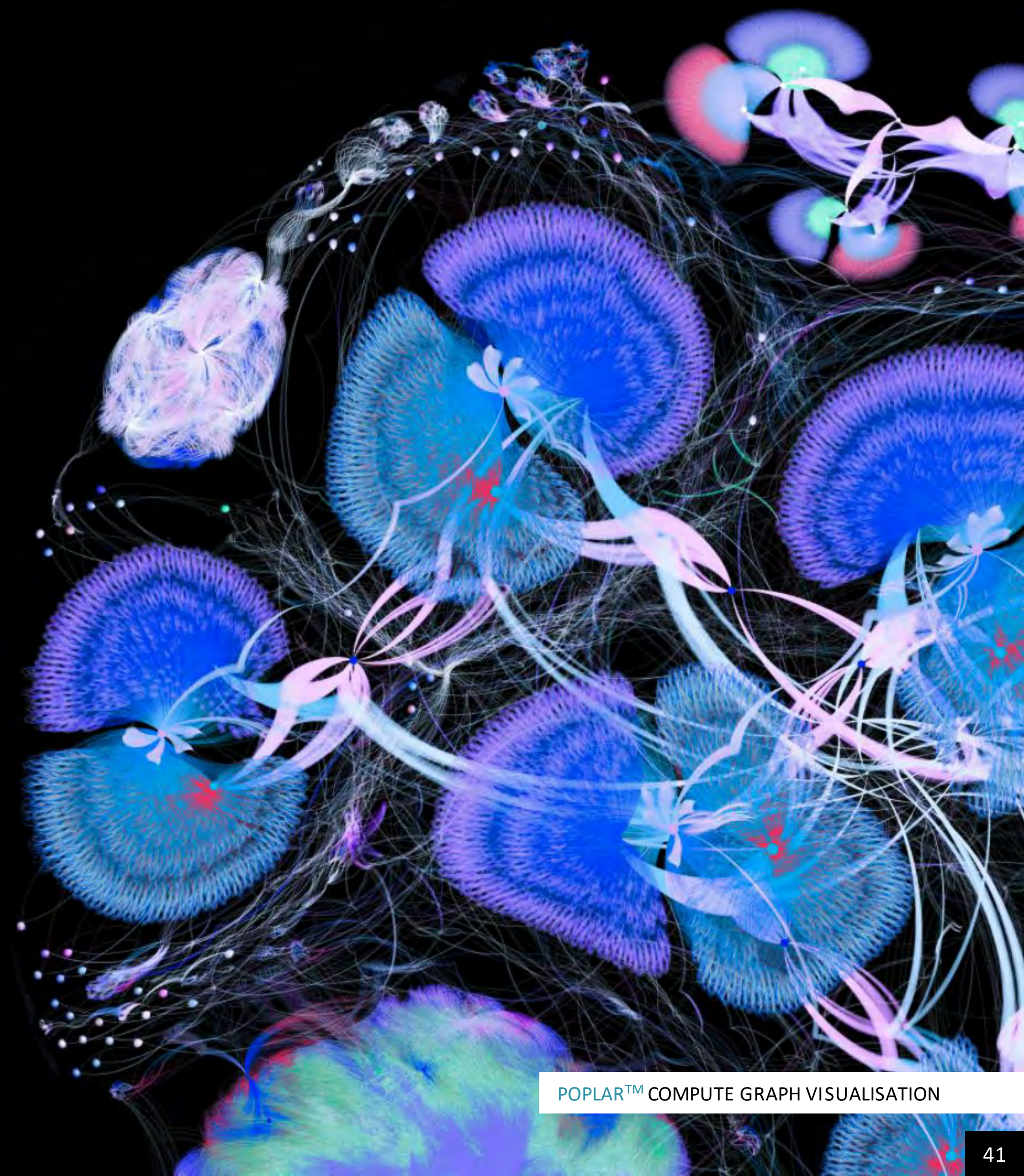
- Muti-trillion parameter model training and inference
- Next-generation, large, conditional and sparse models

AI in Science and Industry

- Healthcare: Genomics | Proteomics | Analysis
- AI-HPC: Simulation | Modeling
- Autonomous systems
- Materials Science | Manufacturing
- Environment: Weather prediction | Smart city

AI in Business

- Language understanding | Process automation | Bots
- Advanced big-data graph analytics and graph databases
- Next generation Recommenders



GRAPHCORE

IPU CLOUD SERVICE

On-Demand
IPU Cloud Infrastructure

Scalable, Flexible, Convenient

Europe focus initially

GET UP & RUNNING ON IPUS RIGHT AWAY

AI COMPUTE ON DEMAND

IMPROVE INNOVATION WITH FASTER & LOWER COST TO TRAIN FOR:
- NLP, CV & GRAPH ML MODELS

EXTENSIVE RANGE OF ML MODELS READY TO RUN ON IPU

IPU-POD AND BOW POD PLATFORMS AVAILABLE FROM POD4 THROUGH TO LARGER SYSTEMS

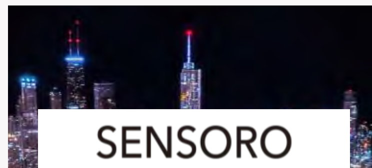
PROVEN IPU ADVANTAGE

IN CASE STUDIES ACROSS MANY INDUSTRIES

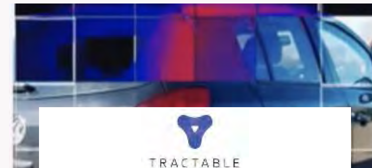
DRUG DISCOVERY
DISCRIMINATIVE NLP
blog

FINANCE
OPTION PRICING
blog paper code

AI SAAS
TEXT ANALYTICS
blog



SMART CITY
OBJECT DETECTION
blog



FINANCE
IMAGE CLASSIFICATION
blog

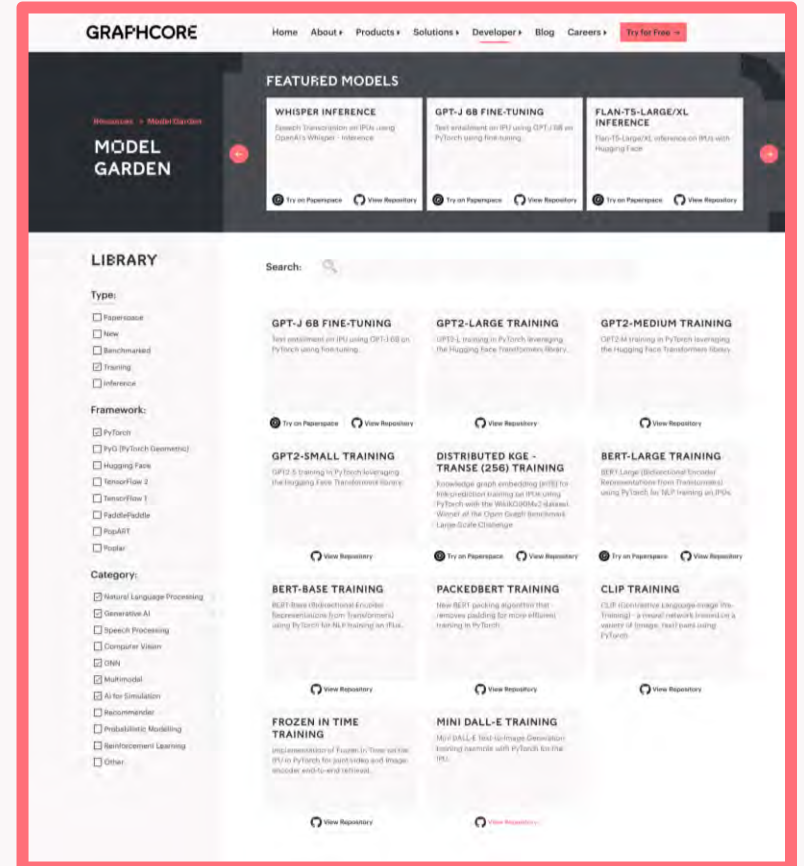
RESEARCH
AI + HPC
blog

RESEARCH
WEATHER FORECASTING
blog

DEFENSE
DECISION MAKING
blog

SOCIAL MEDIA
DYNAMIC GRAPHS
blog code

RESEARCH
COMPUTATIONAL CHEMISTRY
blog



graphcore.ai/resources/model-garden↗

Contact us

Gautier Soubrane gautiers@Graphcore.ai

RECENT ANNOUNCEMENTS & CUSTOMER CASE STUDIES

GRAPHCORE

PROVEN IPU ADVANTAGE

IN LIFE SCIENCES

1st place

OGB-LSC

quantum property prediction

NEURAL INFORMATION PROCESSING SYSTEMS

MOLECULAR MACHINE LEARNING

GPS++ model

SchNet model

PopTorch Geometric

paper

blog

GPS++: An Optimised Hybrid MPNN/Transformer for Molecular Property Prediction

First Place Solution to the 2022 Open Graph Benchmark - Large Scale Challenge

COMPUTATIONAL CHEMISTRY

PySCF$_{IPU}$ library

paper

PySCF$_{IPU}$: Repurposing Density Functional Theory to Suit Deep Learning

Alexander Mathiasen, Hatem Helal, Kerstin Klaser, Paul Balanca, Josef Dean, Carlo Luschi, Dominique Beaini, Andrew Fitzgibbon, Dominic Masters

BESS: Balanced Entity Sampling and Sharing for Large-Scale Knowledge Graph Completion

1st Place OGB-LSC 2022 (WikiKG90Mv2 Track)

1st place

OGB-LSC

link prediction

NEURAL INFORMATION PROCESSING SYSTEMS

KNOWLEDGE GRAPHS

TGN model

NBFNet model

BESS-KGE library

paper

blog

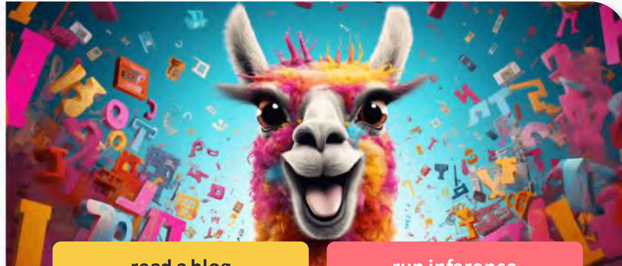
APPLICATIONS

LARGE LANGUAGE MODELS

Aug 01, 2023

LLAMA 2: RUN META'S OPEN SOURCE LARGE LANGUAGE MODEL FOR FREE ON IPUS

Written By:
Tim Santos and Arsalan Uddin

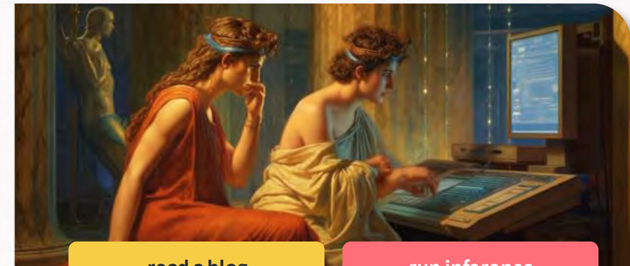


read a blog | run inference

May 31, 2023

OPENASSISTANT FINE-TUNED PYTHIA-12B: OPEN-SOURCE CHATGPT ALTERNATIVE

Written By:
Steve Barlow



read a blog | run inference

Mar 24, 2023

FINE-TUNE GPT-J: A COST-EFFECTIVE GPT-4 ALTERNATIVE FOR MANY NLP TASKS

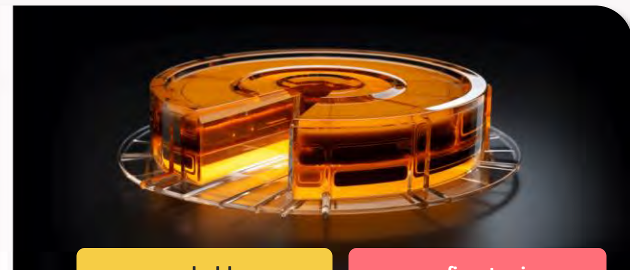
Written By:
Sofia Liguori

read a blog | run text-generation | run fine-tuning

Jul 27, 2023

FINE-TUNING FLAN-T5 XXL - THE POWERFUL AND EFFICIENT LLM

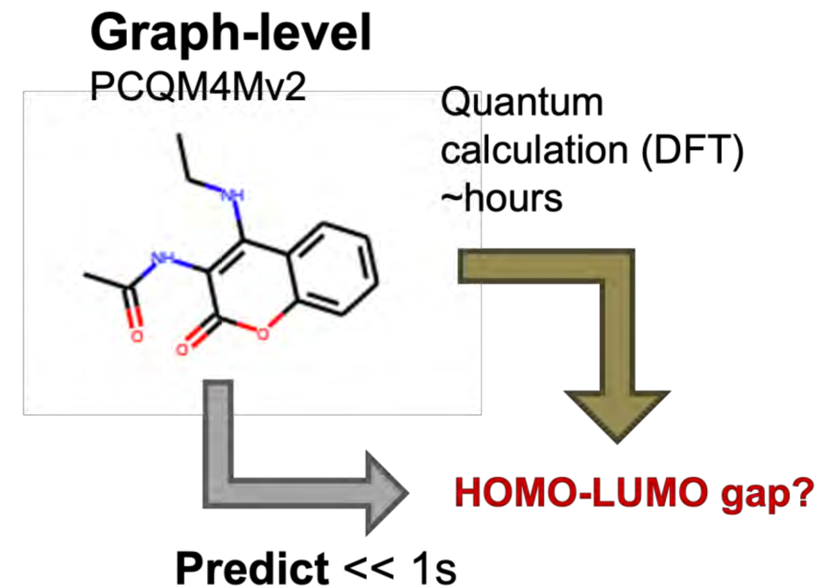
Written By:
Manuele Sigona



read a blog | run fine-tuning

MOLECULAR PROPERTY PREDICTION

- The PCQM4Mv2 track
  - A quantum chemistry dataset generated by computationally expensive molecular simulation in ~hours/molecule.
  - The task is to predict the HOMO-LUMO gap of a given molecule.
  - The metric used is MAE (Mean Absolute Error).



MOLECULAR PROPERTY PREDICTION

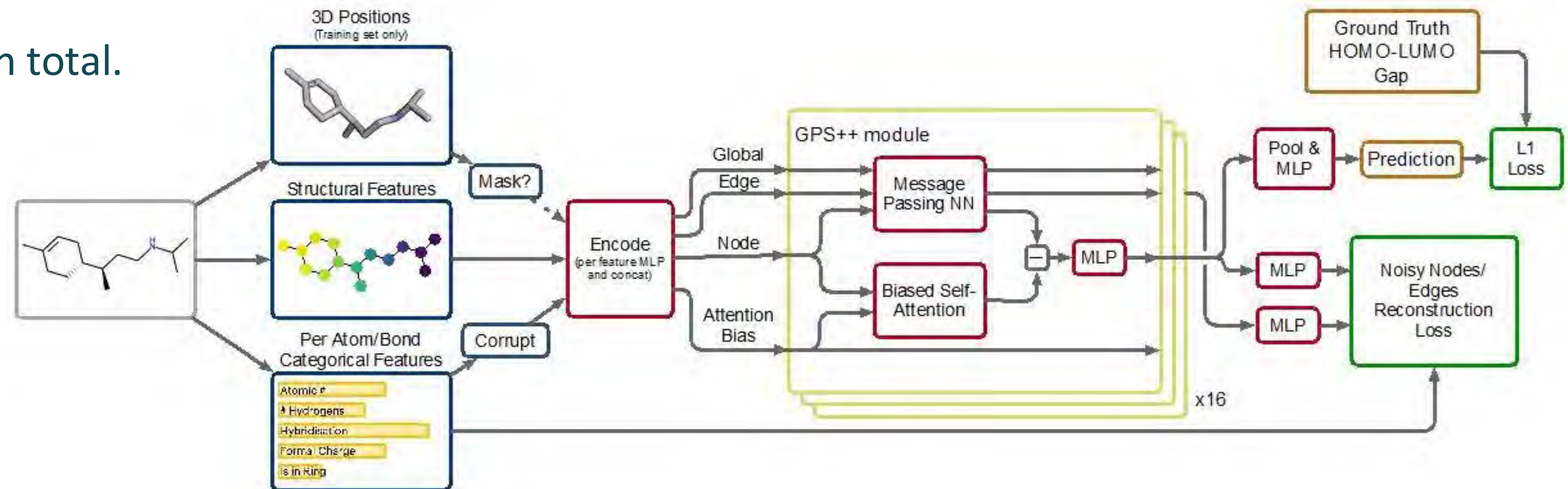
- GPS++
  - An optimised hybrid of MPNN and Transformer for molecular property prediction.
  - Builds on top of General, Powerful, Scalable Graph Transformer (GPS) framework (Rampášek et al. 2022).
  - 44M parameters in total.

Run on Gradient

Inference GPS++

Run on Gradient

Training GPS++



github.com/graphcore/ogb-lsc-pcqm4mv2

MOLECULAR PROPERTY PREDICTION

- SchNet – a GNN model for modelling quantum interactions.
- A collaboration with Pacific Northwest National Laboratory (PNNL), Department of Energy in the US, on improving the performance of the model.

Run on Gradient

Molecular property prediction using SchNet

github.com/graphcore/examples/gnn/schnet

r
neighbor collection
pair-wise distance
distance expansion
e_{ij}
Z
w_{ij}
e^a_{ij}
Embedding
h
MLP
Linear
Propagate
MLP
Interaction Block
$G=(V,E)$
v_j
v_i
e_{ij}
h´
MLP
E_{node}
sum
E_{graph}

ARGONNE ADDS BOW POD SYSTEM TO LEADERSHIP COMPUTING FACILITY (ALCF) AI TESTBED

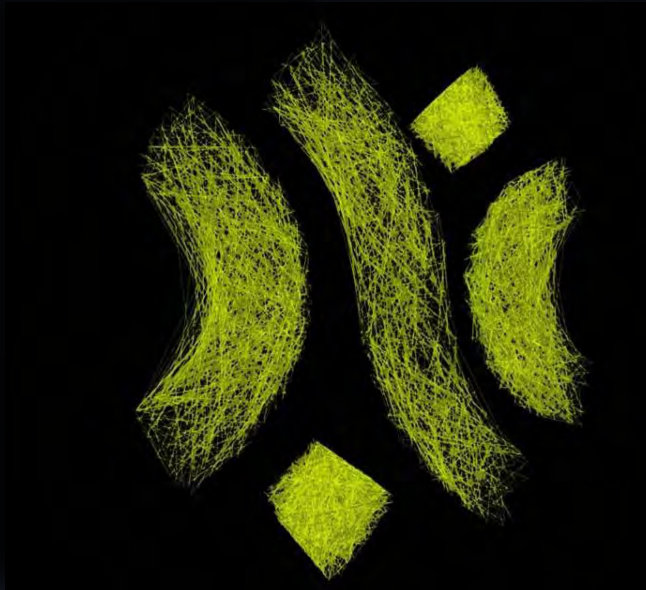
Argonne National Laboratory is adding the latest-generation Bow IPU system to its AI testbed which is enabling pioneering research at the intersection of AI, HPC, and big data, including studies involving climate predictions, drug discovery, and the analysis of large-scale experimental datasets.

“Novel processor architectures like the IPU are facilitating and accelerating new AI techniques and model types. We are excited to work with the research community on ALCF’s latest Bow Pod system to advance AI for science,”

Venkatram Vishwanath, Data Science Team Lead – ALCF

SC22 paper presentation by Argonne covering scientific ML applications (BraggNN & Candle UNO) running on previous generation IPU

ALEPH ALPHA CHATBOT SPARSIFIED & RUNNING ON THE GRAPHCORE IPU



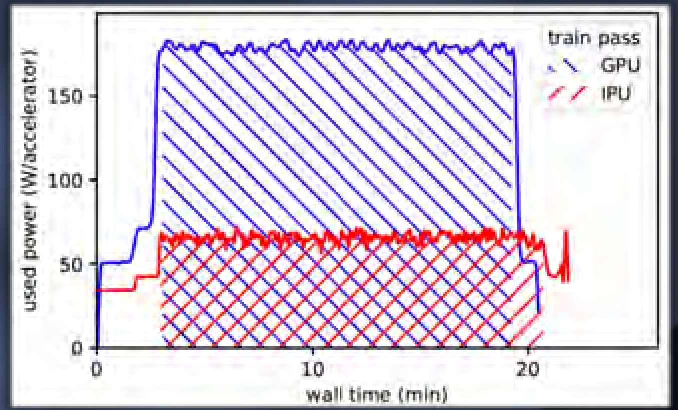
LAWRENCE BERKELEY NATIONAL LABORATORY (LBNL) PAPER PUBLISHED & PRESENTED AT SC22

Time-series ML-regression on Graphcore IPU-M2000 and Nvidia A100

"Abstract—We compare the ML-training performance of a Graphcore IPU-M2000-based system with Nvidia A100 GPU based system on the Perlmutter HPC machine at NERSC/LBL…."

…

"Finally, we compared the energy consumption and observed that IPUs used 2.5 to 3 times less energy while accomplishing the same amount of compute work."



GRAPHCORE