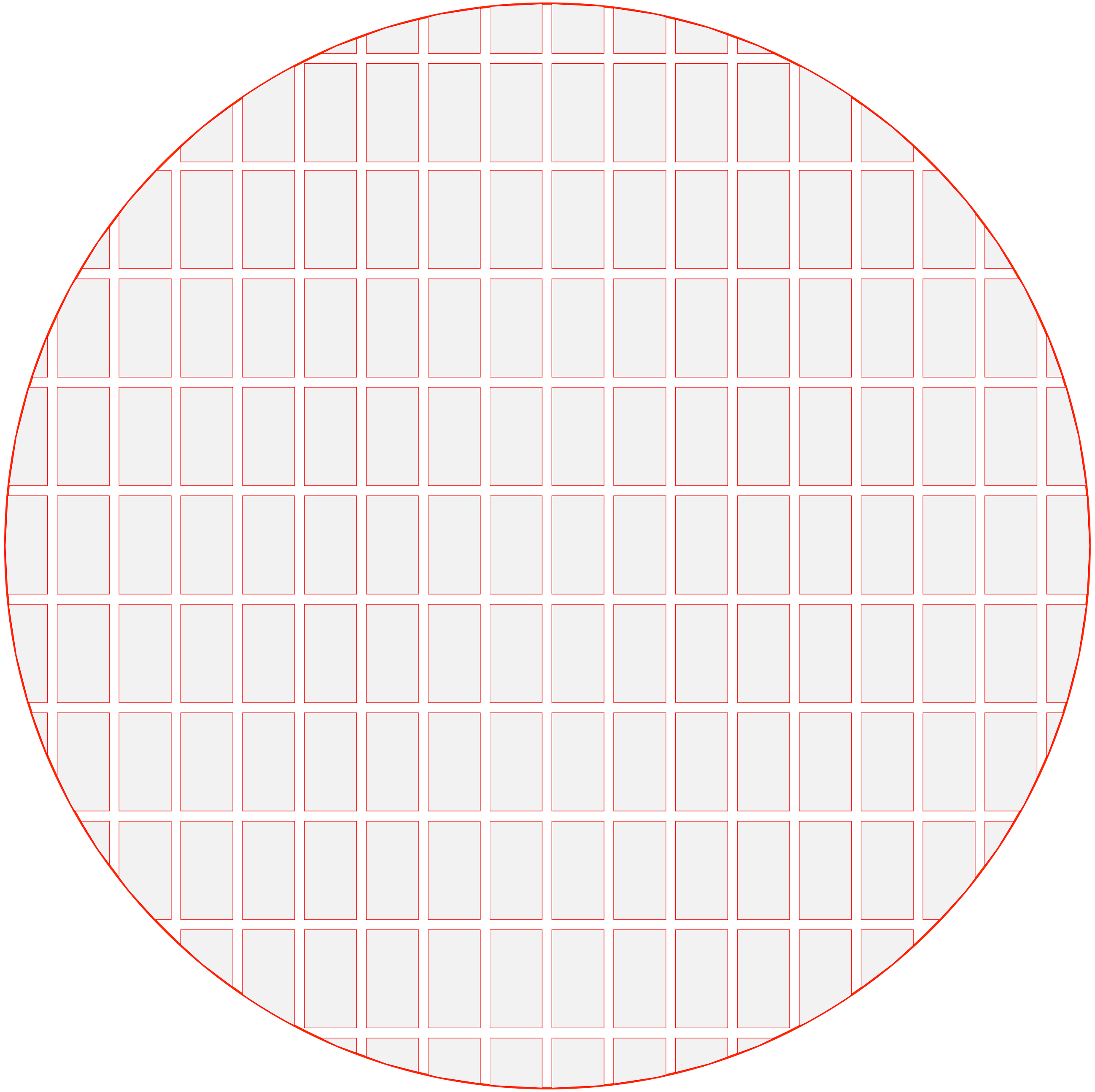
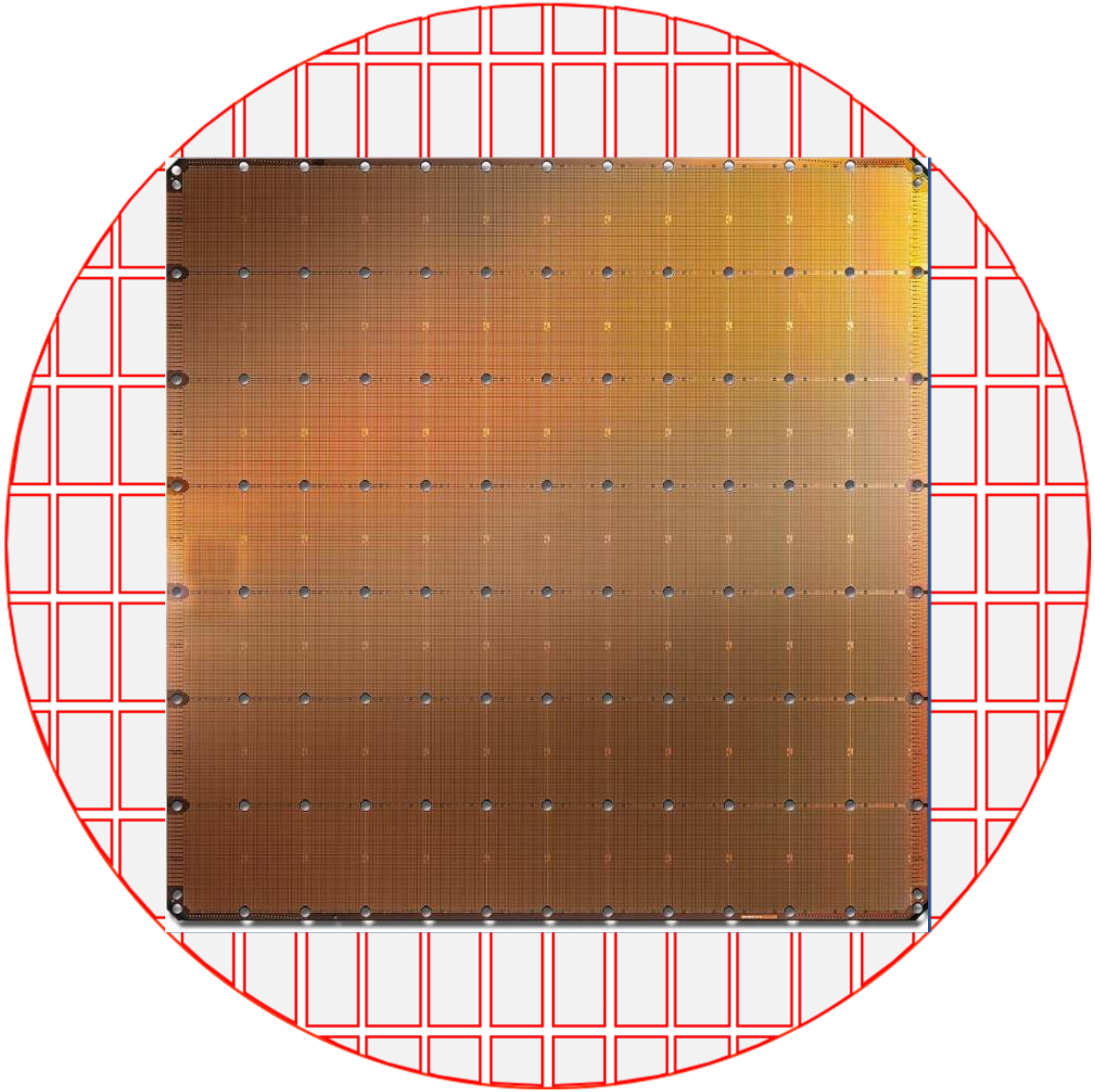
Deep Learning at Scale with the Cerebras CS-1

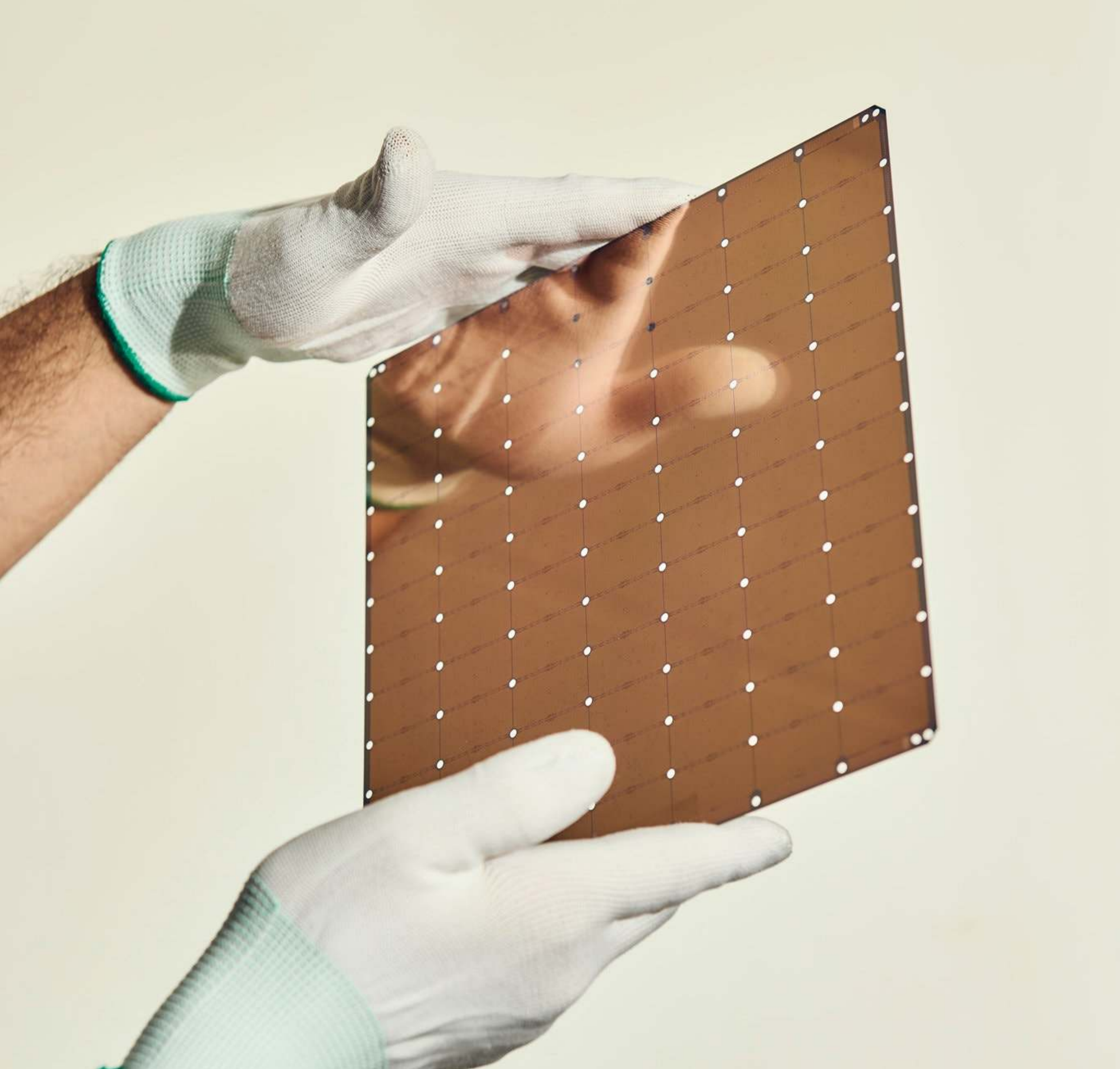
Cerebras Systems

Natalia Vassilieva



cerebras





The Cerebras Wafer Scale Engine (WSE)

The most powerful processor for AI

46,225 mm^2 silicon

1.2 trillion transistors

400,000 AI optimized cores

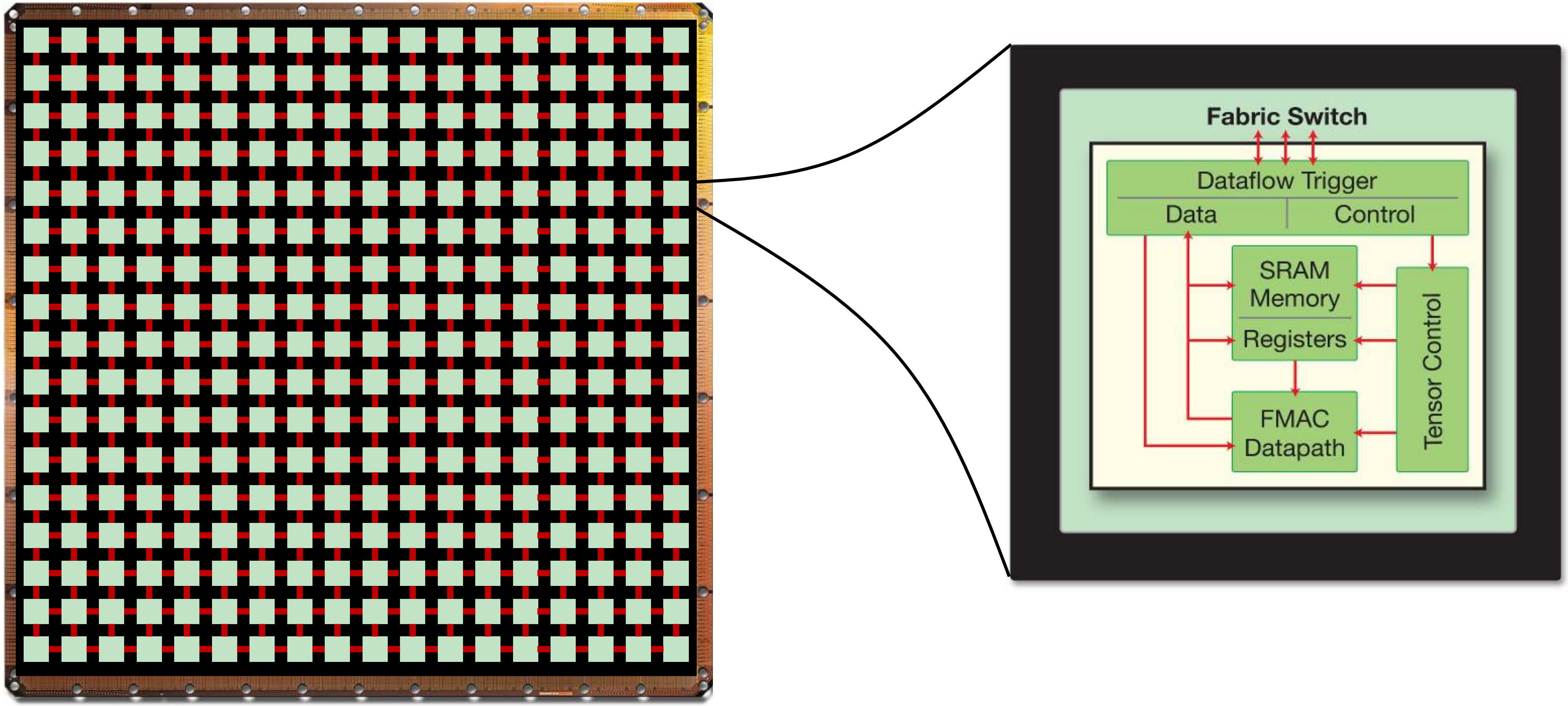
18 Gigabytes of On-chip Memory

9 PByte/s memory bandwidth

100 Pbit/s fabric bandwidth

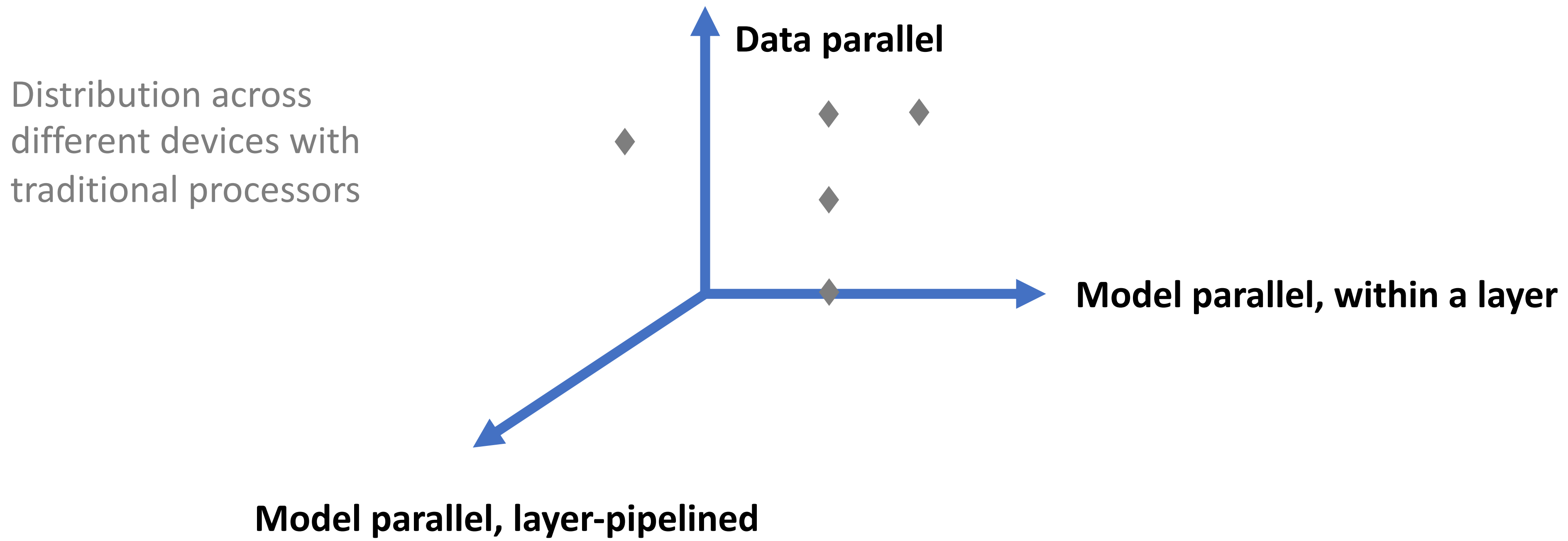
TSMC 16nm process

WSE - 2D mesh of 400,000 fully programmable processing elements



Leverage 400,000 tightly-connected cores to accelerate deep learning

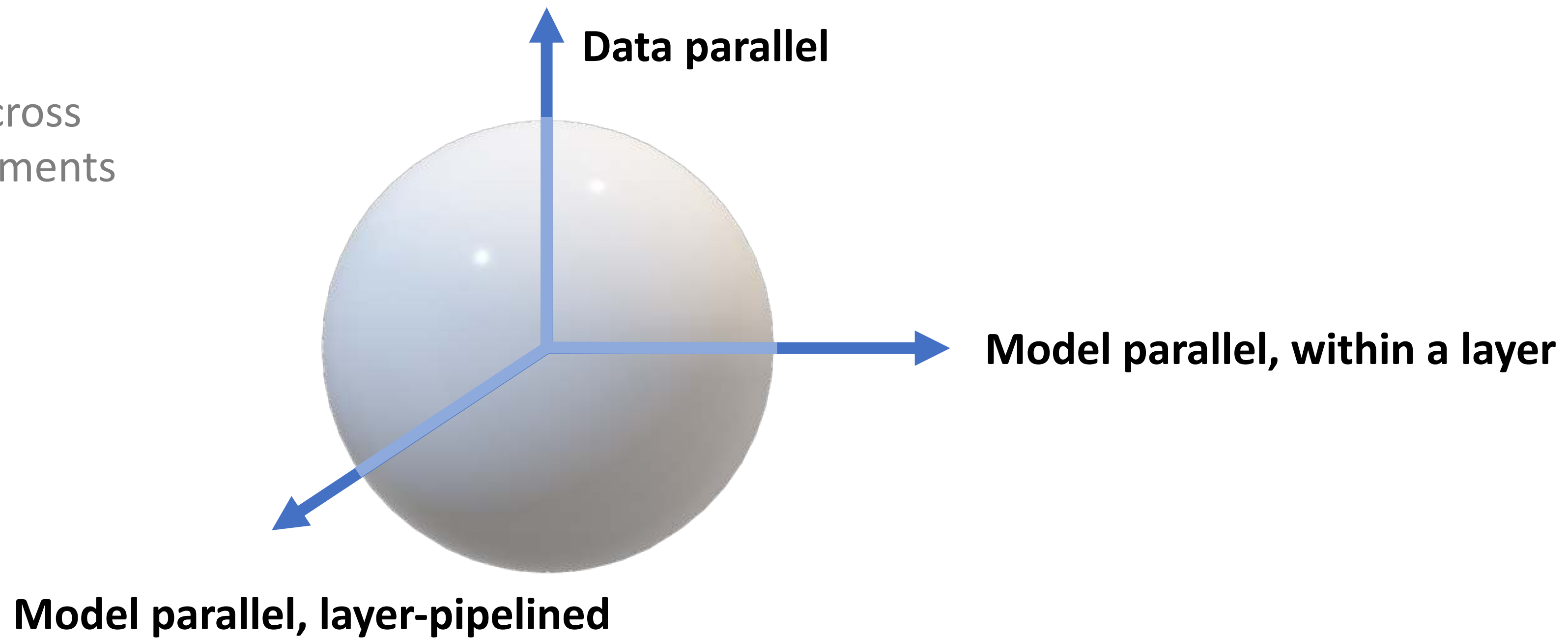
- Use **a blend** of distribution strategies: all types of **model parallel** + **data parallel**
  - Rely on model parallel first, as it doesn't depend on batch size
  - Add data parallel for small models
- Dynamically choose the **execution strategy** optimized for **different models**



Leverage 400,000 tightly-connected cores to accelerate deep learning

- Use **a blend** of distribution strategies: all types of **model parallel** + **data parallel**
  - Rely on model parallel first, as it doesn't depend on batch size
  - Add data parallel for small models
- Dynamically choose the **execution strategy** optimized for **different models**

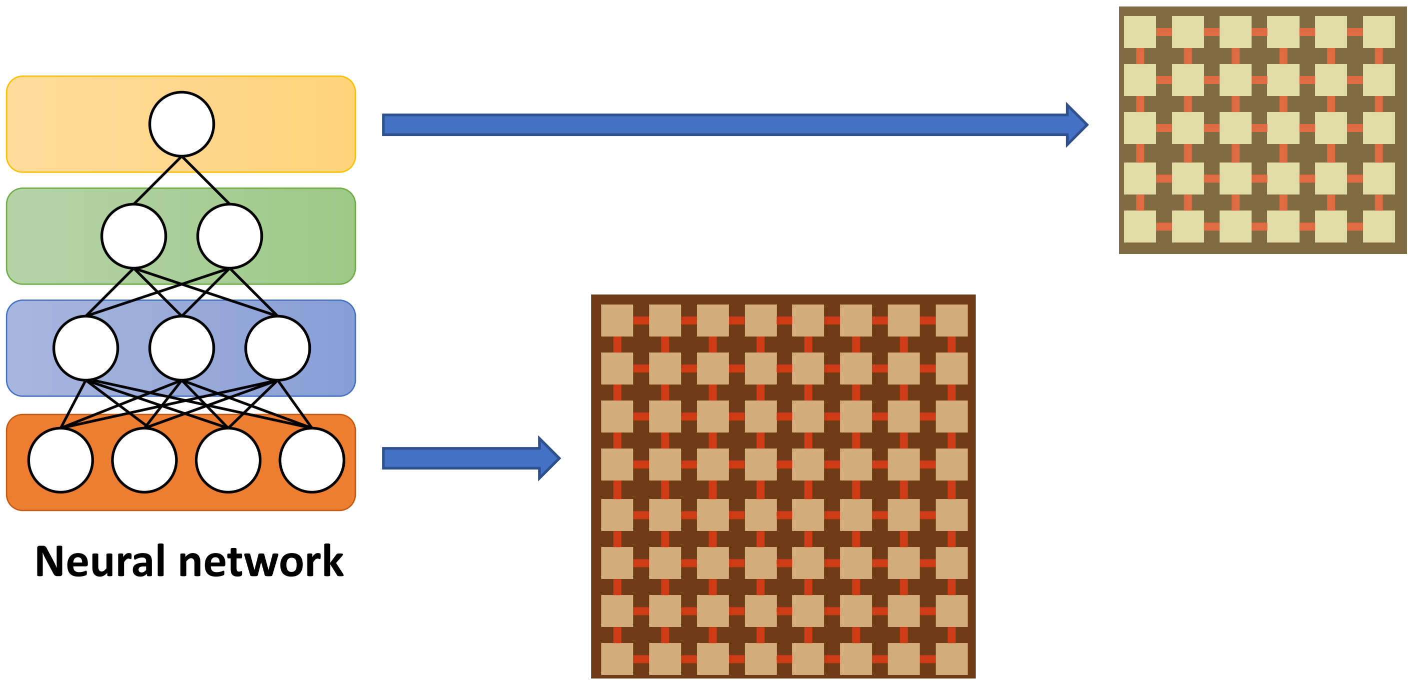
Distribution across processing elements of WSE



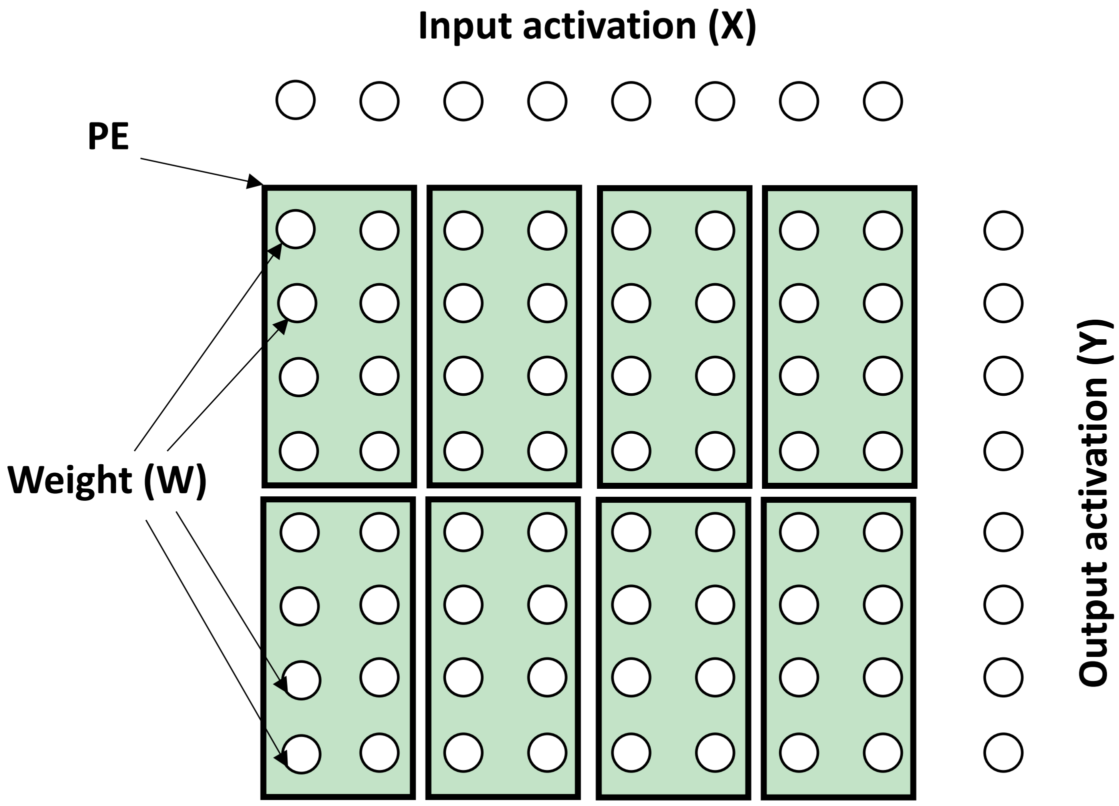
Model parallel within a layer

Distribute execution of a **single layer** across **multiple processing elements** (PEs)

- Compiler chooses an **optimal number** of PEs and **optimal shape** for every layer
- **Compute-heavy** layers get **larger PEs allocations**

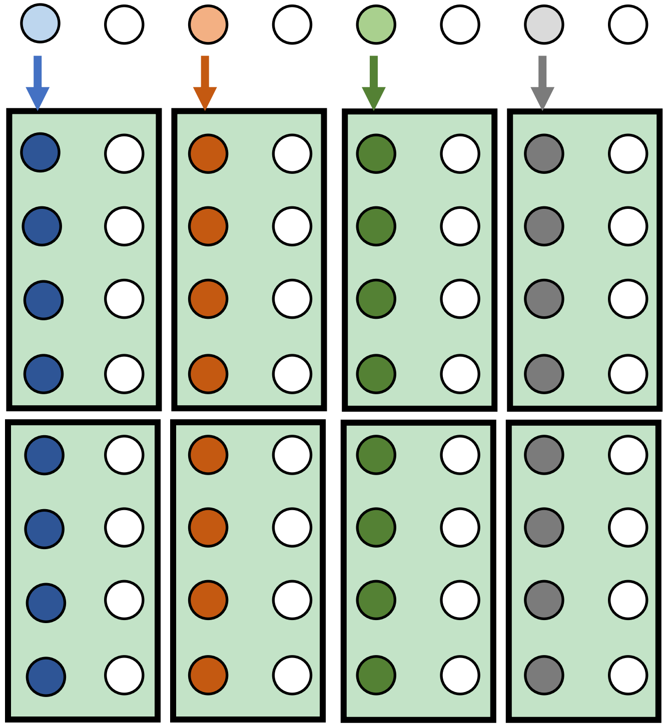


Example: FC Layer (GEMV)



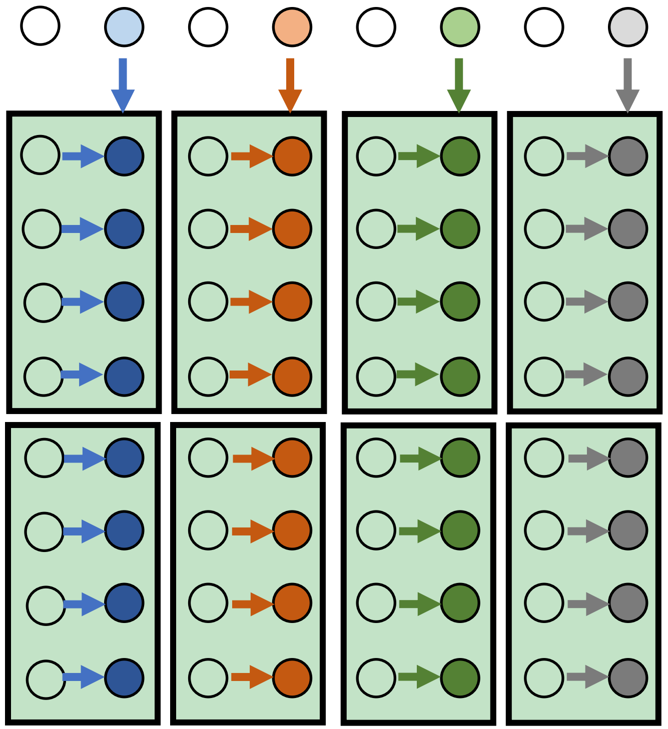
- Weights are stationary
  - Each PE holds a tile of weight matrix
  - Forward and backward pass share the same set of PEs
- Input activation comes in from vertical/horizontal direction
- Output activation goes out from horizontal/vertical direction

Example: FC Layer (GEMV)



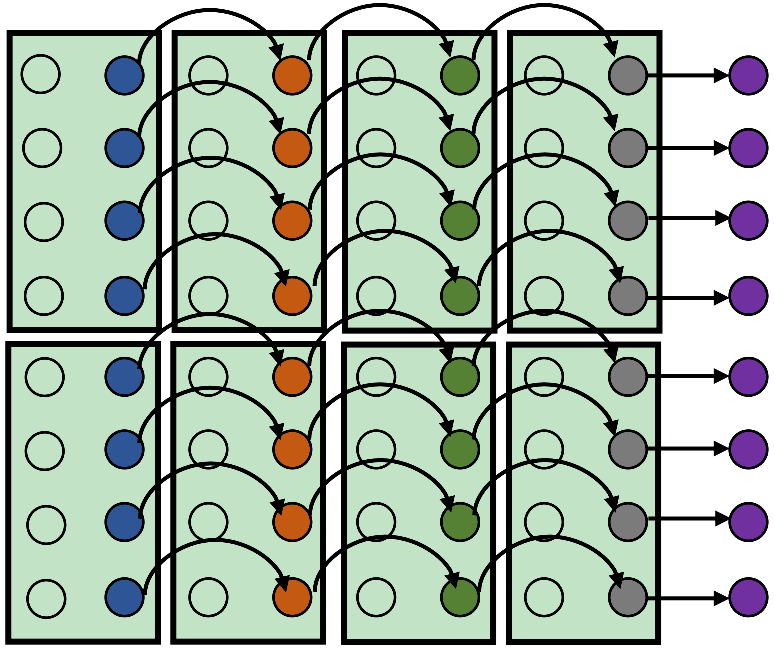
- Each PE works on a subset of input activation
- An input activation element is multiplied to a column of the weight matrix
- The results are accumulated to a set of accumulators (that is reset at the beginning)

Example: FC Layer (GEMV)



- Each PE works on a subset of input activation
- An input activation element is multiplied to a column of the weight matrix
- The results are accumulated to a set of accumulators (that is reset at the beginning)

Example: FC Layer (GEMV)

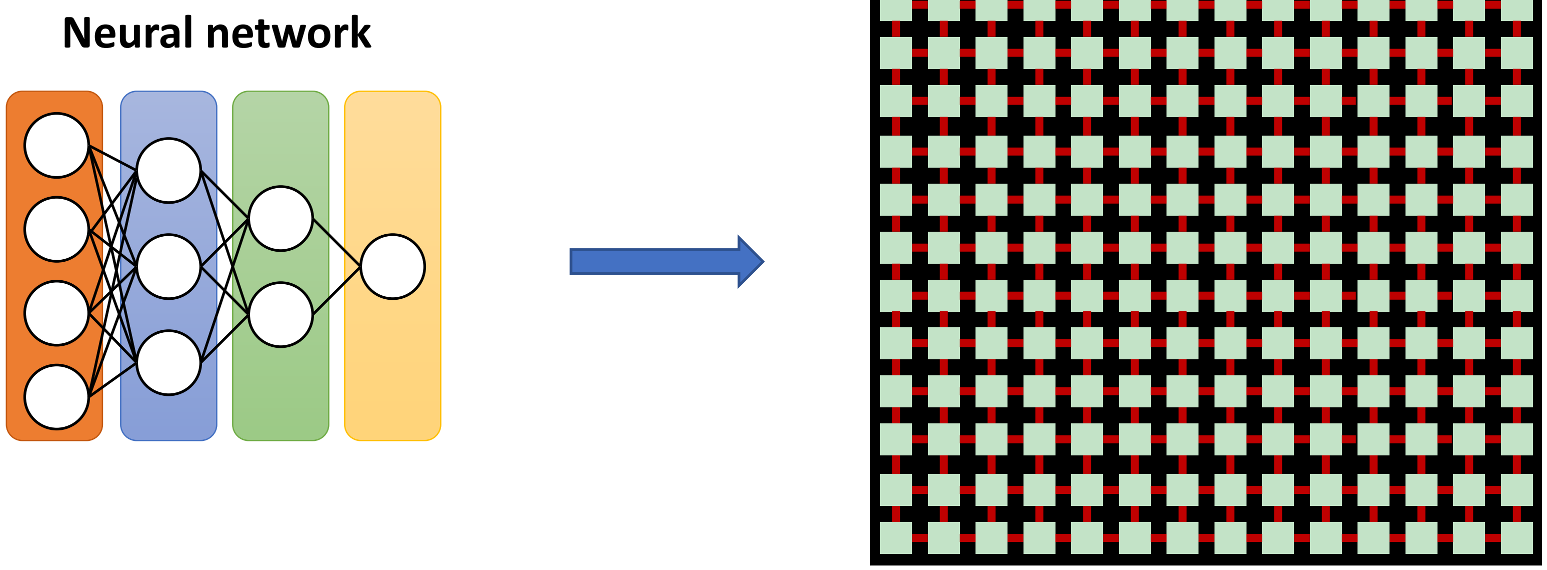


- Each PE has a partial sum of a subset of result (output activation)
- Partial sums are accumulated to produce the final result
- Latency of partial sum accumulation is mitigated with input activation streaming (GEMM)

cerebras

Model parallel, layer-pipelined

Distribute execution of **multiple layers** across **different fabric sections** and keep **entire model in fast on-chip memory**

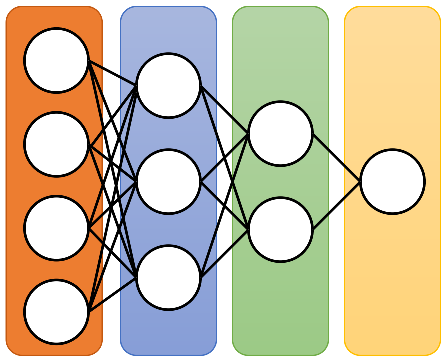


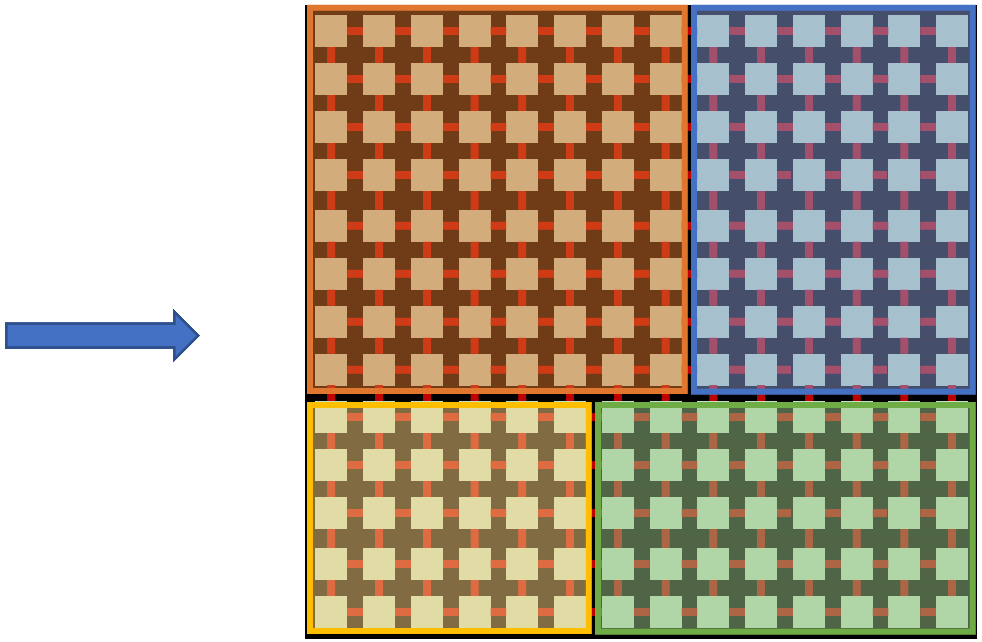
Model parallel, layer-pipelined

Distribute execution of **multiple layers** across **different fabric sections** and keep **entire model in fast on-chip memory**

- Compiler maps layers to the fabric to optimize compute and communication
- Adjacent layers typically placed next to each other

Neural network

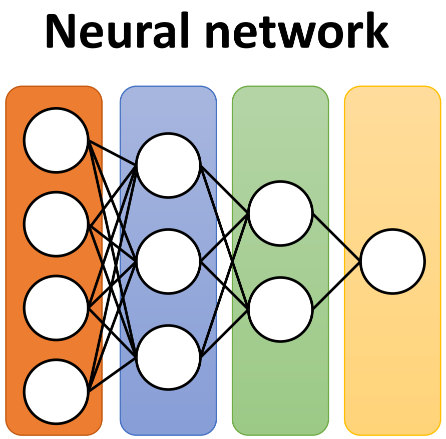


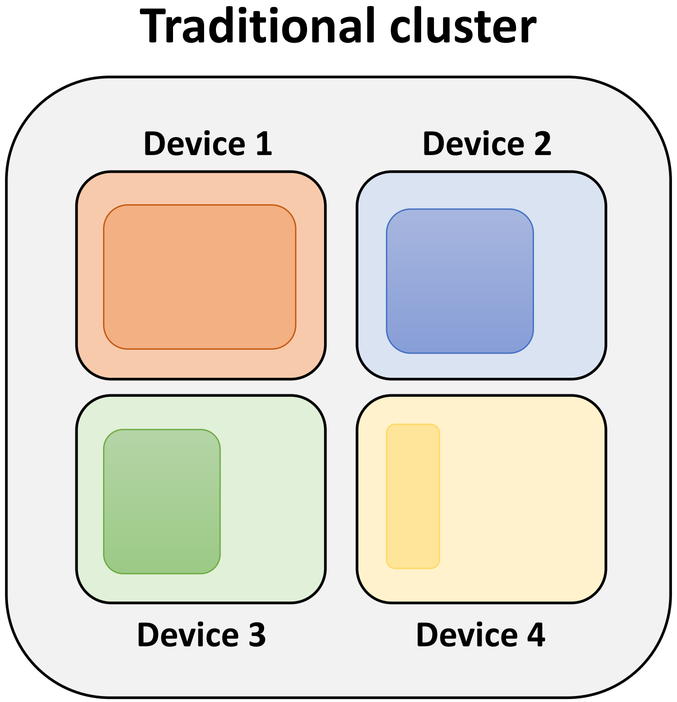


Model parallel, layer-pipelined

Challenging on a traditional cluster:

- Limited communication between devices
- Work should be dividable into fixed units of compute
- ML researcher should choose optimal distribution

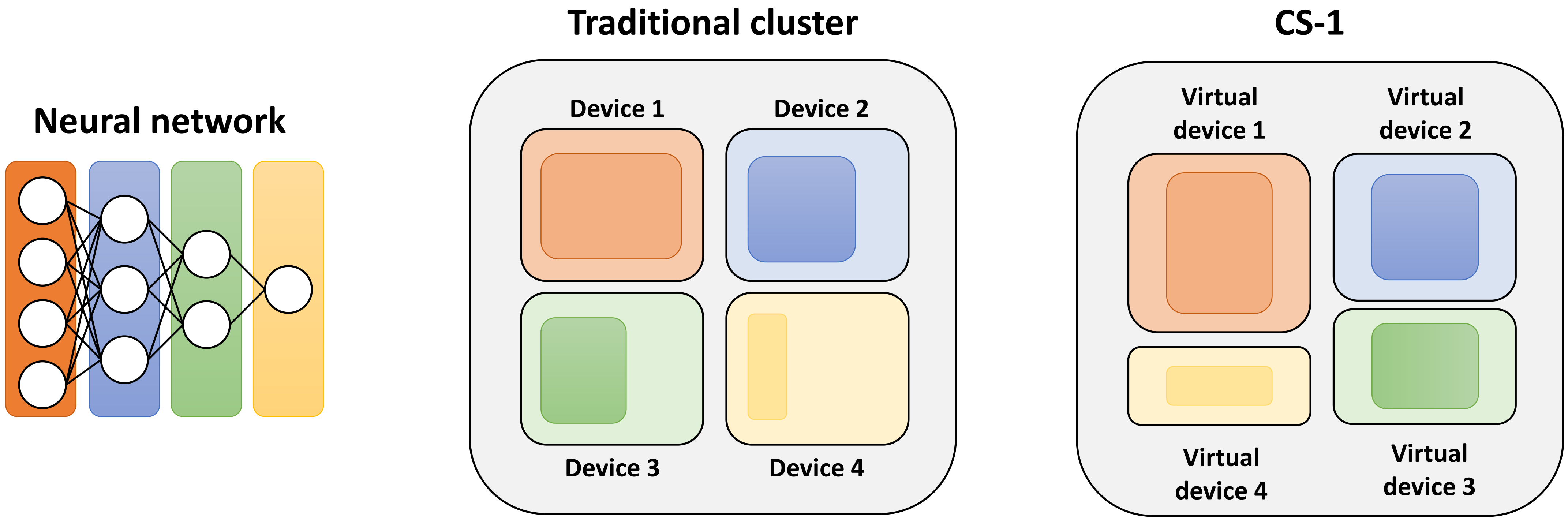




Model parallel, layer-pipelined

Easy and efficient on CS-1:

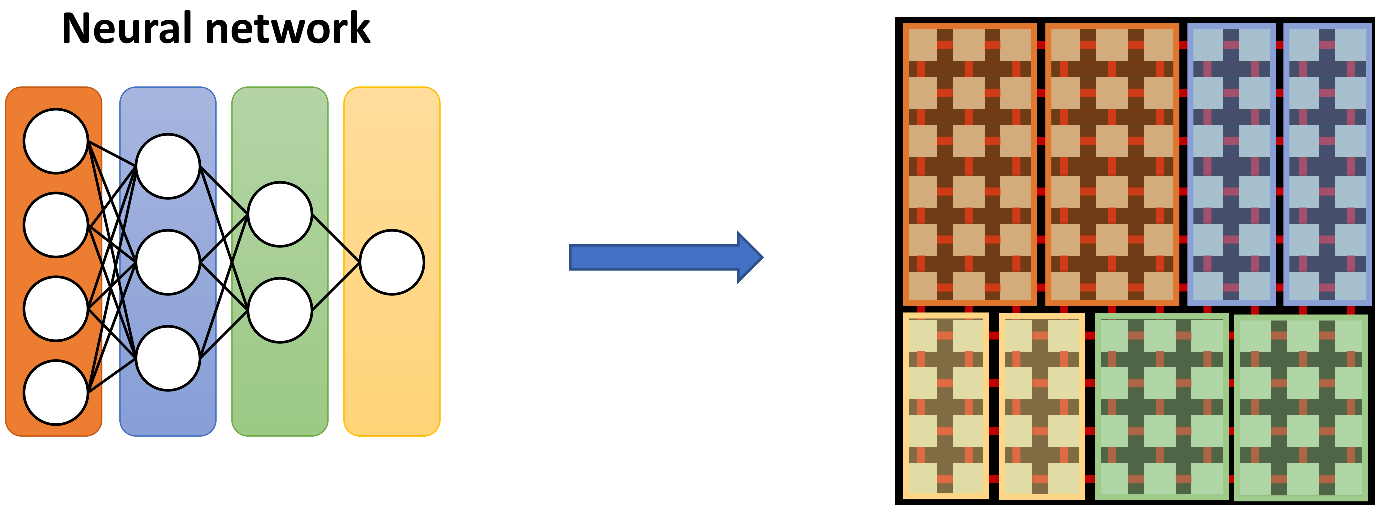
- Low-latency high-bandwidth communication between all cores
- Flexible units of compute
- Cerebras compiler automatically chooses optimal distribution



Data parallel

Replicate layers for higher performance on small models

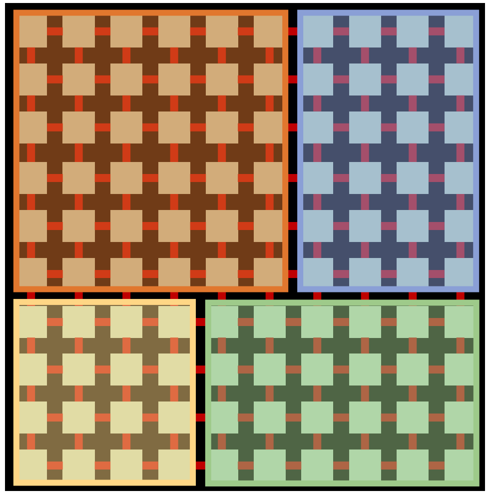
- Use very small batch size (down to 1 sample) per replica
  - Enabled by high bandwidth low latency local memory
  - Result: medium effective batch size
- Place replicas on adjacent fabric sections
  - Low synchronization overheads due to high-bandwidth low-latency connections between PEs



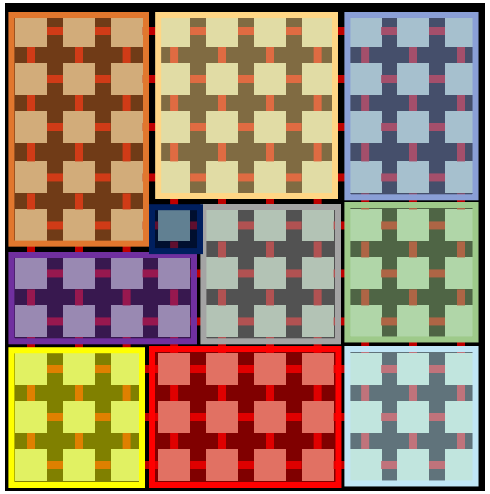
In summary – WSE uses a blend of parallel execution modes

- Single algorithm uses both model and data parallelism in optimization
- Execution strategies optimized for different neural networks

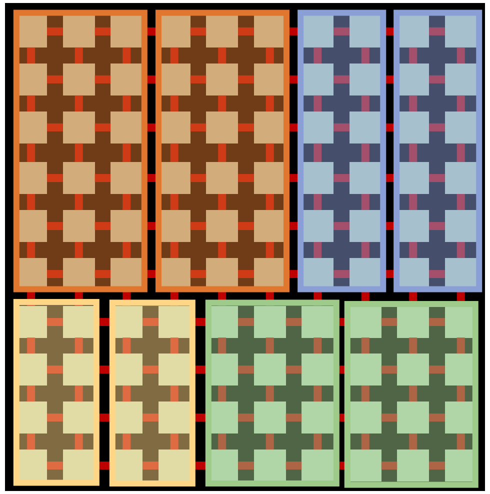
Few large layers: mostly model parallel within each layer



More layers: "more" layer-pipelined



Few small layers: model parallel + data parallel



In summary – WSE uses a blend of parallel execution modes

- Single algorithm uses both model and data parallelism in optimization
- Execution strategies optimized for different neural networks

But scale in deep learning is **not only** about **efficient distribution…**

it's also about **compute flexibility.**

Future path: larger *and* smarter models

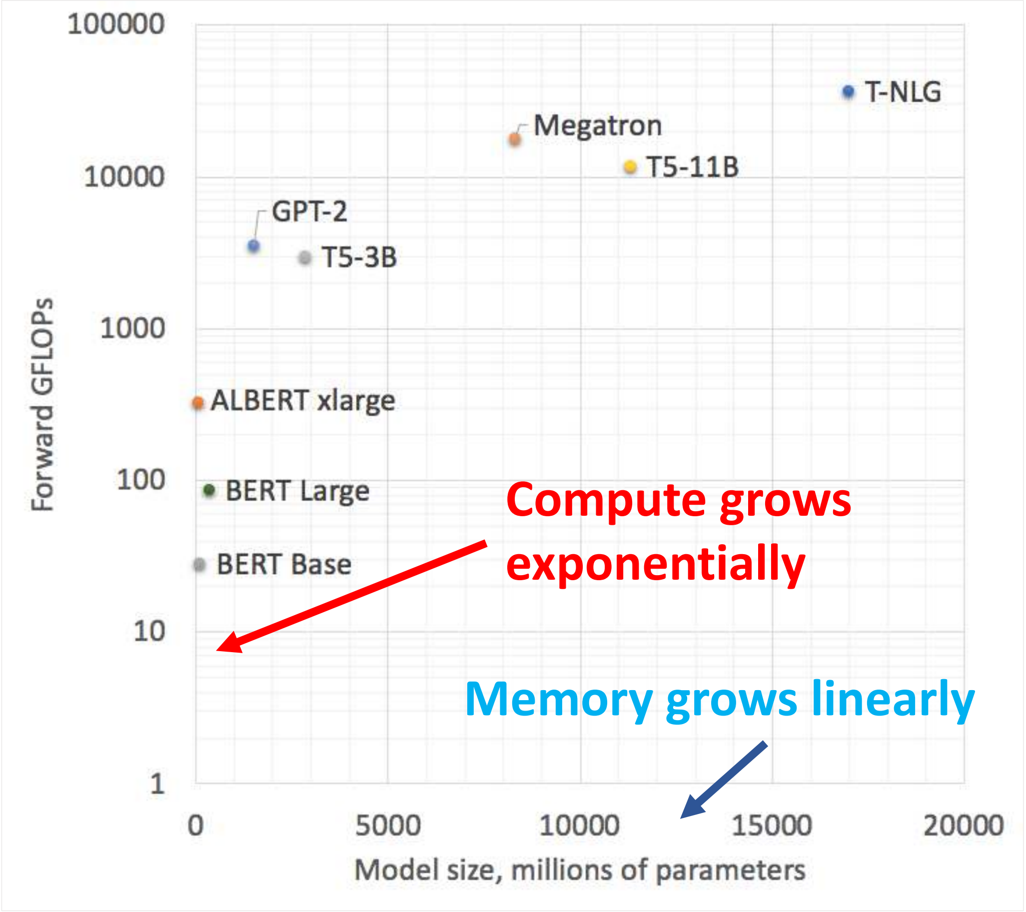
Brute-force scaling is historical path to better models.

- This is **challenging**
- Memory needs to grow
- Compute needs to grow

Algorithmic innovations give more efficient models.

- These are **promising but challenge existing hardware**.

CS-1 delivers both. Extreme scale with fewer nodes.
Flexible compute for smarter, efficient models.



CS-1 is designed to unlock smarter techniques and scale

CS-1 has a data flow architecture

- Flexibility to stream *token by token*
- Inherent sparsity harvesting

CS-1 is a MIMD architecture

- Can program each core independently
- Perform different operations on different data

CS-1 was built to **enable the next generation of models** otherwise limited today.

CS + new techniques → *efficient*, extreme-scale models

	Memory	Compute		SIMD	CS-1	Notes
Shared weights *eg ALBERT*	↓	↑		✓	✓	Same accuracy in only ~20% the size
Dynamic depth: - per batch - per sequence - per token *eg Universal Transformer*	=	↓		✗	✓	FLOPs reduction: • per batch: 11% • per seq: 20% • per token: 50%
Activation sparsity *Eg dropout*	=	↓		✗	✓	Up to 50% FLOPs reduction at negligible accuracy loss
Attention sparsity *eg Sparse Transformers*	=	↓		✗	✓	Attention cost $O(n^2) \to O(n\sqrt{n})$
Irregularity *eg Evolved Transformer*	↓	↓		✗	✓	**Bigger bang for parameter buck**

cerebras

Summary

- Cerebras WSE is a 2D mesh of **400,000 programmable processing elements**
- Cerebras Graph Compiler can automatically choose the **optimal blend of parallel execution strategies** for each given model
- **No communication or memory bottlenecks** due to local memory and high-bandwidth, low-latency fabric
- MIMD + data flow architecture provide **unique flexibility** to enable the **next generation** of models

Performance of a cluster, ease of use of a single device, and unique flexibility

Thank you

natalia@cerebras.net