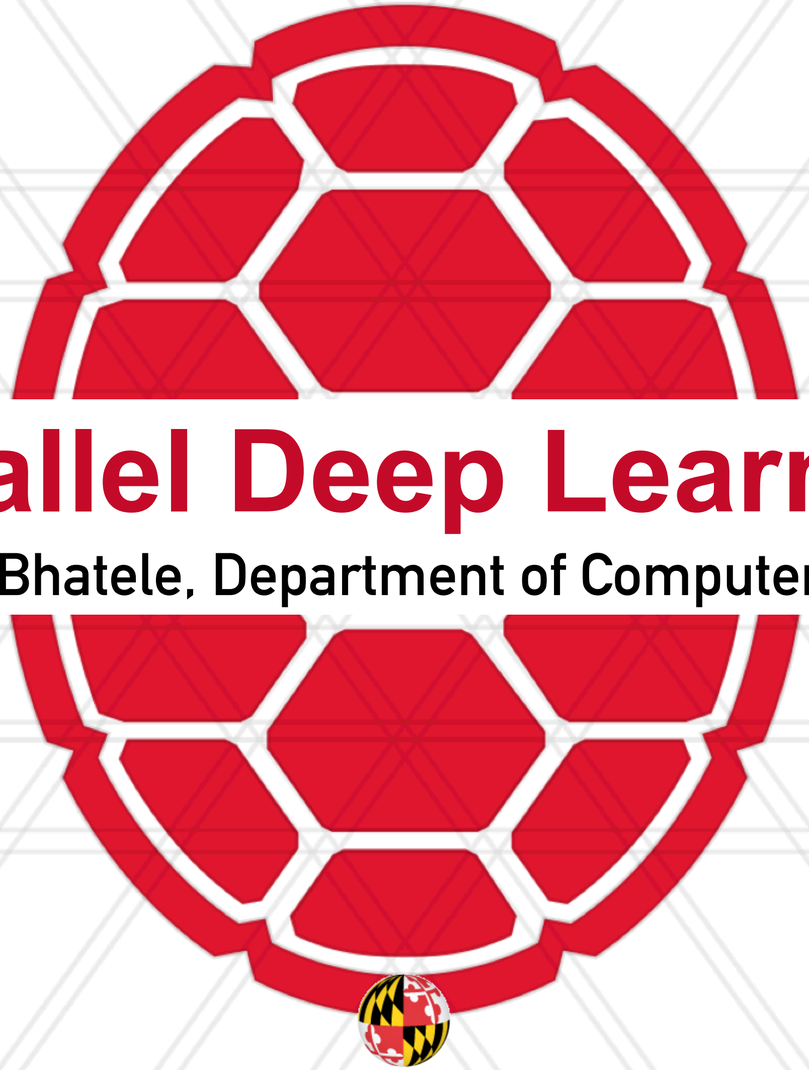
Parallel Computing (CMSC416 / CMSC616)



Parallel Deep Learning

Abhinav Bhatele, Department of Computer Science

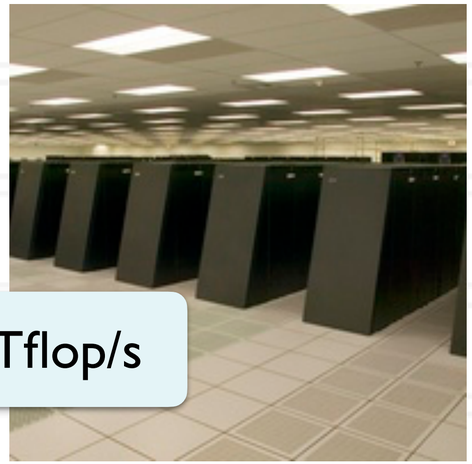
UNIVERSITY OF MARYLAND

Annoucements

- All extra credit assignments have been posted
  - Due on Dec 13 11:59 pm (no extensions for any reason)
- Dec 11 lecture will be a review session
- Final exam scheduled for Dec 15, 10:30 am, in this room

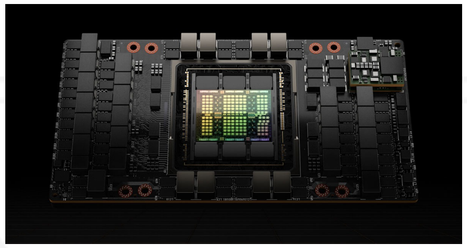
The evolution of HPC systems and rise of a new revolution in AI

- In the last two decades, an enormous amount of compute power has become available
- Large datasets and open source software such as PyTorch have also emerged
- Led to a frenzy in the world of AI and the effects are being felt in almost every other domain



Top500 Rpeak - 91.75 Tflop/s

IBM Blue Gene/L, 2004

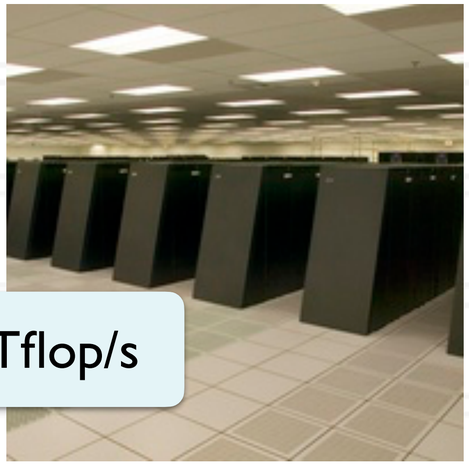


FP64 - 34 Tflop/s

NVIDIA H100, 2024

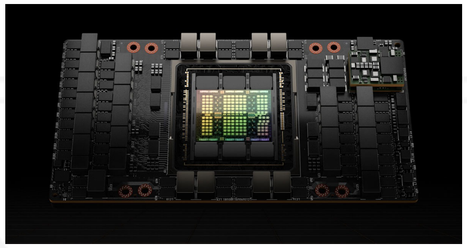
The evolution of HPC systems and rise of a new revolution in AI

- In the last two decades, an enormous amount of compute power has become available
- Large datasets and open source software such as PyTorch have also emerged
- Led to a frenzy in the world of AI and the effects are being felt in almost every other domain



Top500 Rpeak - 91.75 Tflop/s

IBM Blue Gene/L, 2004

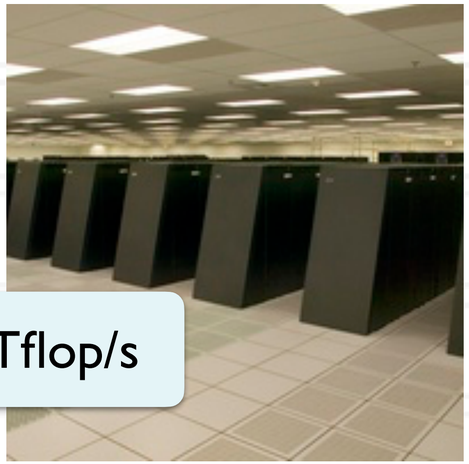


FP16 - 989 Tflop/s

NVIDIA H100, 2024

The evolution of HPC systems and rise of a new revolution in AI

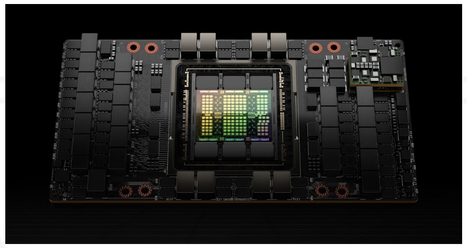
- In the last two decades, an enormous amount of compute power has become available
- Large datasets and open source software such as PyTorch have also emerged
- Led to a frenzy in the world of AI and the effects are being felt in almost every other domain



Top500 Rpeak - 91.75 Tflop/s

IBM Blue Gene/L, 2004

10.63 Exaflop/s!!



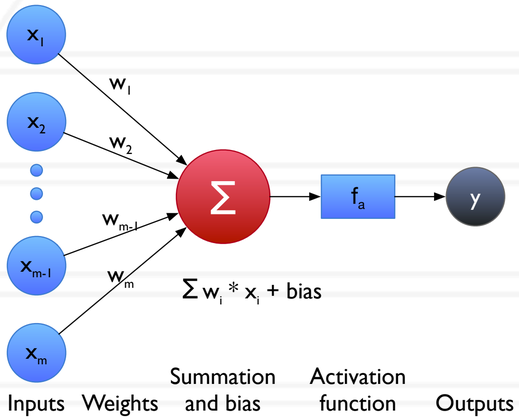
NVIDIA H100, 2024

Deep neural networks (DNNs)

- An area of machine learning that uses artificial neural networks to learn complex functions
  - Often from high-dimensional data: text, images, audio, …
- Widespread use in computer vision, natural language processing, etc.
- Neural networks can be used to model complex functions
- Several layers that process "batches" of the input data

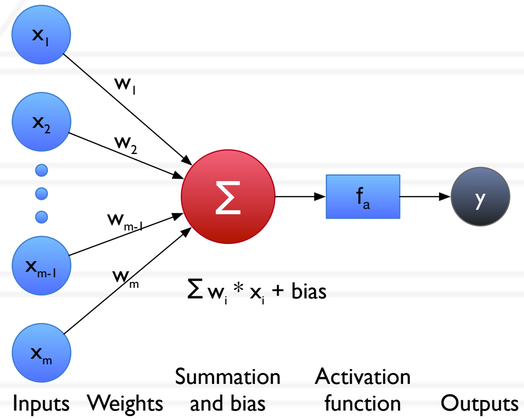
Deep neural networks (DNNs)

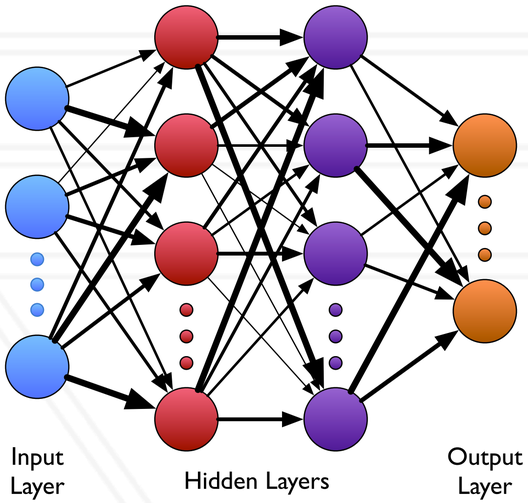
- An area of machine learning that uses artificial neural networks to learn complex functions
  - Often from high-dimensional data: text, images, audio, …
- Widespread use in computer vision, natural language processing, etc.
- Neural networks can be used to model complex functions
- Several layers that process "batches" of the input data



Deep neural networks (DNNs)

- An area of machine learning that uses artificial neural networks to learn complex functions
  - Often from high-dimensional data: text, images, audio, …
- Widespread use in computer vision, natural language processing, etc.
- Neural networks can be used to model complex functions
- Several layers that process "batches" of the input data



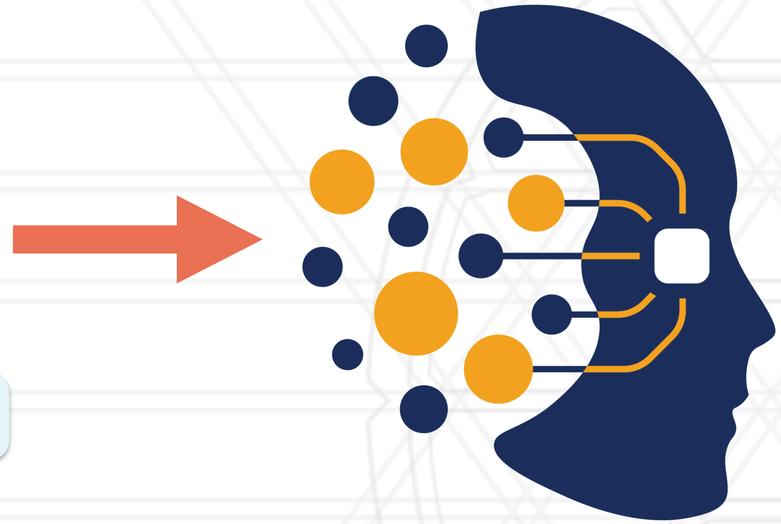


Deep learning and language models

Web data

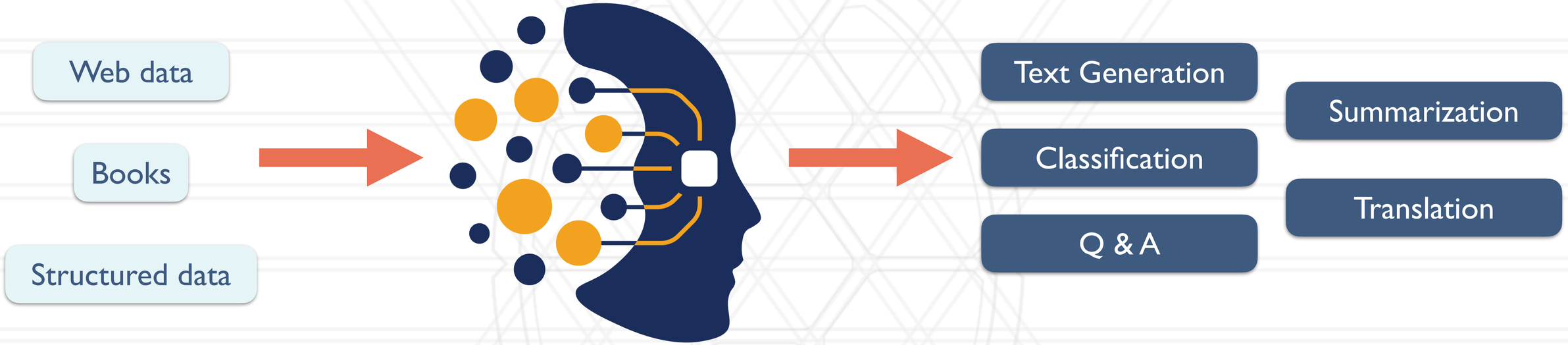
Books

Structured data



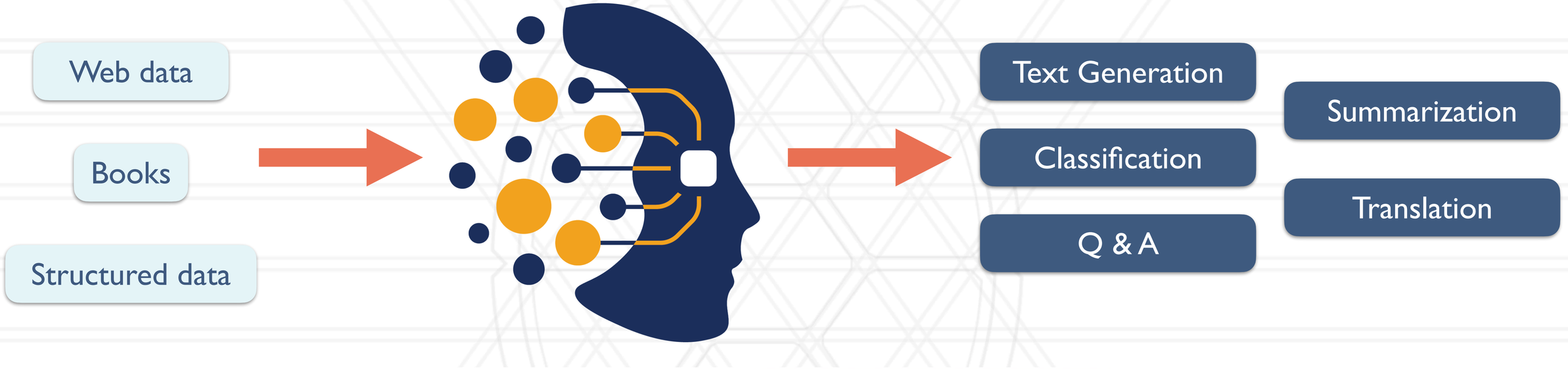
https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

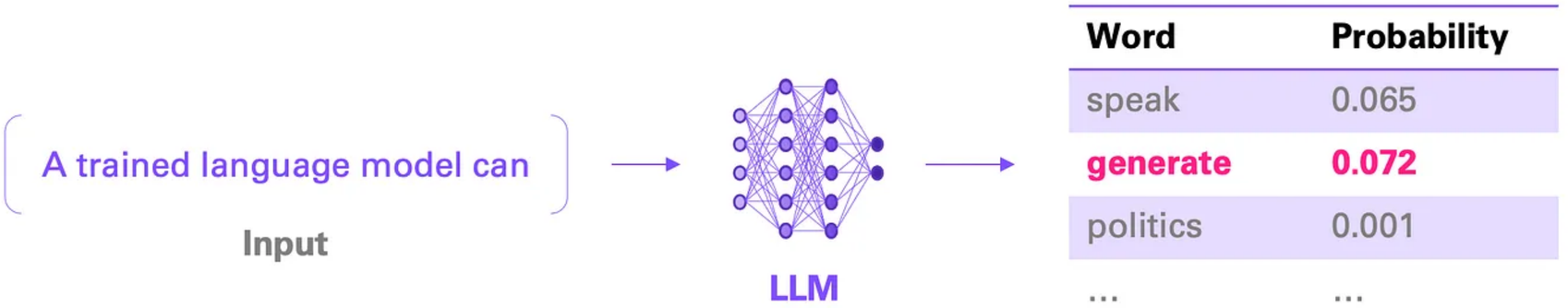
Deep learning and language models



https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

Deep learning and language models





Word	Probability
speak	0.065
generate	**0.072**
politics	0.001
...	...
walk	0.003

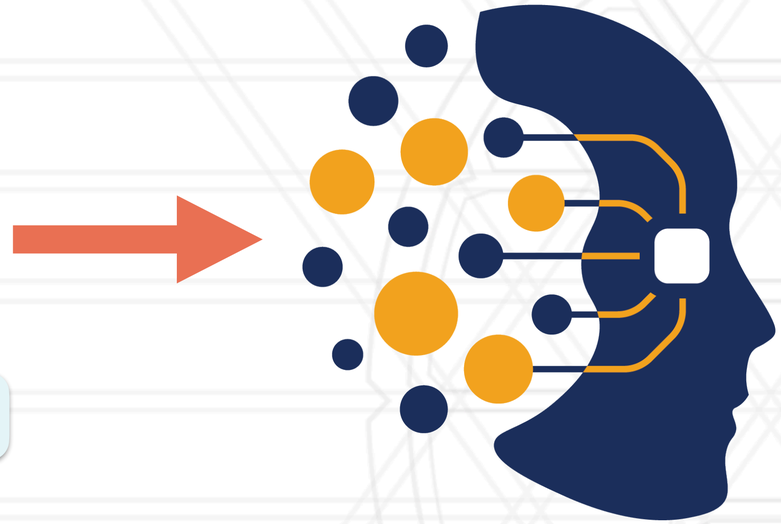
https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

Deep learning and language models

Web data

Books

Structured data



Text Generation

Classification

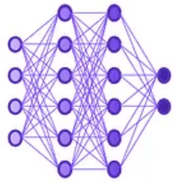
Q & A

Summarization

Translation

A trained language model can

Input



LLM

Word	Probability
speak	0.065
generate	**0.072**
politics	0.001
...	...
walk	0.003

Word	Probability
ability	0.002
text	0.084
coherent	**0.085**
...	...
ideas	0.041

https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

Other definitions

- Learning/training: task of selecting weights that lead to an accurate function
- Loss: a scalar proxy that when minimized leads to higher accuracy
- Gradient descent: process of updating the weights using gradients (derivates) of the loss weighted by a learning rate
- Mini-batch: Small subsets of the dataset processed iteratively
- Epoch: One pass over all the mini-batches

HPC / parallel clusters are critical for AI

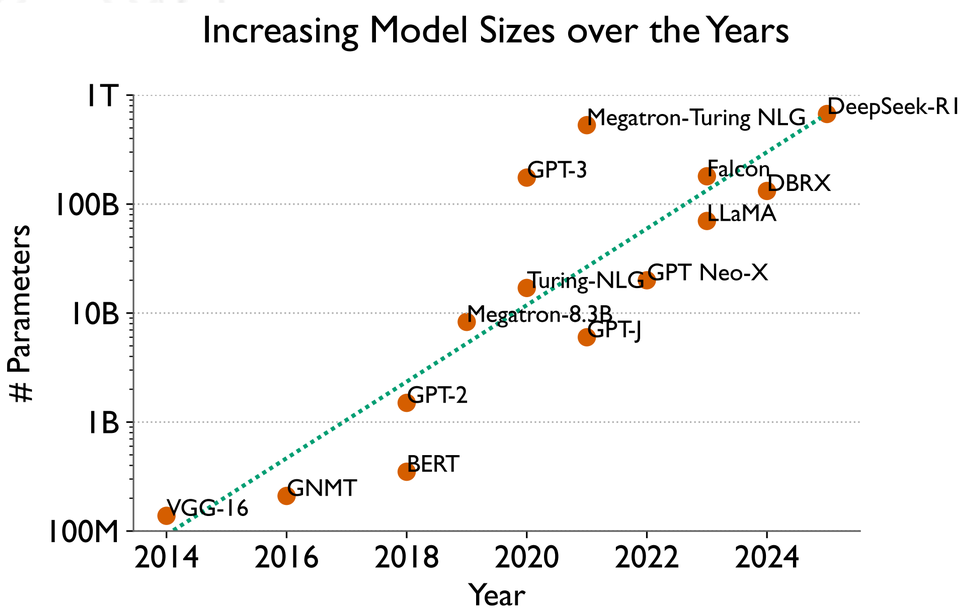
- Largest language model you can train on an H100 96 GB GPU: ~3.5-4 billion parameters
- On a single node (with four H100 GPUs): around ~16 billion parameters model

HPC / parallel clusters are critical for AI

- Largest language model you can train on an H100 96 GB GPU: ~3.5-4 billion parameters
- On a single node (with four H100 GPUs): around ~16 billion parameters model
- Training a 16B parameter would take 33 years!

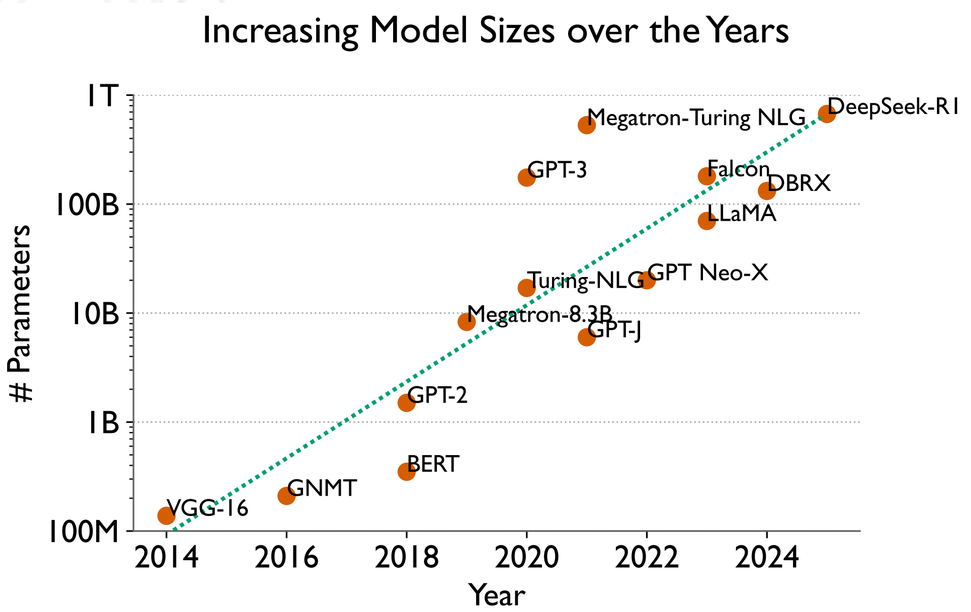
HPC / parallel clusters are critical for AI

- Largest language model you can train on an H100 96 GB GPU: ~3.5-4 billion parameters
- On a single node (with four H100 GPUs): around ~16 billion parameters model
- Training a 16B parameter would take 33 years!



HPC / parallel clusters are critical for AI

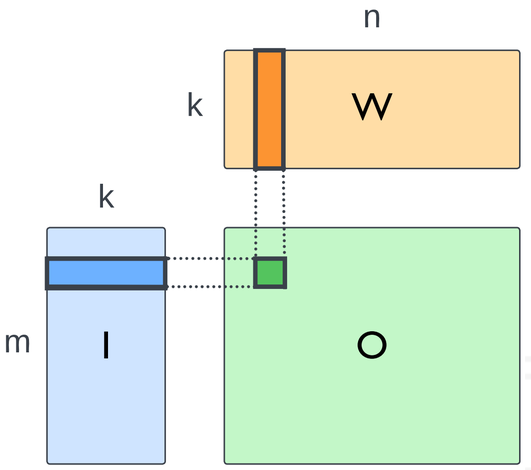
- Largest language model you can train on an H100 96 GB GPU: ~3.5-4 billion parameters
- On a single node (with four H100 GPUs): around ~16 billion parameters model
- Training a 16B parameter would take 33 years!
- OpenAI's GPT 4.0 is estimated to have 1.8 trillion parameters
- Meta's Llama-3.1-405B has more than 400 billion parameters



Scaling distributed AI is challenging

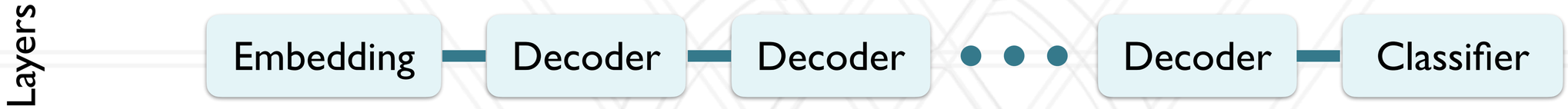
- Single GPU performance: ensure efficient compute kernels
- Multi-node performance: scalable communication, especially collectives
- File I/O: for certain categories of AI models such as image, video, etc.
- We need scalable algorithms AND good practical implementations of them

Sequential LLM training

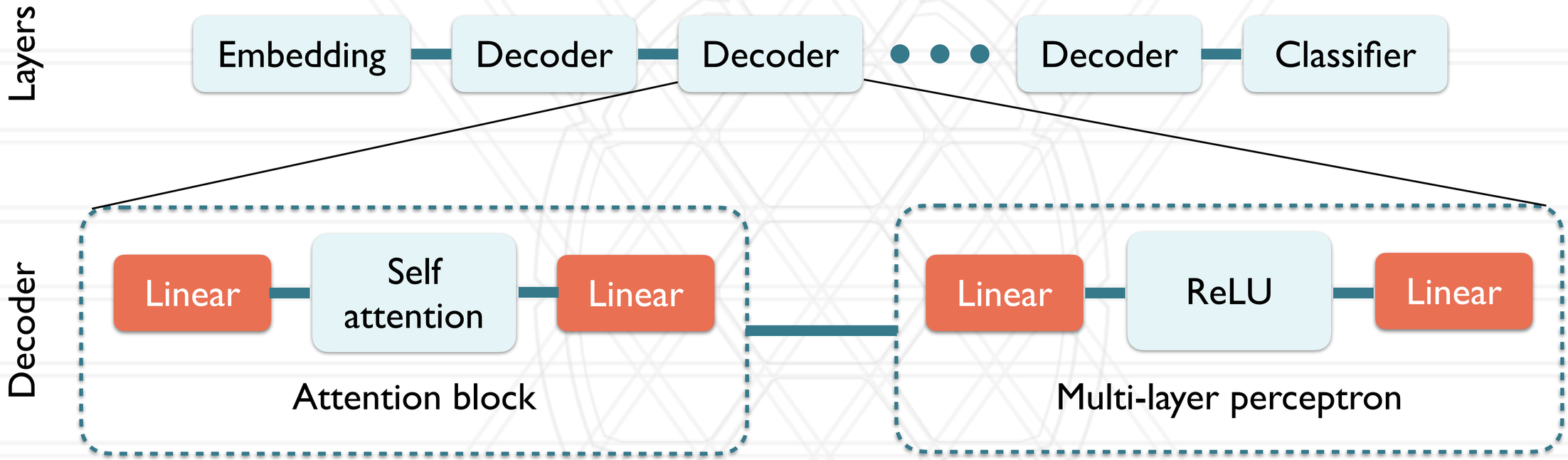



```
while (remaining_batches) {
    Read a single batch

    Forward pass: perform matrix multiplies to compute
        output activations, and a loss on the batch

    Backward pass: matrix multiplies to compute gradients of
        the loss w.r.t. parameters via backpropagation

    Optimizer step: use gradients to update the weights or
        parameters such that loss is gradually reduced
}
```

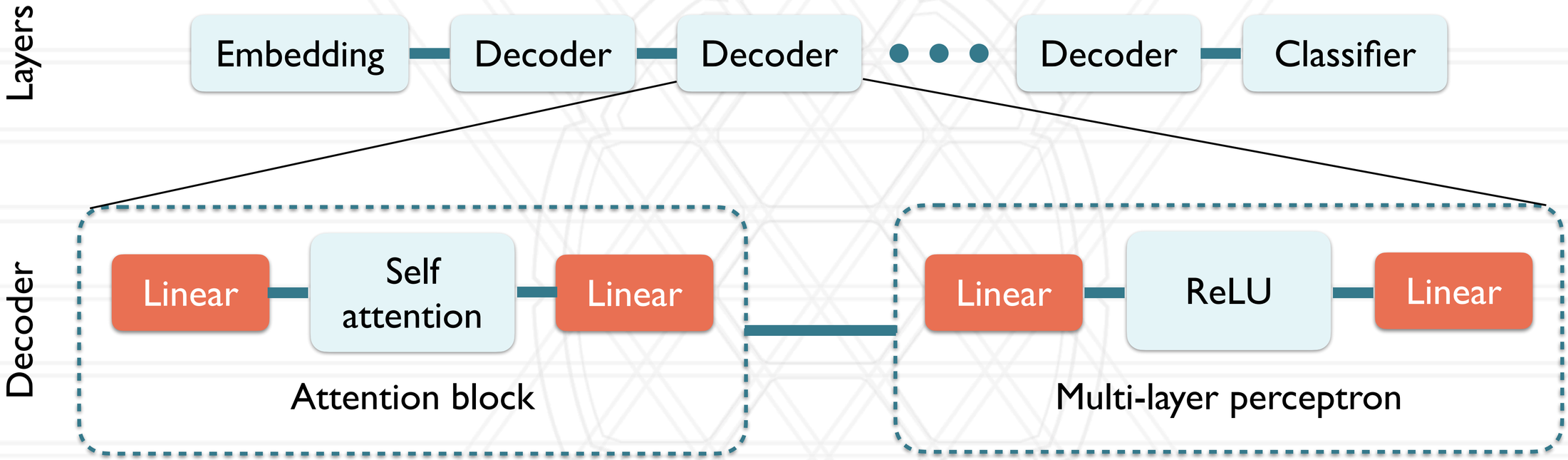
Why is LLM training well-suited for HPC?

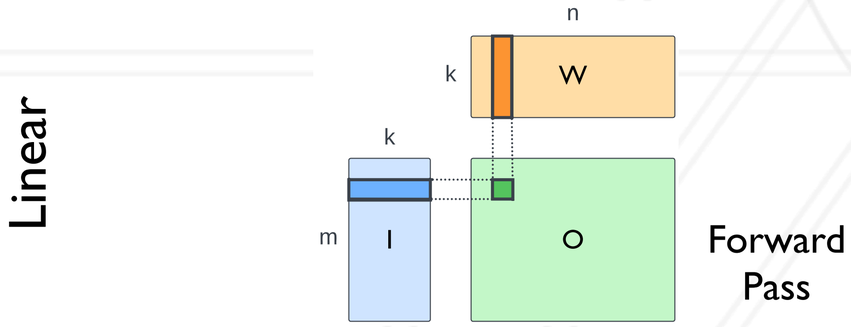


Why is LLM training well-suited for HPC?

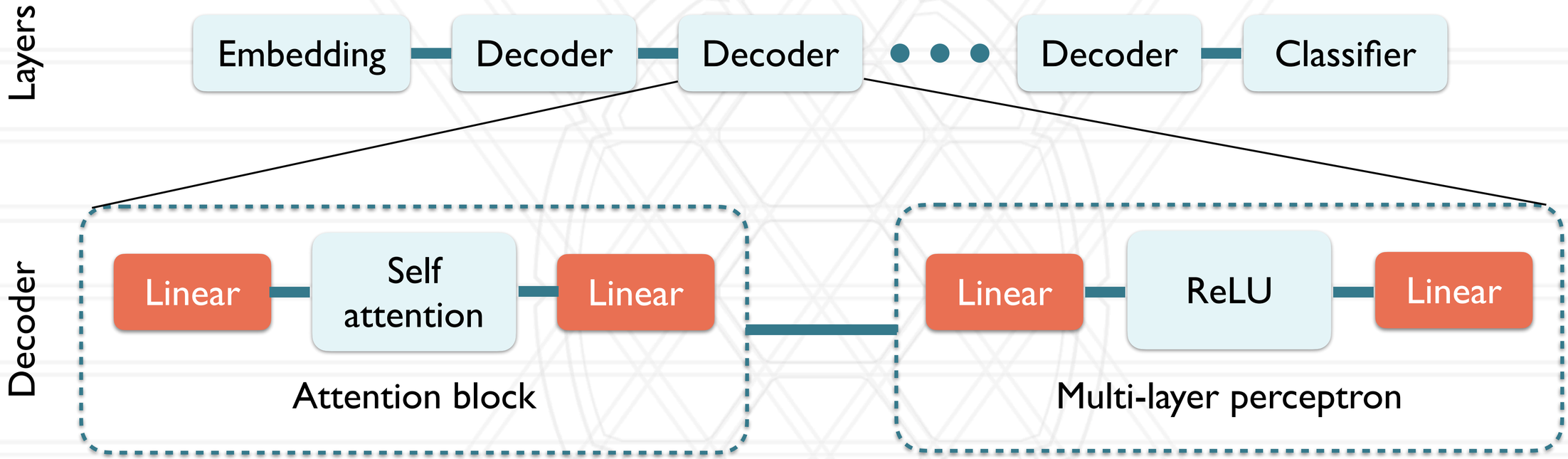


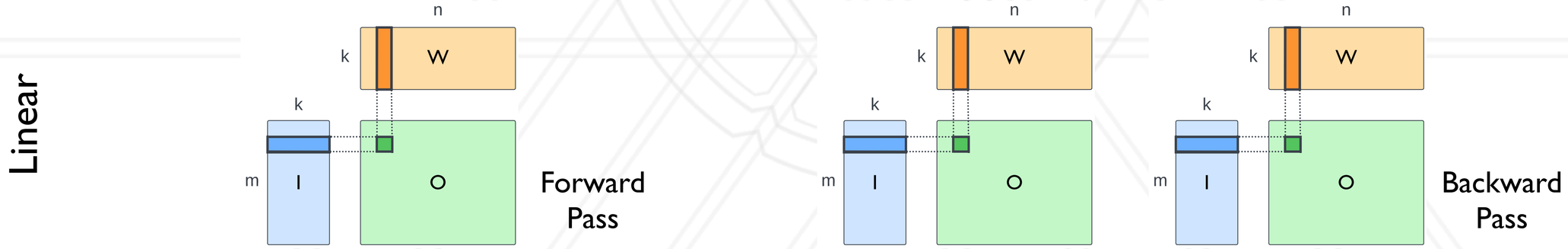
Why is LLM training well-suited for HPC?





Why is LLM training well-suited for HPC?





Parallel/distributed training

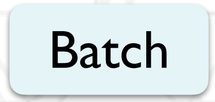
- Many opportunities for exploiting parallelism
- Iterative process of training (epochs)
- Many iterations per epoch (mini-batches)
- Many layers in DNNs

Data parallelism

- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO

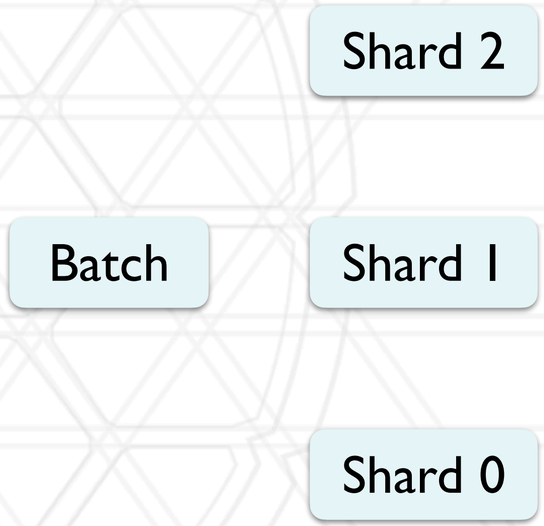
Data parallelism

- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO



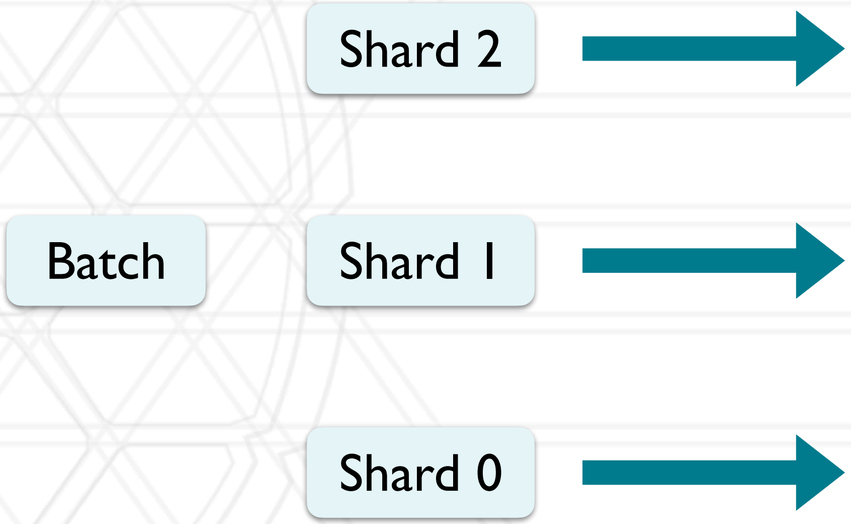
Data parallelism

- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO



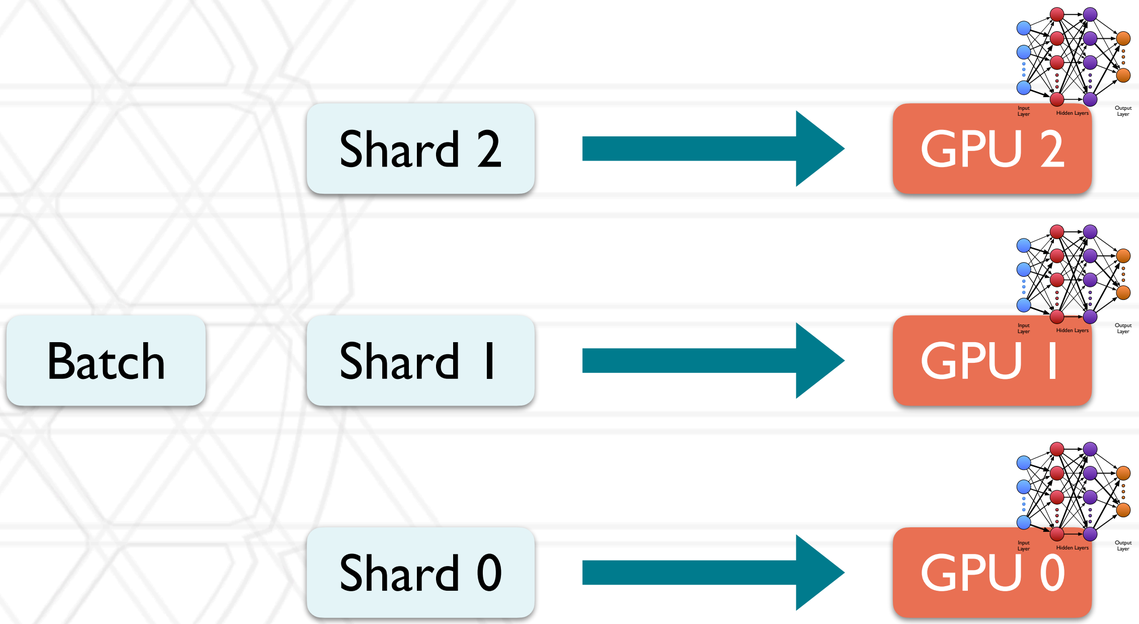
Data parallelism

- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO



Data parallelism

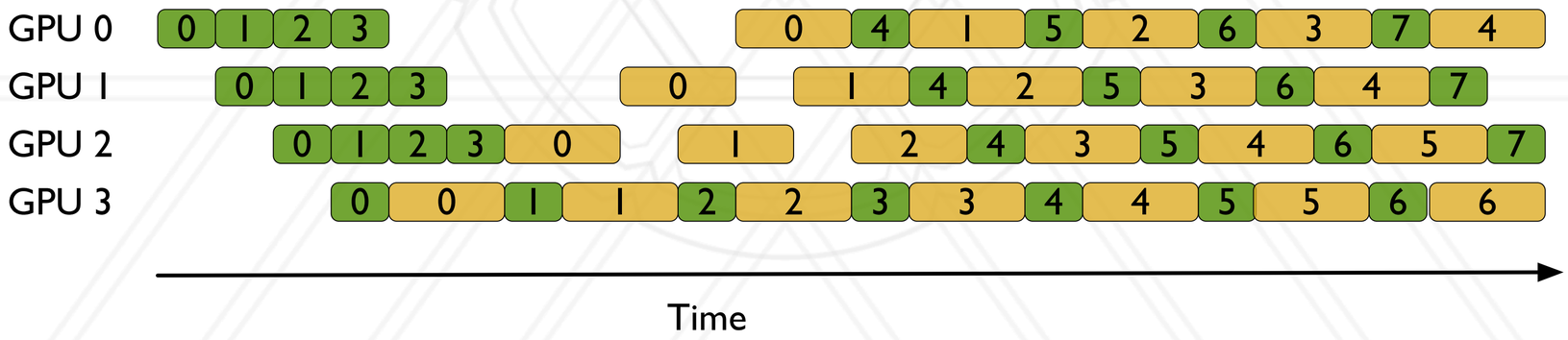
- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO



Data Parallelism

Inter-layer parallelism

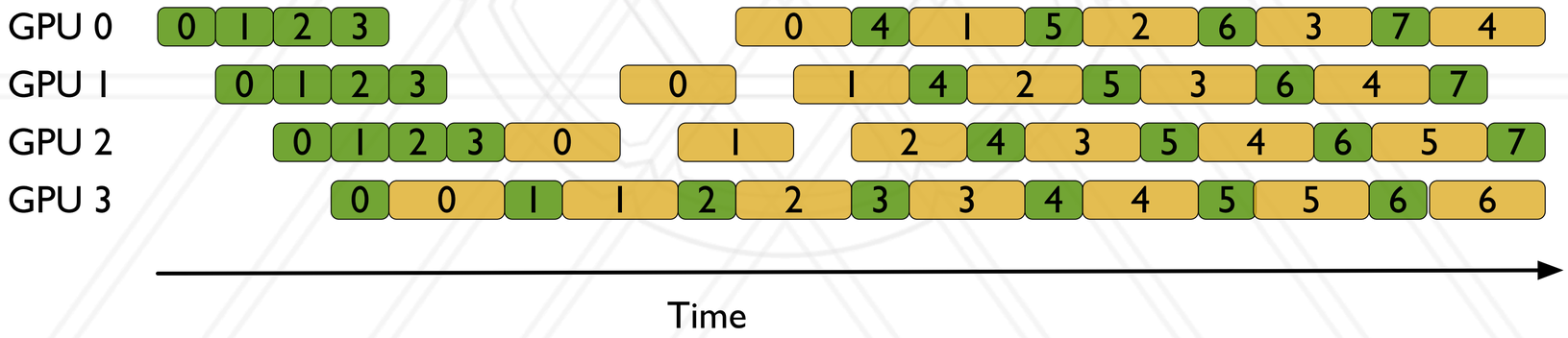
- Assign entire layers to different processes/GPUs
  - Ideally map contiguous subsets of layers
- Point-to-point communication (activations and gradients) between processes/GPUs managing different layers
- Use a pipeline of mini-batches to enable concurrent execution



Inter-layer parallelism

- Assign entire layers to different processes/GPUs
  - Ideally map contiguous subsets of layers
- Point-to-point communication (activations and gradients) between processes/GPUs managing different layers
- Use a pipeline of mini-batches to enable concurrent execution

Pipeline parallelism



Intra-layer parallelism

- Enables training neural networks that would not fit on a single GPU
- Distribute the work within each layer to multiple processes/GPUs
  - Essentially parallelize matrix operations such as matmuls across multiple GPUs
- Example: Megatron-LM

Intra-layer parallelism

Tensor parallelism

- Enables training neural networks that would not fit on a single GPU
- Distribute the work within each layer to multiple processes/GPUs
  - Essentially parallelize matrix operations such as matmuls across multiple GPUs
- Example: Megatron-LM

Hybrid parallelism

- Using two or more approaches together in the same parallel framework
- 3D parallelism: use all three
- Popular serial frameworks: pytorch, tensorflow
- Popular parallel frameworks: DDP, MeshTensorFlow, Megatron-LM, ZeRO, AxoNN

A four-dimensional hybrid parallel approach

- A hybrid parallelism approach

- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)

A four-dimensional hybrid parallel approach

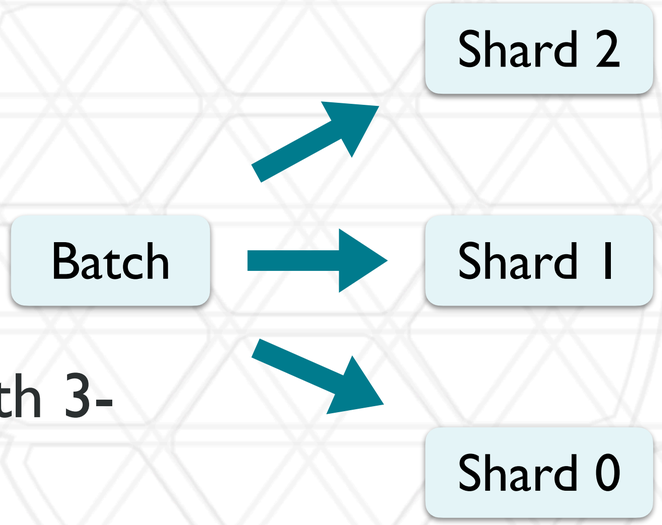
- A hybrid parallelism approach

Batch

- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)

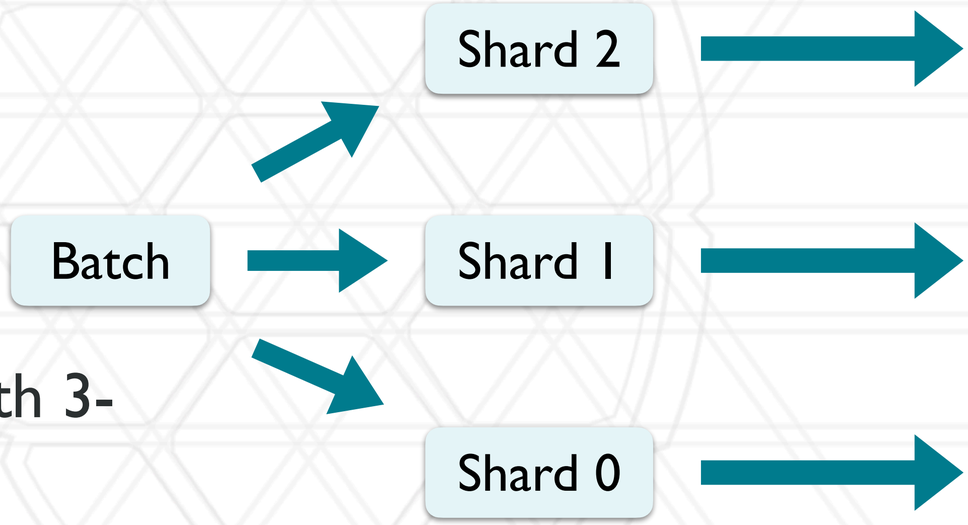
A four-dimensional hybrid parallel approach

- A hybrid parallelism approach
- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)



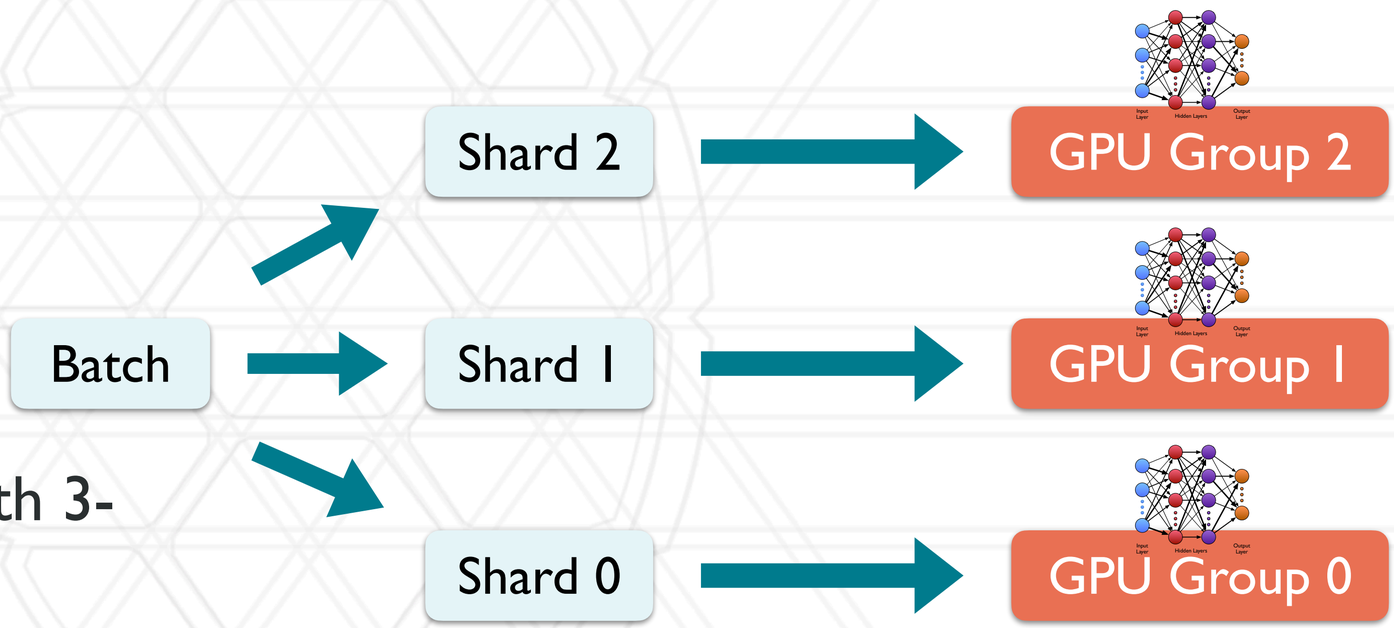
A four-dimensional hybrid parallel approach

- A hybrid parallelism approach
- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)



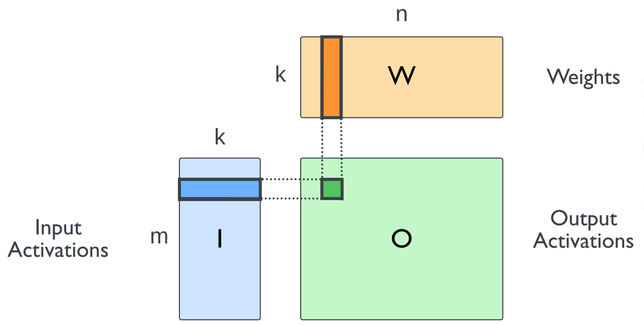
A four-dimensional hybrid parallel approach

- A hybrid parallelism approach
- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)



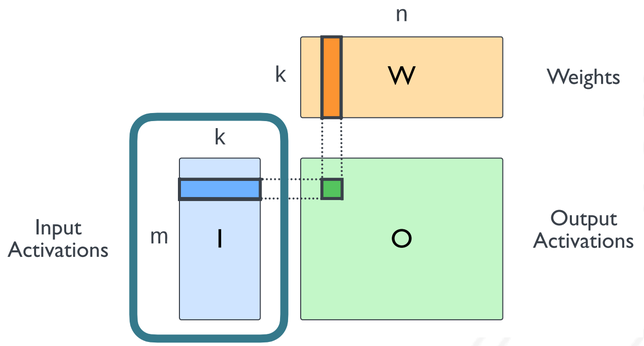
Enabling 3D parallel matrix multiplication in AxoNN

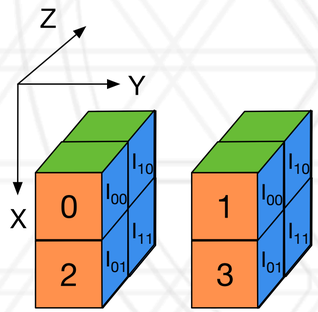
- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU



Enabling 3D parallel matrix multiplication in AxoNN

- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU

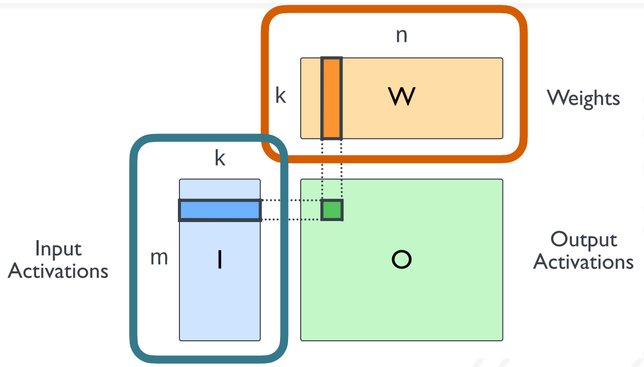


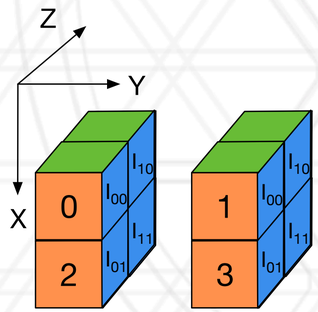


I

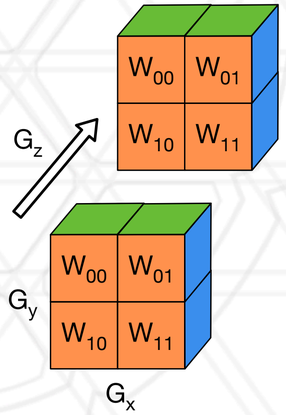
Enabling 3D parallel matrix multiplication in AxoNN

- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU





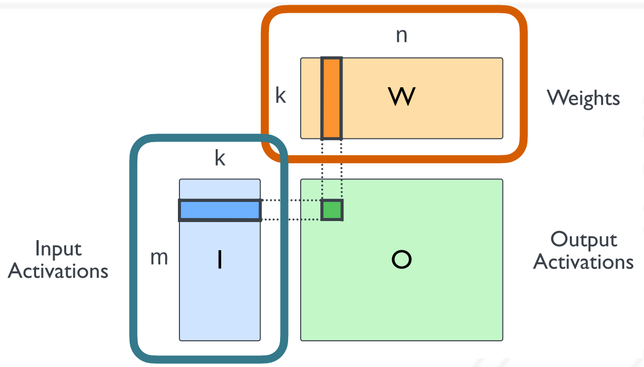
I

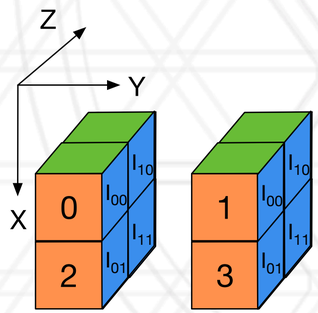


W

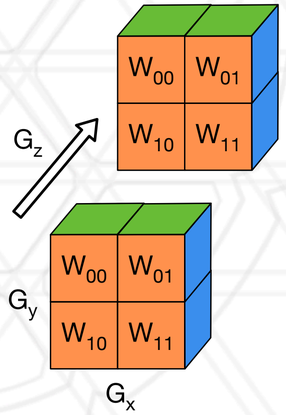
Enabling 3D parallel matrix multiplication in AxoNN

- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU

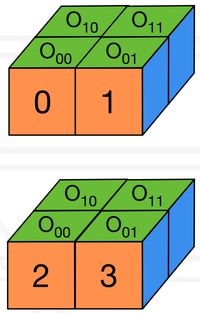




I



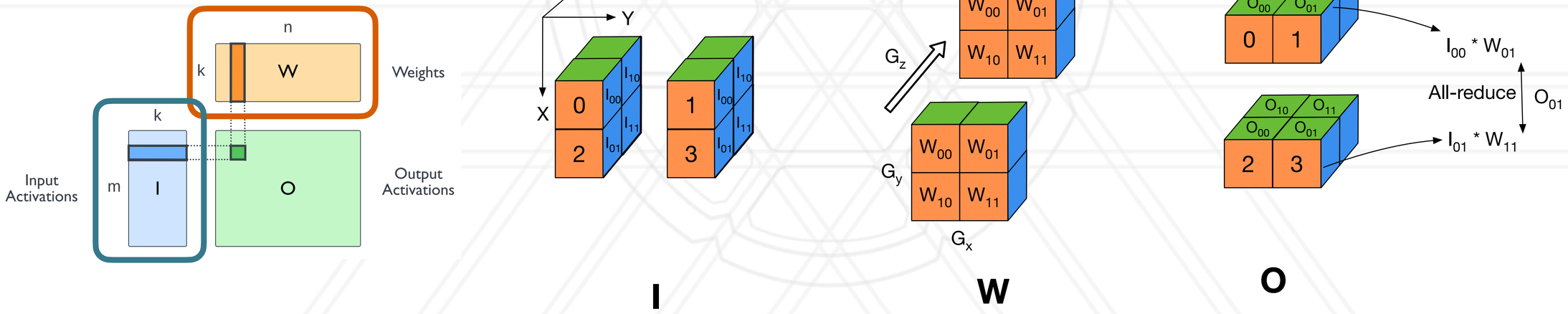
W



O

Enabling 3D parallel matrix multiplication in AxoNN

- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU



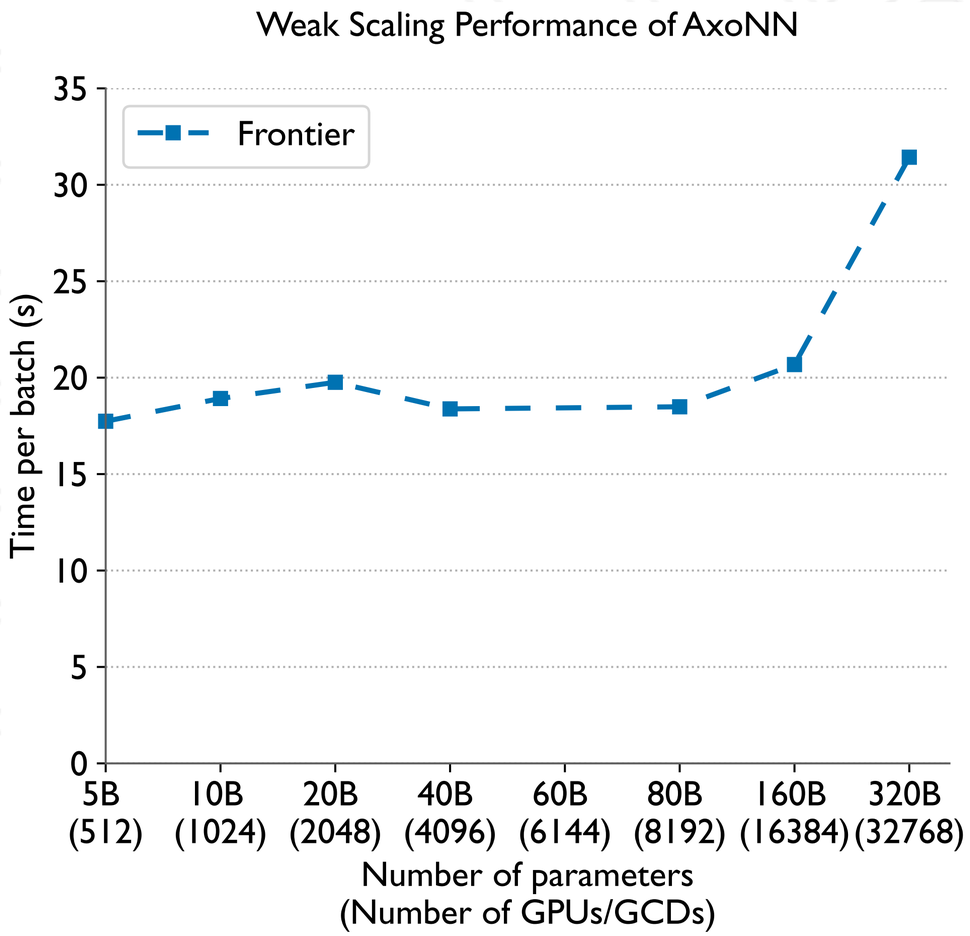
Easy parallelization using AxoNN

- Requires minimal code changes to model architecture (code):

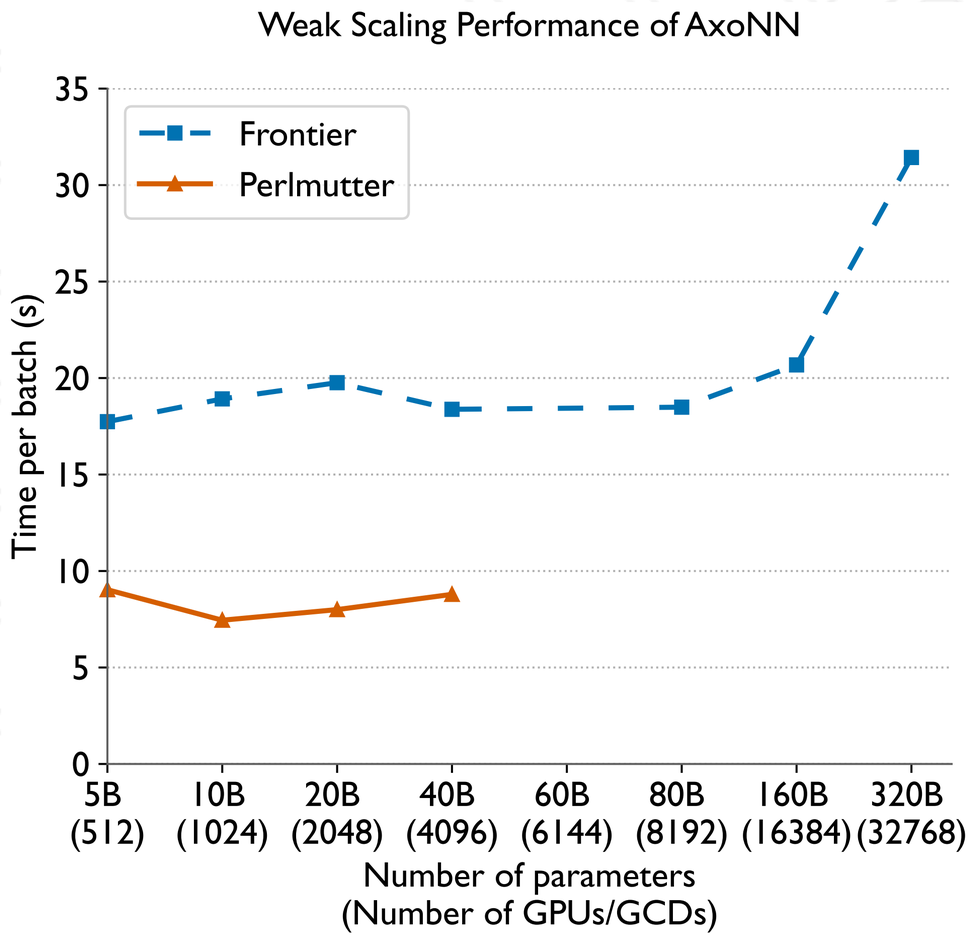
```
from axonn.intra_layer import auto_parallelize

with auto_parallelize():
    net = # declare your sequential model here
```

- AxoNN intercepts all declarations of `torch.nn.Linear`, and parallelizes them
- Our ML collaborators used this mode for the memorization experiments
- We also have backends for `lightning` and `accelerate`

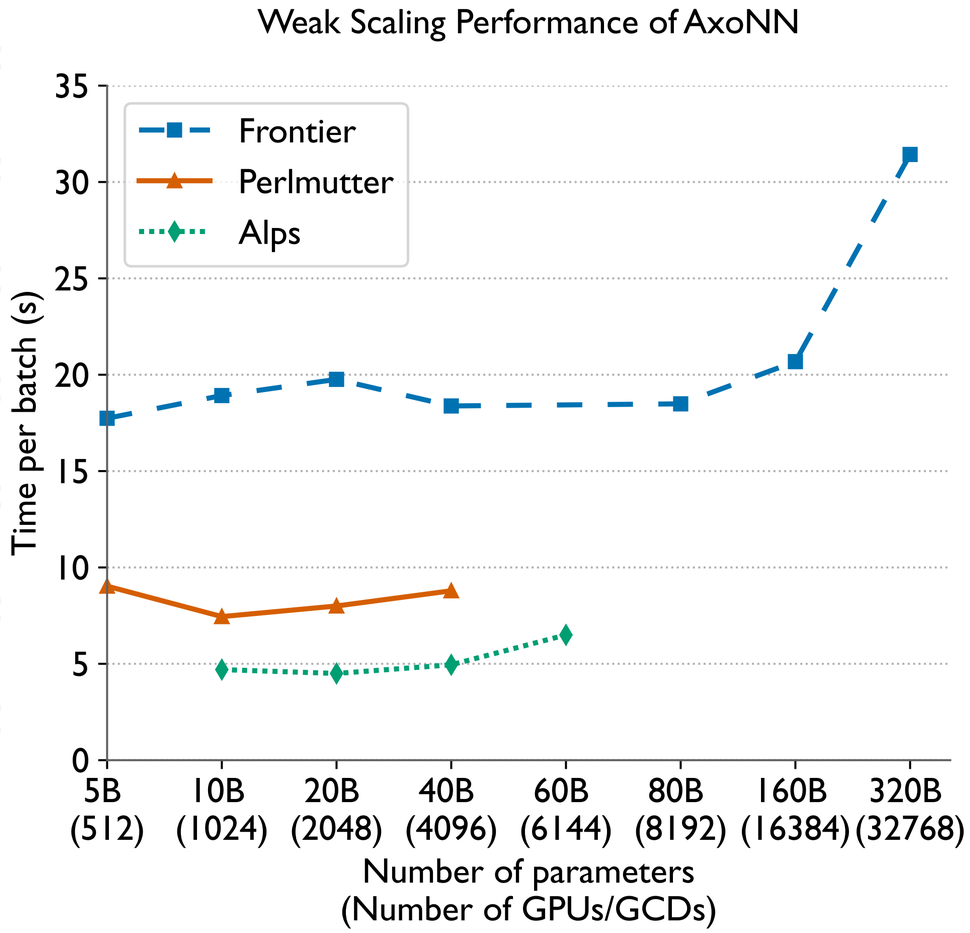
Weak scaling performance



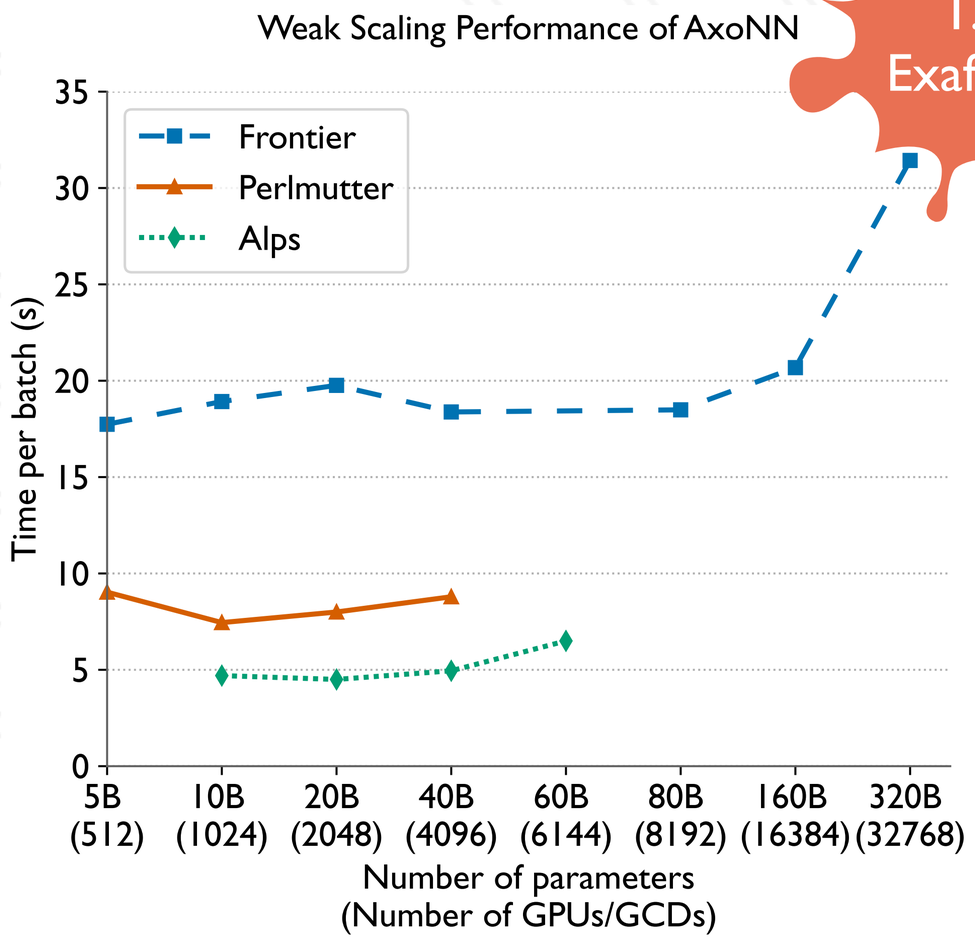
Weak scaling performance



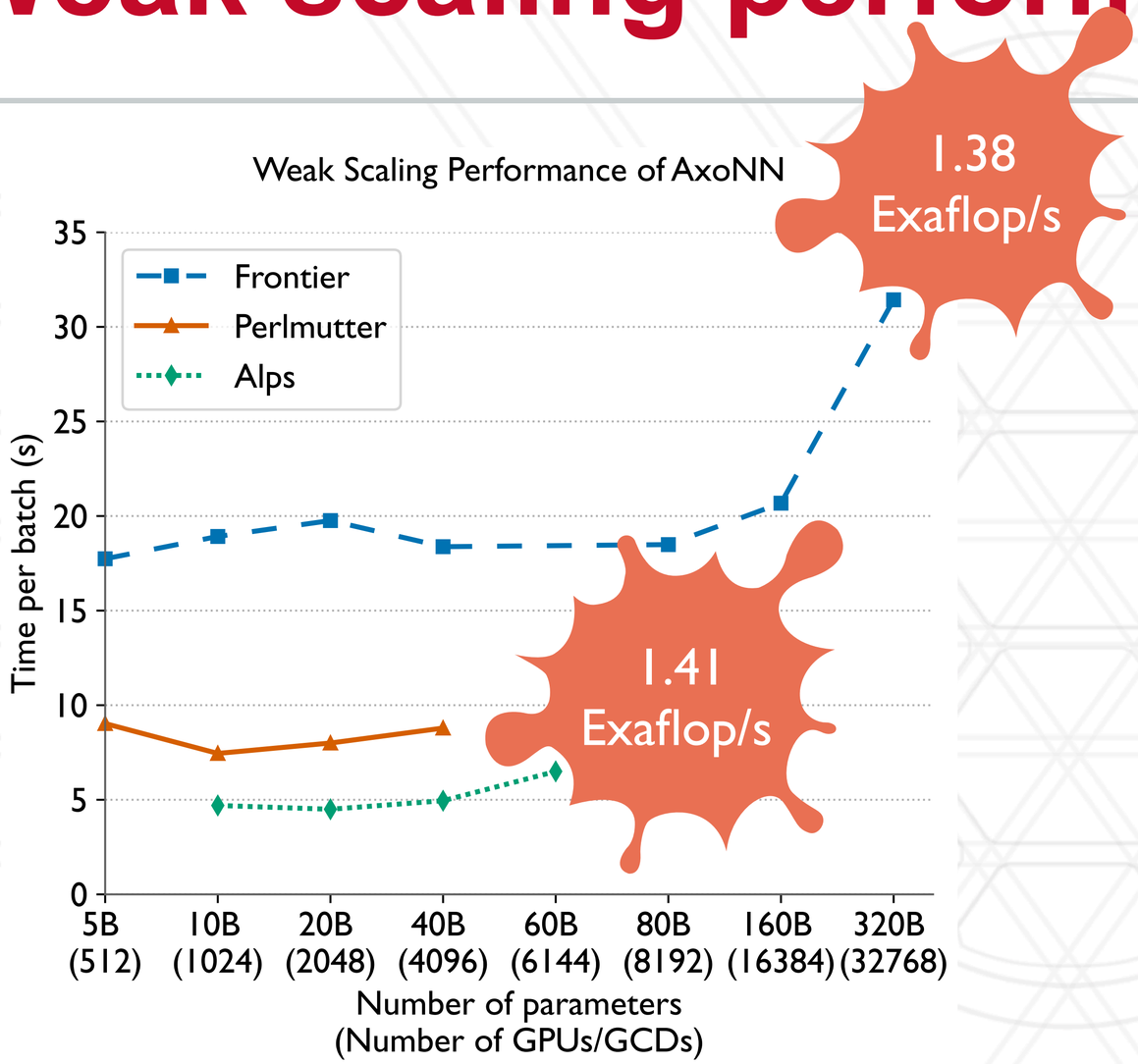
Weak scaling performance



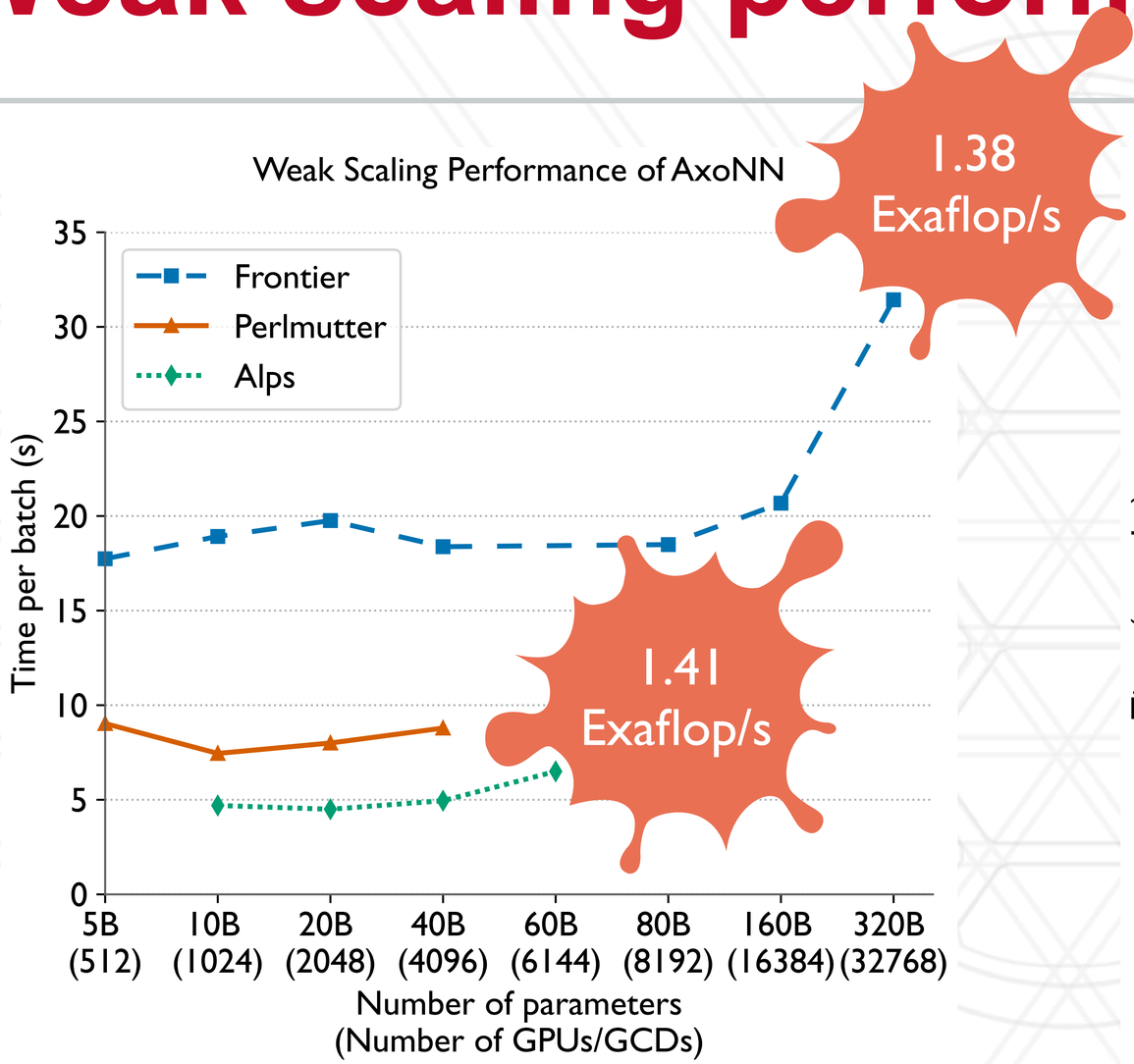
Weak scaling performance

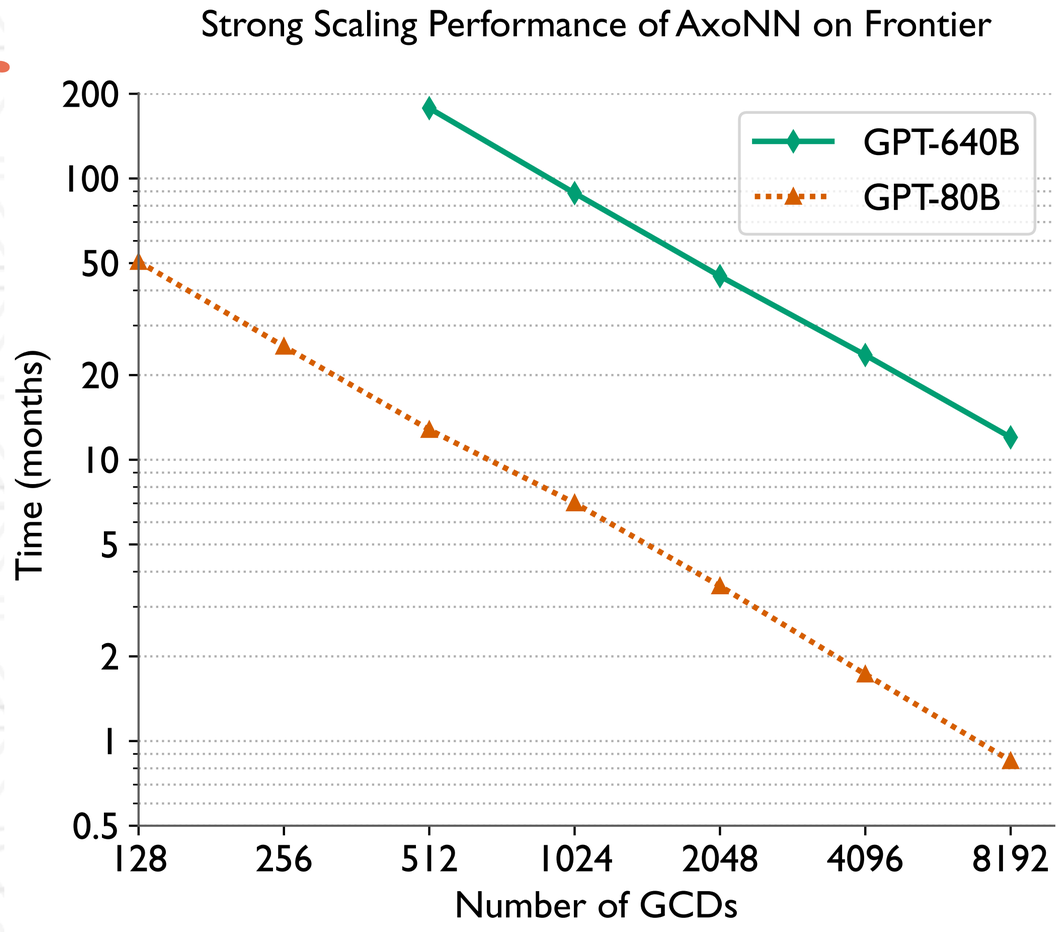


Weak scaling performance



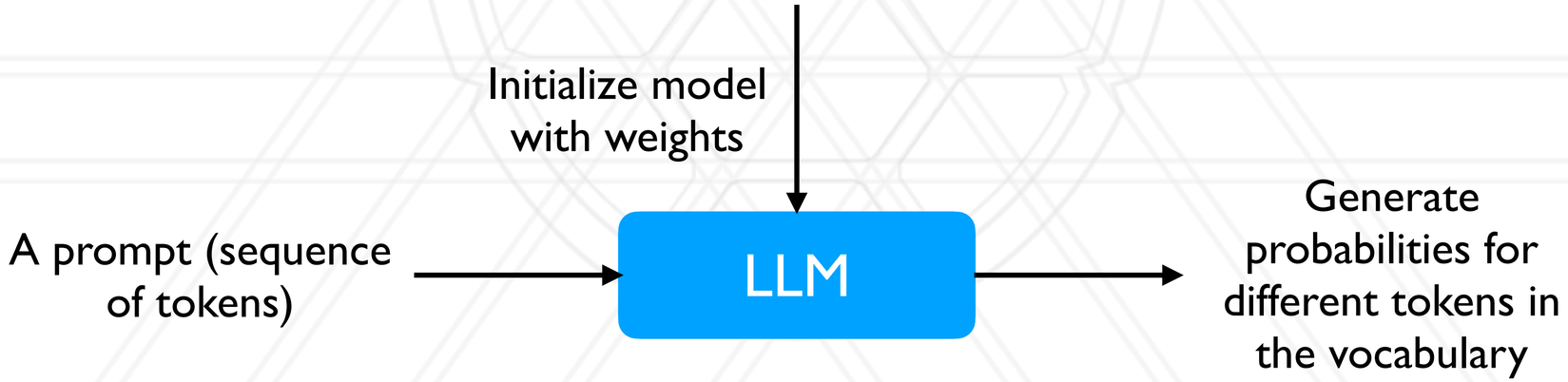
Weak scaling performance





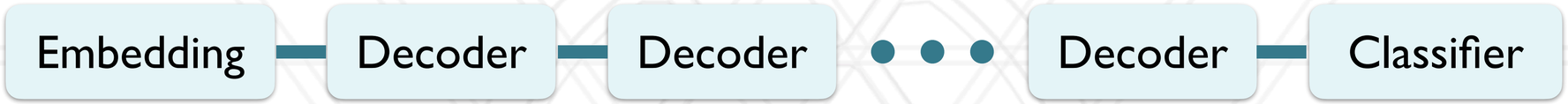
How is inference done?

- Start with initializing a model on a GPU with weights from a pre-trained model
- Input: a user prompt
- Output: a generation of output tokens

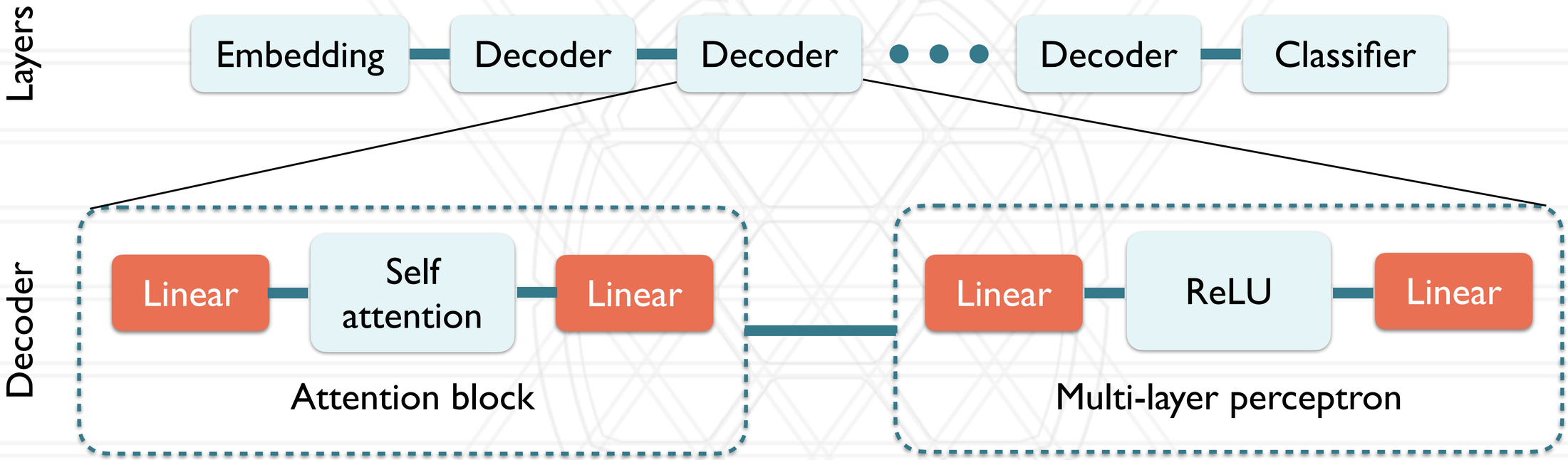


Compute work in transformer models

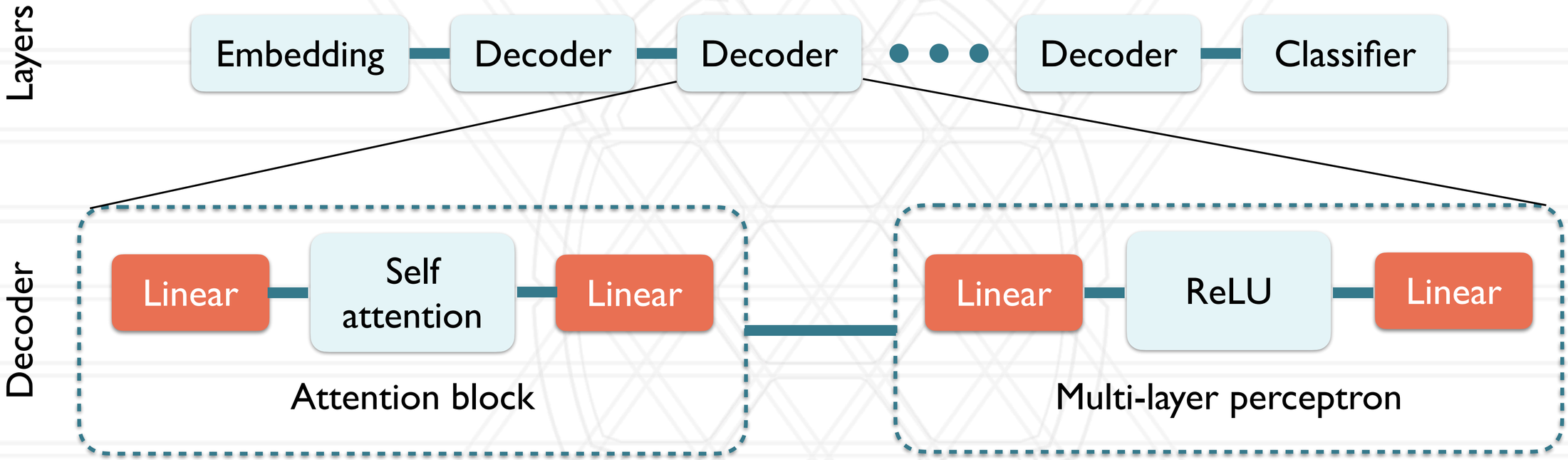
Layers

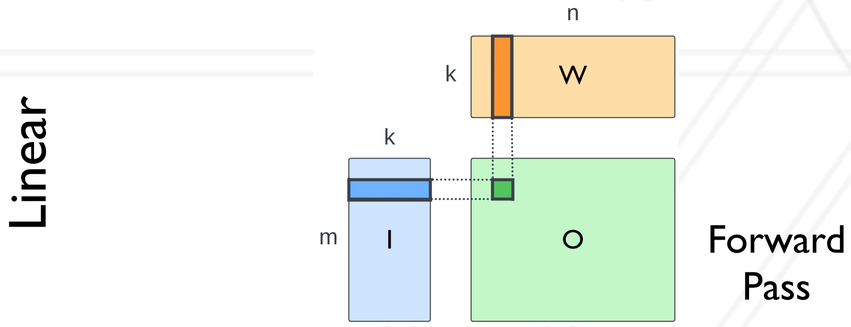


Compute work in transformer models



Compute work in transformer models





Graph Neural Networks

Thomas Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017

Graph Neural Networks

- Type of neural network to learn from graph datasets
- Graph Convolutional Networks (GCNs) have become widely popular in recent years for studying properties of graphs

Thomas Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017

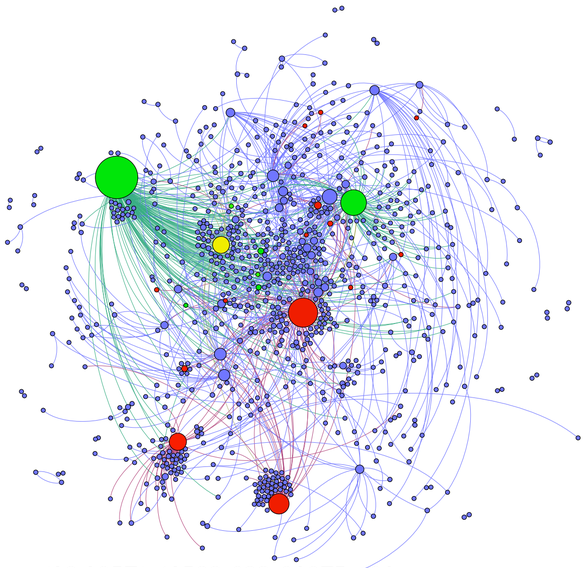
Graph Neural Networks

- Type of neural network to learn from graph datasets
- Graph Convolutional Networks (GCNs) have become widely popular in recent years for studying properties of graphs
- Systems challenges with GNNs:
  - Graphs are irregular as opposed to images or text
  - Input datasets representing extremely large graphs do not fit within memory

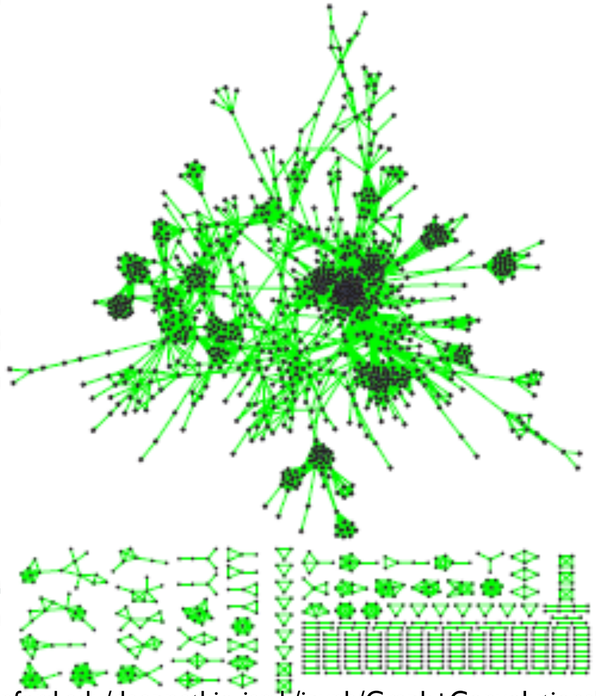
Thomas Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017

Real-world graphs

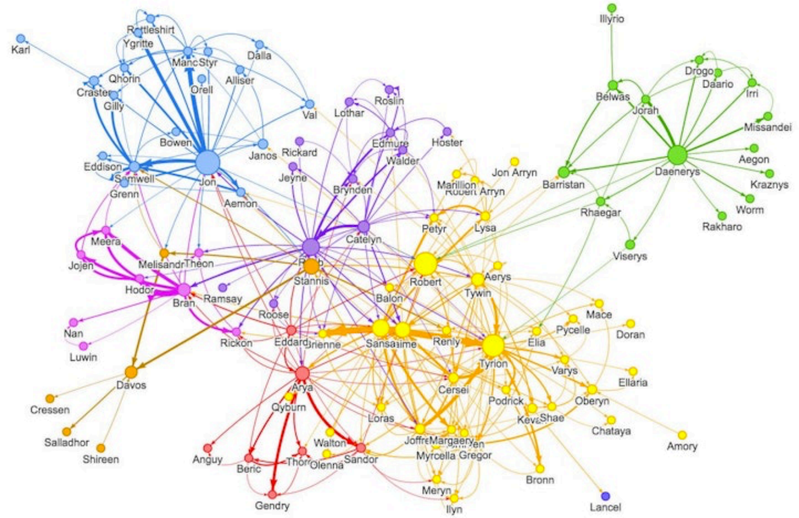
- Graphs are everywhere: financial transaction networks, protein-protein interactions, social networks
- Learn complex properties of and relationships within graphs
- Use cases: fraud detection, bioinformatics, recommendation systems



https://doi.org/10.1057/s41599-022-01075-x01



http://snap.stanford.edu/deepnetbio-ismb/ipynb/Graph+Convolutional+Prediction+of+Protein+Interactions+in+Yeast.html



https://www.cylynx.io/blog/a-comparison-of-javascript-graph-network-visualisation-libraries/

Three steps in a GCN

Aggregation

https://ieeexplore.ieee.org/document/10246394

Three steps in a GCN

Aggregation

$$H = \text{SPMM}(A, F)$$

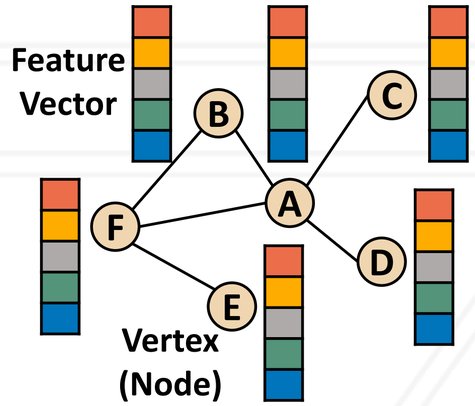
https://ieeexplore.ieee.org/document/10246394

Three steps in a GCN

Aggregation

$$H = \text{SPMM}(A, F)$$

Adjacency Matrix



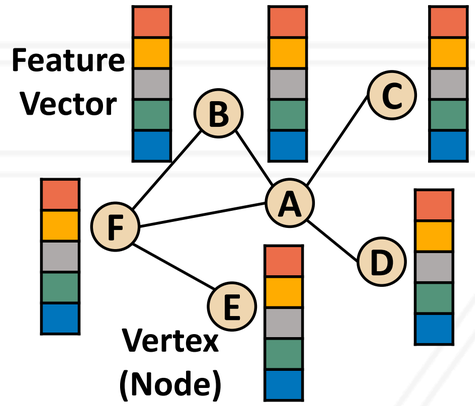
https://ieeexplore.ieee.org/document/10246394

Three steps in a GCN

Aggregation

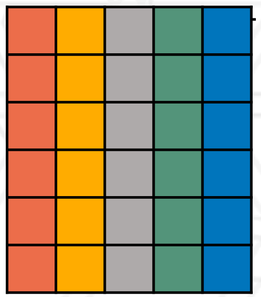
$$H = \text{SPMM}(A, F)$$

Adjacency Matrix



Feature Matrix

N X D

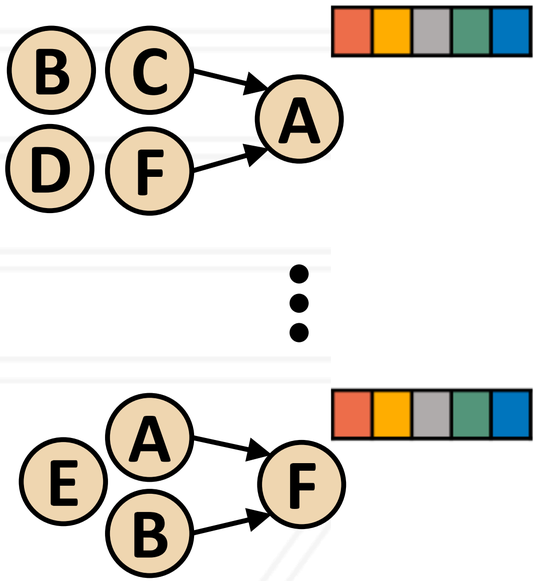


https://ieeexplore.ieee.org/document/10246394

Three steps in a GCN

Aggregation

$$H = \text{SPMM}(A, F)$$



https://ieeexplore.ieee.org/document/10246394

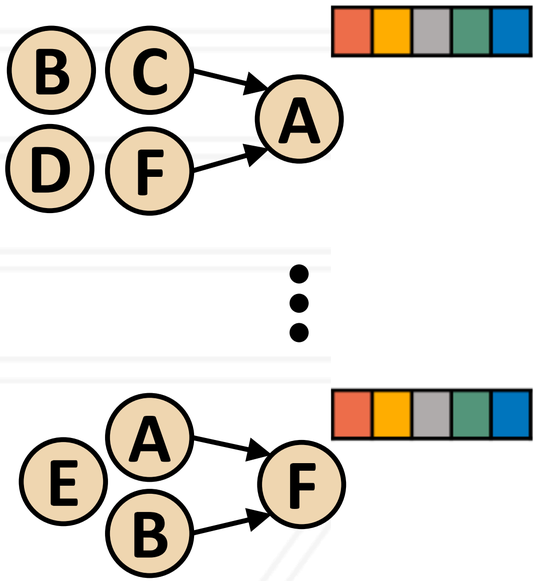
Three steps in a GCN

Aggregation

Combination

$$H = \text{SPMM}(A, F)$$

$$Q = \text{SGEMM}(H, W)$$



https://ieeexplore.ieee.org/document/10246394

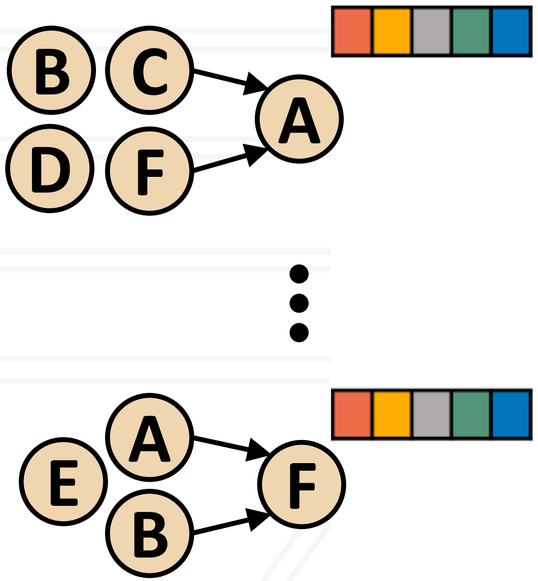
Three steps in a GCN

Aggregation

Combination

$H = \text{SPMM}(A, F)$

$Q = \text{SGEMM}(H, W)$



Weight Matrix

D X D

https://ieeexplore.ieee.org/document/10246394

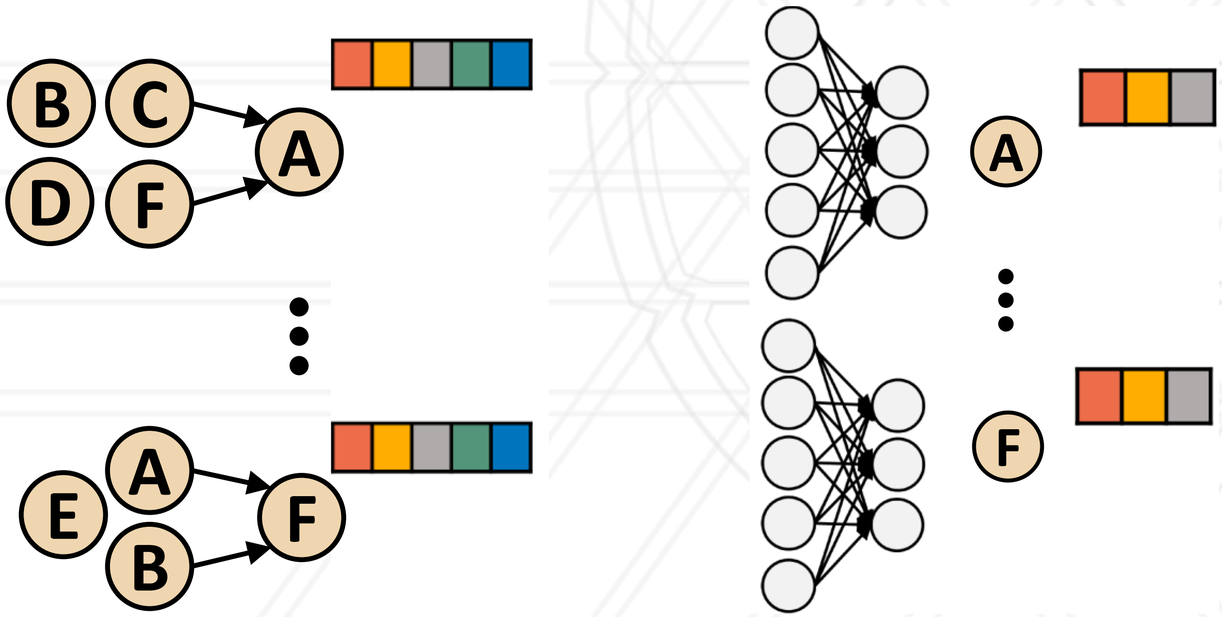
Three steps in a GCN

Aggregation

Combination

$$H = \text{SPMM}(A, F)$$

$$Q = \text{SGEMM}(H, W)$$

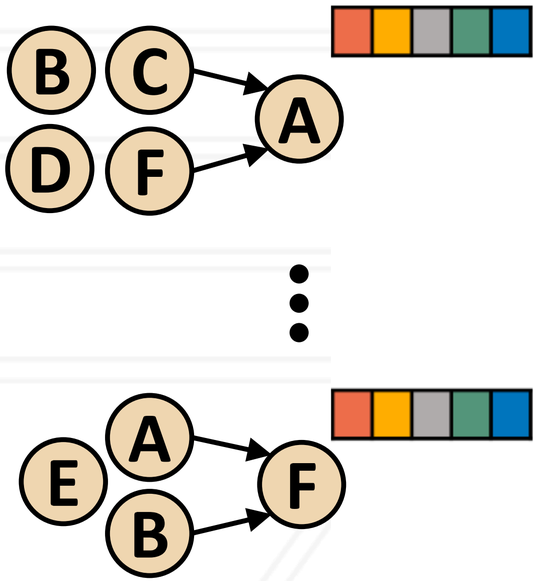


https://ieeexplore.ieee.org/document/10246394

Three steps in a GCN

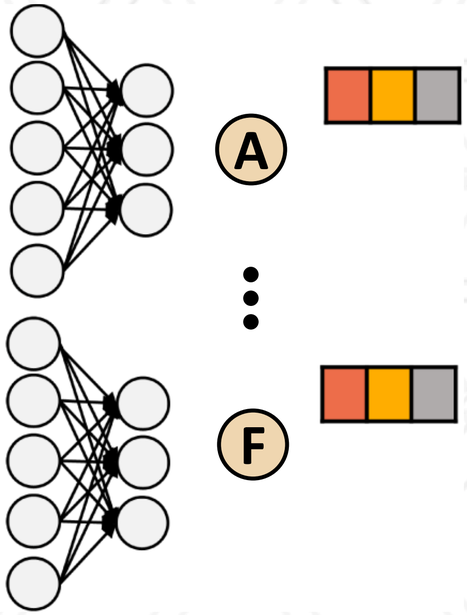
Aggregation

$$H = \text{SPMM}(A, F)$$



Combination

$$Q = \text{SGEMM}(H, W)$$



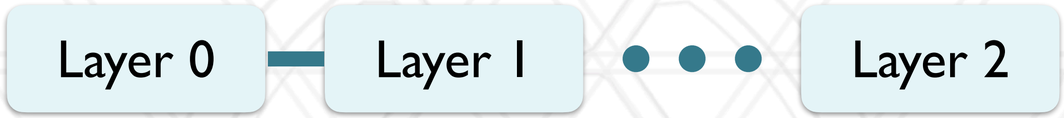
Activation

$$F' = \sigma(Q)$$

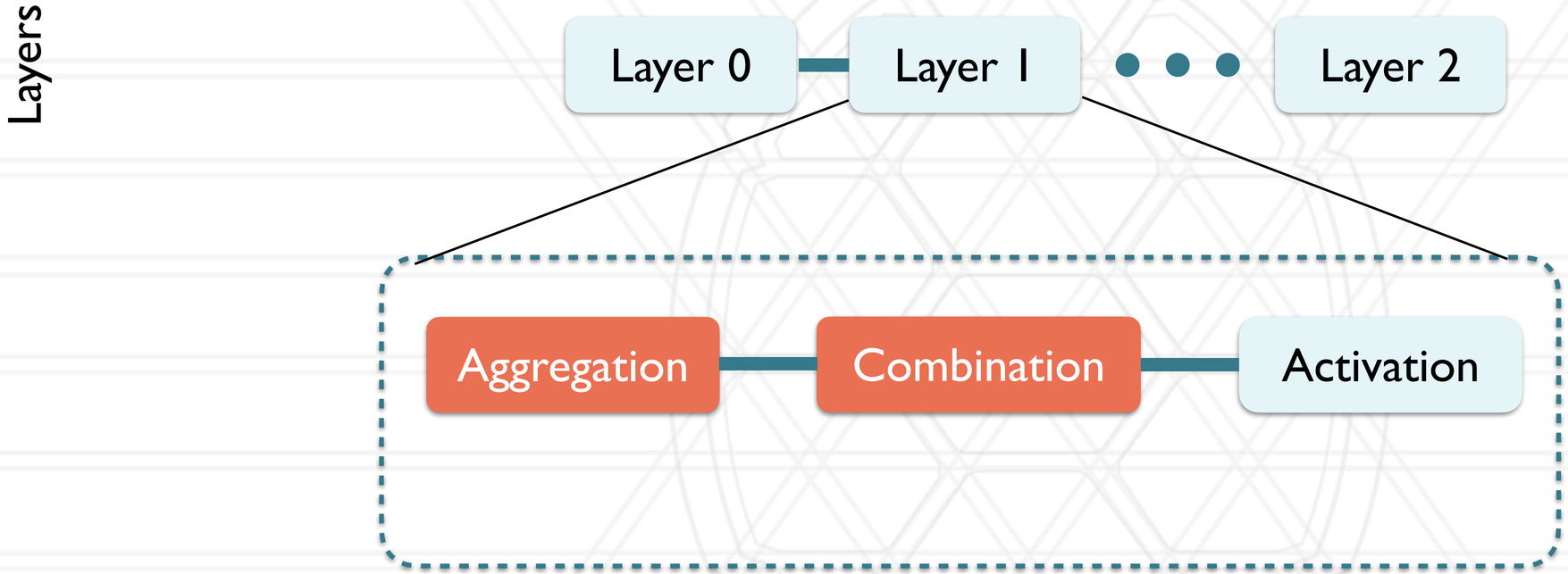
https://ieeexplore.ieee.org/document/10246394

Why is GNN training well-suited for HPC?

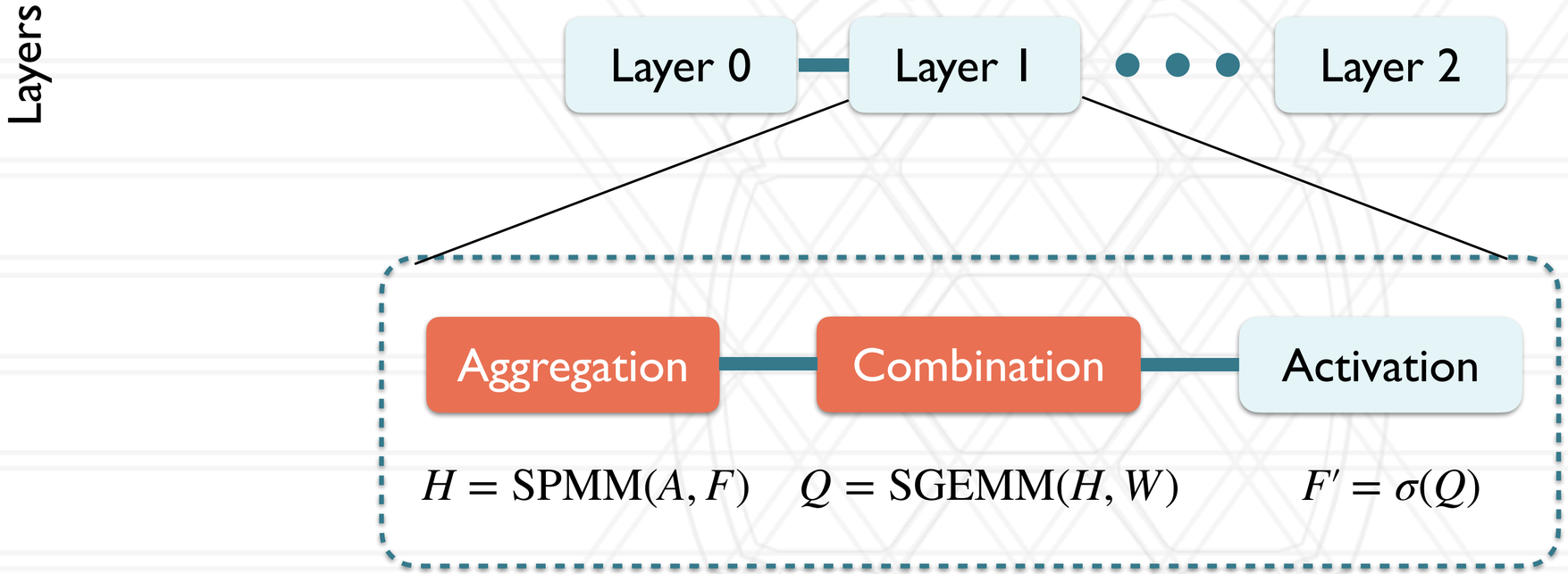
Layers



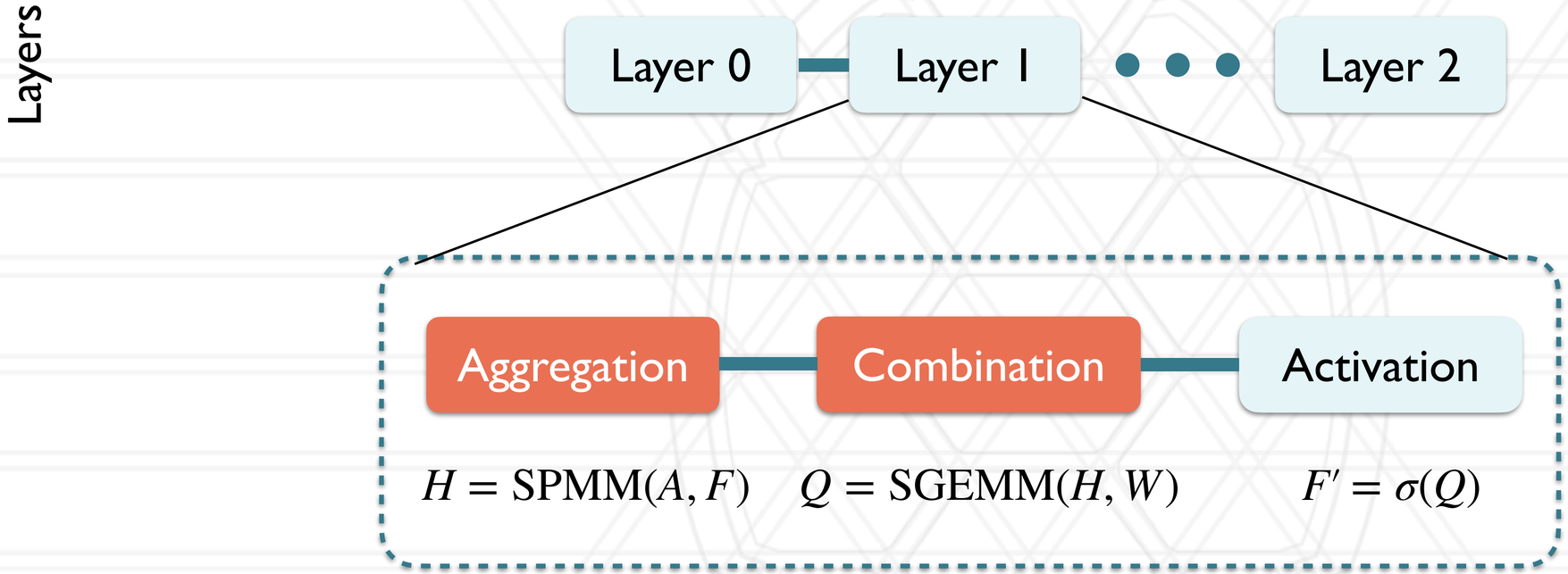
Why is GNN training well-suited for HPC?

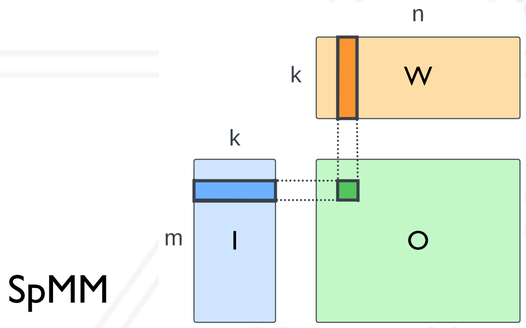


Why is GNN training well-suited for HPC?

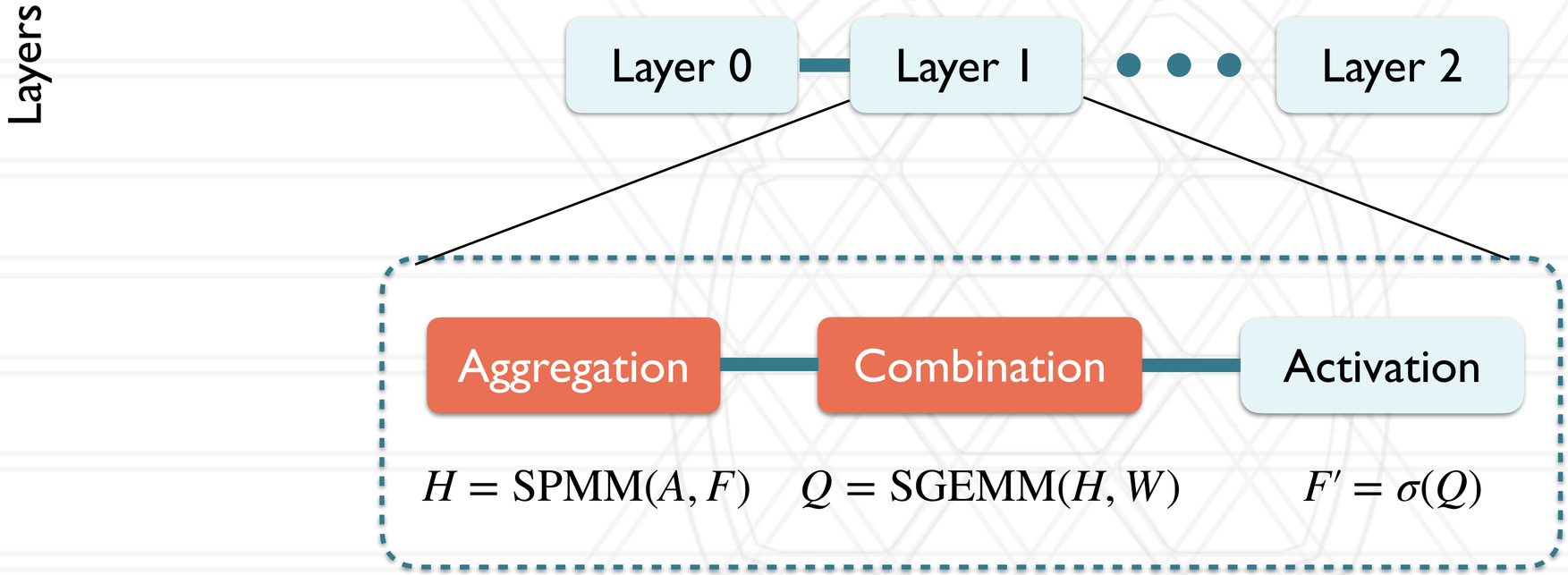


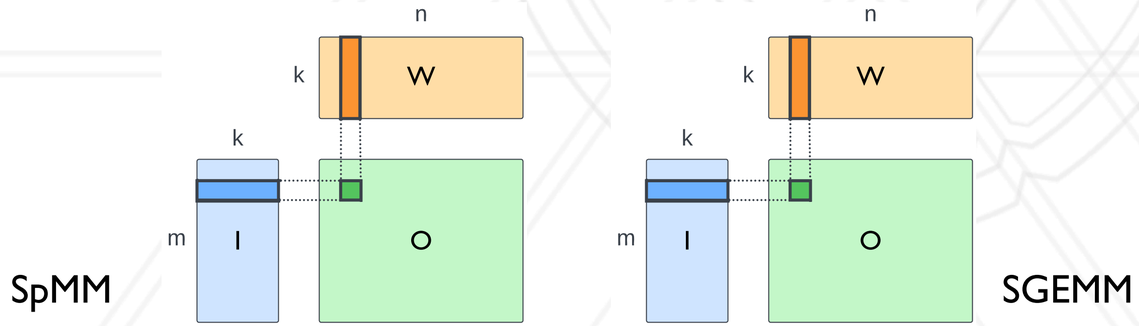
Why is GNN training well-suited for HPC?



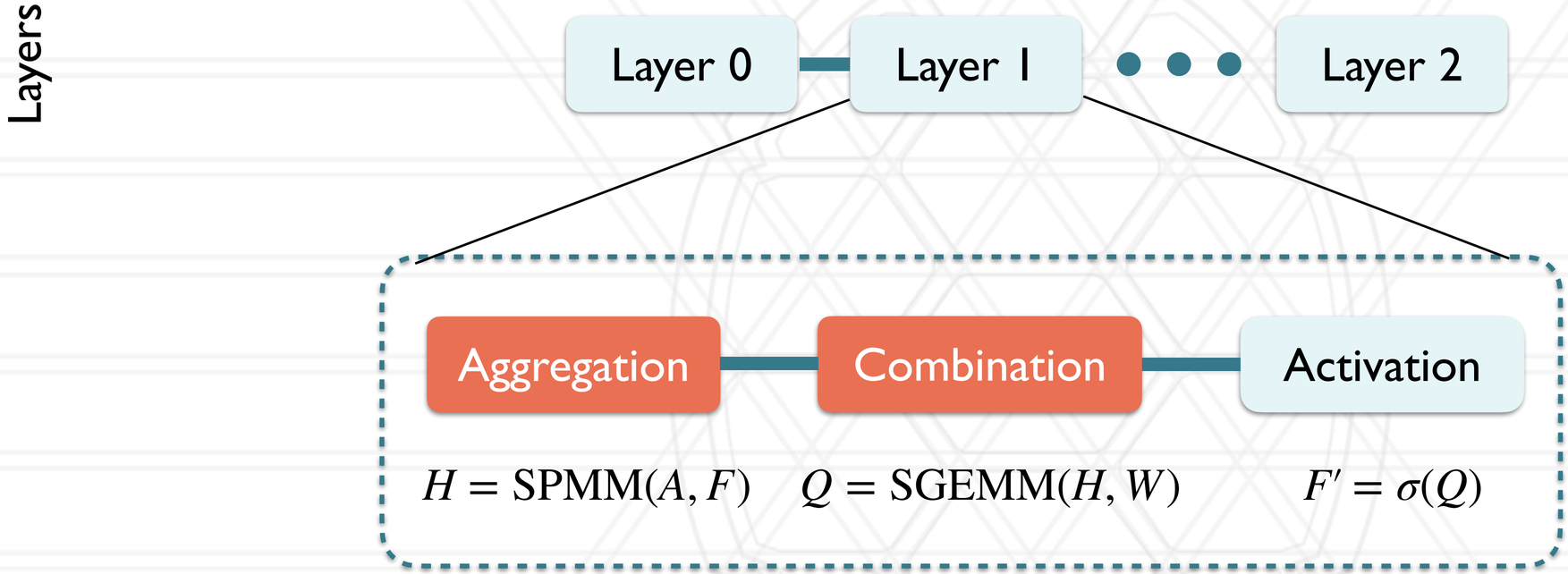


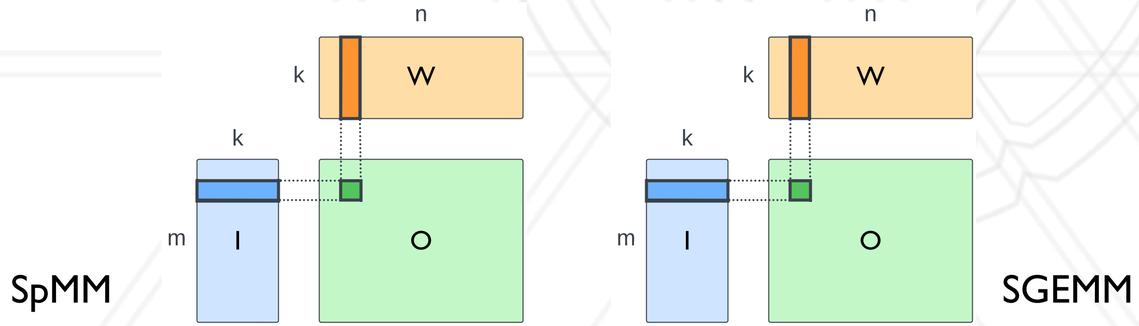
Why is GNN training well-suited for HPC?





Why is GNN training well-suited for HPC?





Backward pass: 2 SGEMMs and 1 SpMM

Challenges with Parallel GNN Training

- Very large graphs require effective parallelization over multiple GPUs
- Irregularity in graph structure leads to highly imbalanced and highly sparse adjacency matrices
  - Leads to load imbalance when distributing work
- Significant communication for synchronizing activations and gradients: N X D

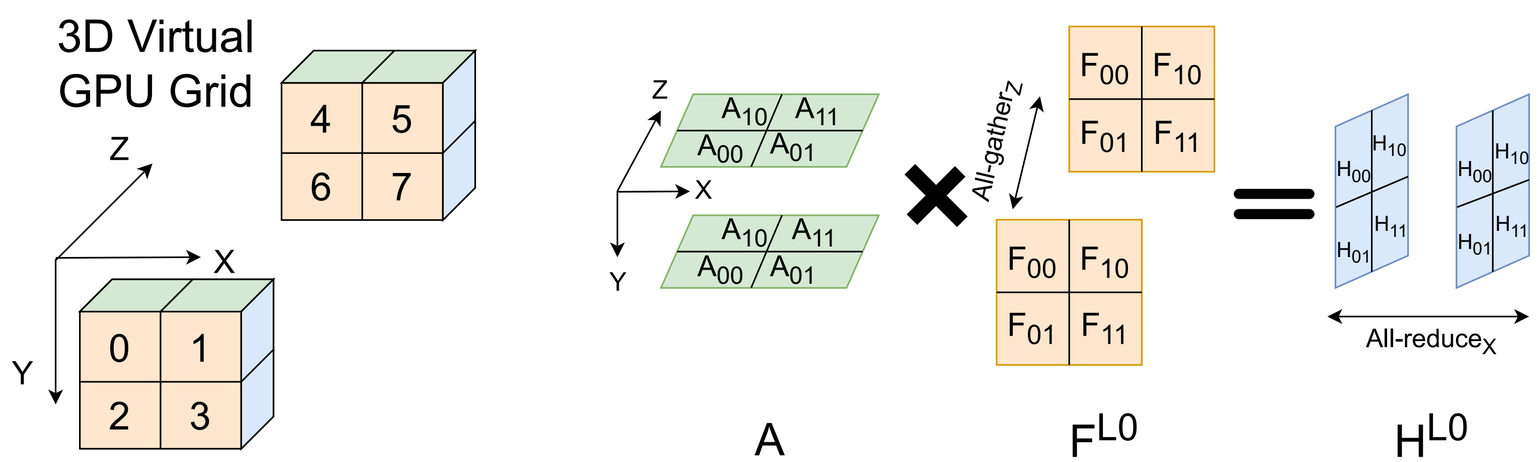
Plexus: 3D Tensor Parallel Approach

- Based on our work on AxoNN and 3D Tensor Parallelism in LLM training
- Divide the Adjacency, Feature and Weight matrices across GPUs

Plexus: 3D Tensor Parallel Approach

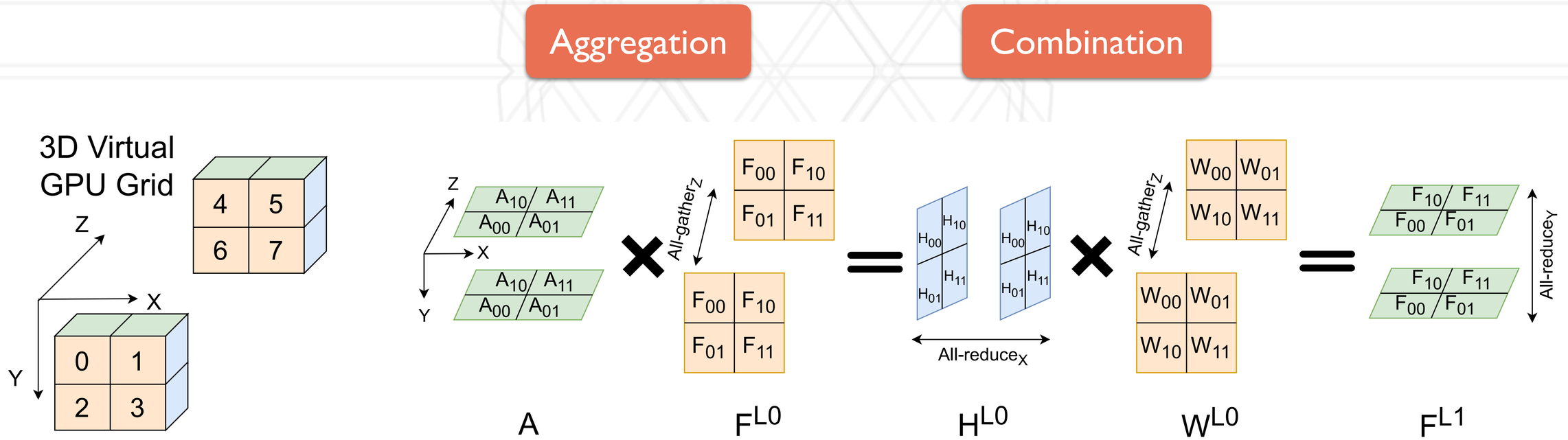
- Based on our work on AxoNN and 3D Tensor Parallelism in LLM training
- Divide the Adjacency, Feature and Weight matrices across GPUs

Aggregation

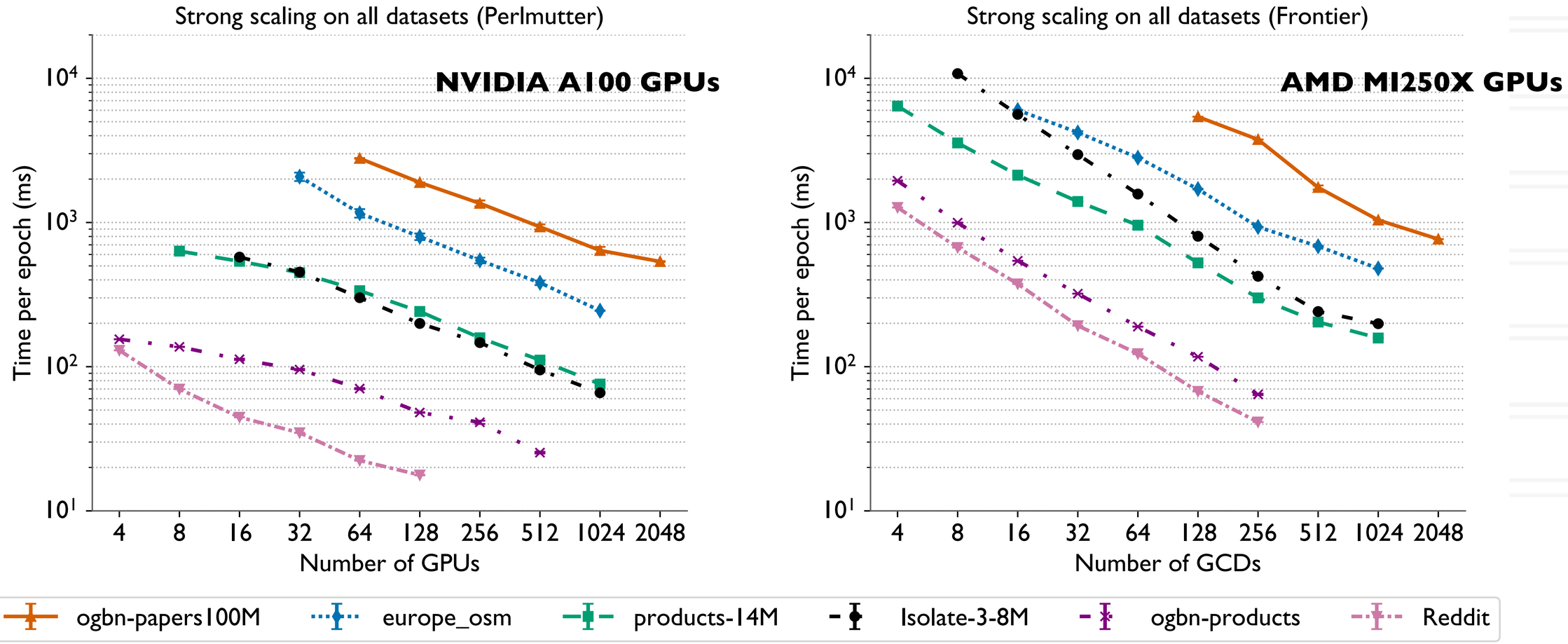


Plexus: 3D Tensor Parallel Approach

- Based on our work on AxoNN and 3D Tensor Parallelism in LLM training
- Divide the Adjacency, Feature and Weight matrices across GPUs



Strong scaling performance of Plexus



UNIVERSITY OF
MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu