CS 498: Machine Learning System Spring 2025

Minjia Zhang

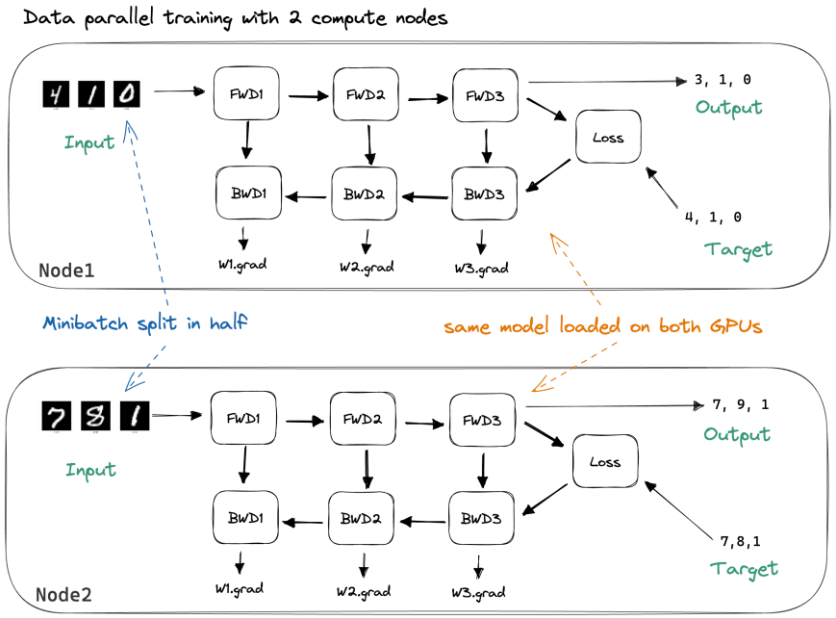
The Grainger College of Engineering

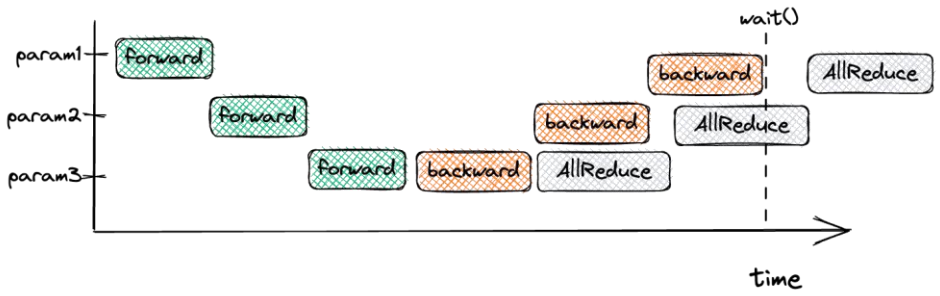
Today

Tensor Slicing Model Parallelism

Data Parallelism

- Partitions a training minibatch across multiple devices
- Linearly scalable
- Slicing activation only
- Does not help with excessive model size

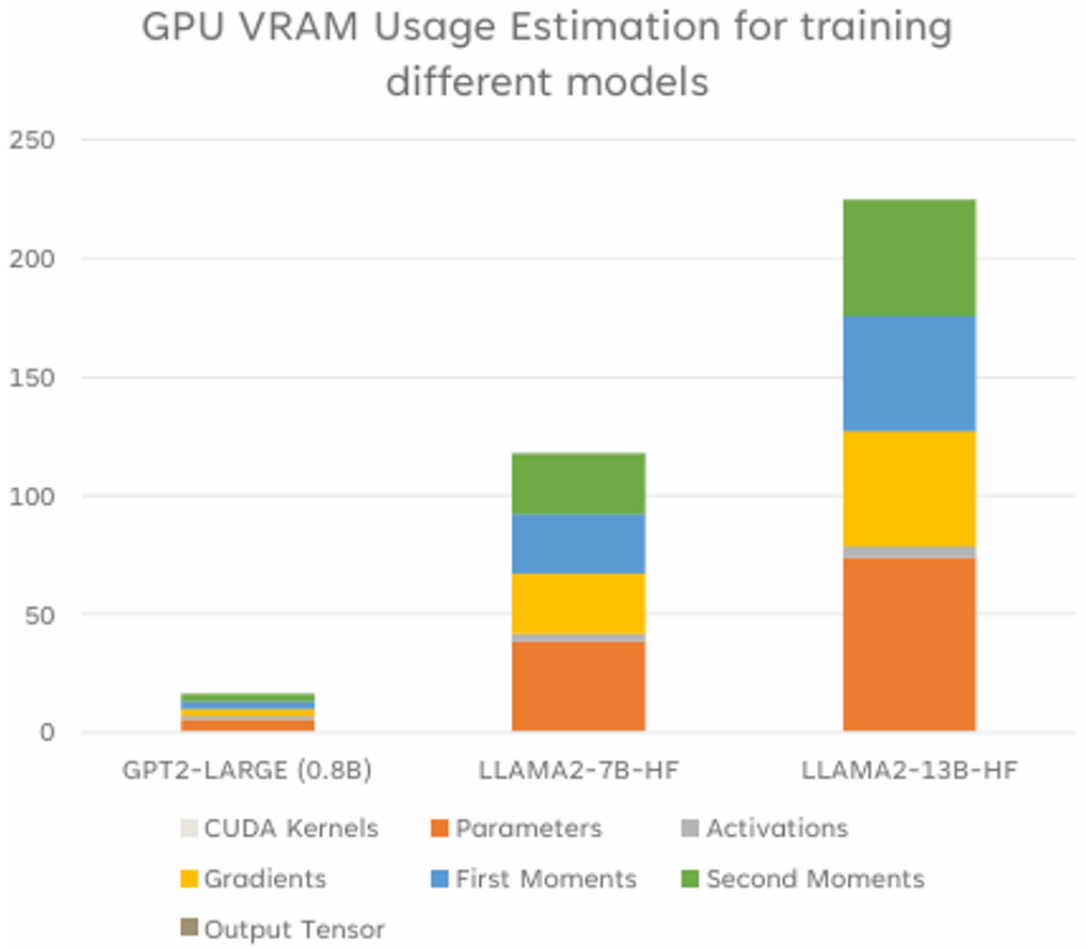




Large Models Require Large Memory

- Training >> Inference
- Adam >> SGD
- Larger minibatch
- More parameters
- …

Question: How do we speed up training or simply enable training?

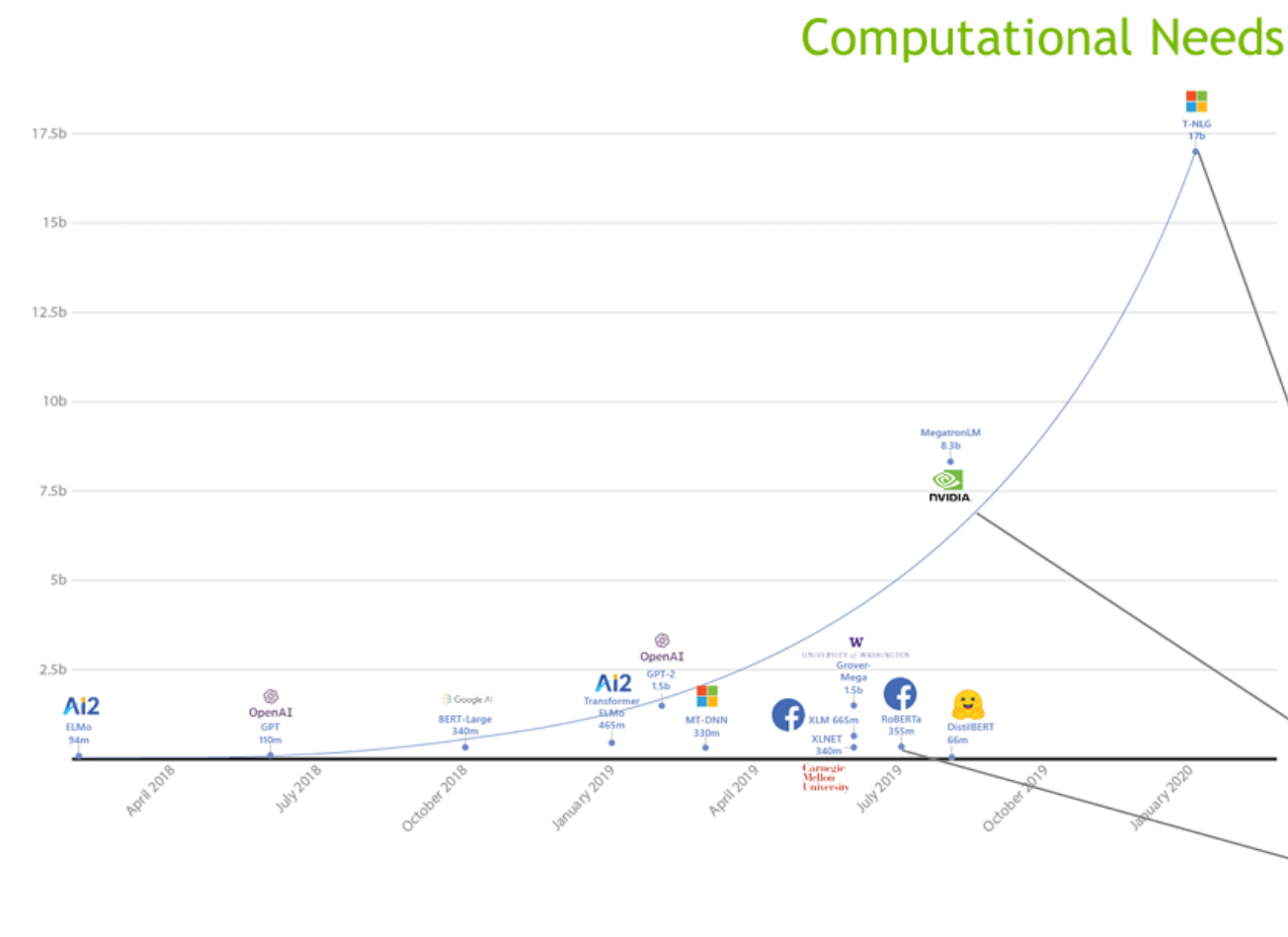


Megatron-LM

- Paper: https://arxiv.org/abs/1909.08053
- Repo: https://github.com/NVIDIA/Megatron-LM

- NVIDIA's framework, released in 2019, for efficiently training large-scale language models

Motivation



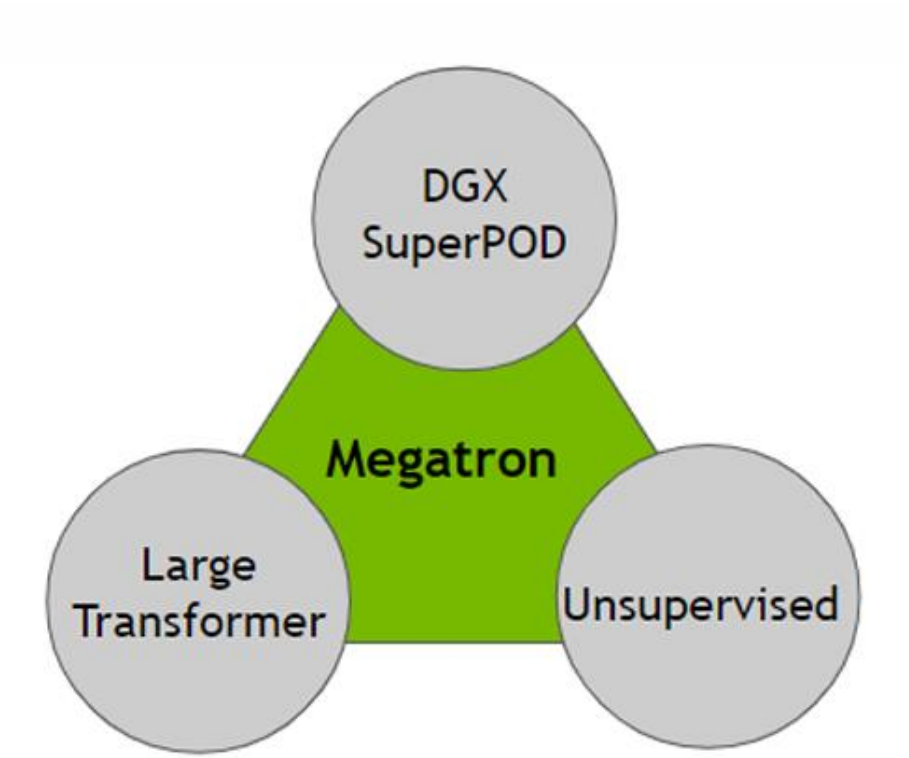
Model	# of Parameters	# of Iterations	# of GPUs	Training Time
T-NLG (MS + NV)	17.2 B	300 K	400	60 days
Megatron-GPT2 (NV)	8.3 B	300 K	512	10 days
Megatron-BERT (NV)	3.9 B	2 M	512	25 days
RoBERTa (FB)	345M	4 M	1024	1 day

Motivation: Why Megatron?

Training larger transformer-based language models became an important way to advance SoTA NLP applications.

Unsupervised language models such as GPT-2, BERT, and XLNet demonstrate the power of scaling language models trained on a huge corpus.

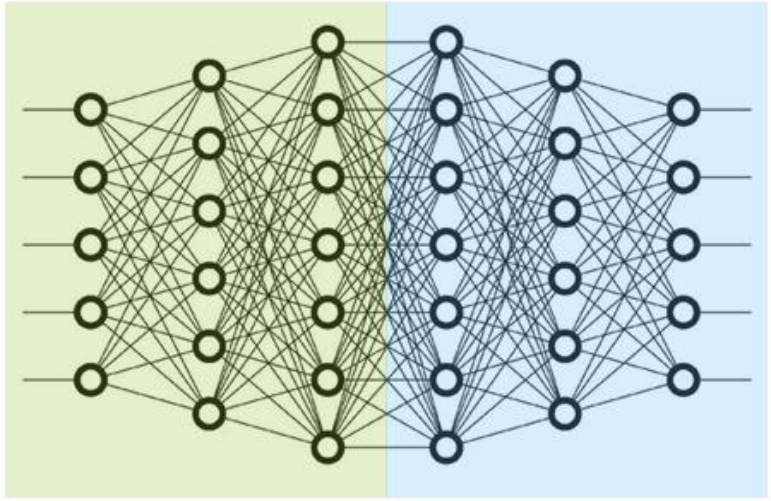
Nvidia DGX boxes optimized for deep learning provides a unique opportunity for training very large models.

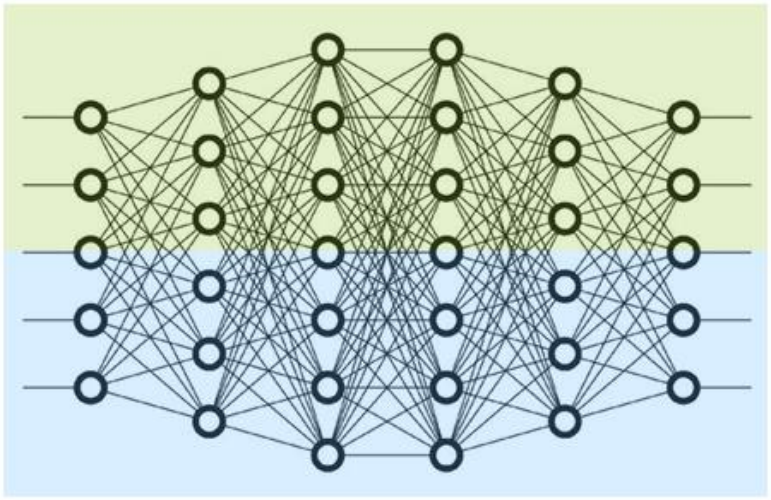


Model Parallelism

Inter-layer (Pipeline) parallelism
- Split sets of layers across multiple devices
- Layer 0, 1, 2 and layer 3, 4, 5 are on different devices

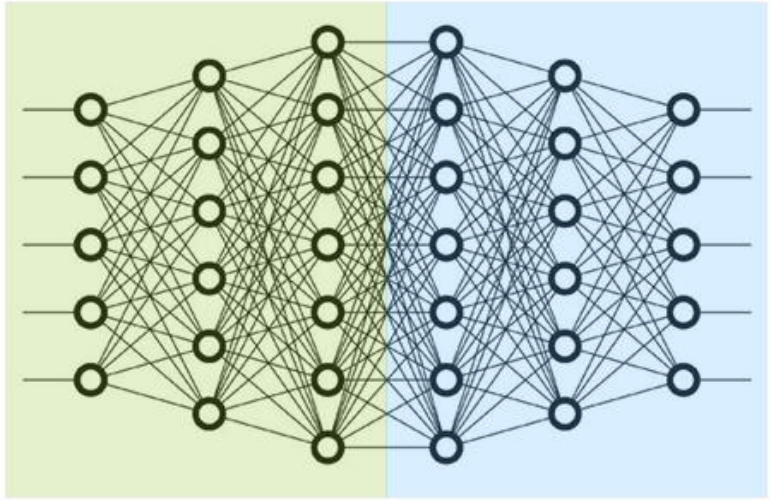
Intra-layer (Tensor) parallelism
- Split individual layers across multiple devices
- Both devices compute different parts of layer 0, 1, 2, 3, 4, 5



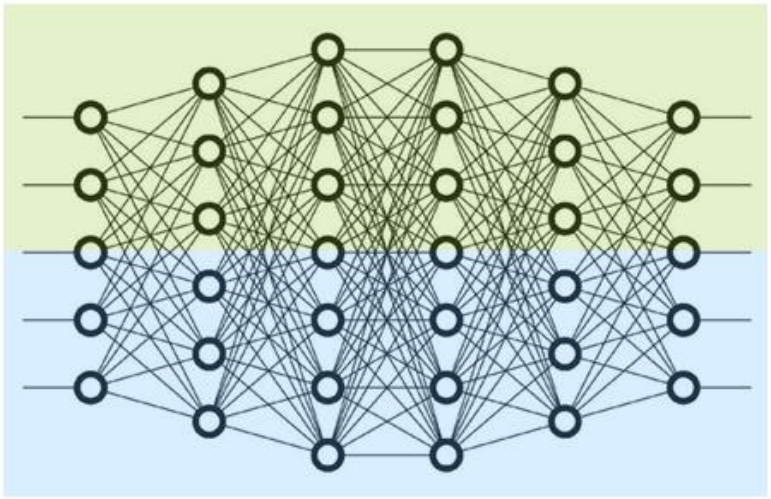


Model Parallelism

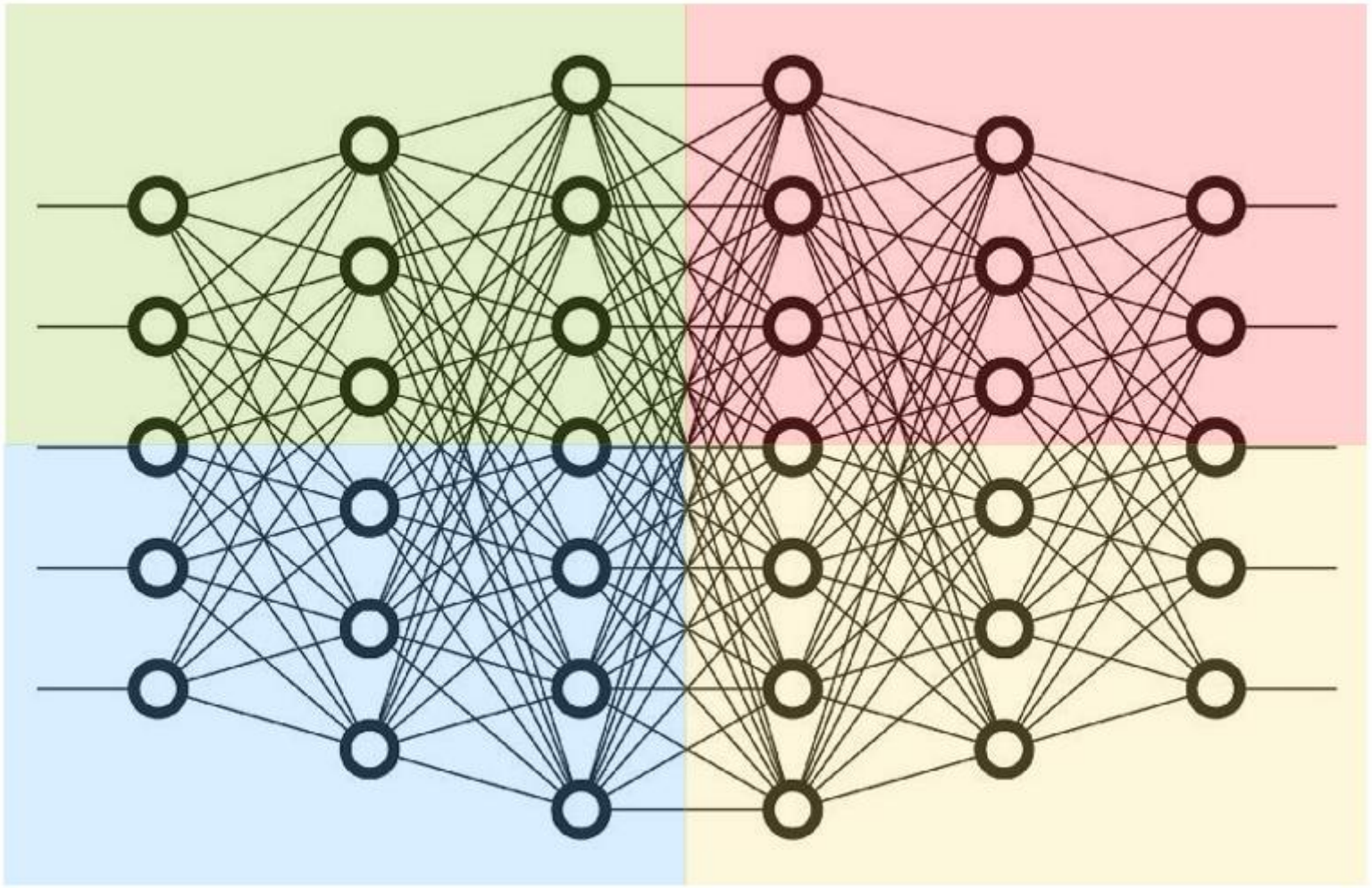
Inter-layer (Pipeline) parallelism
- Split sets of layers across multiple devices
- Layer 0, 1, 2 and layer 3, 4, 5 are on different devices



Intra-layer (Tensor) parallelism
- Split individual layers across multiple devices
- Both devices compute different parts of layer 0, 1, 2, 3, 4, 5

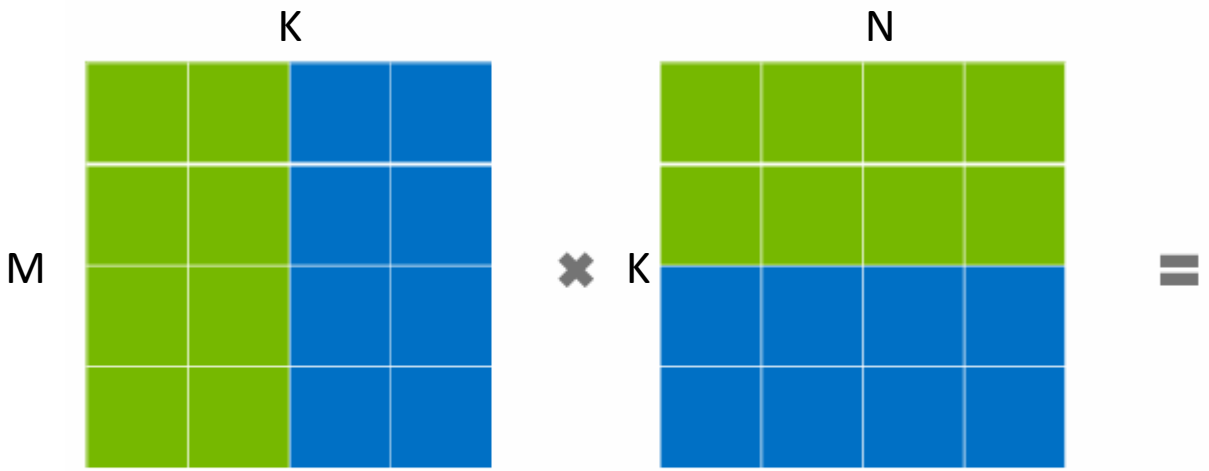


Complementary Types of Model Parallelism



Inter + Intra Parallelism

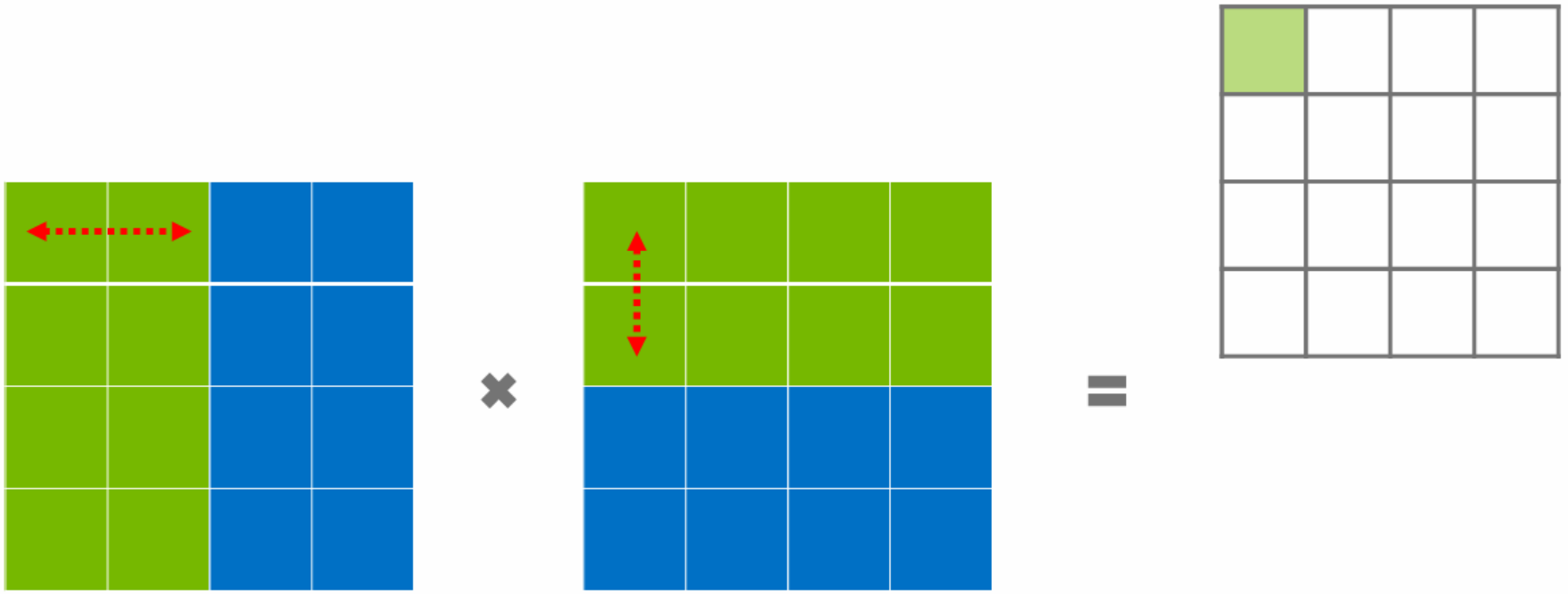
Tensor Parallel: Parallel GeMM



$$X = [X_1, X_2] \qquad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$Y = Y_1 + Y_2$$

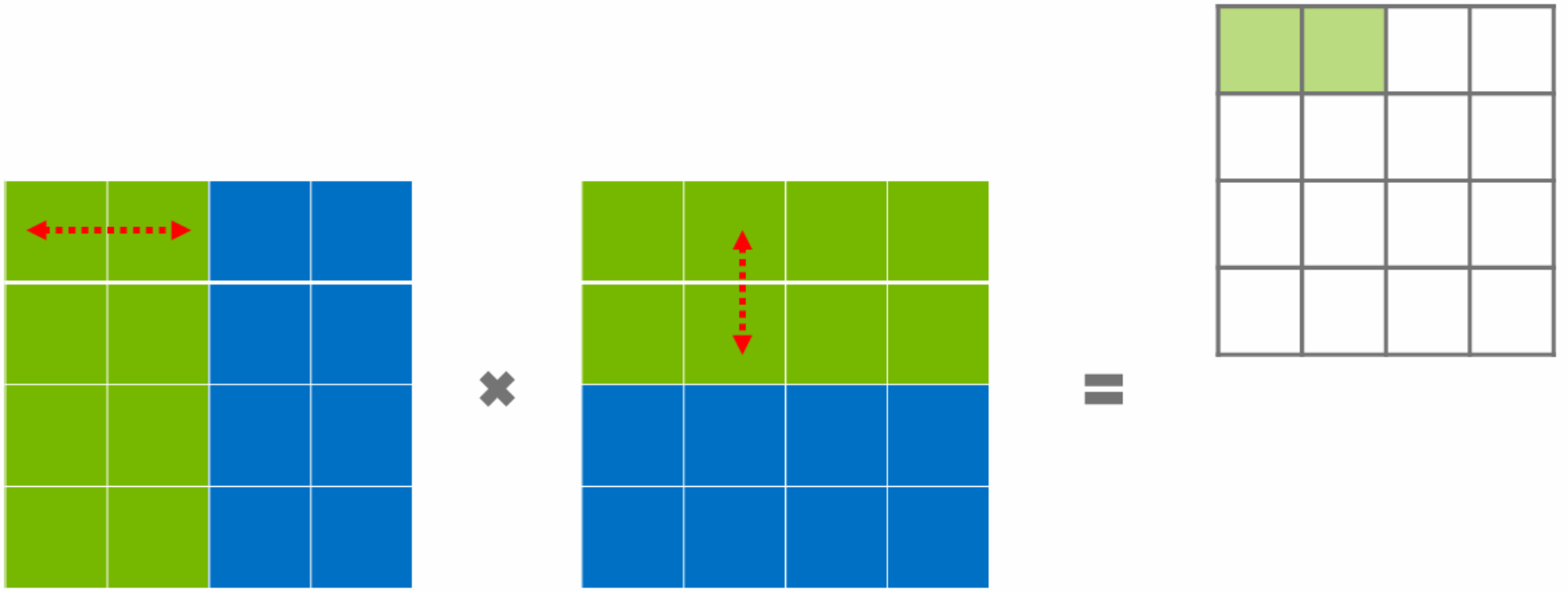
Tensor Parallel: Parallel GeMM



$$X = [X_1, X_2] \qquad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$Y = Y_1 + Y_2$$

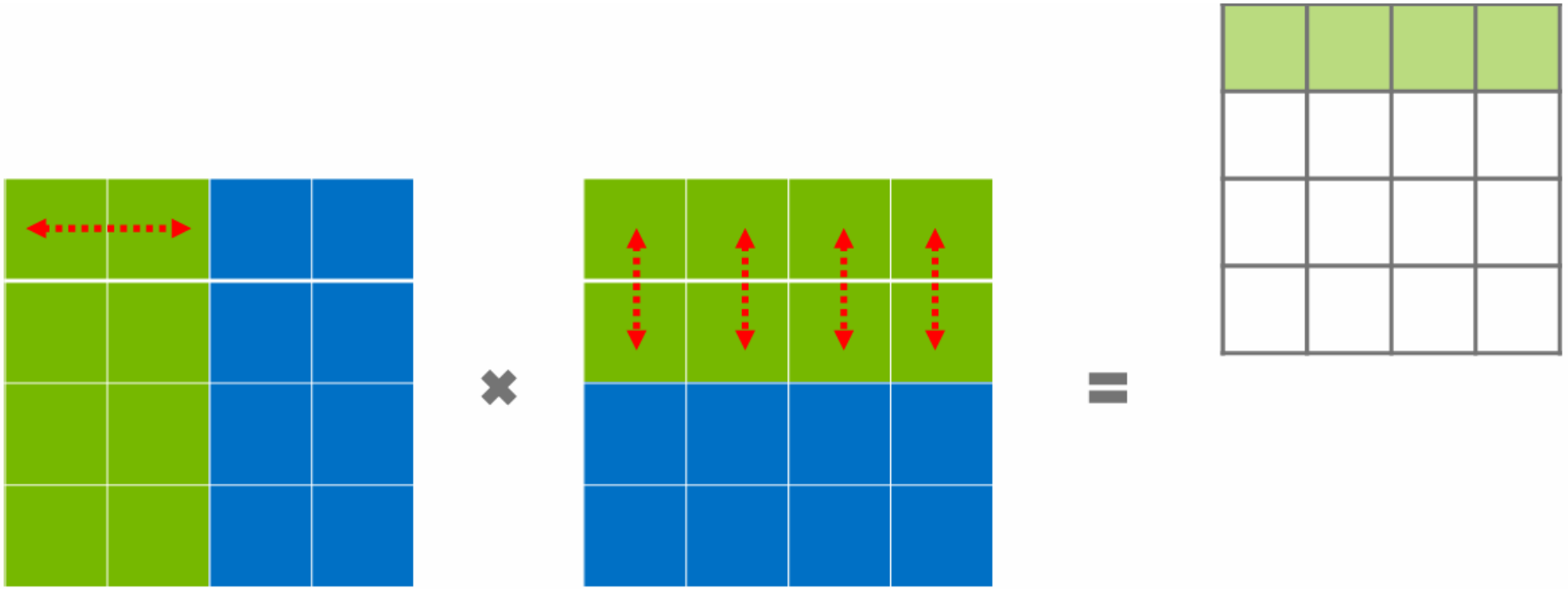
Tensor Parallel: Parallel GeMM



$$X = [X_1, X_2] \qquad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$Y = Y_1 + Y_2$$

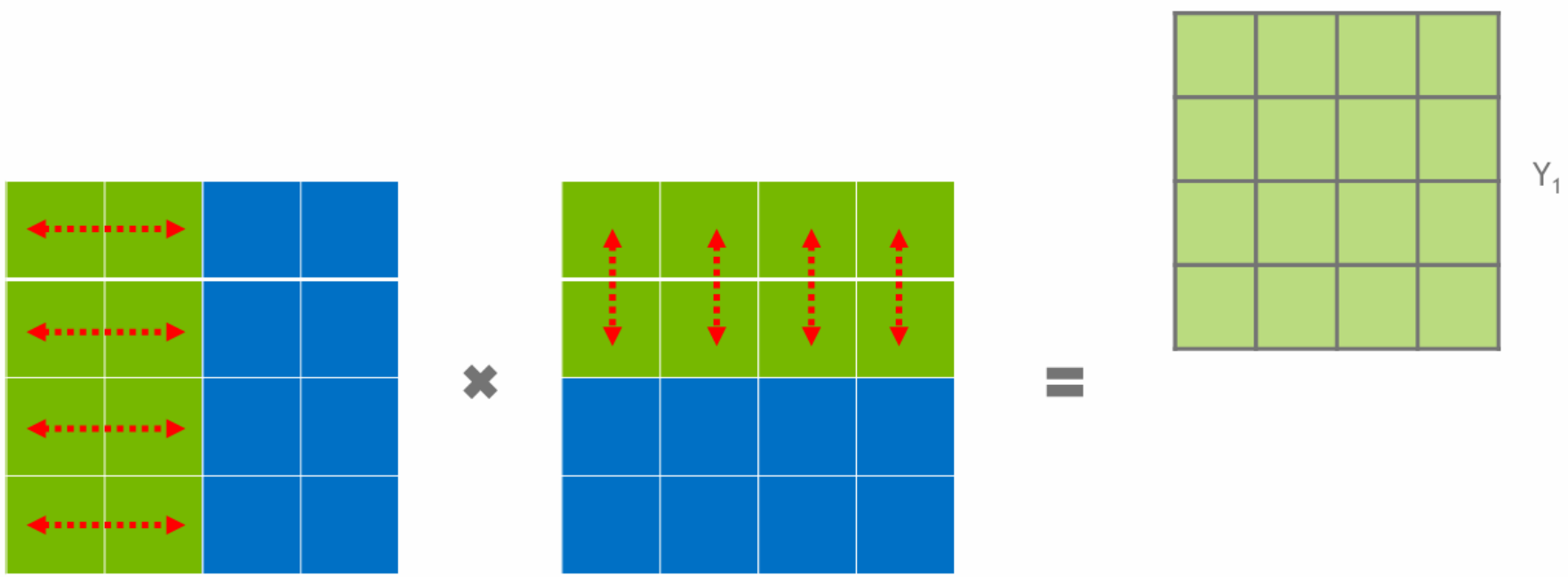
Tensor Parallel: Parallel GeMM



$$X = [X_1, X_2] \qquad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$Y = Y_1 + Y_2$$

Tensor Parallel: Parallel GeMM

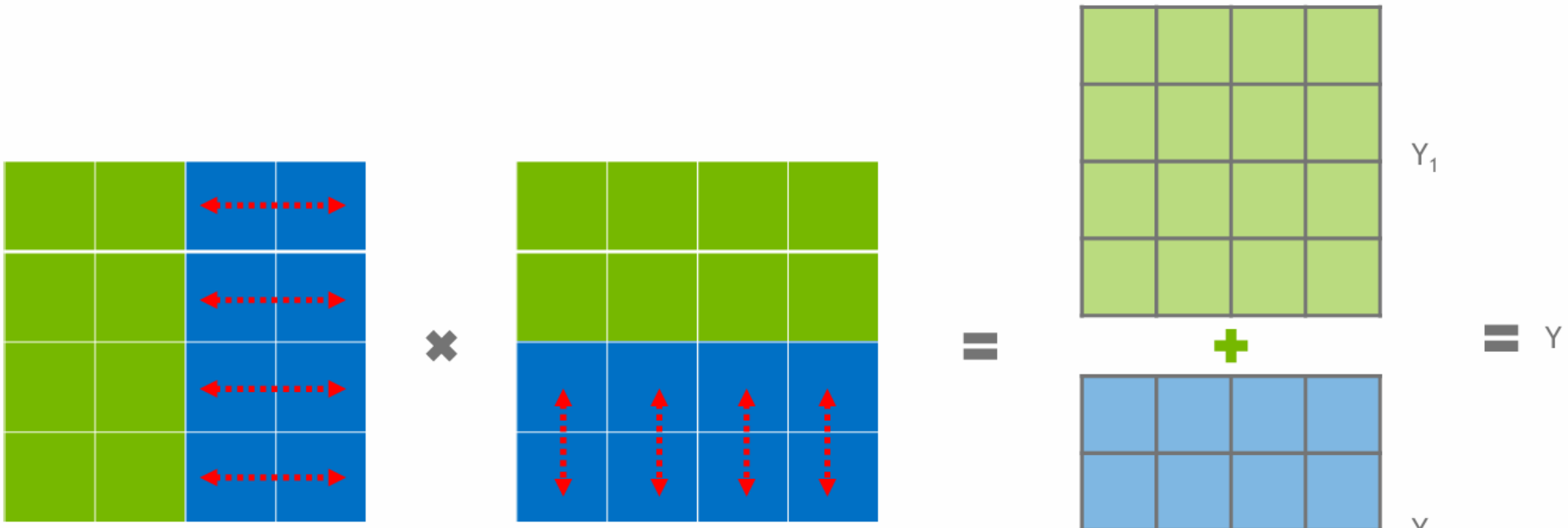


$$X = [X_1, X_2]$$

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$Y = Y_1 + Y_2$$

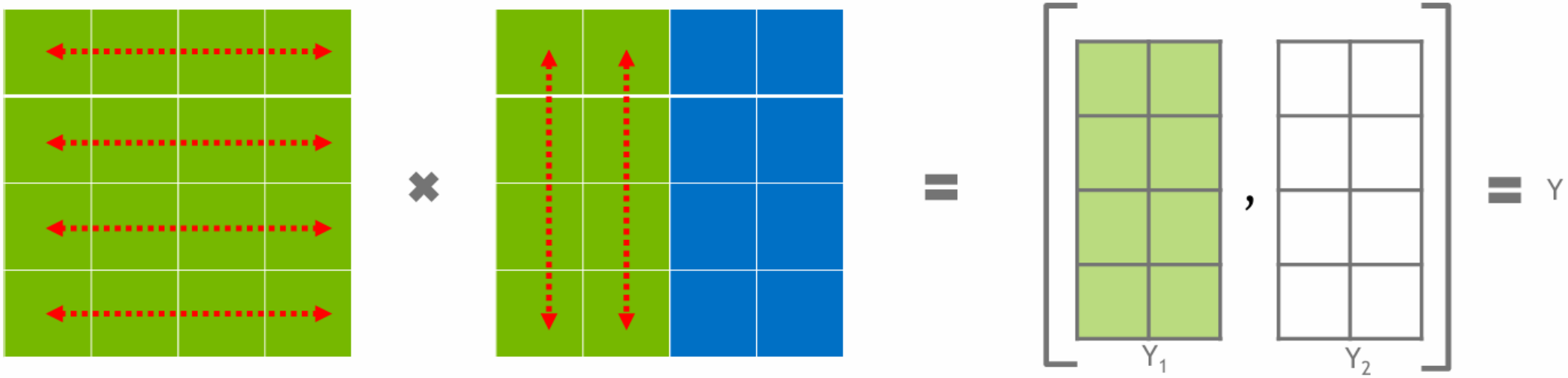
Tensor Parallel: Parallel GeMM



$$X = [X_1, X_2] \qquad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

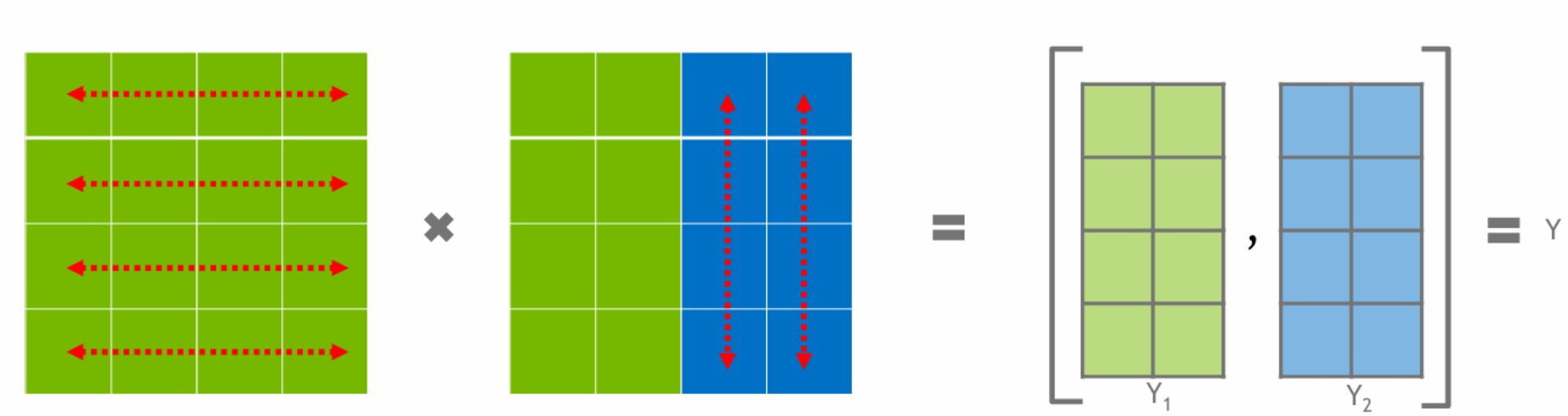
$$Y = Y_1 + Y_2$$

Tensor Parallel: Parallel GeMM



X $\quad A = [A_1, A_2]$ $\quad Y = [Y_1, Y_2]$

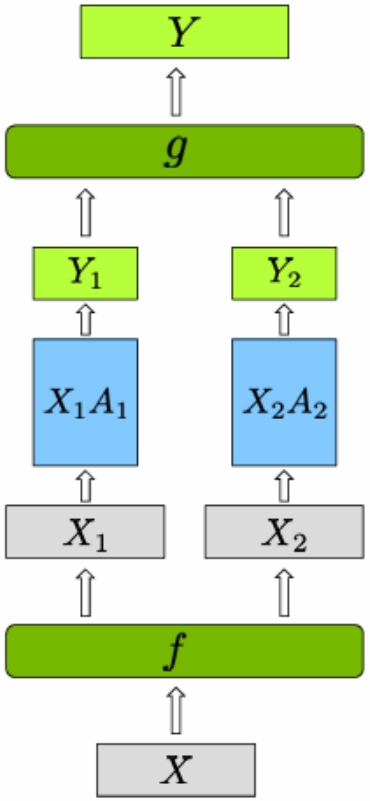
Tensor Parallel: Parallel GeMM



X $\quad A = [A_1, A_2]$ $\quad Y = [Y_1, Y_2]$

Tensor Parallelism

Row Parallel Linear Layer

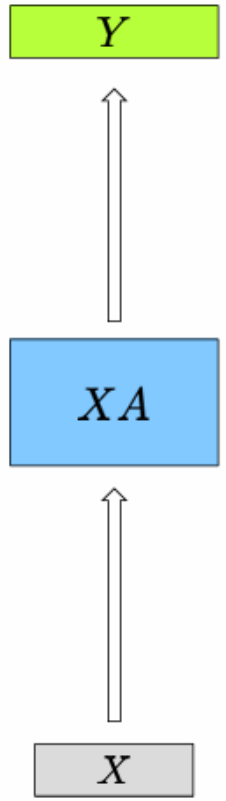


g : forward: $Y = Y_1 + Y_2$ (all-reduce)
backward: $\frac{\partial L}{\partial Y_i} = \frac{\partial L}{\partial Y}$ (identity)

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$X = [X_1, X_2]$$

f : forward: X_i (split)
backward: $\frac{\partial L}{\partial X} = [\frac{\partial L}{\partial X_1}, \frac{\partial L}{\partial X_2}]$ (all-gather)

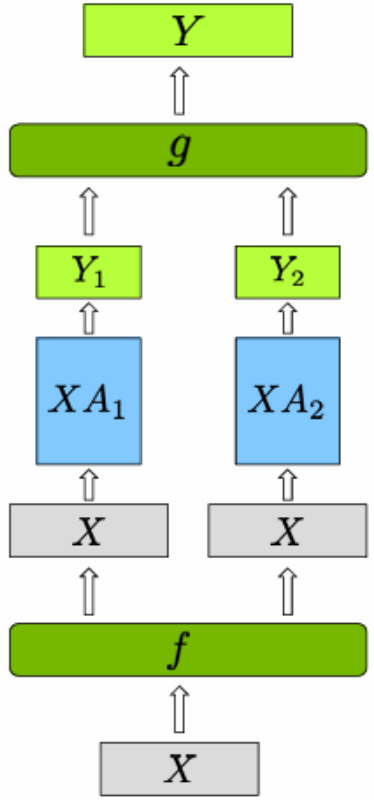


Column Parallel Linear Layer

g : forward: $Y = [Y_1, Y_2]$ (all-gather)
backward: $\frac{\partial L}{\partial Y_i}$ (split)

$$A = [A_1, A_2]$$

f : forward: X (identity)
backward: $\frac{\partial L}{\partial X} = \frac{\partial L}{\partial X}|_1 + \frac{\partial L}{\partial X}|_2$ (all-reduce)



Tensor Parallelism

Normal

Operation: $Y_{n\times n} = X_{n\times n}A_{n\times n}$

Flops: $2n^3$

Bandwidth: $6n^2$

Intensity: $\frac{1}{3}n$

Parallel

Operation: $Y_{n\times (n/p)} = X_{n\times n}A_{n\times (n/p)}$

Flops: $2n^3/p$

Bandwidth: $2n^2(1+2/p)$

Intensity: $\frac{1}{2+p}n$

Ratio between serial and parallel

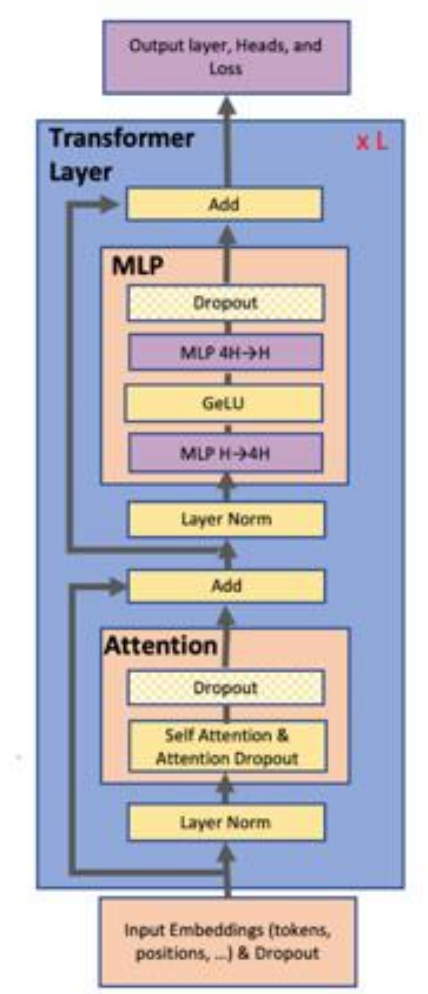
Flops: $1/p$ 

Bandwidth: $\frac{1+2/p}{3}$ 

Intensity: $\frac{3}{2+p}$ 

Approach and Goals

- Tailored for transformer networks
- Transformer layer
  - Self-attention block
  - Two-layer MLP
- Targets:
  - Reduce synchronization cost
  - Simple to implement with a few highly optimized collectives NCCL provides



Tensor Parallelism: MLP

- MLP:

$$Y = \mathrm{GeLU}(XA)$$
$$Z = \mathrm{Dropout}(YB)$$

- Approach 1: split X column-wise and A row-wise

$$X = [X_1, X_2] \quad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \longrightarrow Y = \mathrm{GeLU}(X_1A_1 + X_2A_2)$$

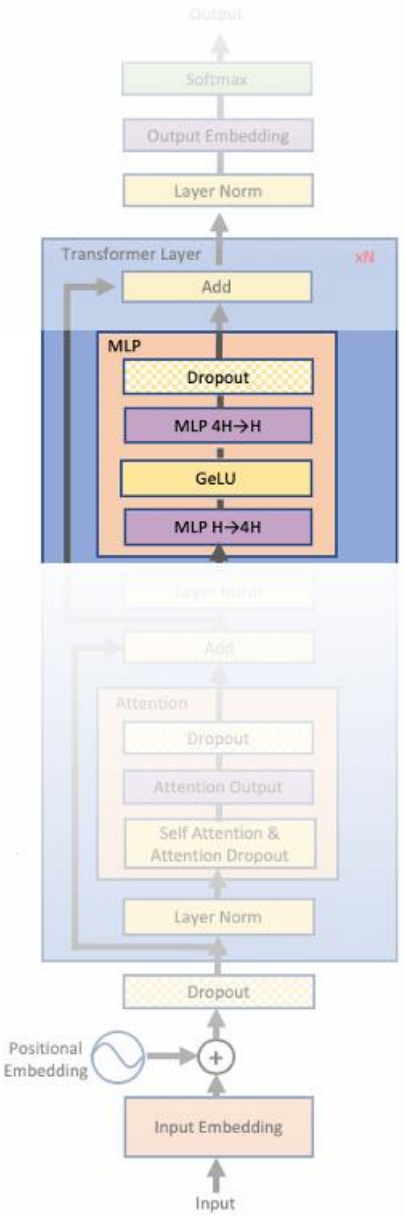
  - Requires synchronization before GeLU

GeLU of sums != sum of GeLUs

- Approach 2: split A column-wise

$$A = [A_1, A_2] \longrightarrow [Y_1, Y_2] = [\mathrm{GeLU}(XA_1), \mathrm{GeLU}(XA_2)]$$

  - No synchronization necessary



Tensor Parallelism: MLP

- MLP:

$$Y = \mathrm{GeLU}(XA)$$
$$Z = \mathrm{Dropout}(YB)$$

- Approach 1: split X column-wise and A row-wise

$$X = [X_1, X_2] \quad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \longrightarrow Y = \mathrm{GeLU}(X_1A_1 + X_2A_2)$$

  - Requires synchronization before GeLU

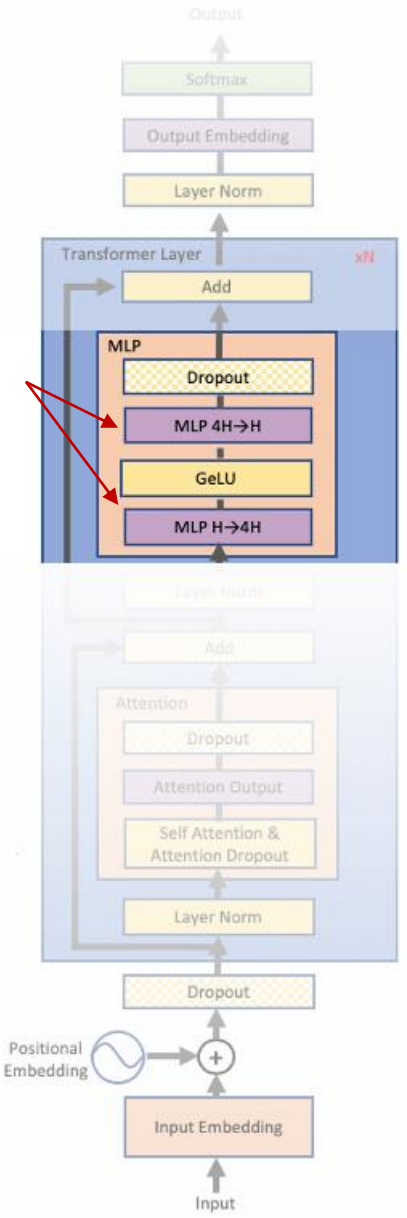
GeLU of sums != sum of GeLUs

- Approach 2: split A column-wise

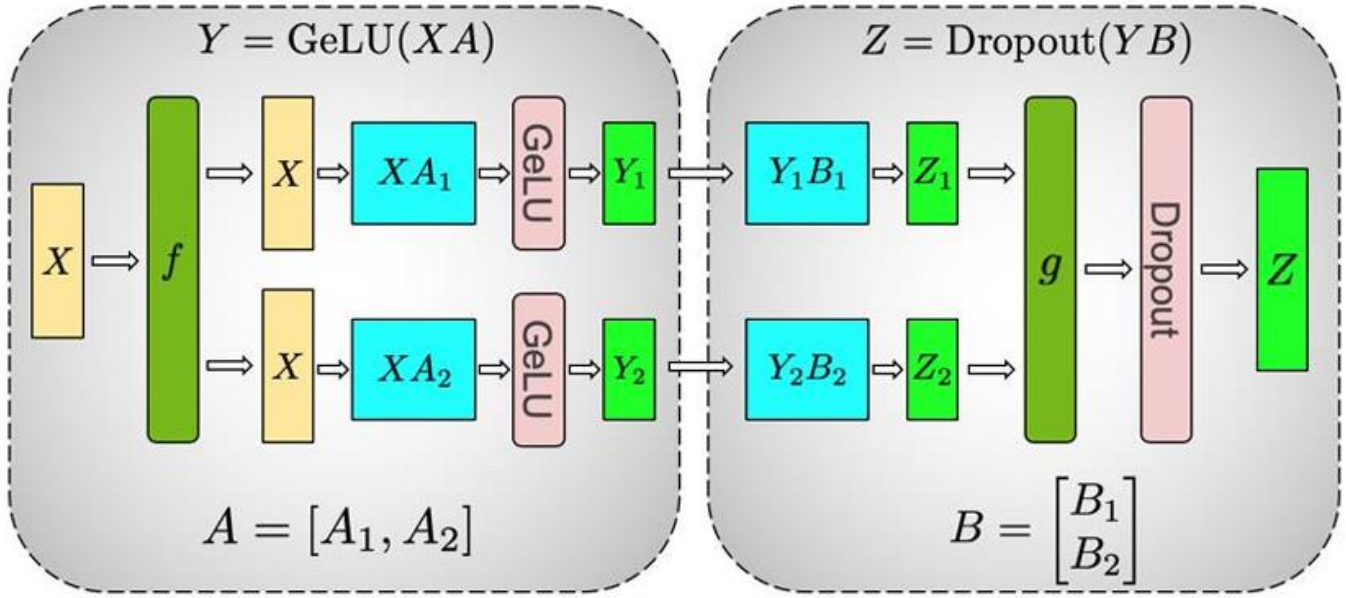
$$A = [A_1, A_2] \longrightarrow [Y_1, Y_2] = [\mathrm{GeLU}(XA_1), \mathrm{GeLU}(XA_2)]$$

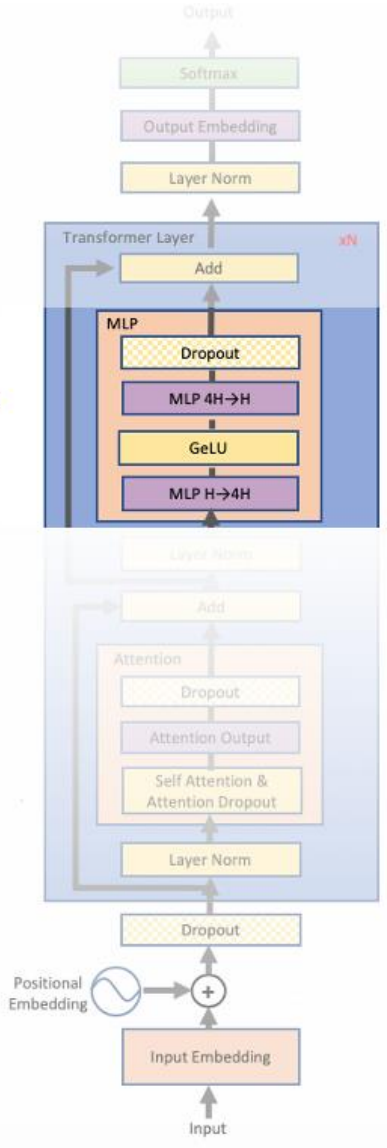
  - No synchronization necessary

Shape(Y) = [B, S, 4H]!



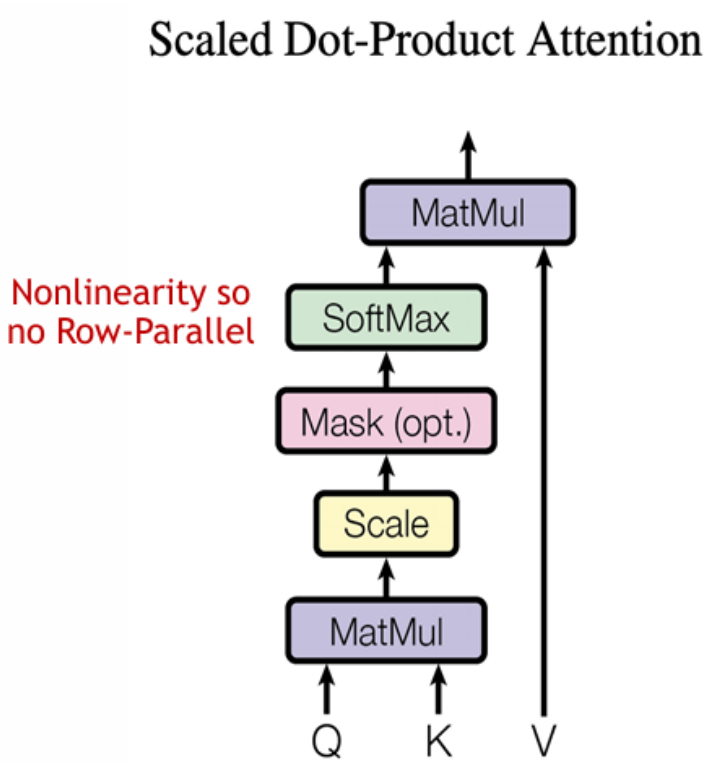
Tensor Parallelism: MLP

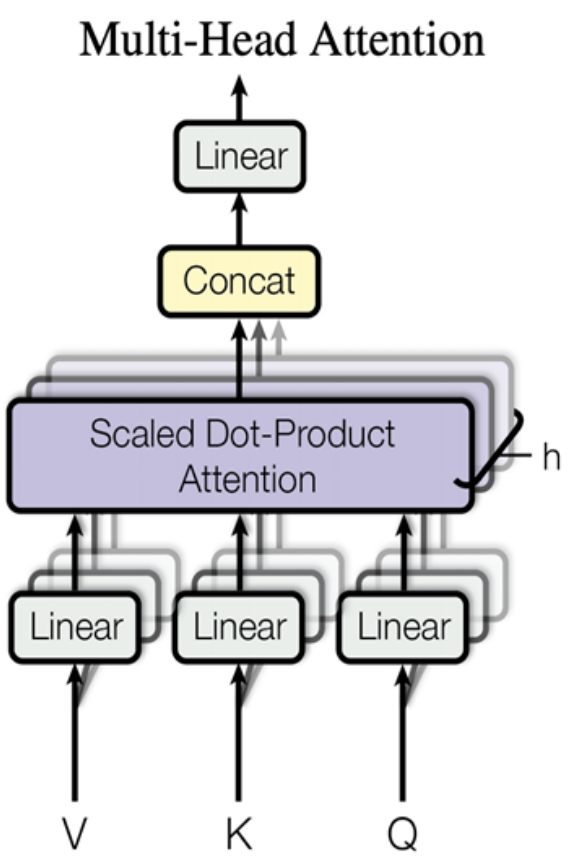


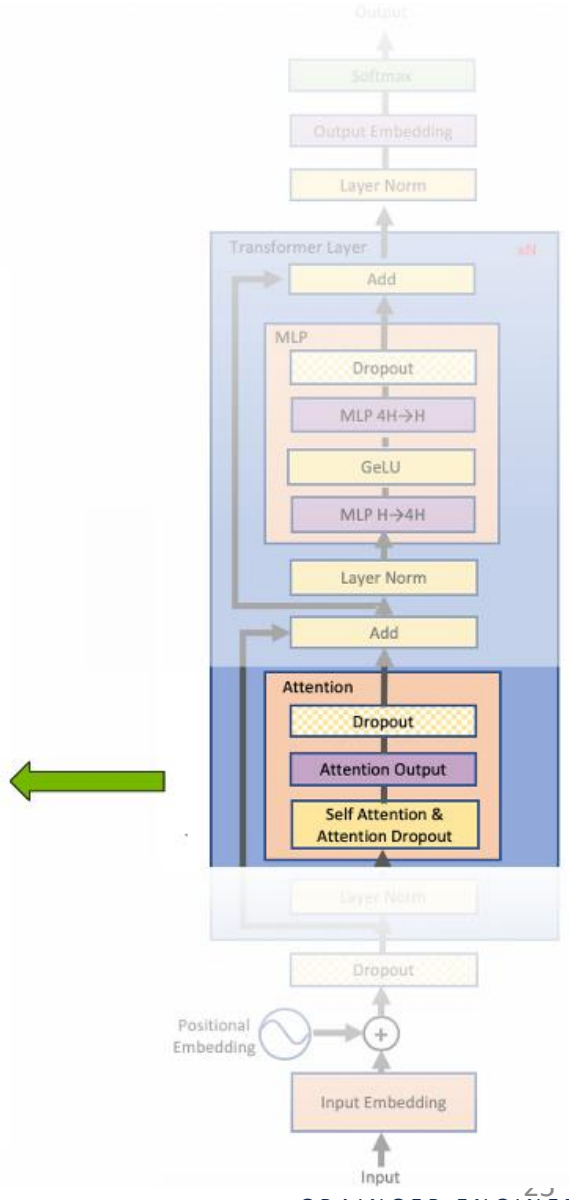


f and ***g*** are **conjugate**, ***f*** is **identity** operator in the forward pass and **all-reduce** in the backward pass while ***g*** is **all-reduce** in forward and **identity** in backward.

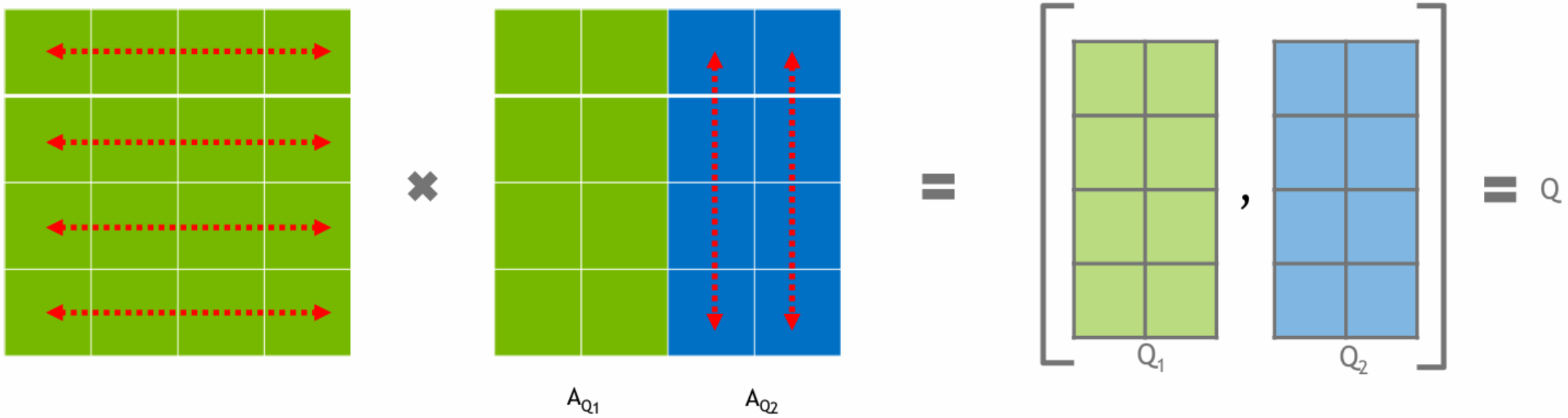
Tensor Parallelism: Self-Attention







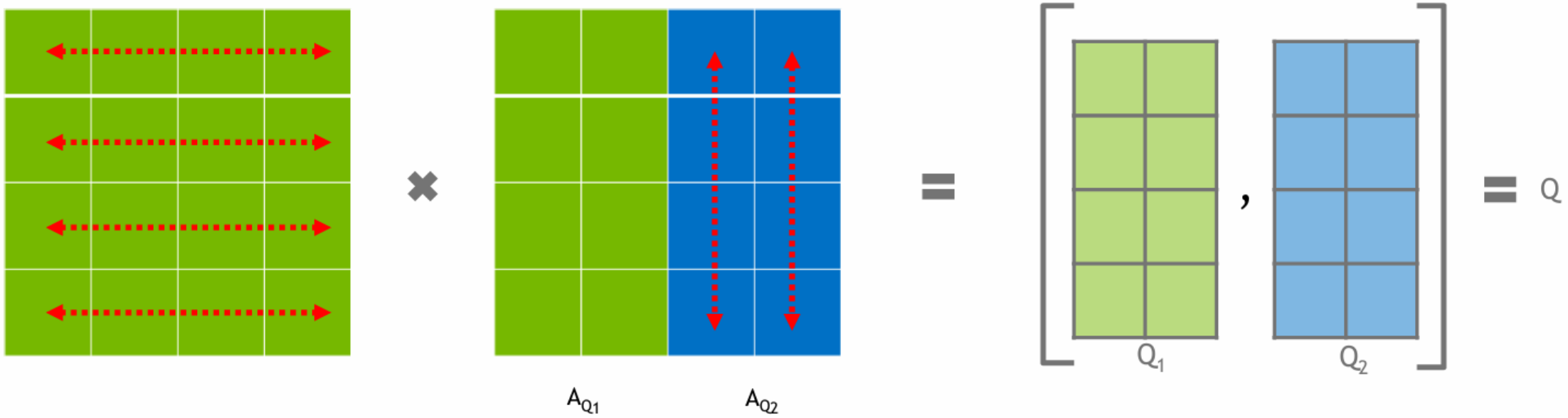
Tensor Parallelism: Self-Attention



X $\quad A = [A_1, A_2]$ $\quad Y = [Y_1, Y_2]$

Attention heads can be parallelized with Column Parallel GEMMs (ex. Query head 1 (Q_1) and Query head 2 (Q_2))

Tensor Parallelism: Self-Attention

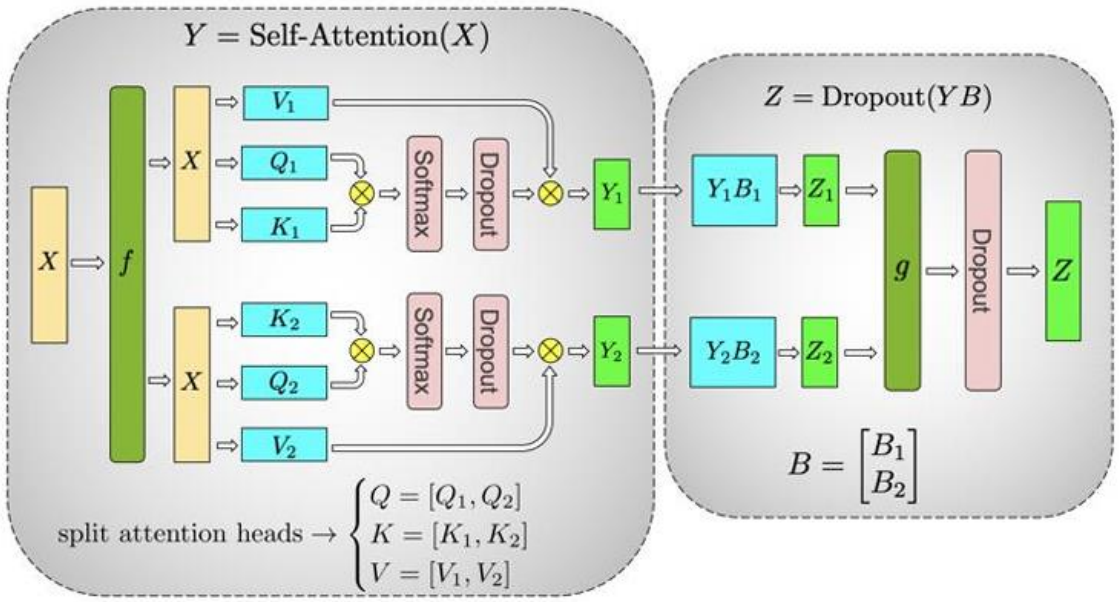


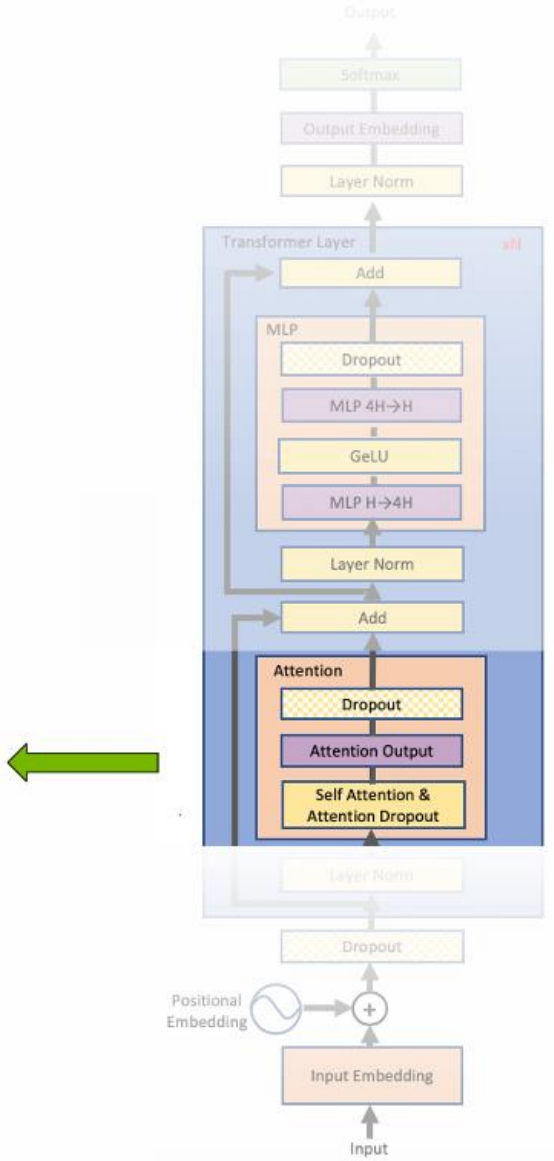
X $A = [A_1, A_2]$ $Y = [Y_1, Y_2]$

Attention heads can be parallelized with Column Parallel GEMMs (ex. Query head 1 (Q_1) and Query head 2 (Q_2))

Question: What if we have more GPUs than the number of heads?

Tensor Parallelism: Self-Attention

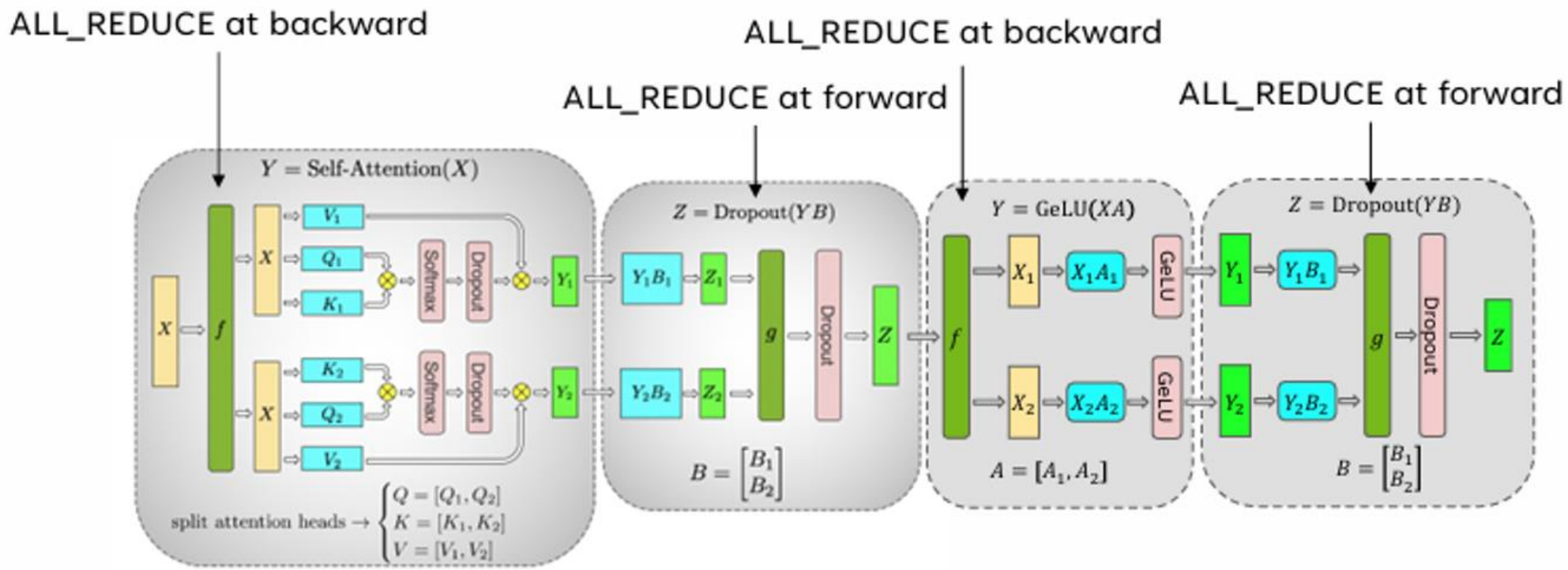




f and ***g*** are **conjugate**, ***f*** is **identity** operator in the forward pass and **all-reduce** in the backward pass while ***g*** is **all-reduce** in forward and **identity** in backward.

Tensor Parallelism: Communication Cost

- 4 total communication operations in 1 forward + backward pass



Experiments

- 32 DGX-2H servers
  - 512 V100 SXM3 32GB GPUs
- Intra-server connection
  - 300 GB/s NVSwitch
- Inter-server connection
  - 100 GB/s InfiniBand (8 per server)

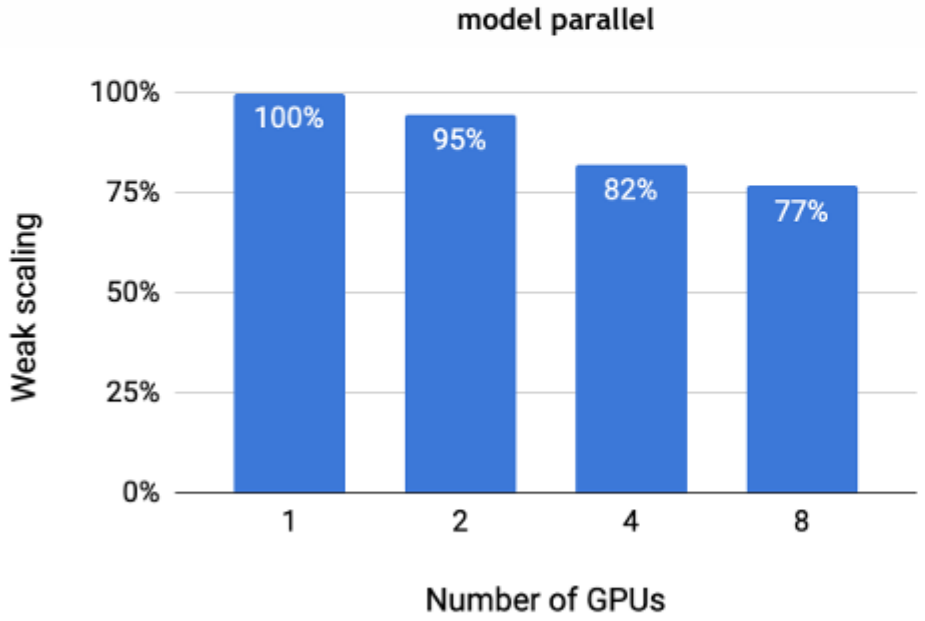
- GPT-2 & BERT Models
- TP (+ DP)
- Mixture of datasets

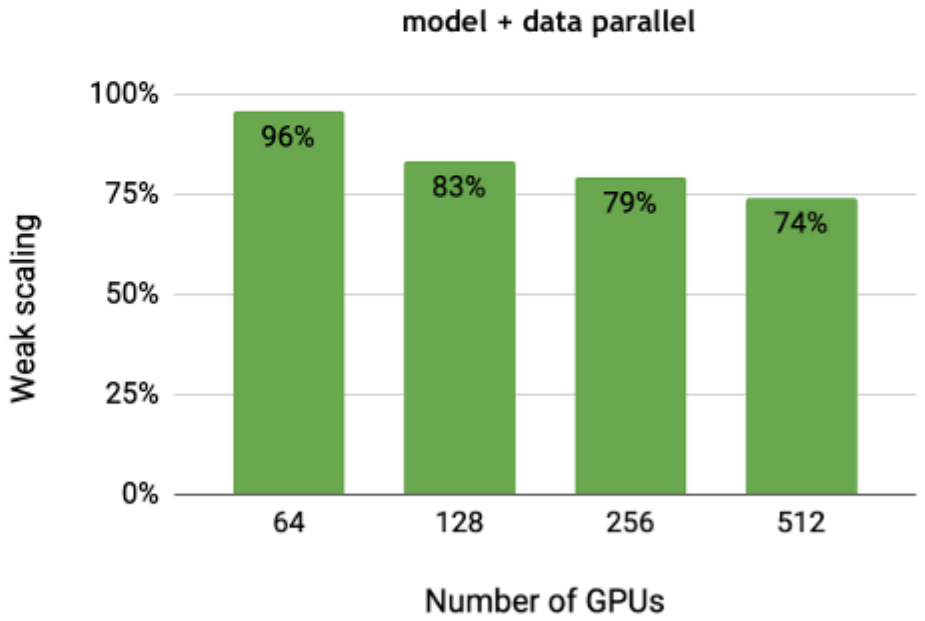
Experiments: Scalability

- GPT-2: 1B-8B
- Hidden size: 1536-3072
- Parameters/GPU: ~1.B

Config	Hidden size	Attention heads	Number of layers	Number of parameters (billions)	Model parallel GPUs	Model+data parallel GPUs
1	1536	16	40	1.2	1	64
2	1920	20	54	2.5	2	128
3	2304	24	64	4.2	4	256
4	3072	32	72	8.3	8	512

- Weak scaling@512 GPUs: 74%

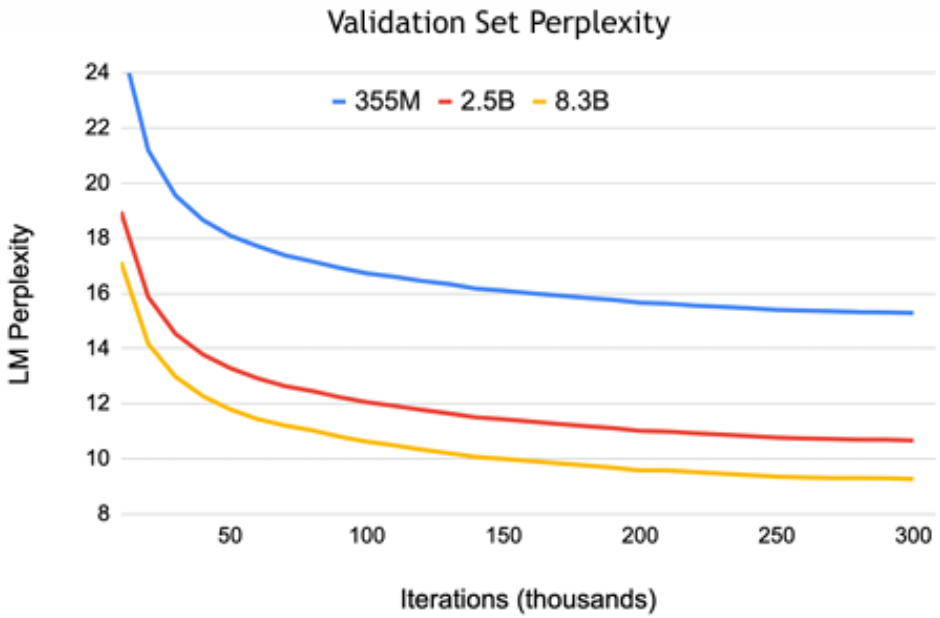




Baseline (1.2B parameters on a single GPU) achieves 39 TeraFLOPs per second, i.e. 30% of the theoretical peak during the entire training process

Experiments: GPT-2

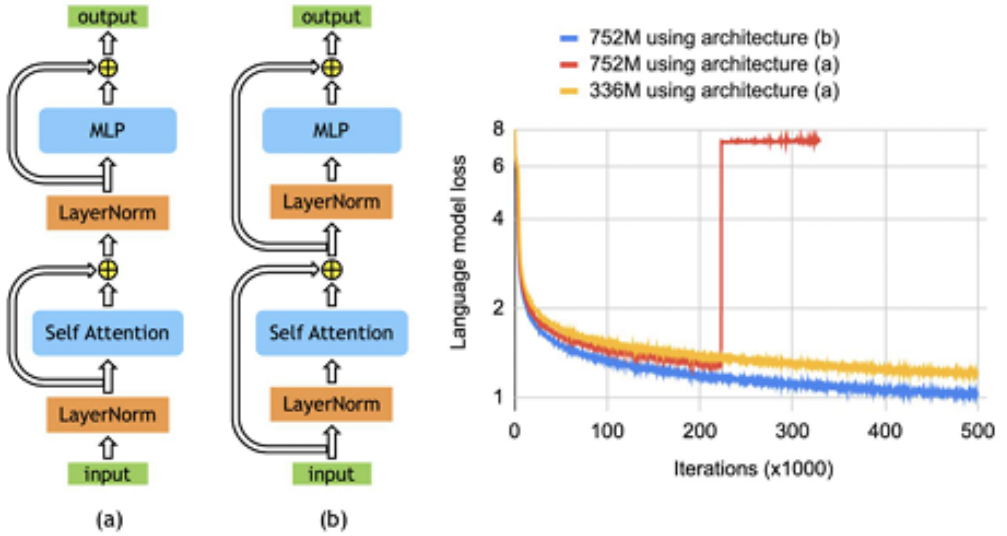
- Training data: 174 GB WebText/CC-Stories/Wikipedia/RealNews
- 3 model sizes: 355 million, 2.5 billion, and 8.3 billion
- Zero-shot evaluation results for Wikitext-103 perplexity and Lambada cloze accuracy



Model Size	Wikitext-103 (Perplexity ↓)	Lambada (Accuracy ↑)
355 M	19.22	46.26
2.5 B	12.68	61.52
8.3 B	**10.81**	**66.51**
Previous SOTA	16.43*	63.24**

Experiments: Megatron-BERT

- Scaling BERT to larger sizes is also possible
- Reordering residual connections to stabilize training
- Megatron-BERT 3.9B, 12x larger than BERT-Large, over 2 million iterations @ batch size 1024



Parameter Count	Parameter Multipler	Hidden Size	Attention Heads	Layers	Model Parallel GPUs	Model + Data Parallel GPUs
334M	1x	1024	16	24	1	128
1.3B	4x	2048	32	24	1	256
3.9B	12x	2506	40	48	4	512

Experiments: SQUAD & RACE

Model	Trained tokens (ratio)	MNLI† m/mm accuracy	QQP† accuracy	SQuAD 1.1† F1/EM	SQuAD 2.0† F1/EM	RACE m/h* accuracy
RoBERTa	2	90.2 / 90.2	92.2	94.6 / 88.9	89.4 / 86.5	86.5 / 81.3
ALBERT	3	90.8	92.2	94.8 / 89.3	90.2 / 87.4	89.0 / 85.5
XLNet	2	90.8 / 90.8	92.3	95.1 / 89.7	90.6 / 87.9	88.6 / 84.0
Megatron-334M	1	89.7 / 90.0	92.3	94.2 / 88.0	88.1 / 84.8	86.9 / 81.5
Megatron-1.3B	1	90.9 / 91.0	92.6	94.9 / 89.1	90.2 / 87.1	90.4 / 86.1
Megatron-3.9B	**1**	**91.4 / 91.4**	**92.7**	**95.5 / 90.0**	**91.2 / 88.5**	**91.8 / 88.6**

Median single model downstream results on Dev† and Test* sets.
State of the art results are bolded.

Questions?