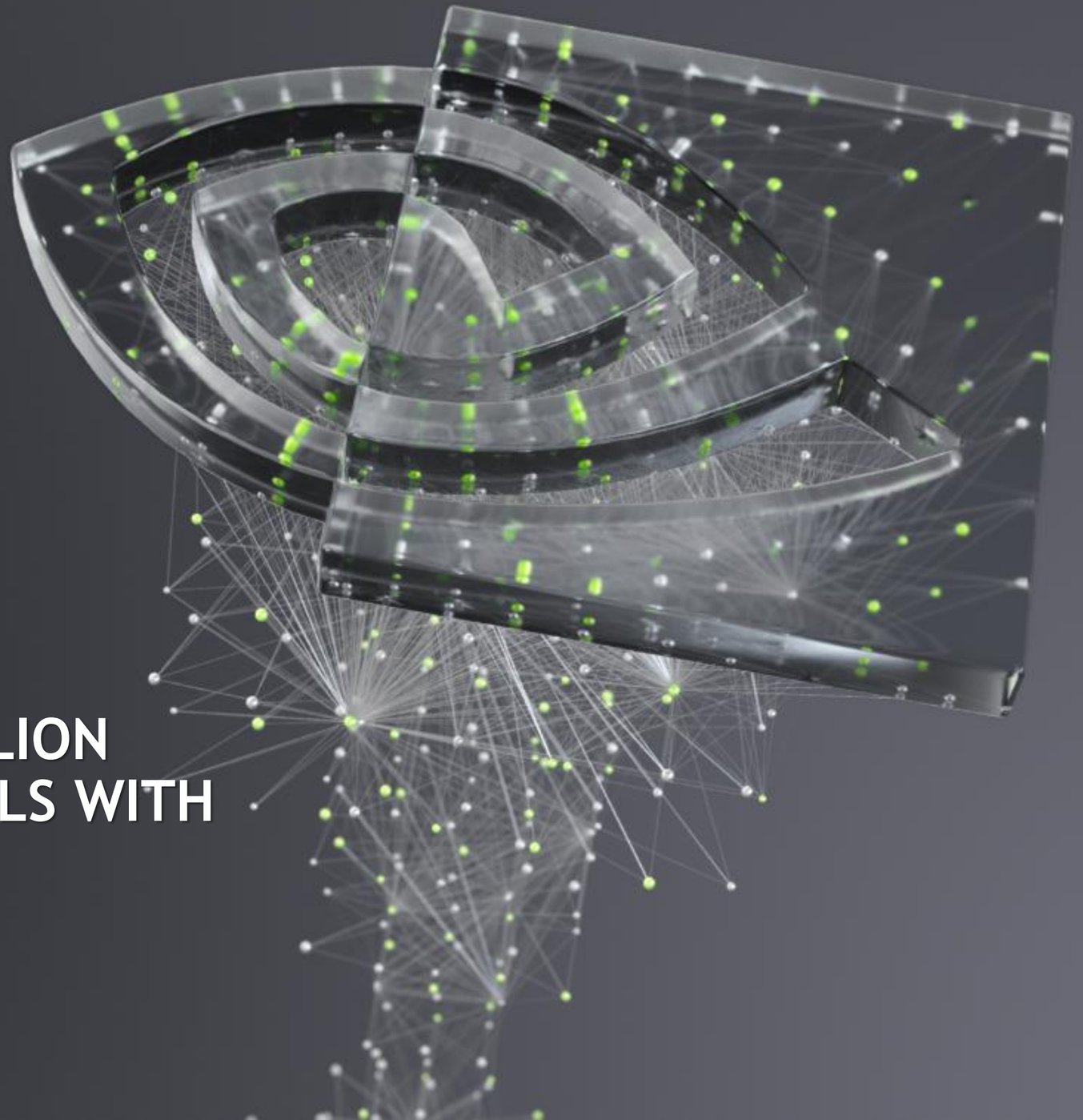


MEGATRON-LM: TRAINING BILLION PARAMETER LANGUAGE MODELS WITH GPU MODEL PARALLELISM

Raul Puri, 03/06/2020

OUR TEAM

Raul Puri

Mohammad Shoeybi

Mostofa Patwary

Patrick LeGresley

Jared Casper

Bryan Catanzaro

AGENDA

Megatron

Training Large Scale Language Models Using Model Parallelism on GPUs

---

Reading Comprehension with Megatron

Applying Megatron to question answering and question generation

---

Conversing with Megatron

Modeling user conversations at scale with controllable personality transfer


MEGATRON-LM

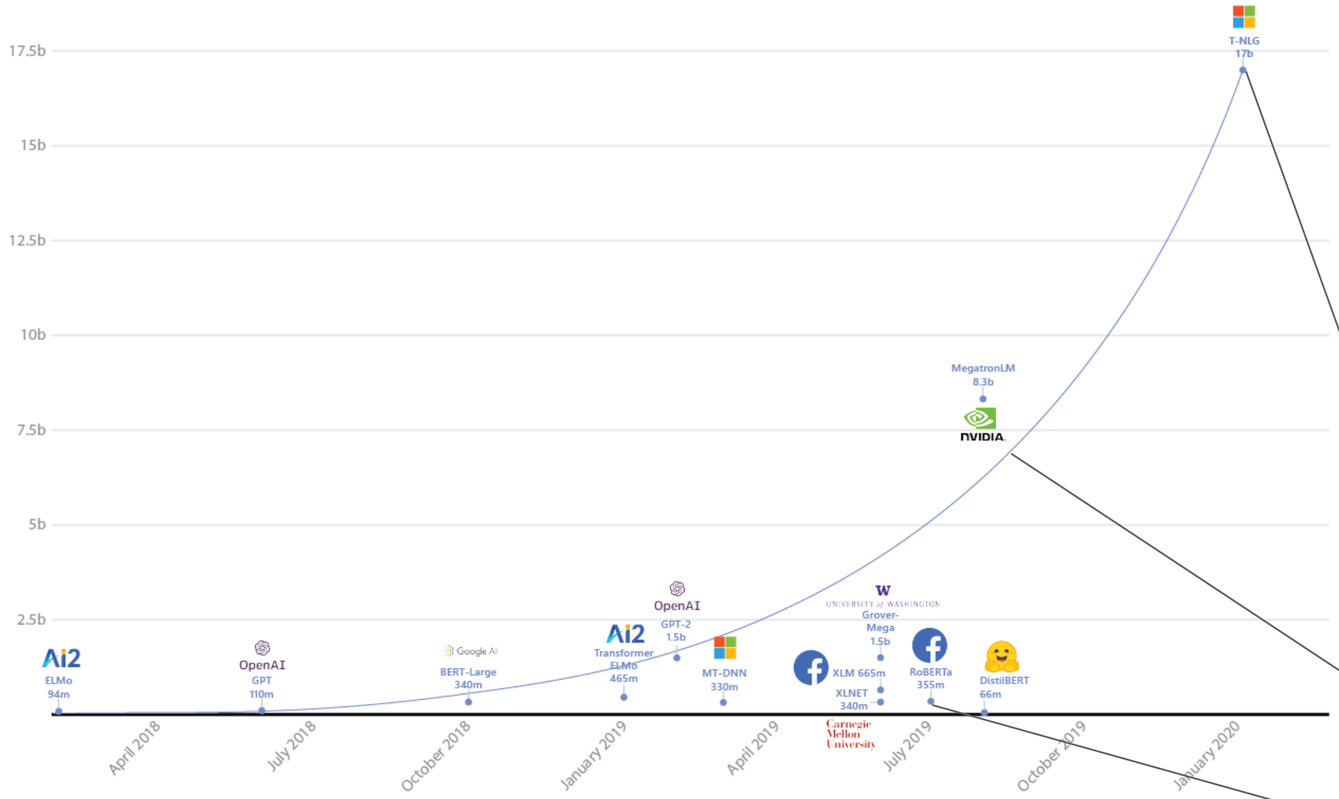
WHAT IS MEGATRON?

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

NVIDIA's framework for efficiently training the world's largest language models

MOTIVATION

Computational Needs of NLP

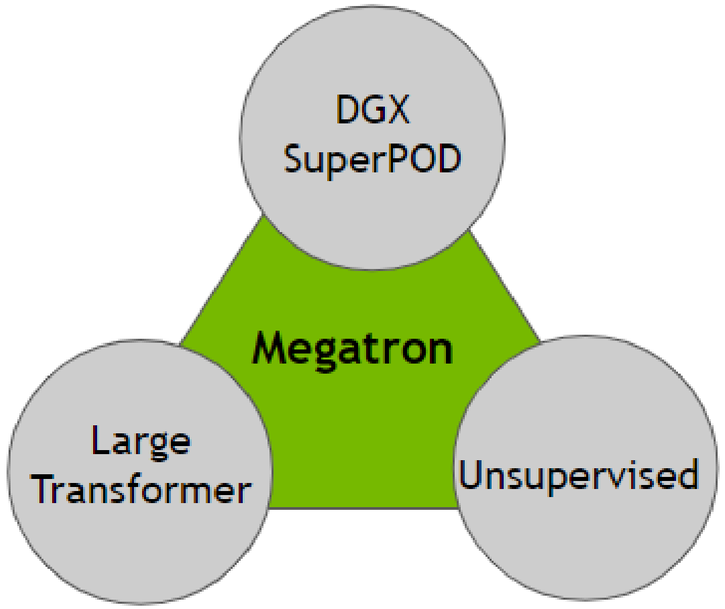


Model	# of Parameters	# of Iterations	# of GPUs	Training Time
T-NLG (MS + NV)	17.2 B	300 K	400	60 days
Megatron-GPT2 (NV)	8.3 B	300 K	512	10 days
Megatron-BERT (NV)	3.9 B	2 M	512	25 days
RoBERTa (FB)	345M	4 M	1024	1 day

MOTIVATION

Why Megatron?

- Training the **largest transformer** based language model has recently been the best way to advance the state of the art in NLP applications.
- **Unsupervised** Language Models such as GPT-2, BERT, and XLNet demonstrate the power of large language models trained on a huge corpus
- NVIDIA **DGX SuperPOD** optimized for Deep Learning and HPC provides a unique opportunity for training very large models



GOALS & CHALLENGES

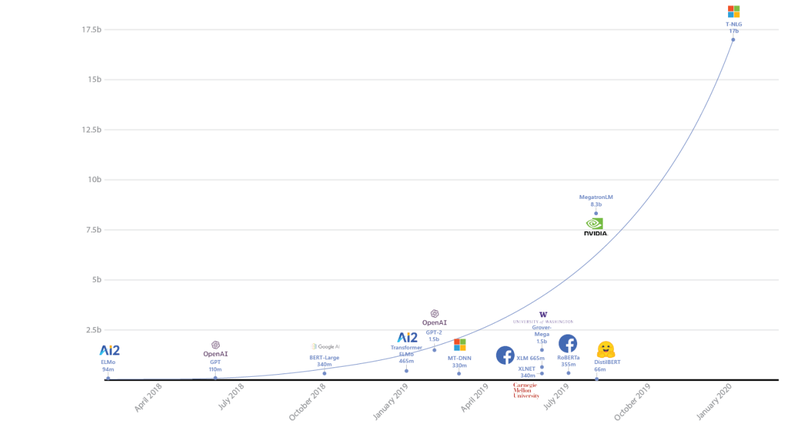
What would we like to do with Megatron?

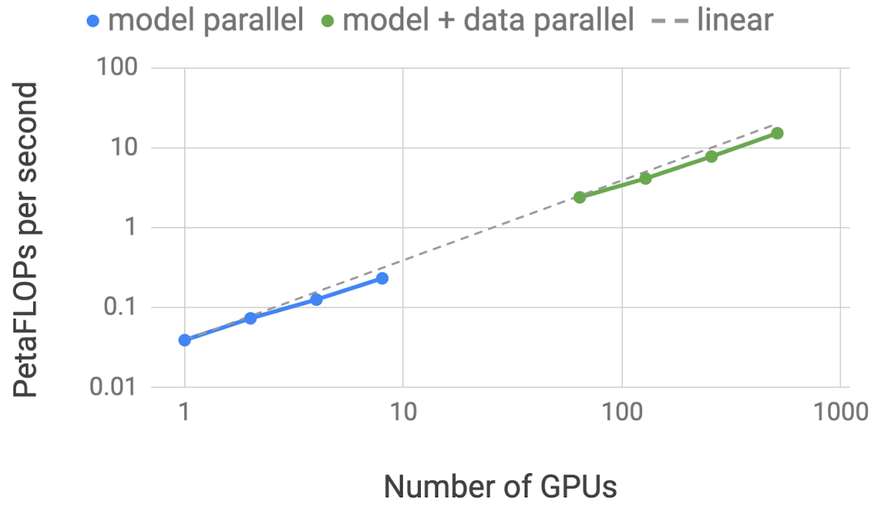
- Training of transformer-based language models with **billions of parameters**
  - Requires **model parallelism** to fit in GPU memory
- Achieving high utilization and scaling up to **hundreds of GPUs**
- Devising simple methods that require minimal changes to our existing code-base (**reducing barrier to entry**)
- Using the developed methodology to scale out Transformer language models such as **BERT** and **GPT-2** and to explore their representation capabilities

ACHIEVEMENTS

What have we done with Megatron?

- **World's largest transformer based language models** are trained using Megatron
- Achieved **15.1 PetaFLOPs per second** sustained performance over the entire application using 512 GPUs at **76% scaling efficiency** compared to a strong single GPU baseline that achieves **39 TeraFLOPs per second**
- **SOTA** for a variety of language modeling tasks such as Wikitext-103 (**10.81** compare to 16.4 perplexity) and reading comprehension tasks such as RACE (**90.9** compared to 89.4 accuracy)
- Significant advancements in downstream applications: **reading comprehension**, **question answering**, and **dialogue modeling systems**.





LANGUAGE MODELS & TRANSFORMERS

LANGUAGE MODELING BASICS

What is a Language Model?

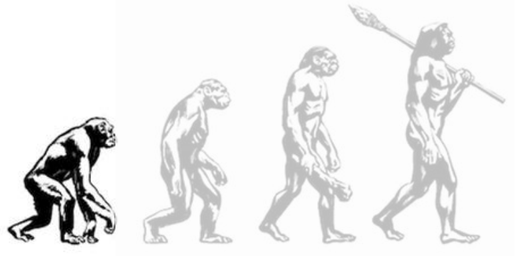
$$P(w_1, w_2, \ldots, w_{T-1}, w_T) = \prod_{t=1}^{T} P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1)$$

the	cat	sat	on	the	mat	$P(w_1)$
the	**cat**	sat	on	the	mat	$P(w_2 \mid w_1)$
the	cat	**sat**	on	the	mat	$P(w_3 \mid w_2, w_1)$
the	cat	sat	**on**	the	mat	$P(w_4 \mid w_3, w_2, w_1)$
the	cat	sat	on	**the**	mat	$P(w_5 \mid w_4, w_3, w_2, w_1)$
the	cat	sat	on	the	**mat**	$P(w_6 \mid w_5, w_4, w_3, w_2, w_1)$

Slide Credit: Piotr Mirowski

LANGUAGE MODELING BASICS

LM Evolution: N-Grams



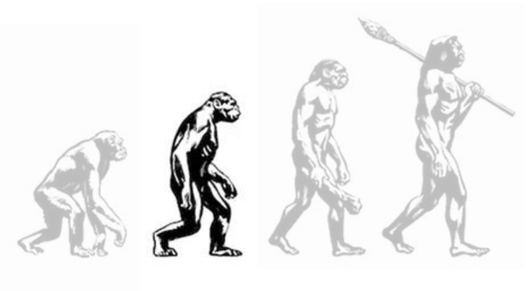
the cat sat on the mat. it was a bad **cat**

bi-grams:

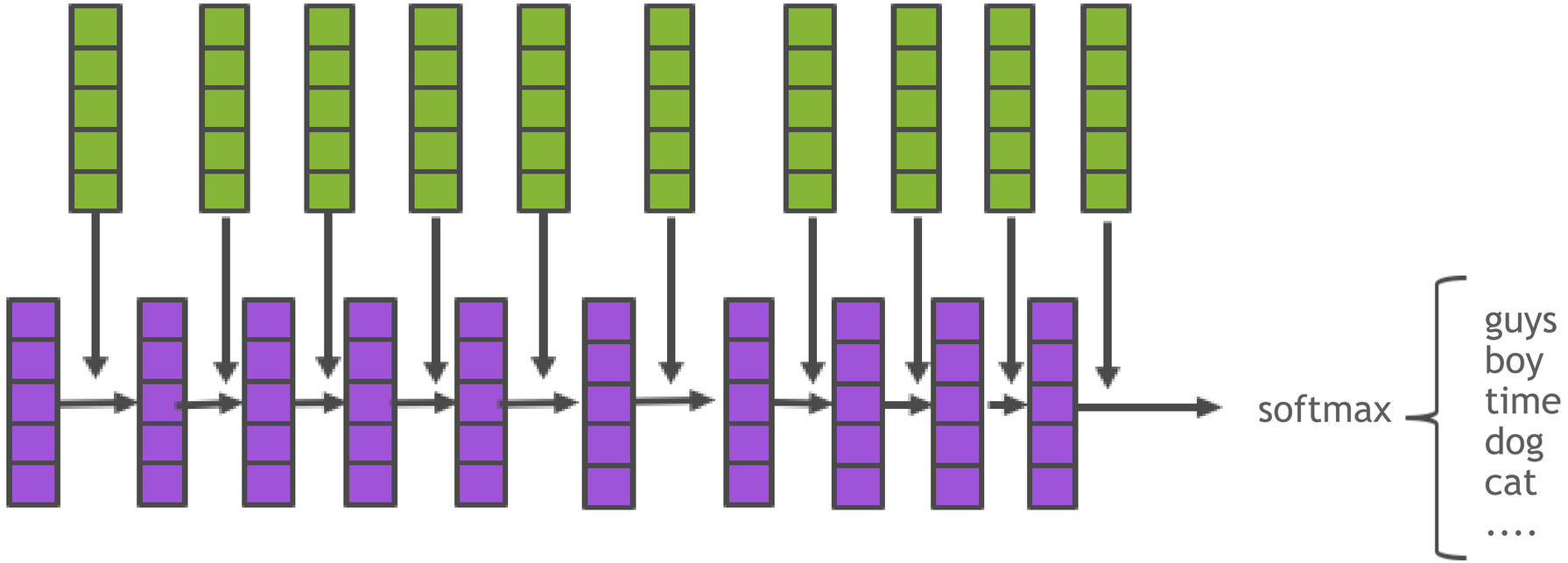
bad guy:	0.30
bad boys:	0.05
bad times:	0.10
bad dog:	0.50
bad cat:	0.04
...	

LANGUAGE MODELING BASICS

LM Evolution: RNNs

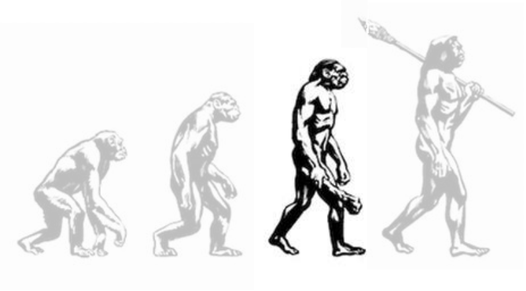


the cat sat on the mat. it was a bad **cat**

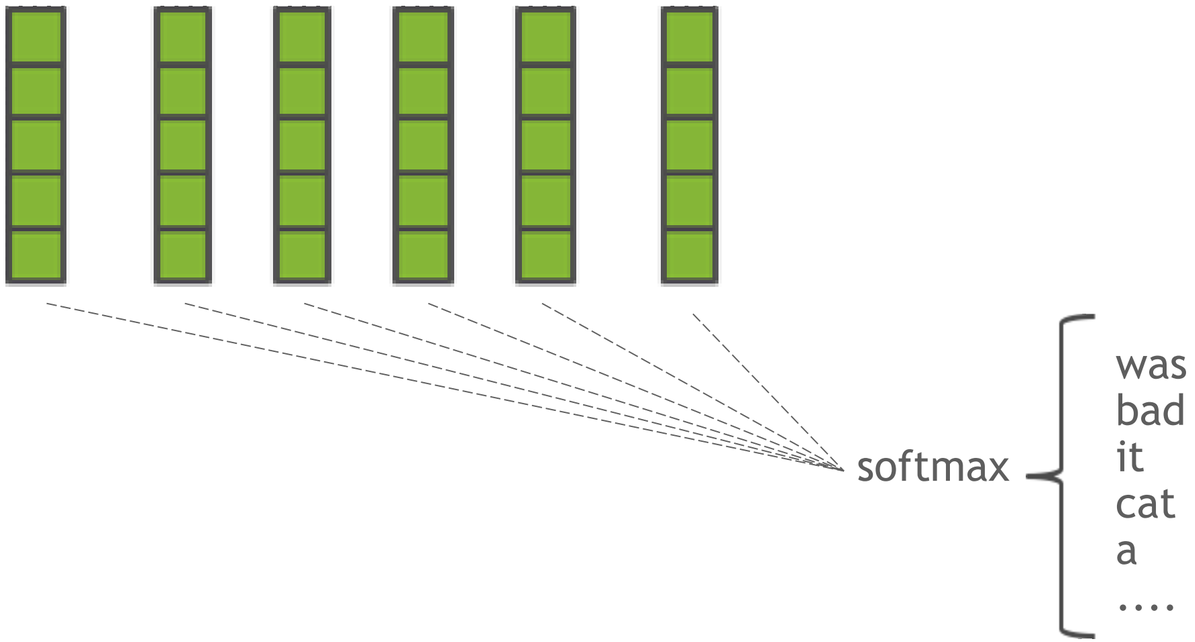


LANGUAGE MODELING BASICS

LM Evolution: Transformers

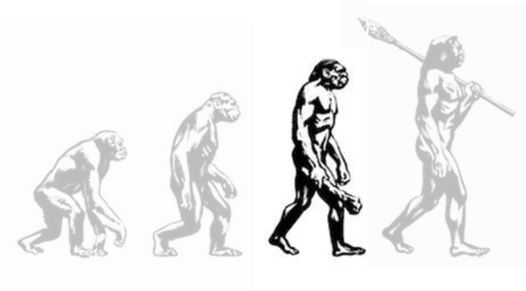


the cat sat on the mat. **it**

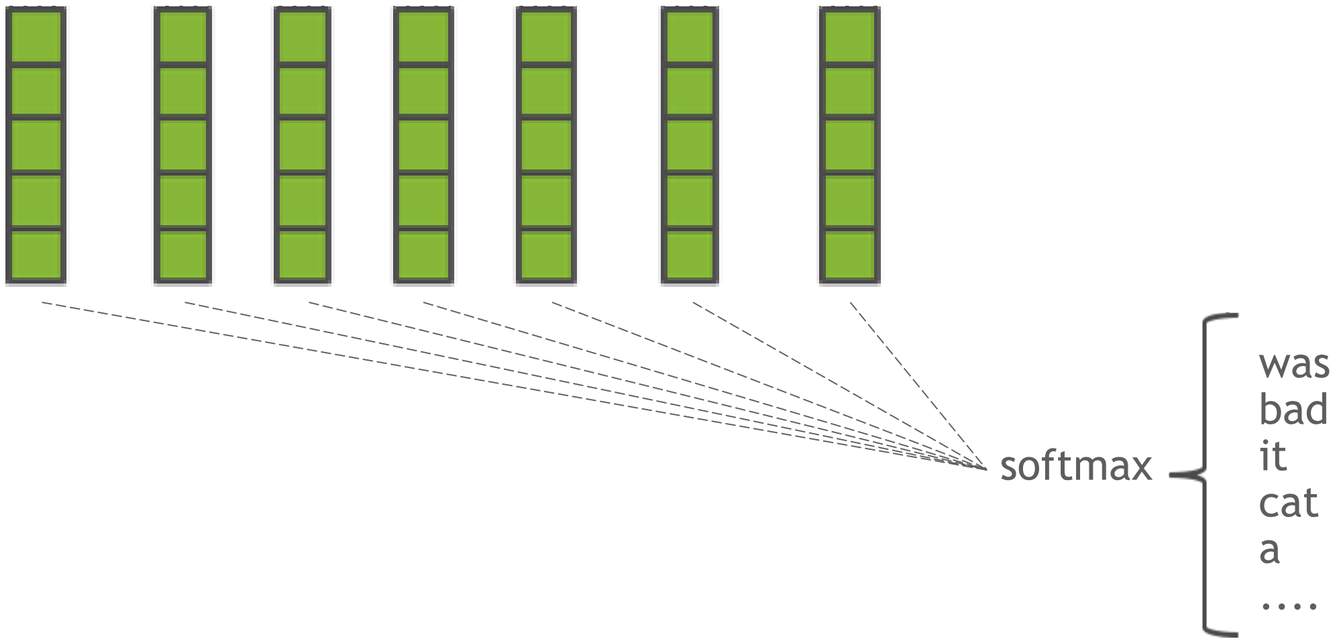


LANGUAGE MODELING BASICS

LM Evolution: Transformers

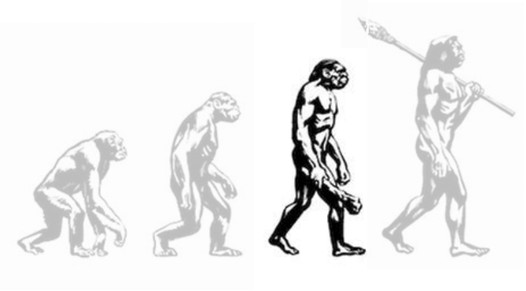


the cat sat on the mat. it **was**

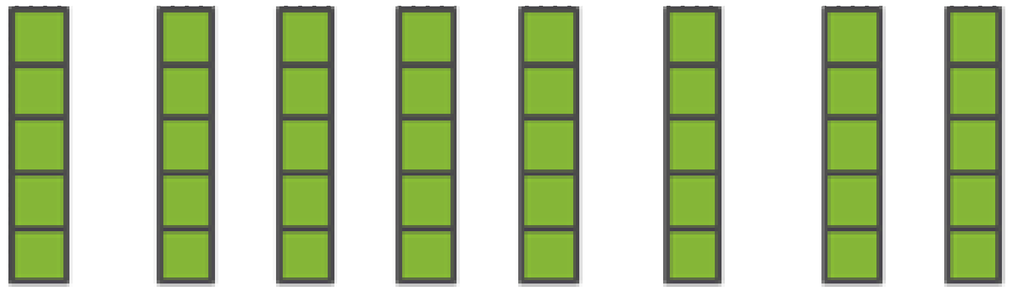


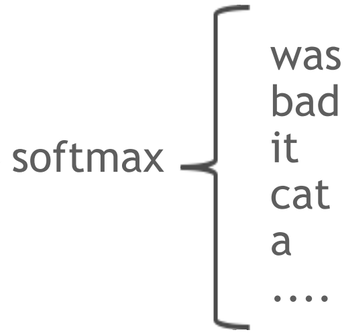
LANGUAGE MODELING BASICS

LM Evolution: Transformers



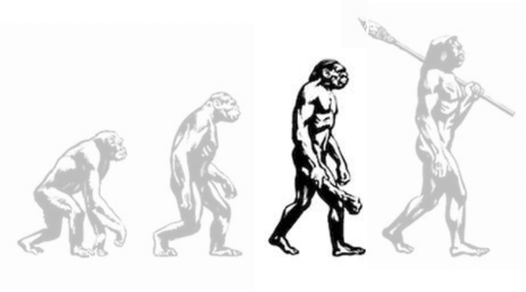
the cat sat on the mat. it was **a**



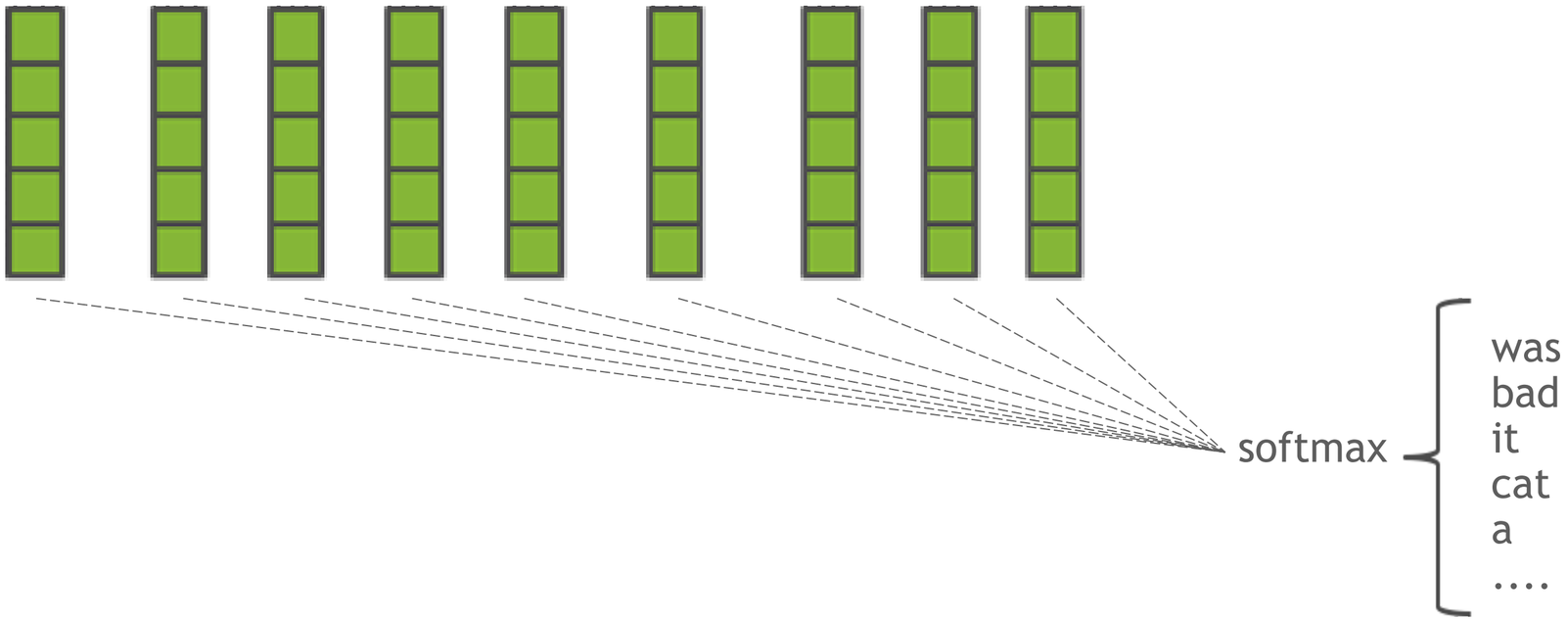


LANGUAGE MODELING BASICS

LM Evolution: Transformers

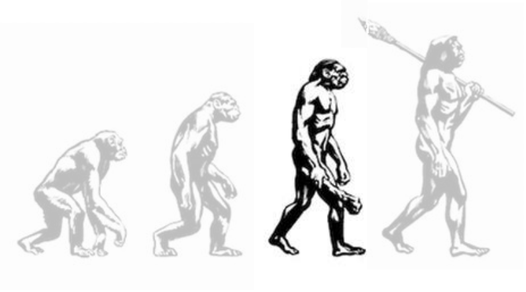


the cat sat on the mat. it was a **bad**

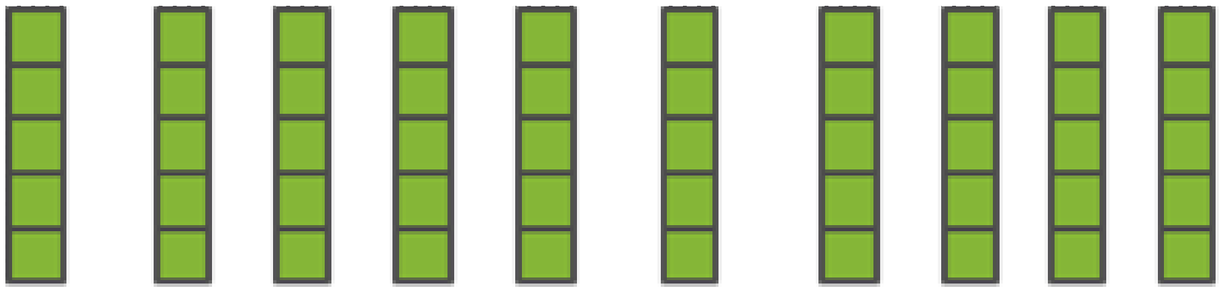


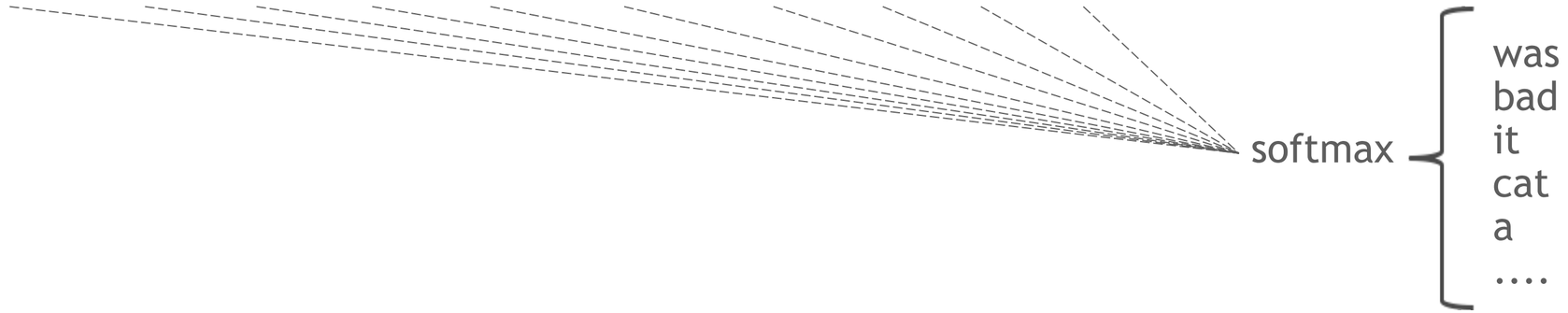
LANGUAGE MODELING BASICS

LM Evolution: Transformers



the cat sat on the mat. it was a bad **cat**





LANGUAGE MODELING BASICS

Large Transformers

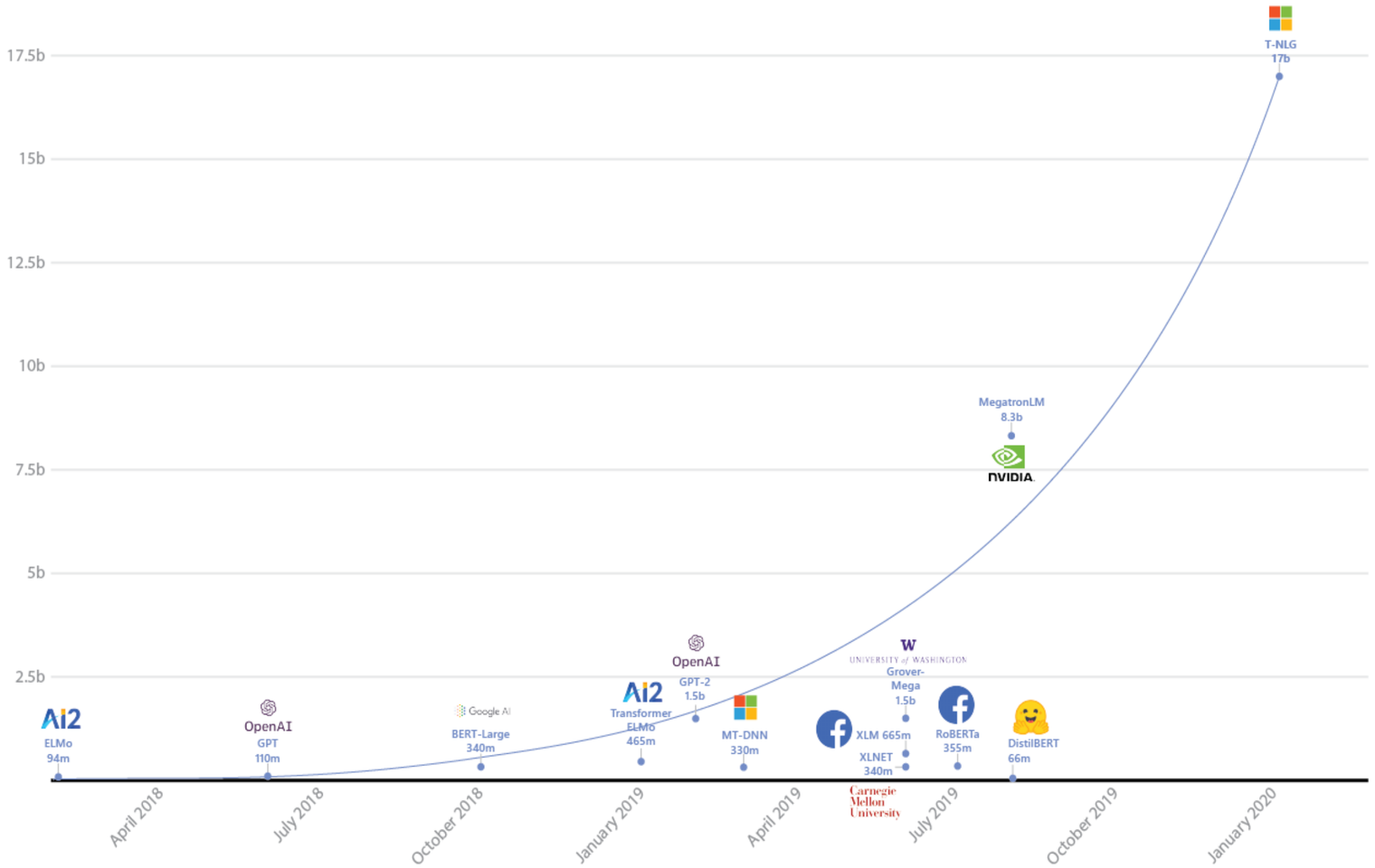
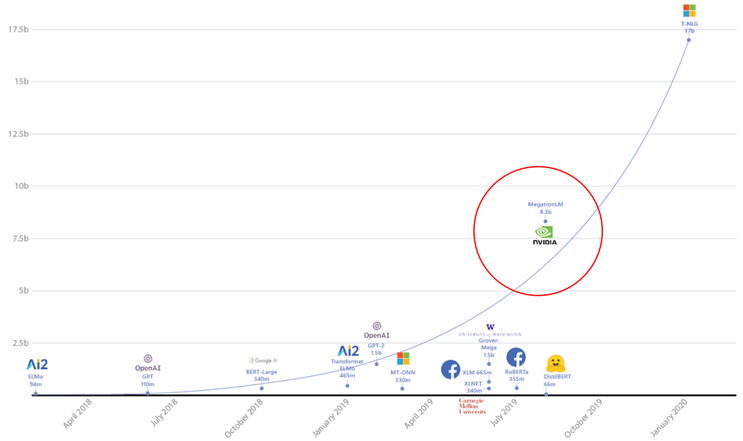


*Figure courtesy of Huggingface and Microsoft blog posts.

TRANSFORMER COSTS

Big Transformers Are Resource Intensive



- Model parameters:

Batch Size (b)	Sequence Length (s)	Hidden Size (h)	Number of Layers (l)	Vocabulary Size (v)	Number of Parameters (Millions)	Batch Size (b)
512	1024	3072	72	51,200	8,317	512

- Required memory in GB:

Parameters	Gradient	Optimizer	total
32.5	32.5	65.0	**130.0**

- Required FLOPS:

PetaFlops Per Iteration	ZettaFlops For Training (300k Iterations)
65.0	**130.0**

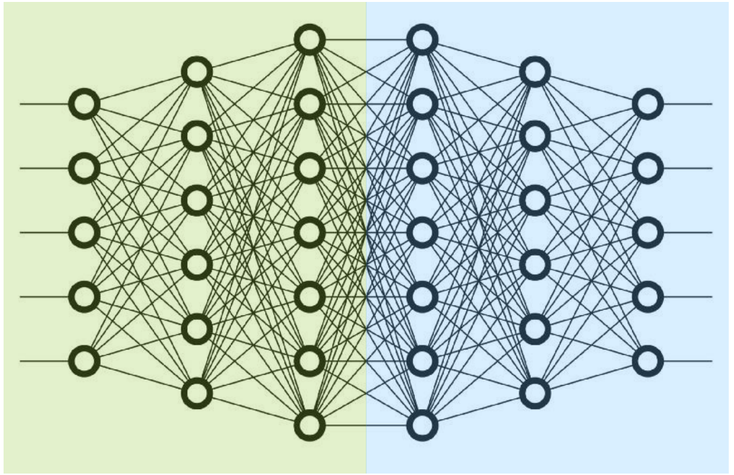
Prefix		Base 10
Name	Symbol	
yotta	Y	10^{24}
zetta	Z	10^{21}
exa	E	10^{18}
peta	P	10^{15}
tera	T	10^{12}
giga	G	10^{9}
mega	M	10^{6}

MODEL PARALLELISM

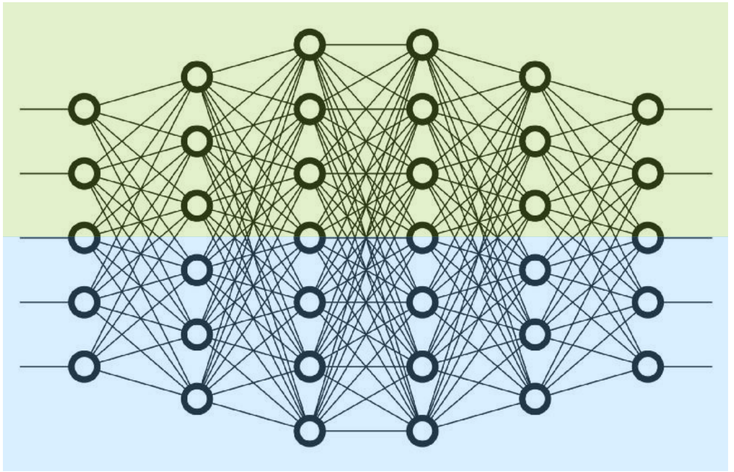
MODEL PARALLELISM

Complementary Types of 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



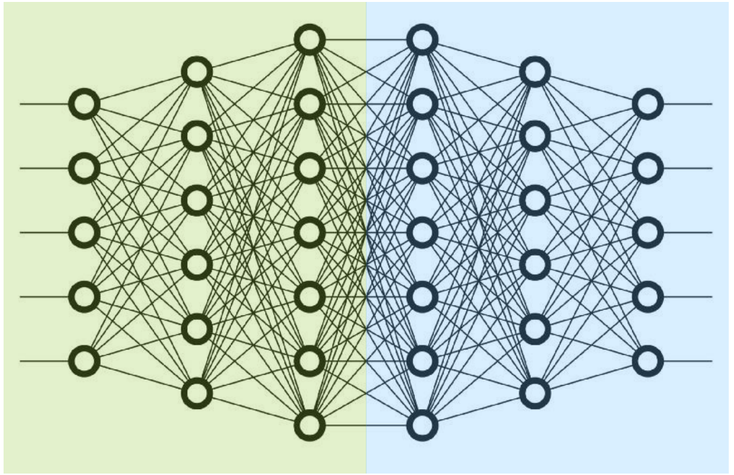
- Intra-Layer (Tensor) Parallelism
  - Split individual layers across multiple devices
  - Both devices compute different parts of Layer 0,1,2,3,4,5



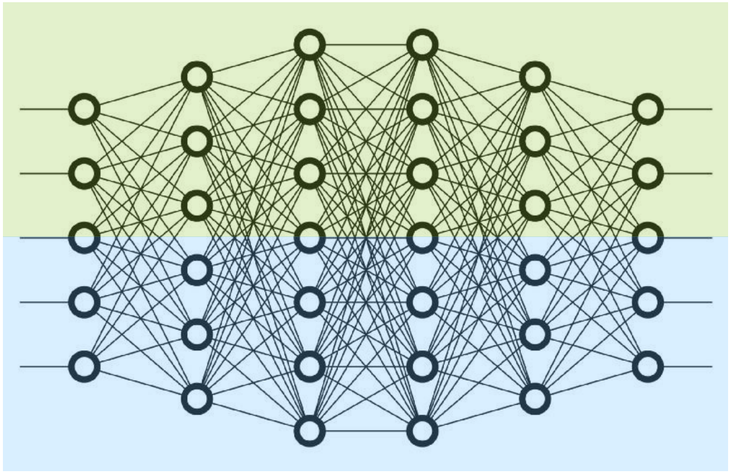
MODEL PARALLELISM

Complementary Types of 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

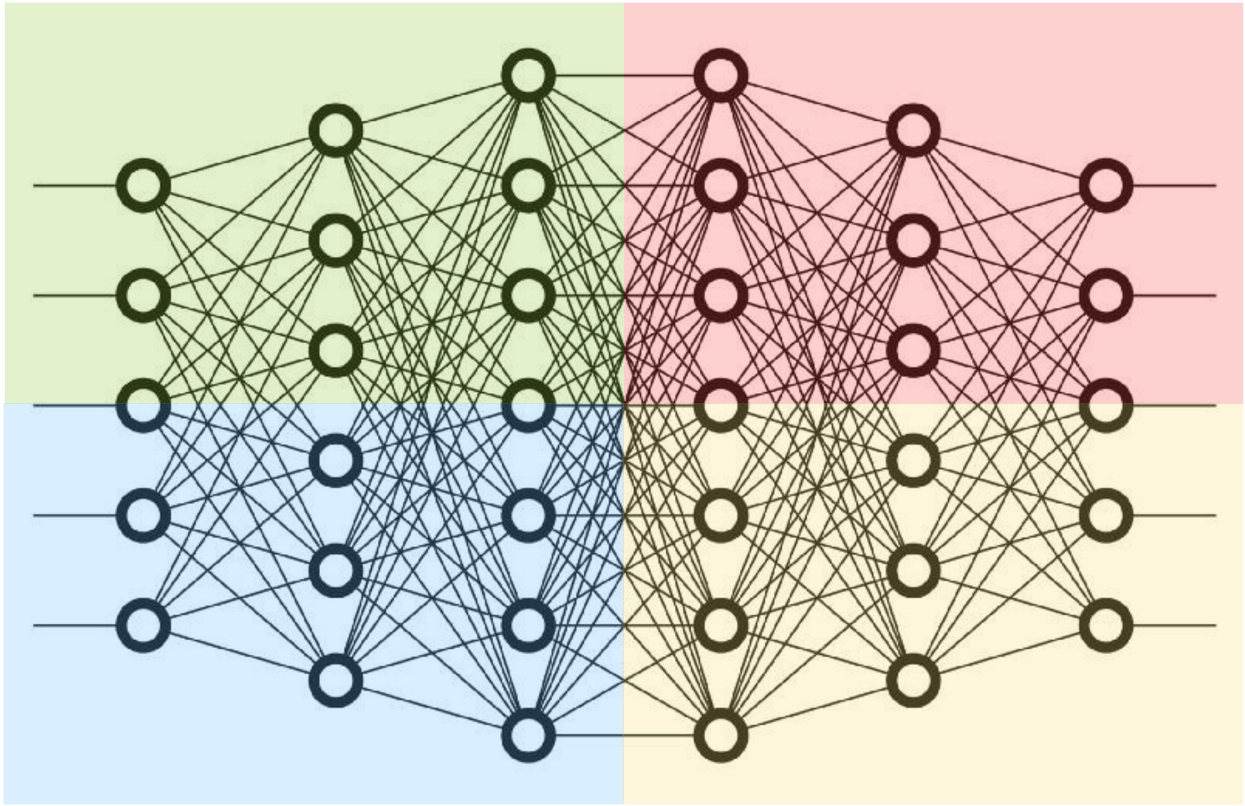


- Intra-Layer (Tensor) Parallelism
  - Split individual layers across multiple devices
  - Both devices compute different parts of Layer 0,1,2,3,4,5



MODEL PARALLELISM

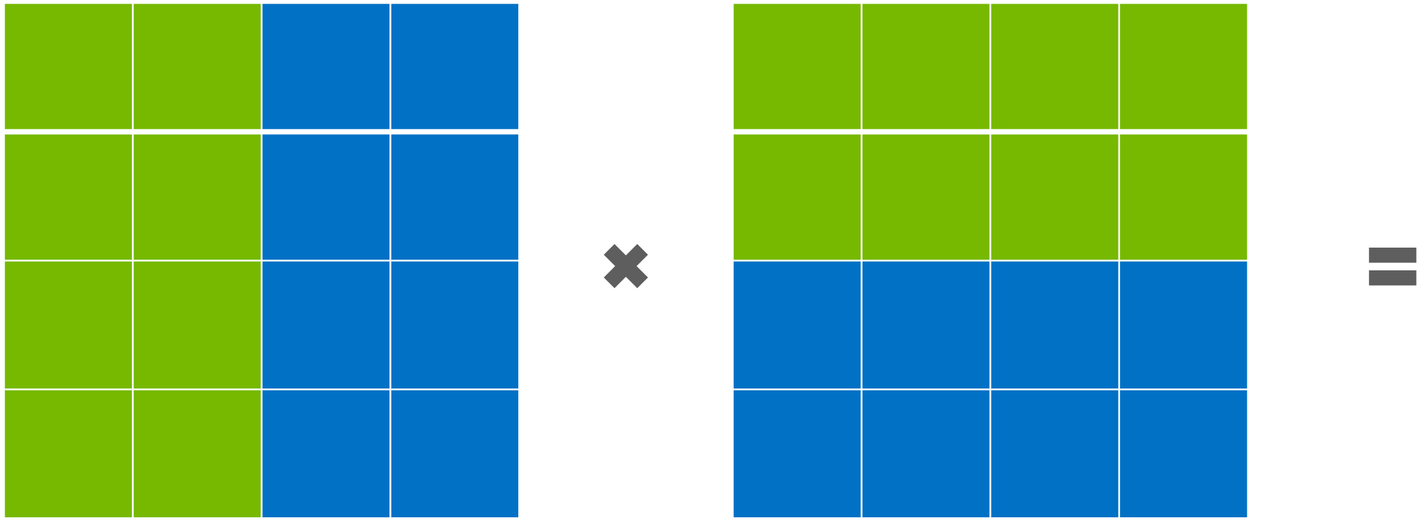
Complementary Types of Model Parallelism



Inter + Intra Parallelism

MODEL PARALLELISM

Parallel GEMMs



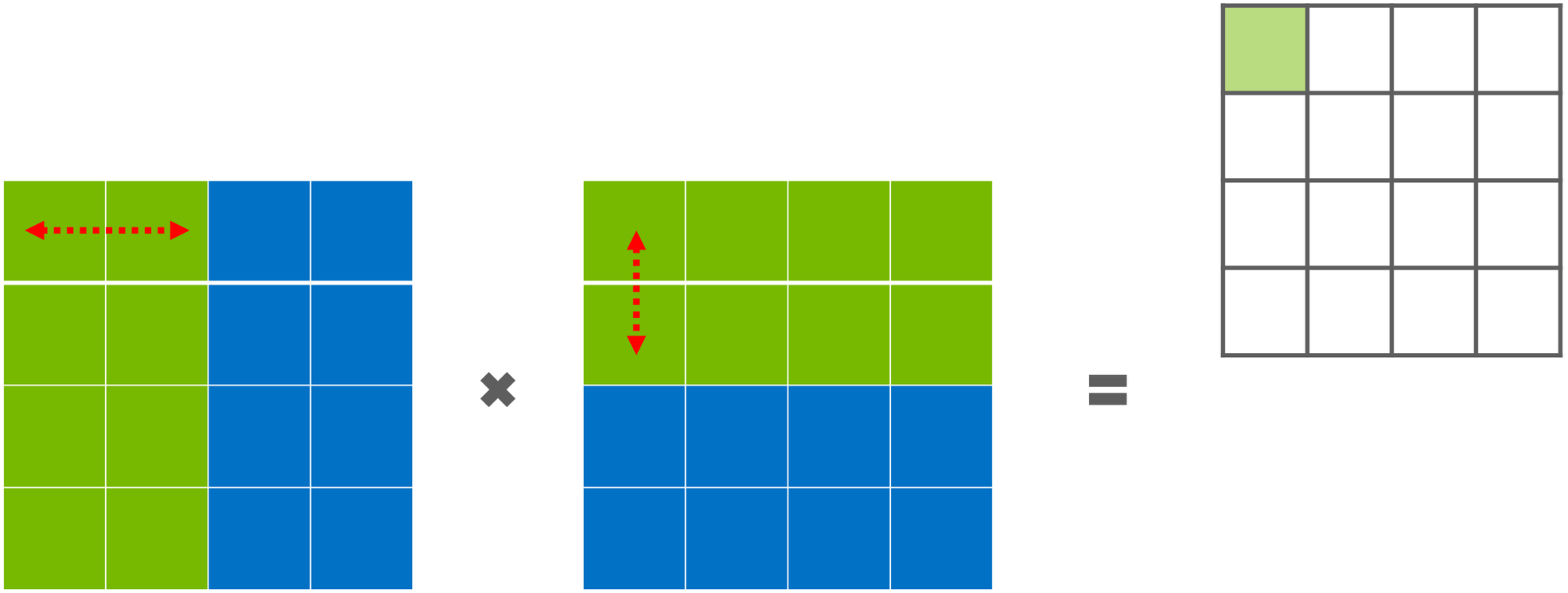
$$X = [X_1, X_2]$$

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

$$Y = Y_1 + Y_2$$

MODEL PARALLELISM

Parallel GEMMs



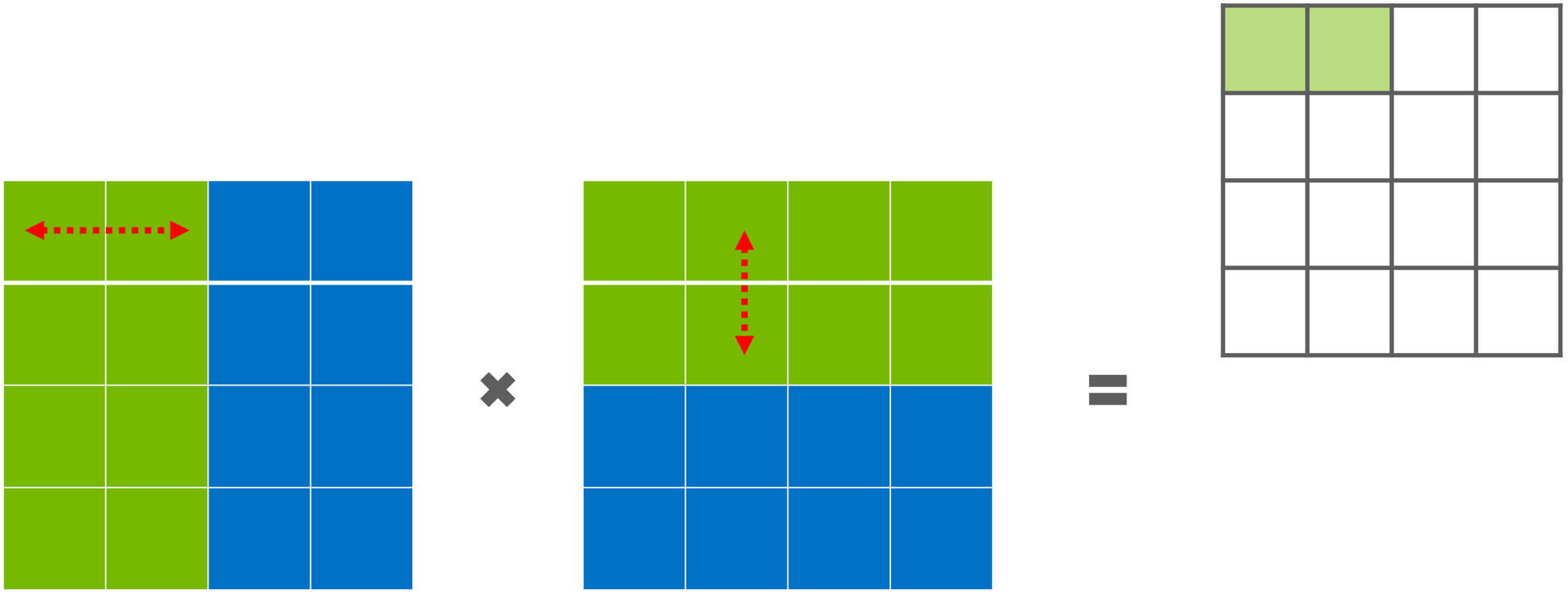
$$X = [X_1, X_2]$$

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

$$Y = Y_1 + Y_2$$

MODEL PARALLELISM

Parallel GEMMs



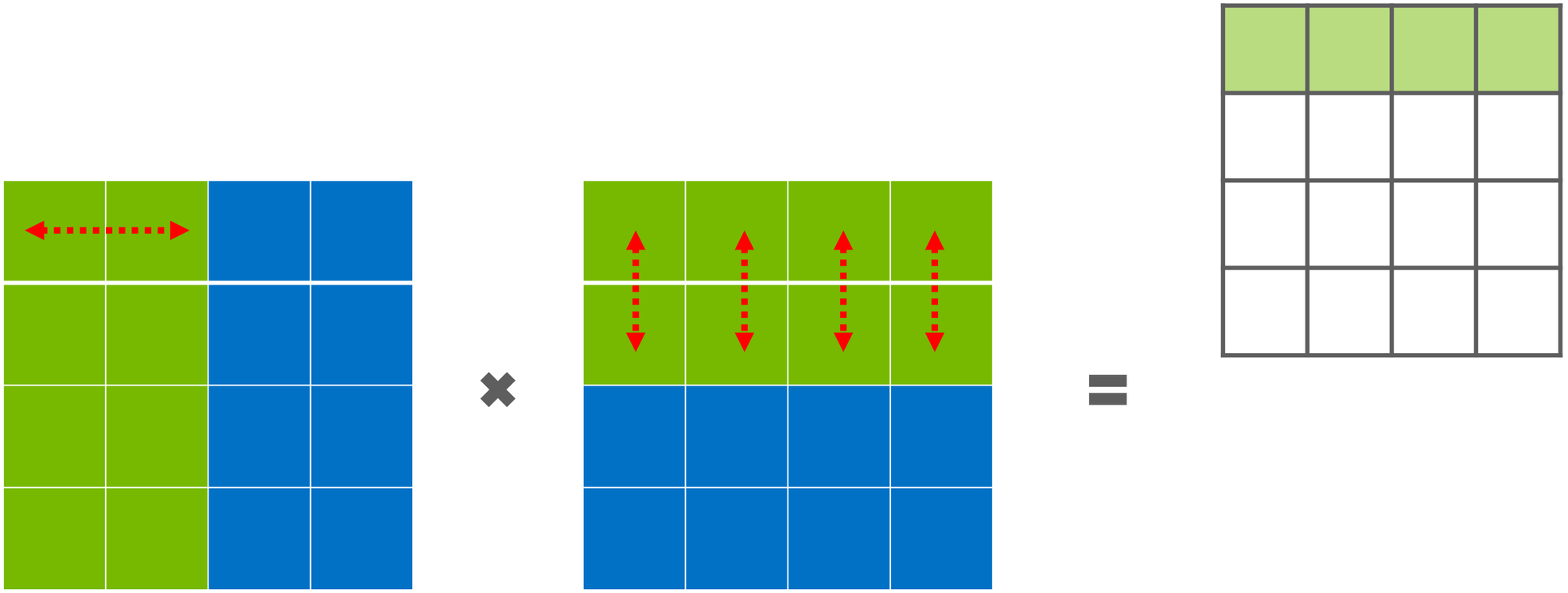
$$X = [X_1, X_2]$$

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

$$Y = Y_1 + Y_2$$

MODEL PARALLELISM

Parallel GEMMs



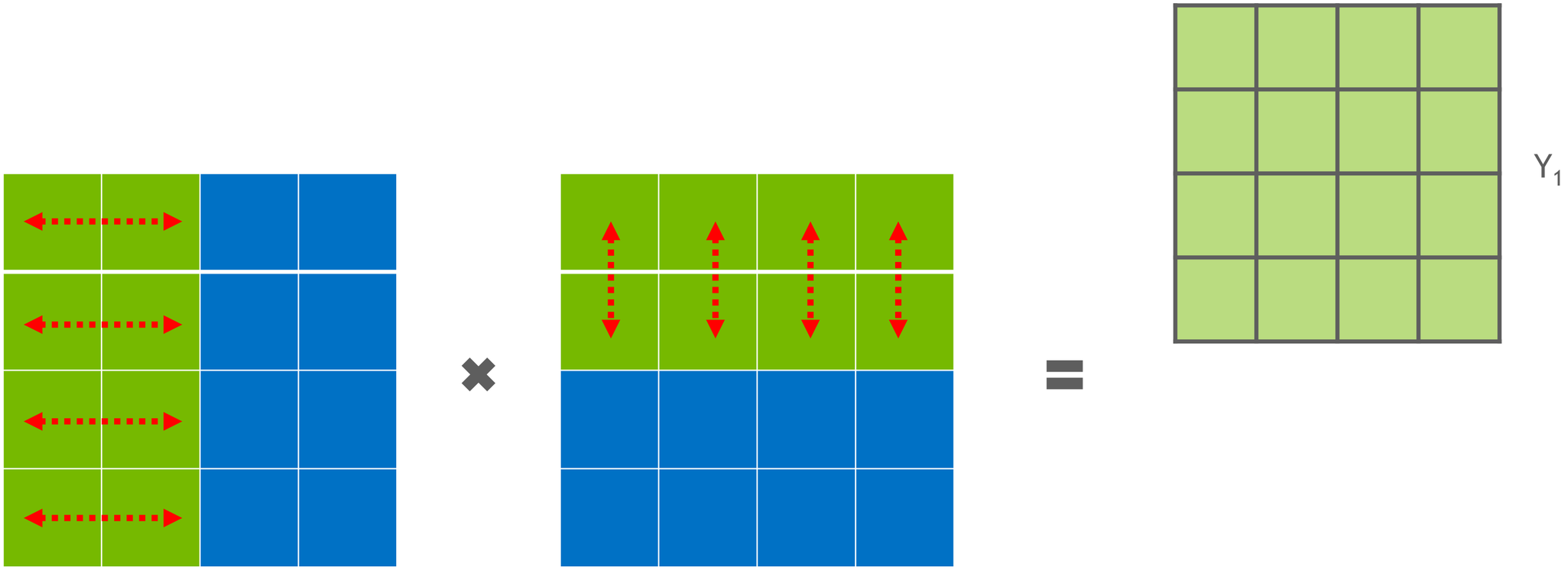
$$X = [X_1, X_2]$$

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

$$Y = Y_1 + Y_2$$

MODEL PARALLELISM

Parallel GEMMs



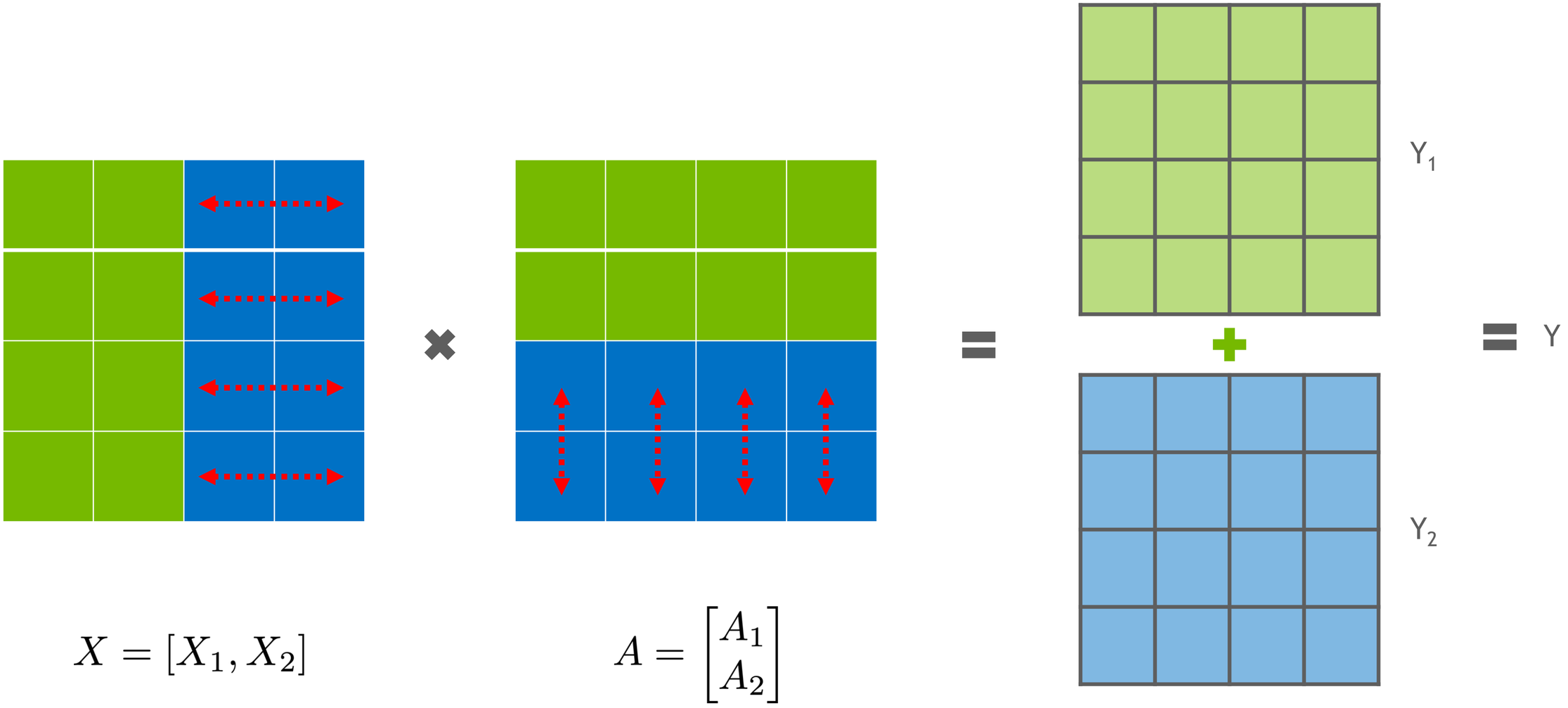
$$X = [X_1, X_2]$$

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

$$Y = Y_1 + Y_2$$

MODEL PARALLELISM

Parallel GEMMs



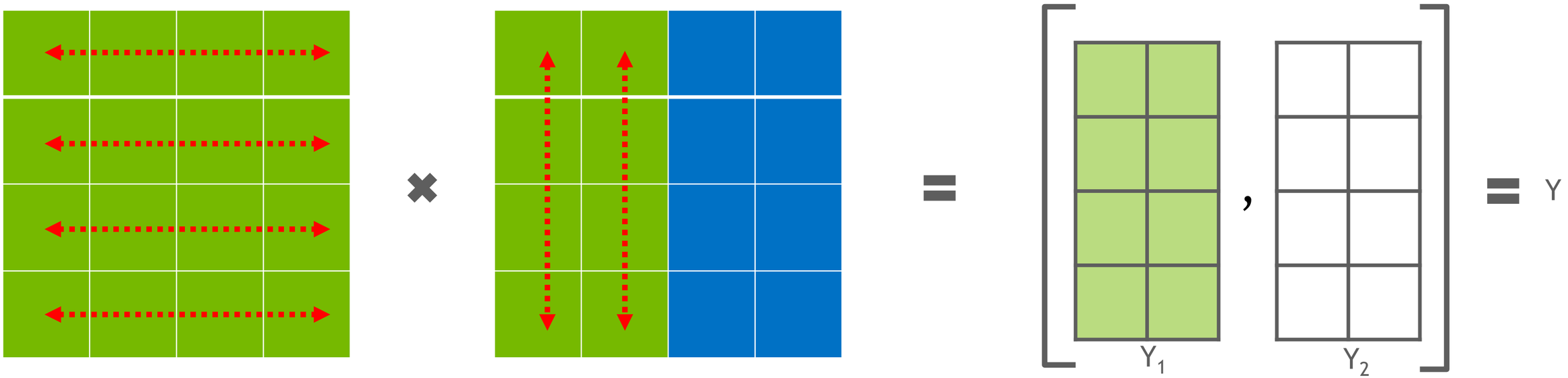
$$X = [X_1, X_2]$$

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

$$Y = Y_1 + Y_2$$

MODEL PARALLELISM

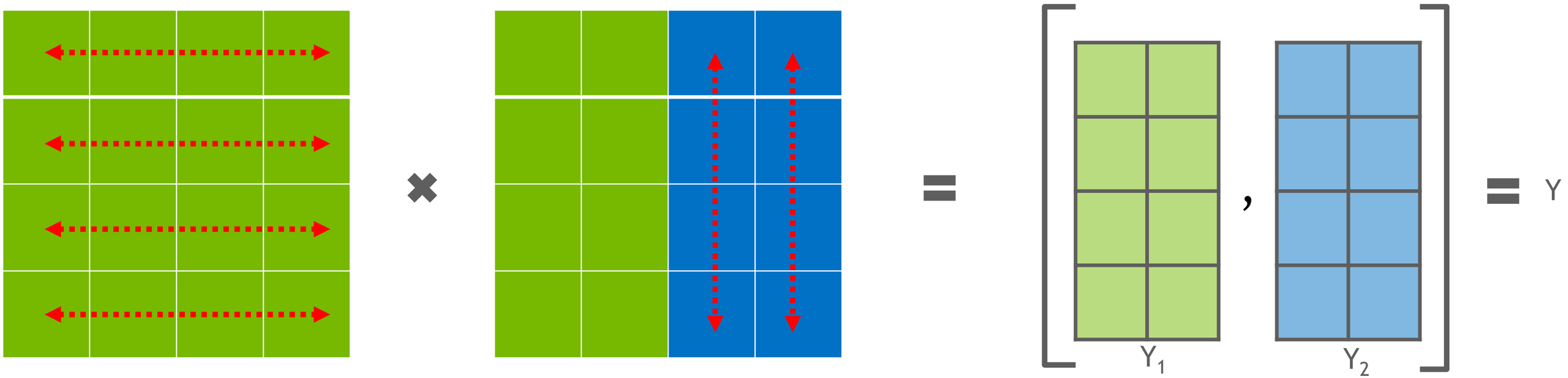
Parallel GEMMs



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

MODEL PARALLELISM

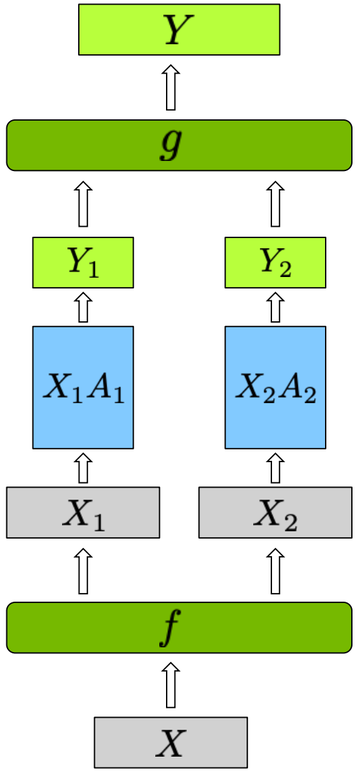
Parallel GEMMs



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

MODEL PARALLELISM

Row Parallel Linear Layer

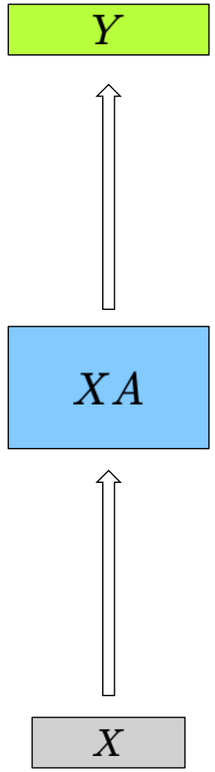


g : forward: $Y = Y_1 + Y_2$ (all-reduce)
backward: $\frac{\partial L}{\partial Y_i} = \frac{\partial L}{\partial Y}$ (identity)

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

$$X = [X_1, X_2]$$

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)

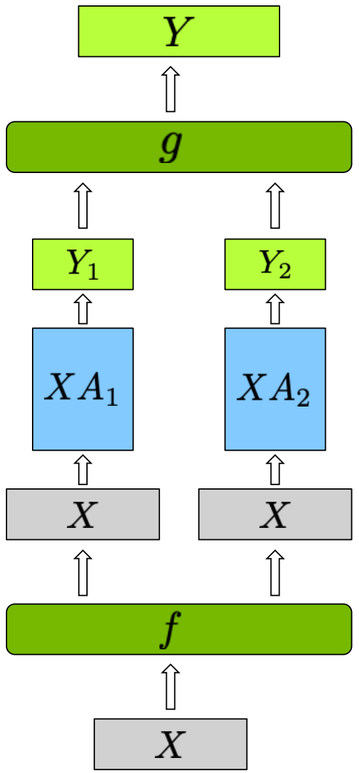


Column Parallel Linear Layer

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)



MODEL PARALLELISM

Challenges & Opportunities

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: $1/p$ 

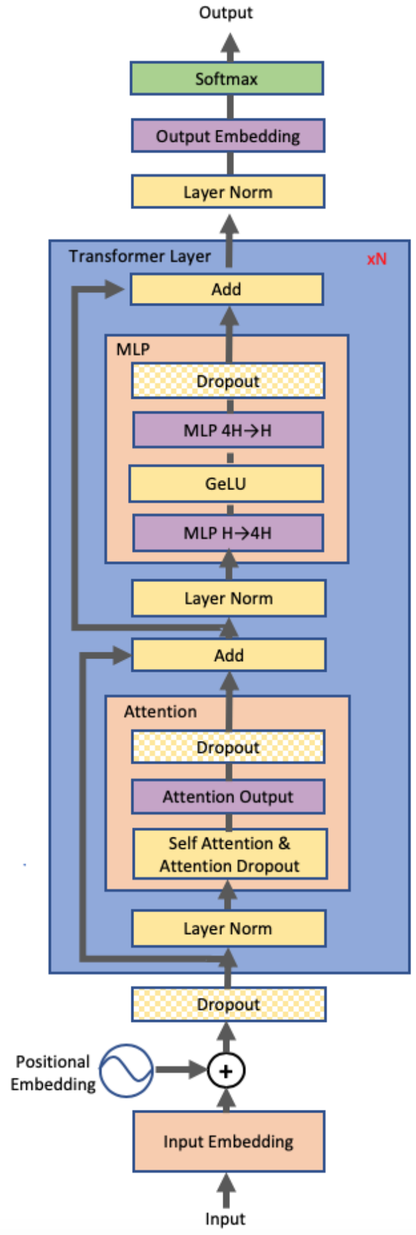
Bandwidth: $\frac{1+2/p}{3}$ 

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

APPROACH

Transformer Goals

- Group math heavy operations (such as GEMMs) to minimize parallel sync points
- Develop an approach that can be fully implemented with the insertion of a few simple collectives
  - Rely on pre-existing NCCL/PyTorch operations for a native PyTorch implementation
- Use Volta's tensor cores for mixed precision training



APPROACH

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

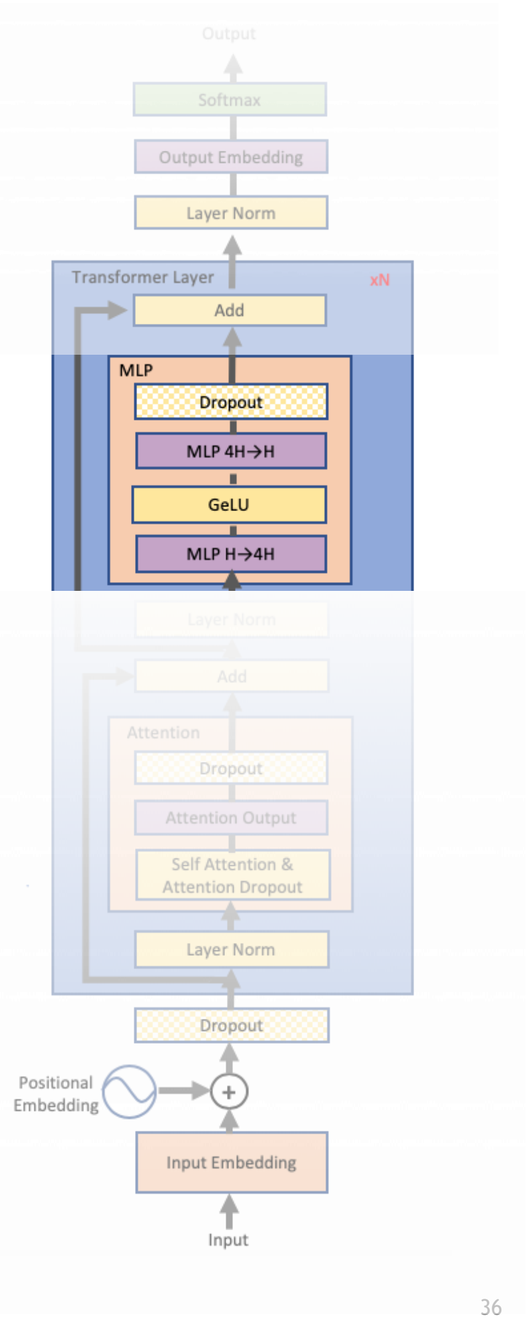
GeLU of sums != sum of GeLUs

- 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

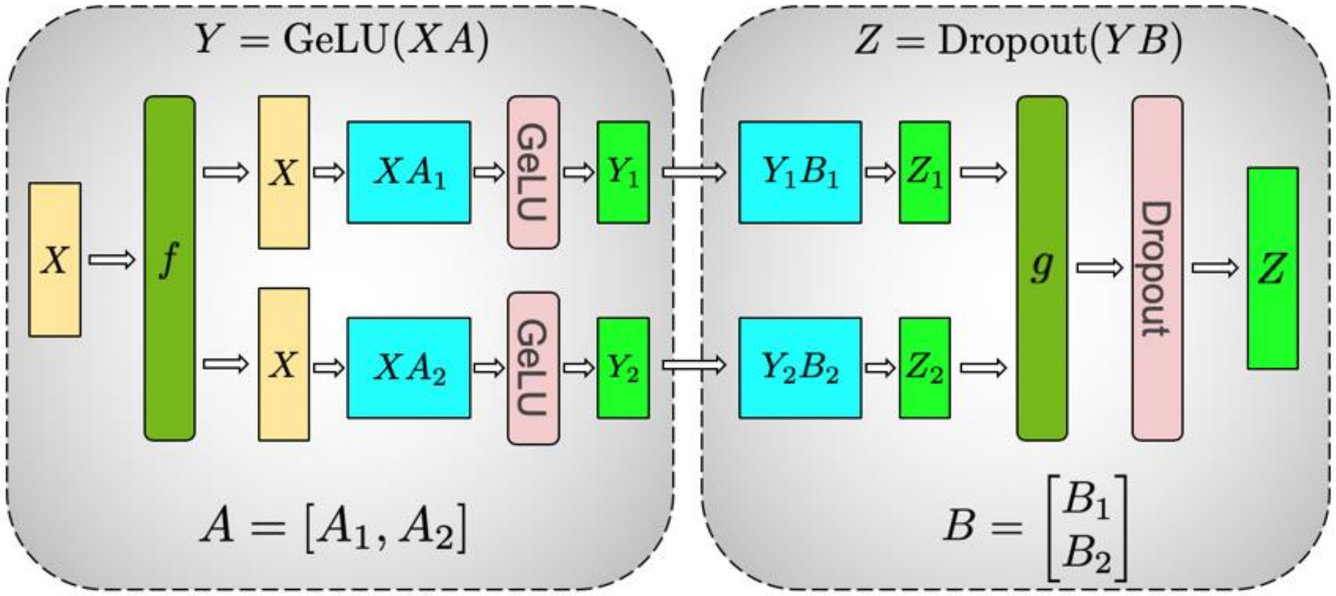
- Gather/all-reduce rely on pre-existing NCCL/PyTorch operations for a native PyTorch implementation 👍

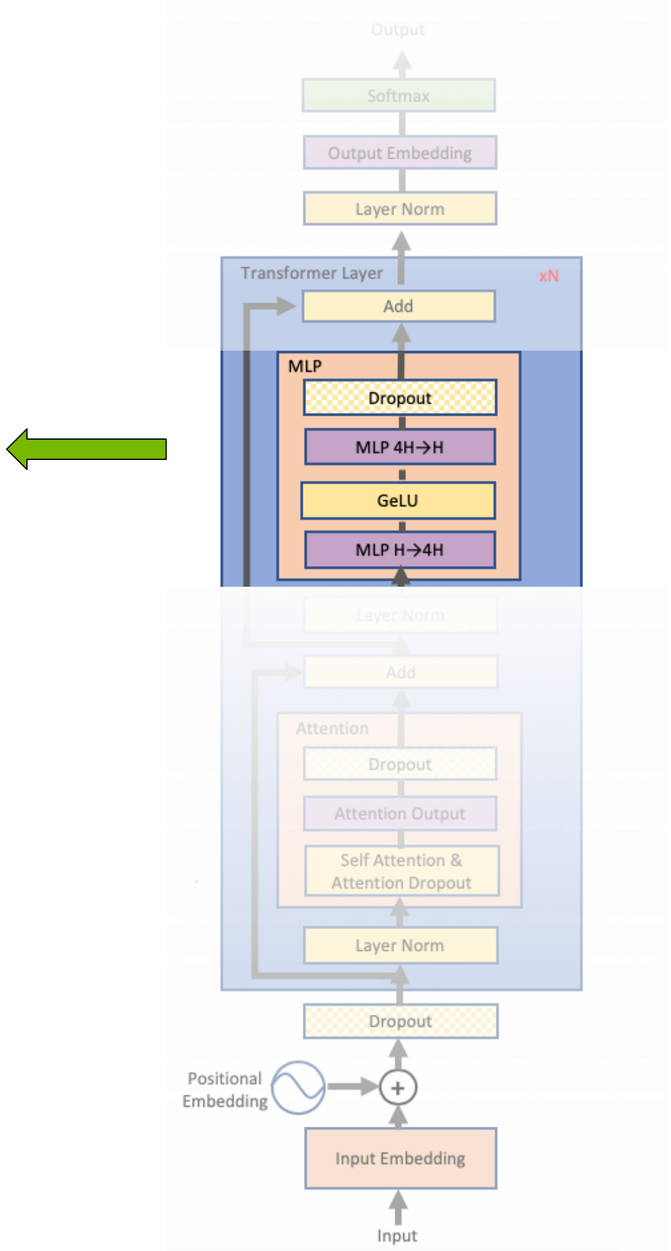


NVIDIA

APPROACH

Fused MLP



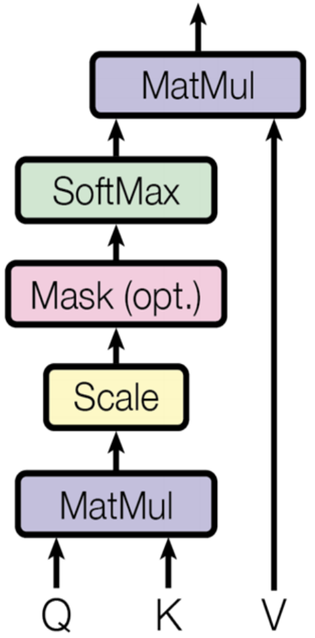


f and ***g*** are **conjugate**, ***f*** is **identity** operator in the forward pass and **all-reduce** in the backward pass while ***g*** is **all-reduce** in forward and **identity** in backward.

APPROACH

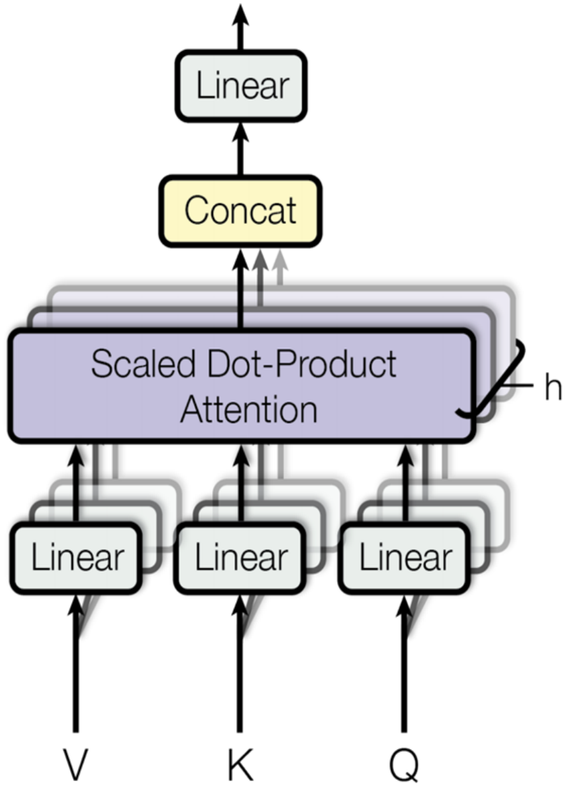
Fused Self-Attention

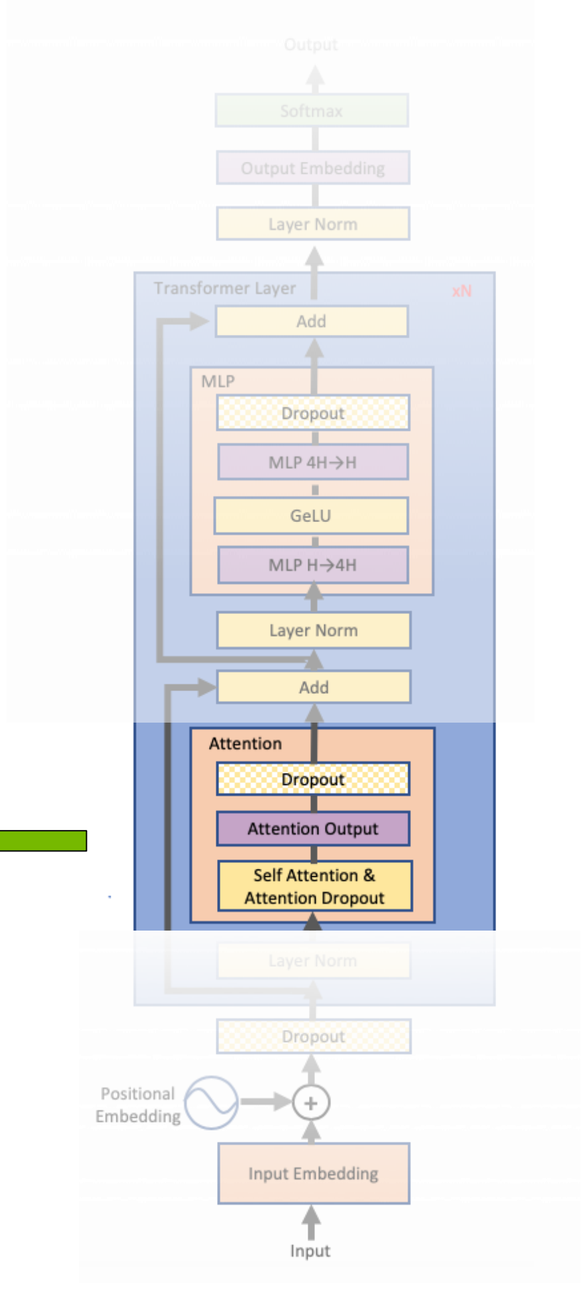
Scaled Dot-Product Attention



Nonlinearity so no Row-Parallel

Multi-Head Attention

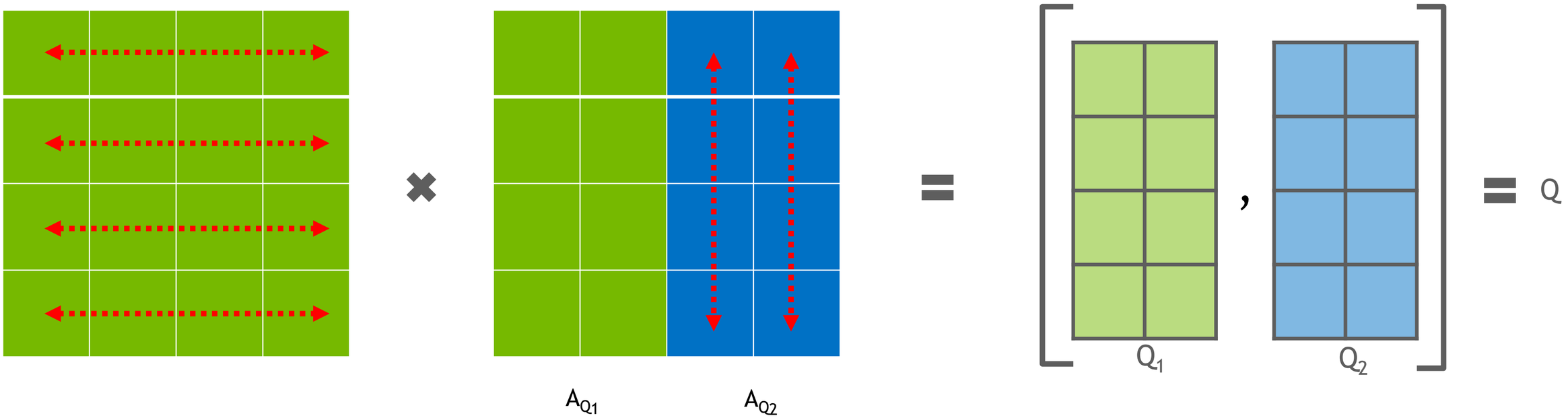




NVIDIA

APPROACH

Fused Self-Attention

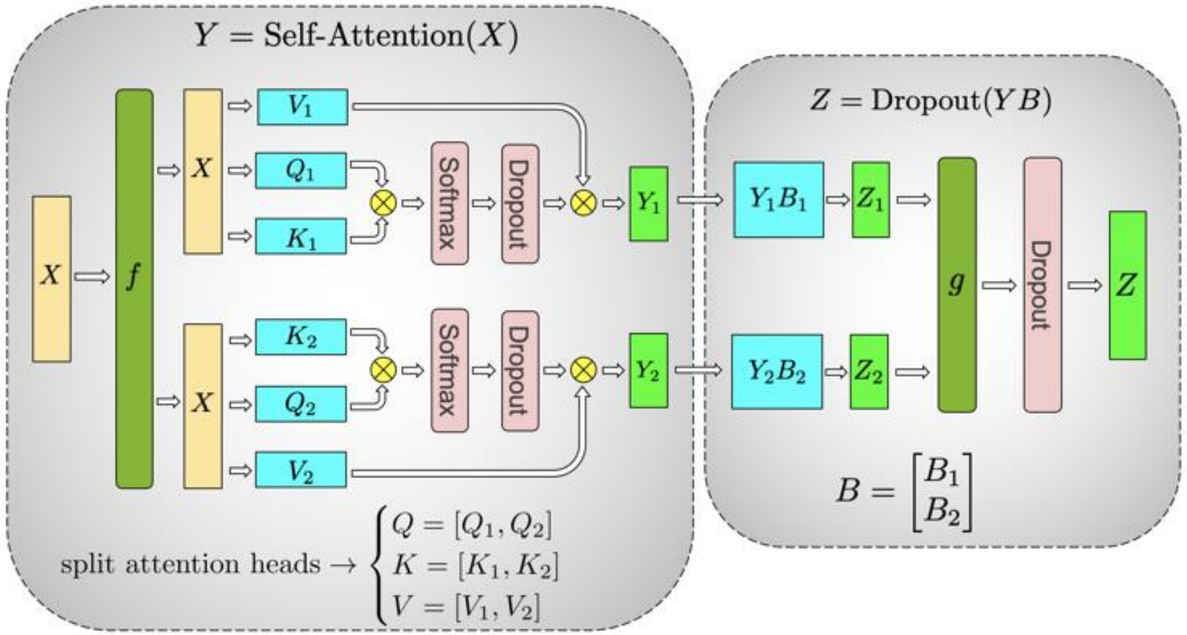


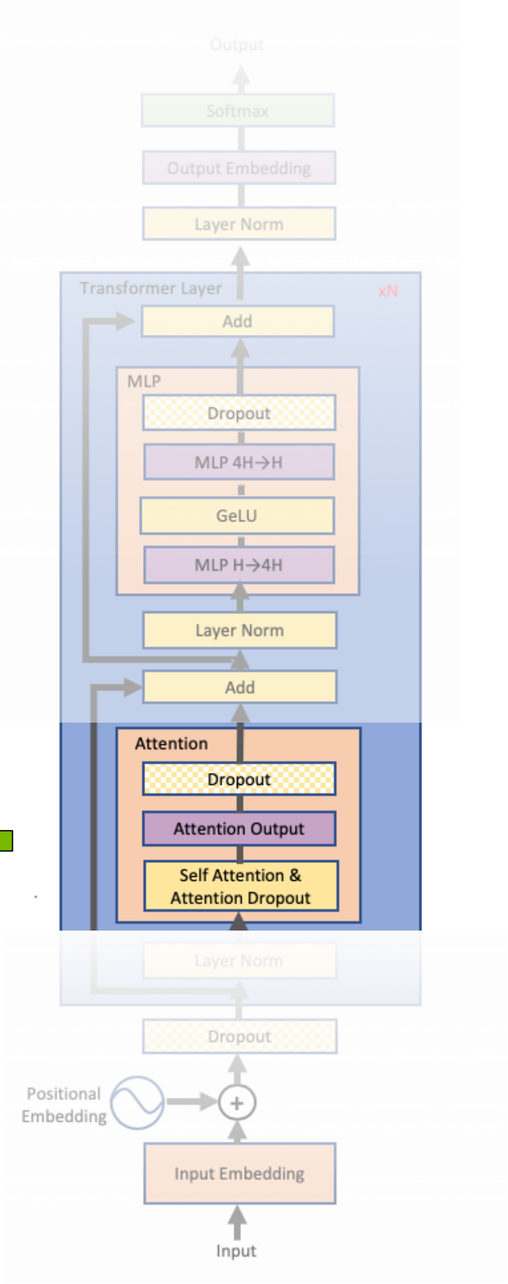
X $\quad A = [A_1, A_2]$ $\quad Y = [Y_1, Y_2]$

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

APPROACH

Fused Self-Attention

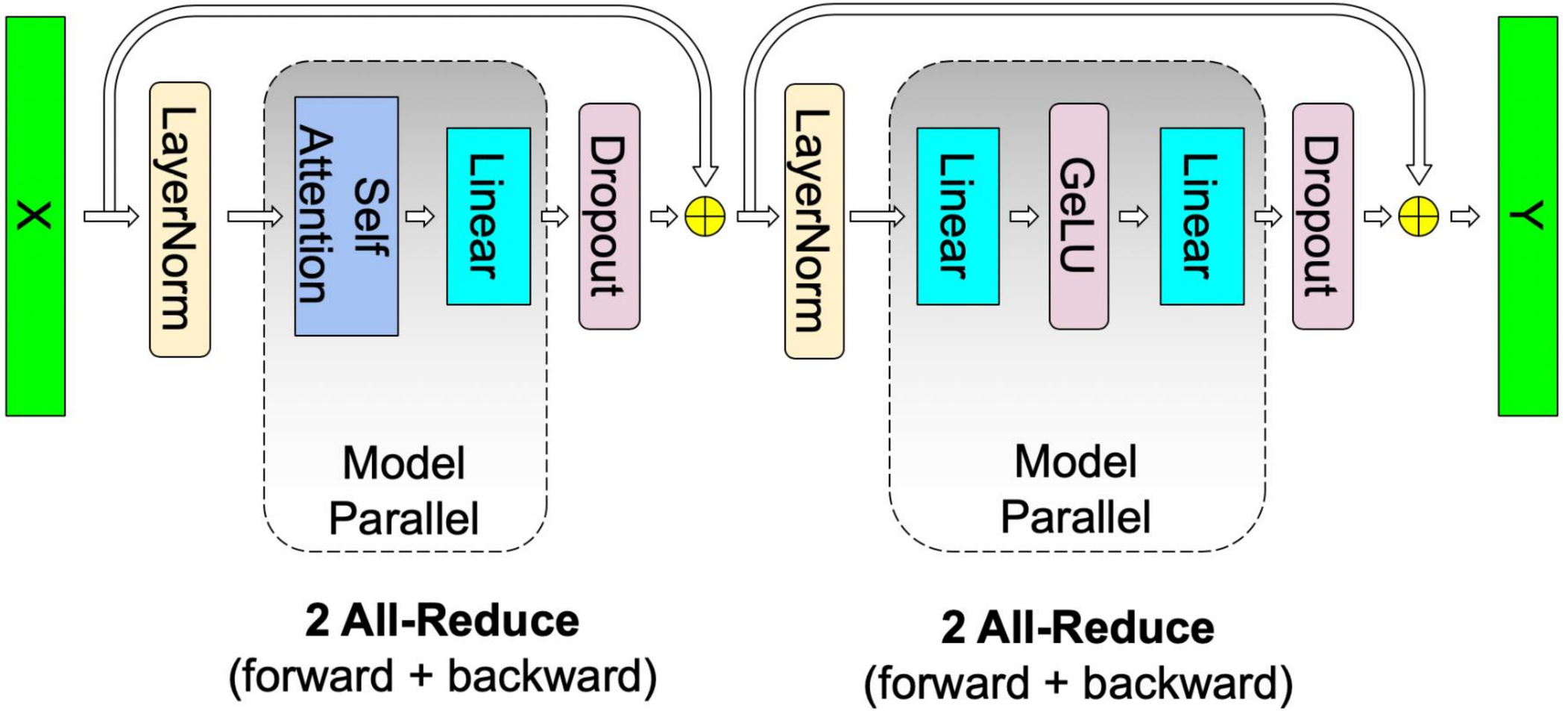




f and ***g*** are **conjugate**, ***f*** is **identity** operator in the forward pass and **all-reduce** in the backward pass while ***g*** is **all-reduce** in forward and **identity** in backward.

APPROACH

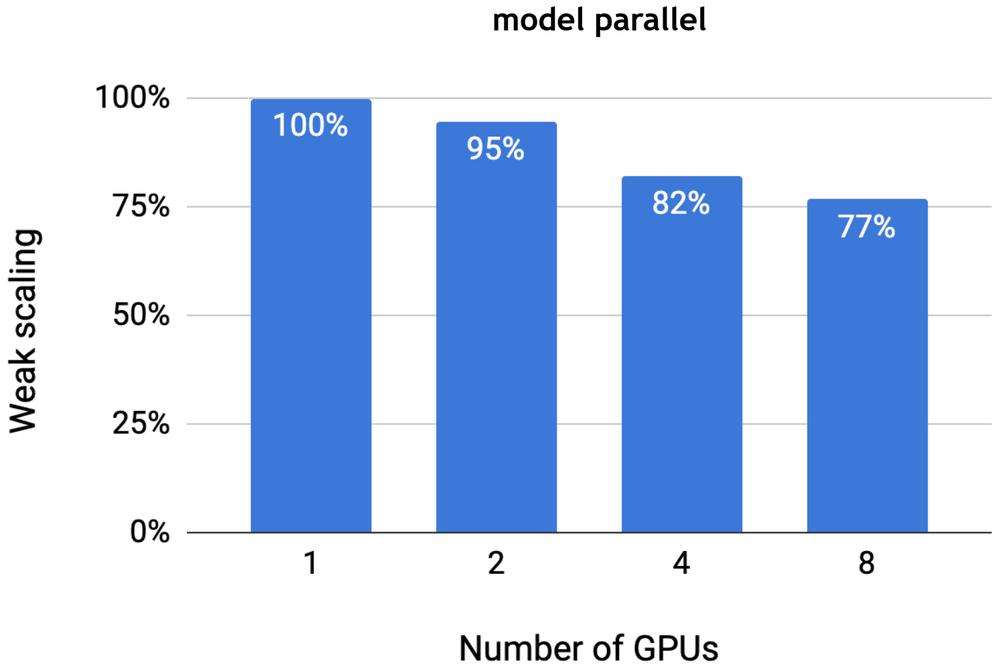
Putting It All Together: Parallel Transformer Layer

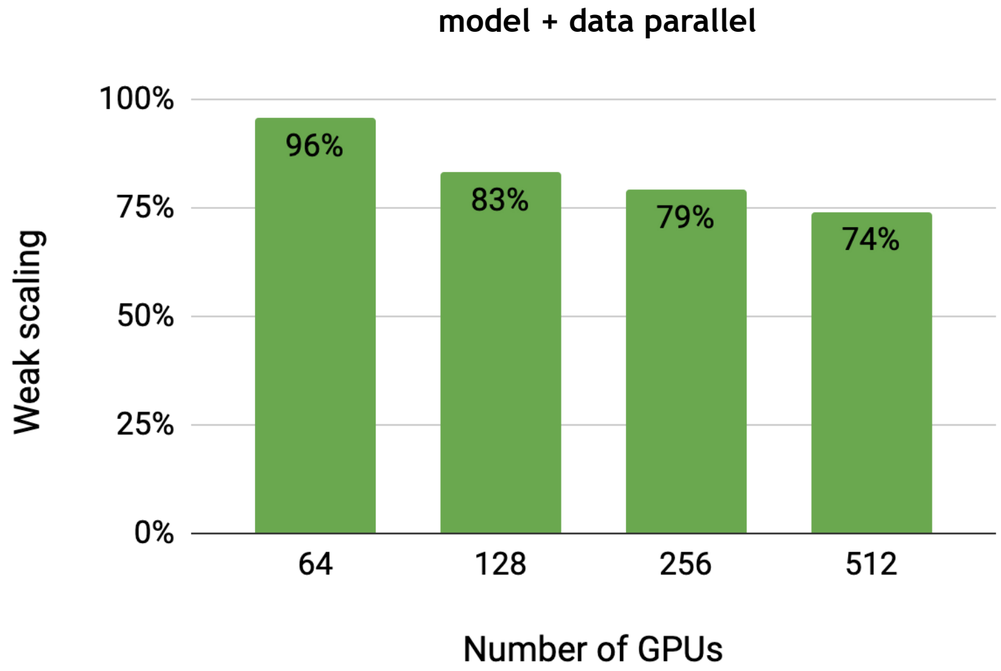


APPROACH

Weak Scaling

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



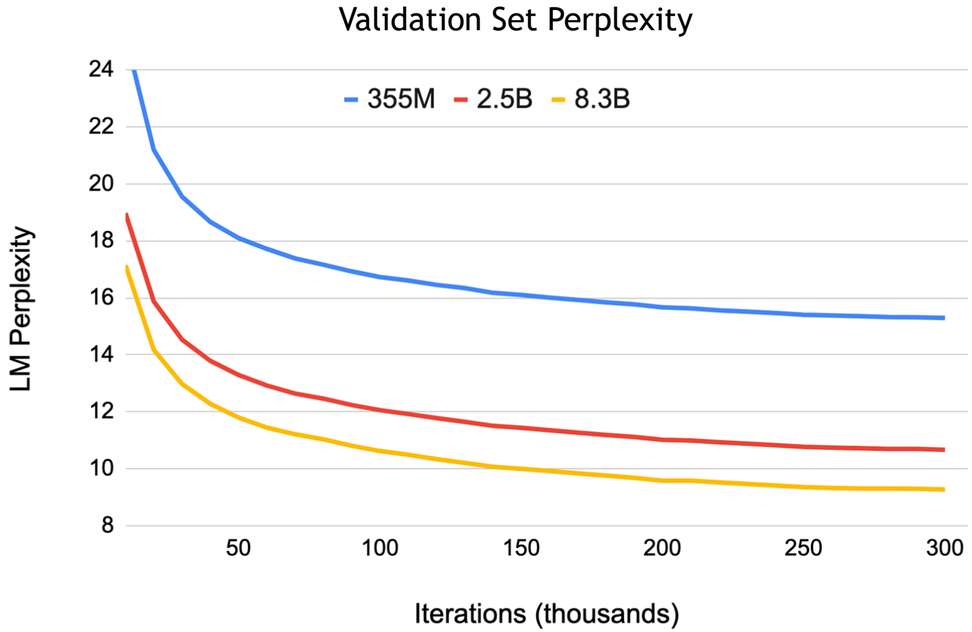


Baseline (1.2B parameters on a single GPU) achieves **39 TeraFLOPs per second**, i.e. **30% of the theoretical peak** during the entire training process

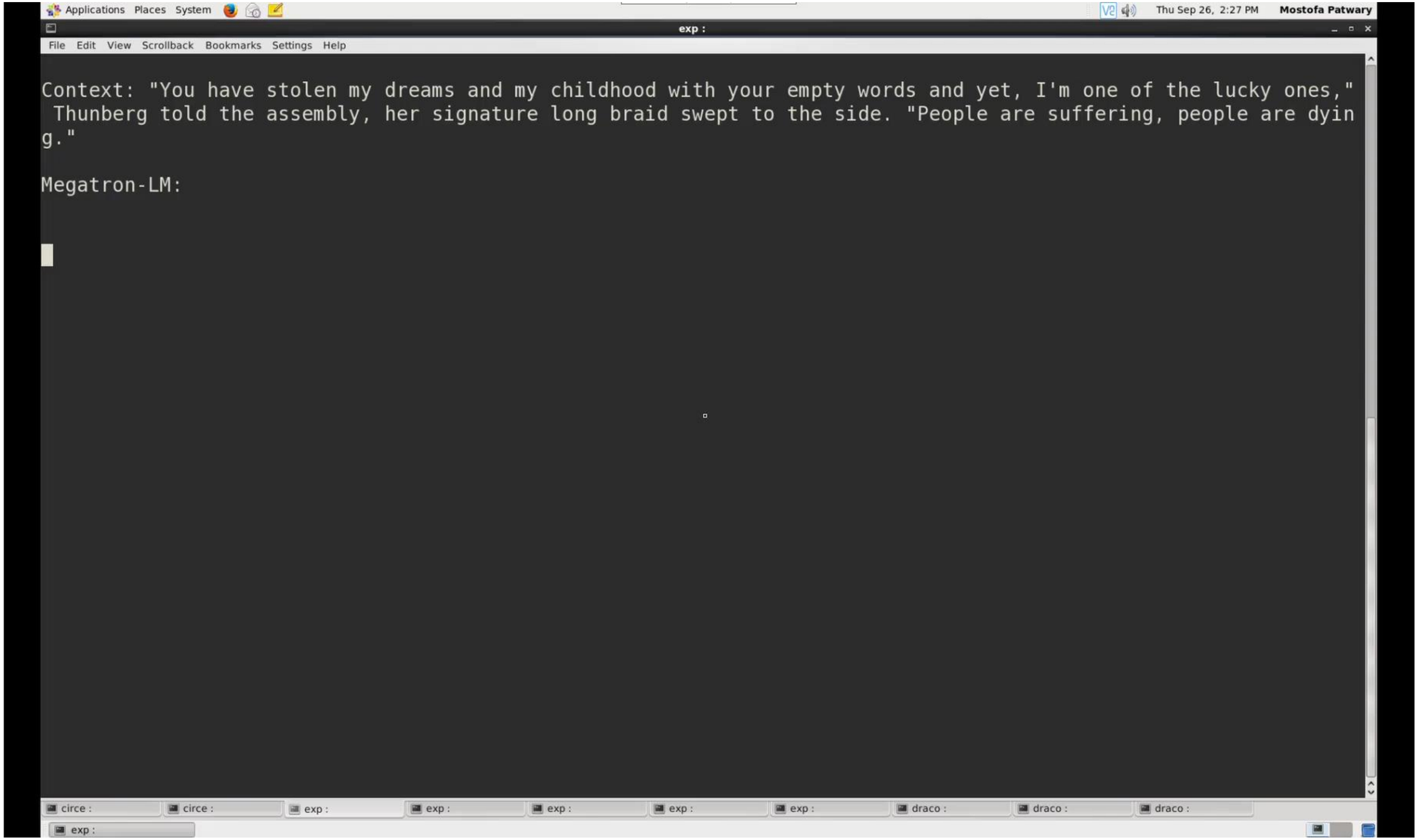
APPROACH

SOTA GPT-2 Results

- 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**

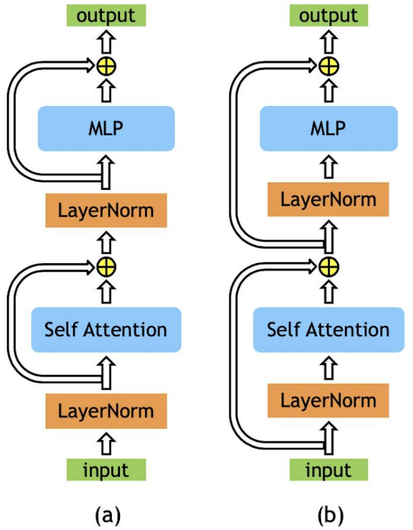


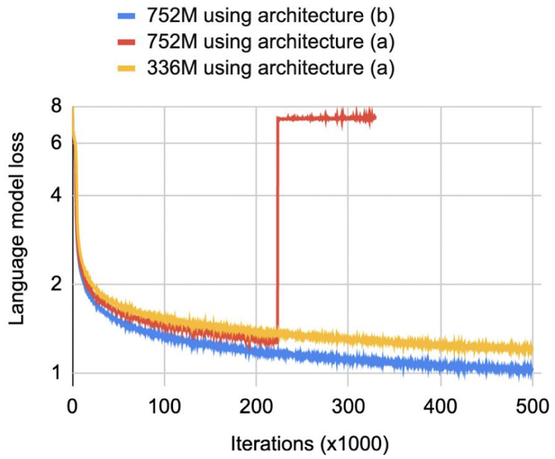
Text Generation: 8.3B parameter model knows lots of fact associations

MEGATRON-BERT

Training The World's Largest BERT Model

- Unlike prior work we find that scaling BERT to larger sizes is possible.
- Training the world's largest BERT model requires reordering residual connections to stabilize training.
- We trained Megatron-BERT-3.9B, at 12x the size of BERT-Large, over 2 million iterations @ batch size 1024.





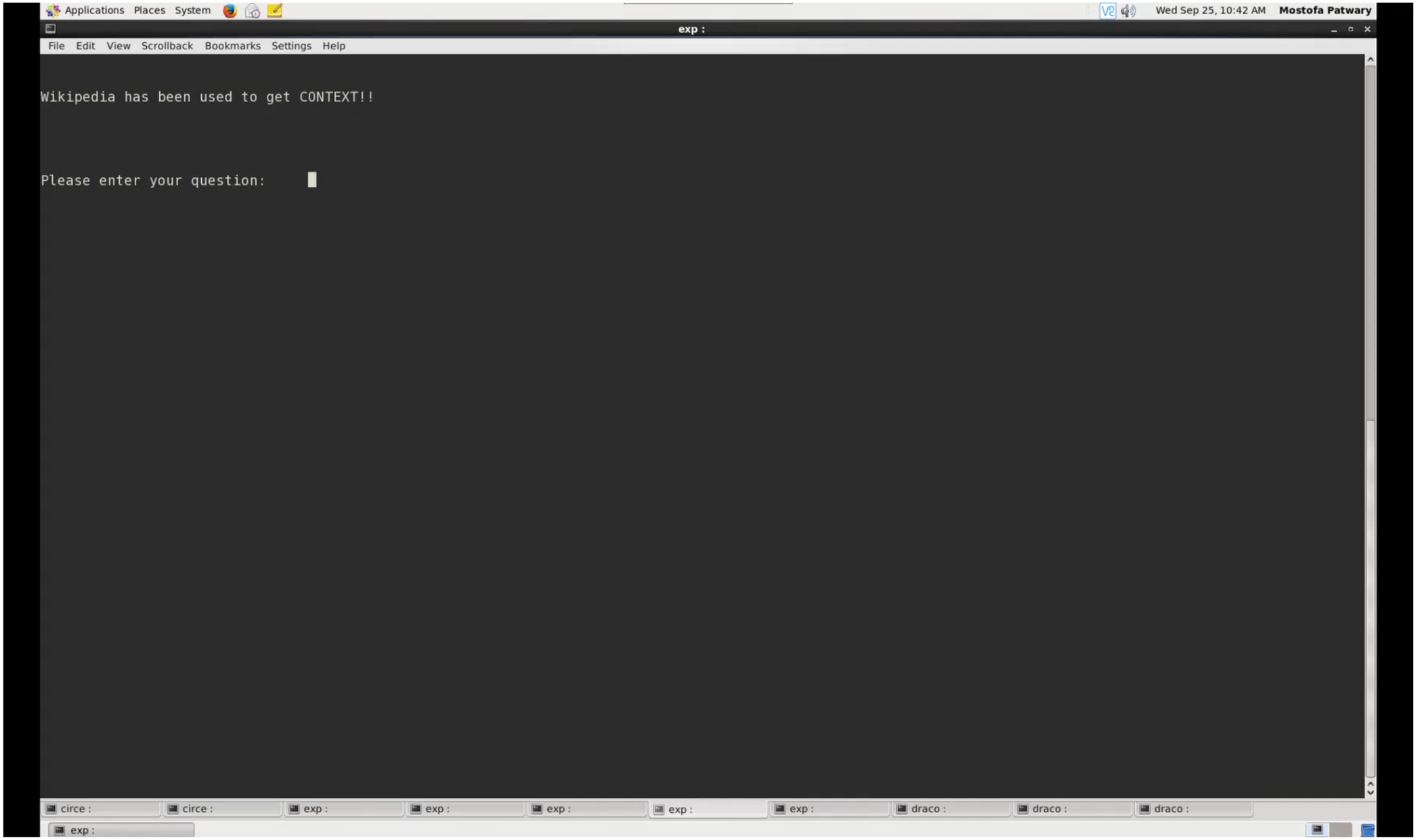
Parameter Count	Parameter Multipler	Hidden Size	Attention Heads	Layers	Model Parallel GPUs	Model + Data Parallel GPUs
334M	1x	1024	16	24	1	128
1.3B	4x	2048	32	24	1	256
3.9B	12x	2506	40	48	4	512

SQUAD & RACE

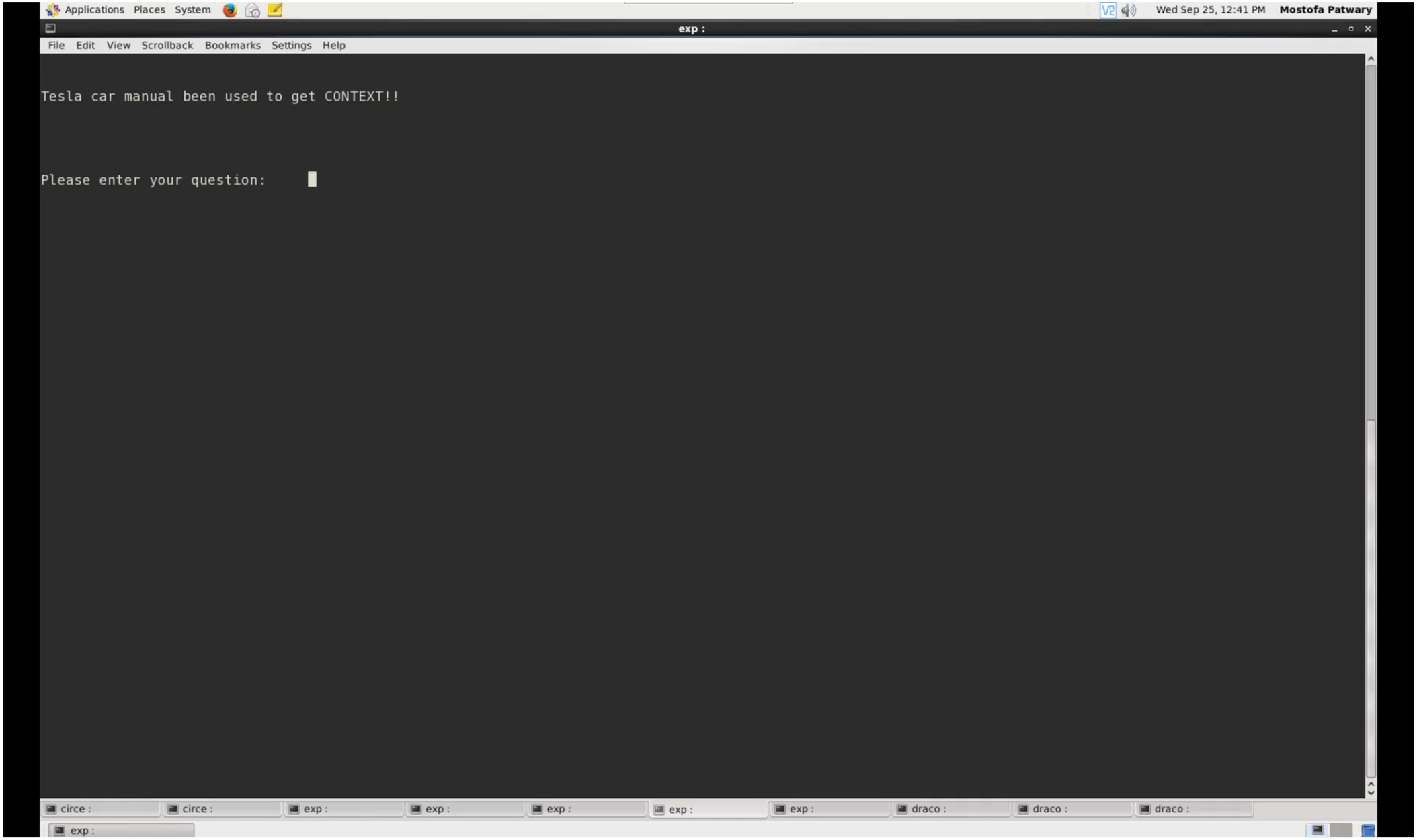
Model	Trained tokens (ratio)	MNLI† m/mm accuracy	QQP† accuracy	SQuAD 1.1† F1/EM	SQuAD 2.0† F1/EM	RACE m/h* accuracy
RoBERTa	2	90.2 / 90.2	92.2	94.6 / 88.9	89.4 / 86.5	86.5 / 81.3
ALBERT	3	90.8	92.2	94.8 / 89.3	90.2 / 87.4	89.0 / 85.5
XLNet	2	90.8 / 90.8	92.3	95.1 / 89.7	90.6 / 87.9	88.6 / 84.0
Megatron-334M	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.

READING COMPREHENSION



Question Answering: Over Wikipedia Knowledge Base



Question Answering: Over Tesla Car Manual

TRAINING READING COMPREHENSION MODELS

Standard Practices

1. Collect and establish corpus
2. Collect queries over corpus
3. Collect labeled answers for query
4. Train a QA Model with supervision

Problems

1. Cost prohibitive labeling
2. Quality of labeling

QUESTION GENERATION

We Taught Transformers to…

1. Generate Text

$$\hat{c} \sim p(c)$$

Context: "I Got Mine" is a song by American rapper 50 Cent from his debut studio album "Get Rich or Die Tryin'" (2003). The song features a guest appearance from fellow New York City rapper Nas, who was also featured on the previous single from "Get Rich or Die Tryin'", "Hate Me Now".

Paper: https://arxiv.org/abs/2002.09599

QUESTION GENERATION

We Taught Transformers to…

1. Generate Text
$\hat{c} \sim p(c)$

Context: "I Got Mine" is a song by American rapper 50 Cent from his debut studio album "**Get Rich or Die Tryin'**" (2003). The song features a guest appearance from fellow New York City rapper Nas, who was also featured on the previous single from "Get Rich or Die Tryin'", "Hate Me Now".

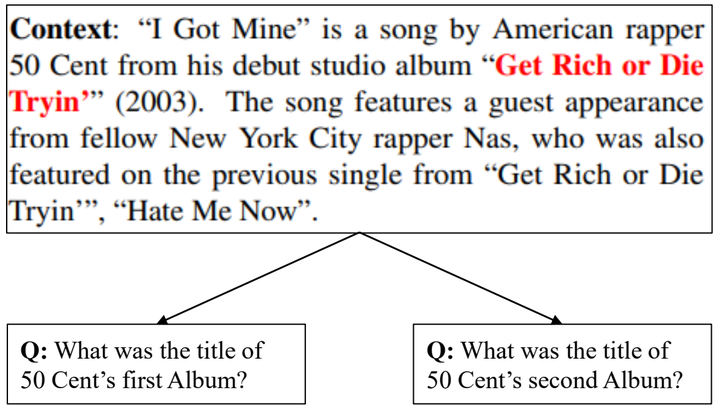
2. Extract Answers From Text
$\hat{a} \sim p(a|\hat{c})$

Paper: https://arxiv.org/abs/2002.09599

QUESTION GENERATION

We Taught Transformers to…

1. Generate Text
$\hat{c} \sim p(c)$



2. Extract Answers From Text
$\hat{a} \sim p(a|\hat{c})$

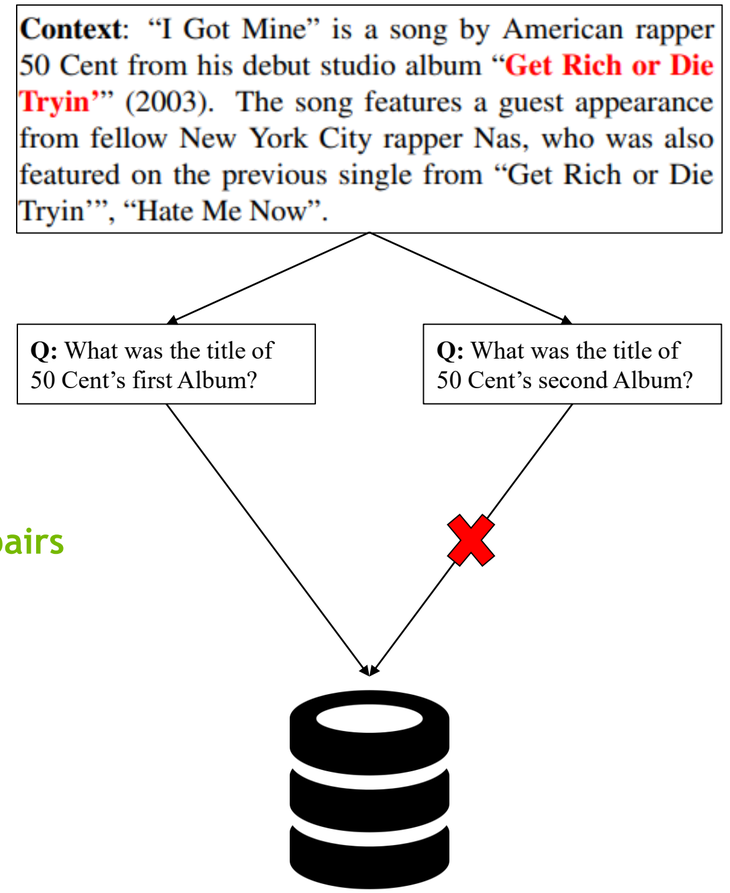
3. Pose Questions From Answers
$\hat{q} \sim p(q|\hat{a}, \hat{c})$

Paper: https://arxiv.org/abs/2002.09599

QUESTION GENERATION

We Taught Transformers to…

1. Generate Text
$\hat{c} \sim p(c)$



2. Extract Answers From Text
$\hat{a} \sim p(a|\hat{c})$

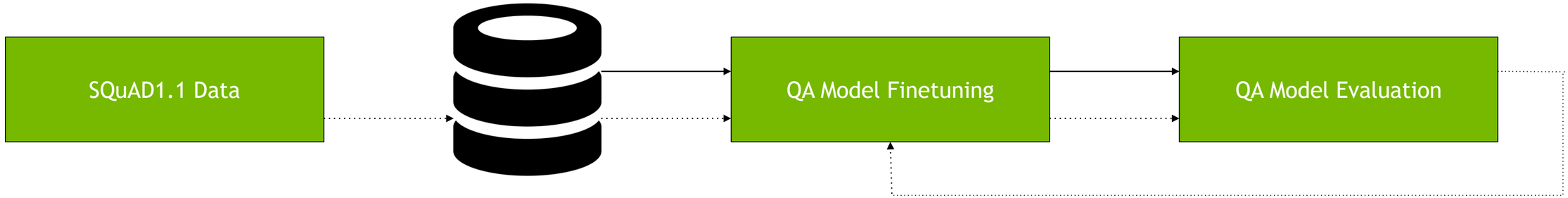
3. Pose Questions From Answers
$\hat{q} \sim p(q|\hat{a}, \hat{c})$

4. Filter out bad question answer pairs
$\hat{a} \stackrel{?}{=} a^* \sim p(a|\hat{c}, \hat{q})$

Paper: https://arxiv.org/abs/2002.09599

QUESTION GENERATION

Train SOTA Transformers with Synthetic Data



The corpus of **synthetically generated data can be used to finetune new SOTA QA models** and can be used in conjunction with ground truth human-labeled data.

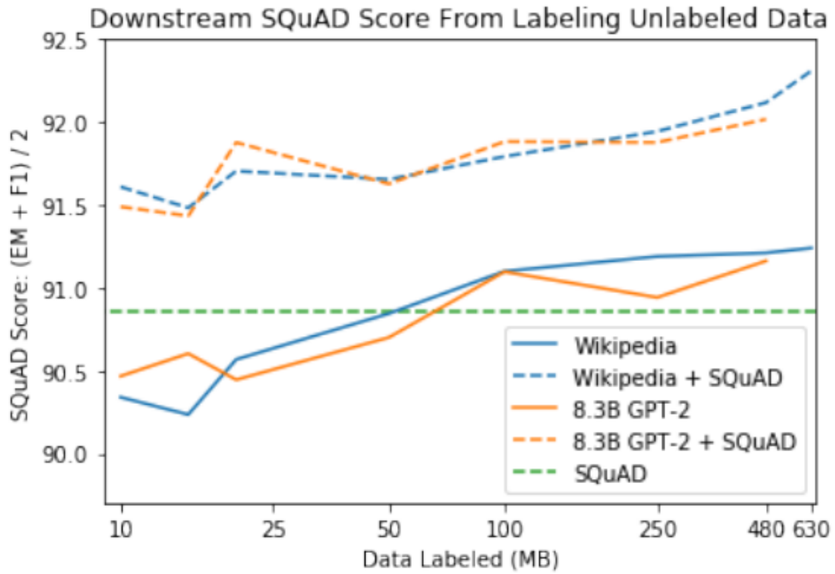
Paper: https://arxiv.org/abs/2002.09599

QUESTION GENERATION

More Synthetic Data is Better.

Using Only Synthetic Data Can Also Beat Using Real Data.

Text Source	Source Data Size	finetune data	# Questions	EM	F1
Wikipedia	638 MB	Synthetic	19,925,130	88.4	94.1
		+SQuAD	20,012,729	**89.4**	**95.2**
8.3B GPT-2	480 MB	Synthetic	17,400,016	88.4	93.9
		+SQuAD	17,487,615	**89.1**	**94.9**
SQuAD1.1	14MB	SQuAD	87,599	87.7	94.0



Using synthetic data from synthetic text outperforms finetuning on real SQuAD data. Finetuning on real data after finetuning on synthetic data boosts performance even further.

Paper: https://arxiv.org/abs/2002.09599

QUESTION GENERATION

Bigger Models = Better Questions

Question Generator	# Questions	EM	F1
117M	42345	76.6	85.0
345M (Klein & Nabi, 2019)	-	75.4	84.4
345M (w/ BERT QA model)	42414	76.6	84.8
345M	42414	80.7	88.6
768M	42465	81.0	89.0
1.2B	42472	83.4	90.9
8.3B	**42478**	**84.9**	**92.0**
Human Generated Data	42472	86.3	93.2

Ground Truth Answers are used to generate questions, with bigger models generating better questions and better QA models.

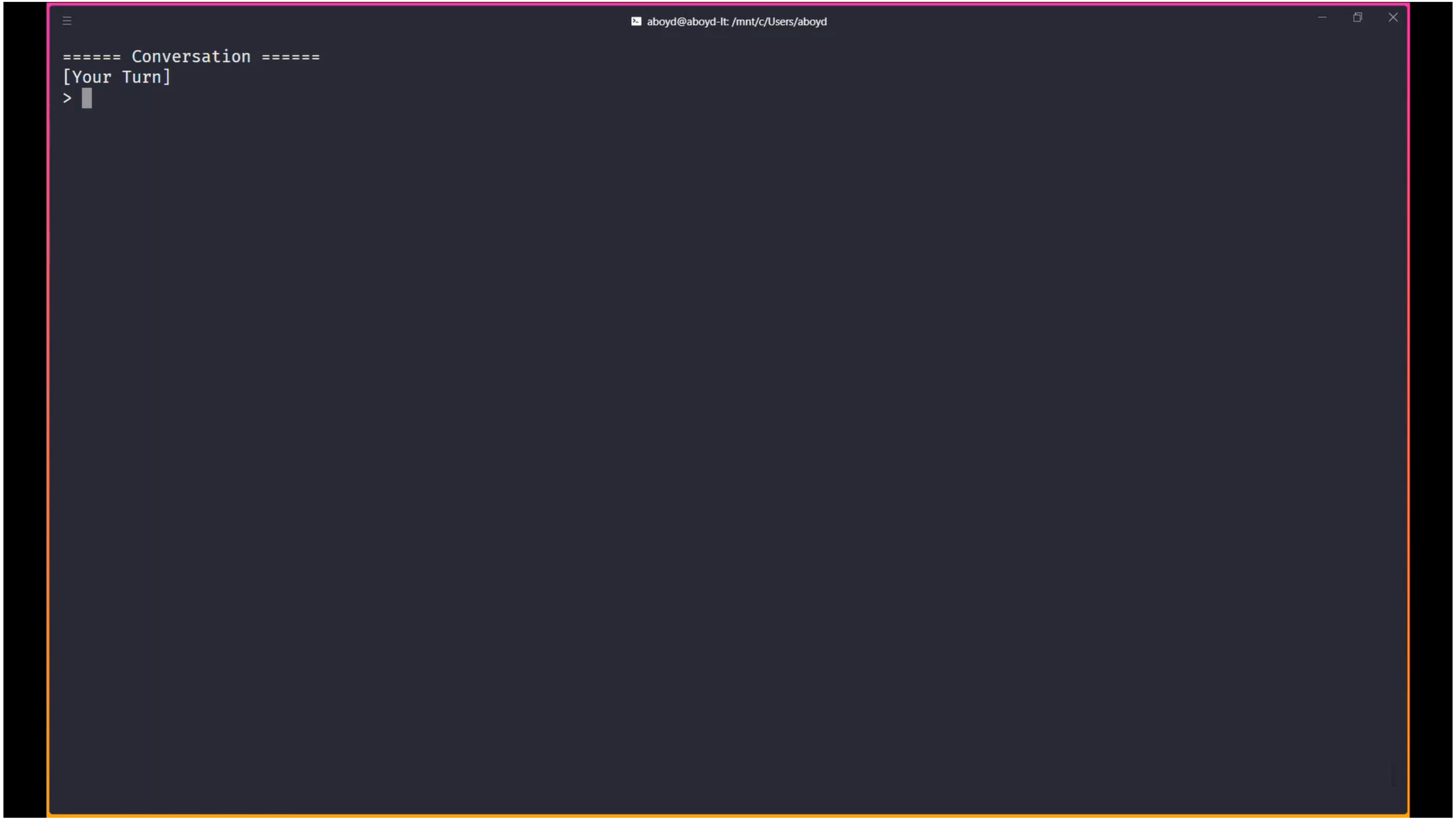
Text	Albert Einstein is known for his theories of special relativity and general relativity. He also made important contributions to statistical mechanics, especially his mathematical treatment of Brownian motion, his resolution of the paradox of specific heats, and his connection of fluctuations and dissipation. Despite his reservations about its interpretation, Einstein also made contributions to quantum mechanics and, indirectly, **quantum field theory**, primarily through his theoretical studies of the photon.
117M	Which two concepts made Einstein's post on quantum mechanics relevant?
768M	Albert Einstein also made significant contributions to which field of theory?
8.3B	Because of his work with the photon, what theory did he indirectly contribute to?
Human	What theory did Einstein have reservations about?

As model size grows, question quality becomes increasingly coherent, complex, and factually relevant.

More Samples →

Paper: https://arxiv.org/abs/2002.09599

CONVERSATION



Chatbot: Trained on 6 months of Reddit Data (Human vs. Agent)

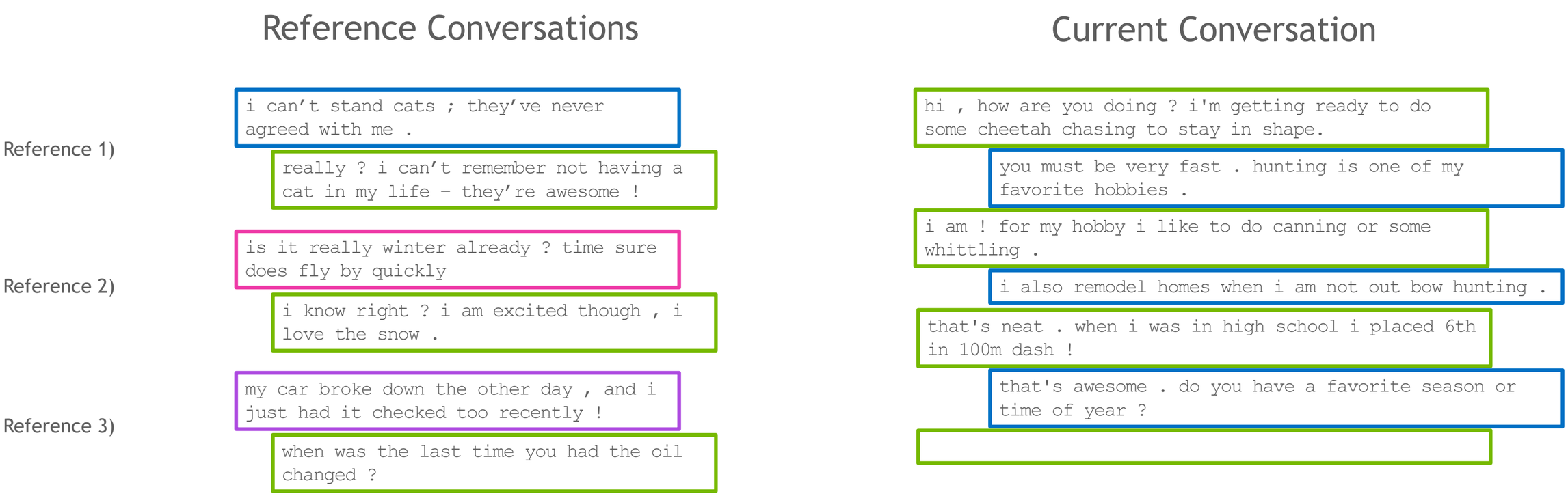
GENERATIVE CONVERSATION CONTROL (GCC)

A New Large Transformer Conversation Model

	# Parameters	Data Source	Persona Control
DLGNet	345 Million	Movie Triples / Ubuntu Dialogue Corpus	👎
DialoGPT	768 Million	Reddit (2005-2017)	👎
Meena	2.6 Billion	Public Social Media	👎
GCC	8.3 Billion	Reddit (2019)	👍

GENERATIVE CONVERSATION CONTROL (GCC)

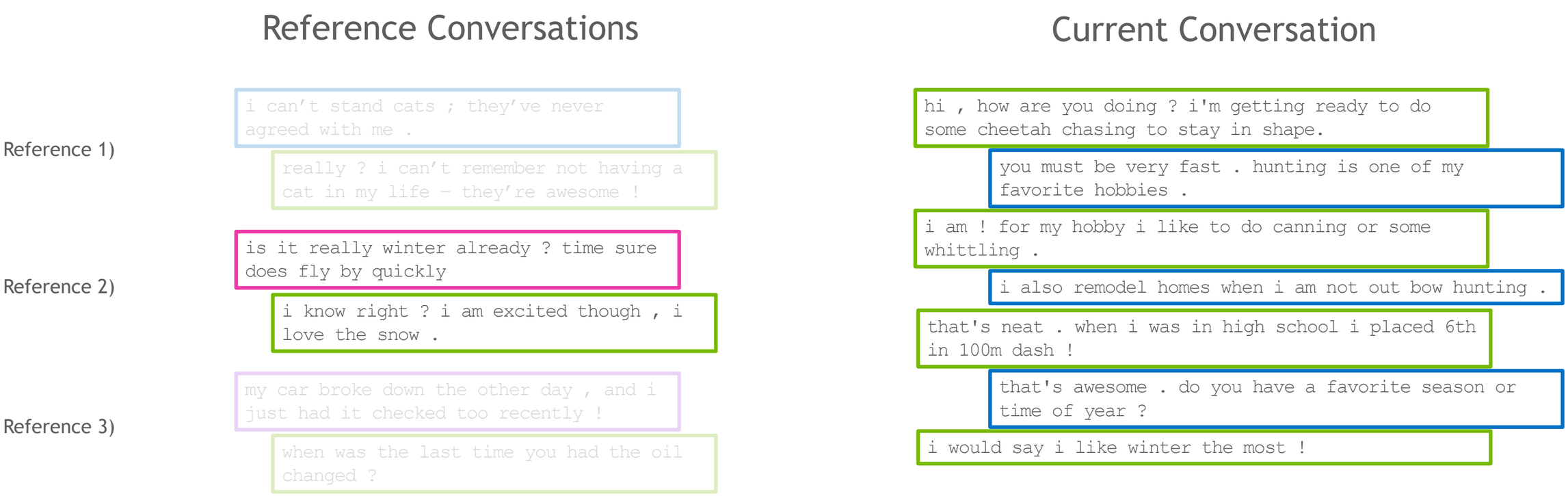
What Is Persona Control?



Multi-turn conversation modeling with the speakers’ response conditioned on reference replies from their past conversations.

GENERATIVE CONVERSATION CONTROL (GCC)

What Is Persona Control?

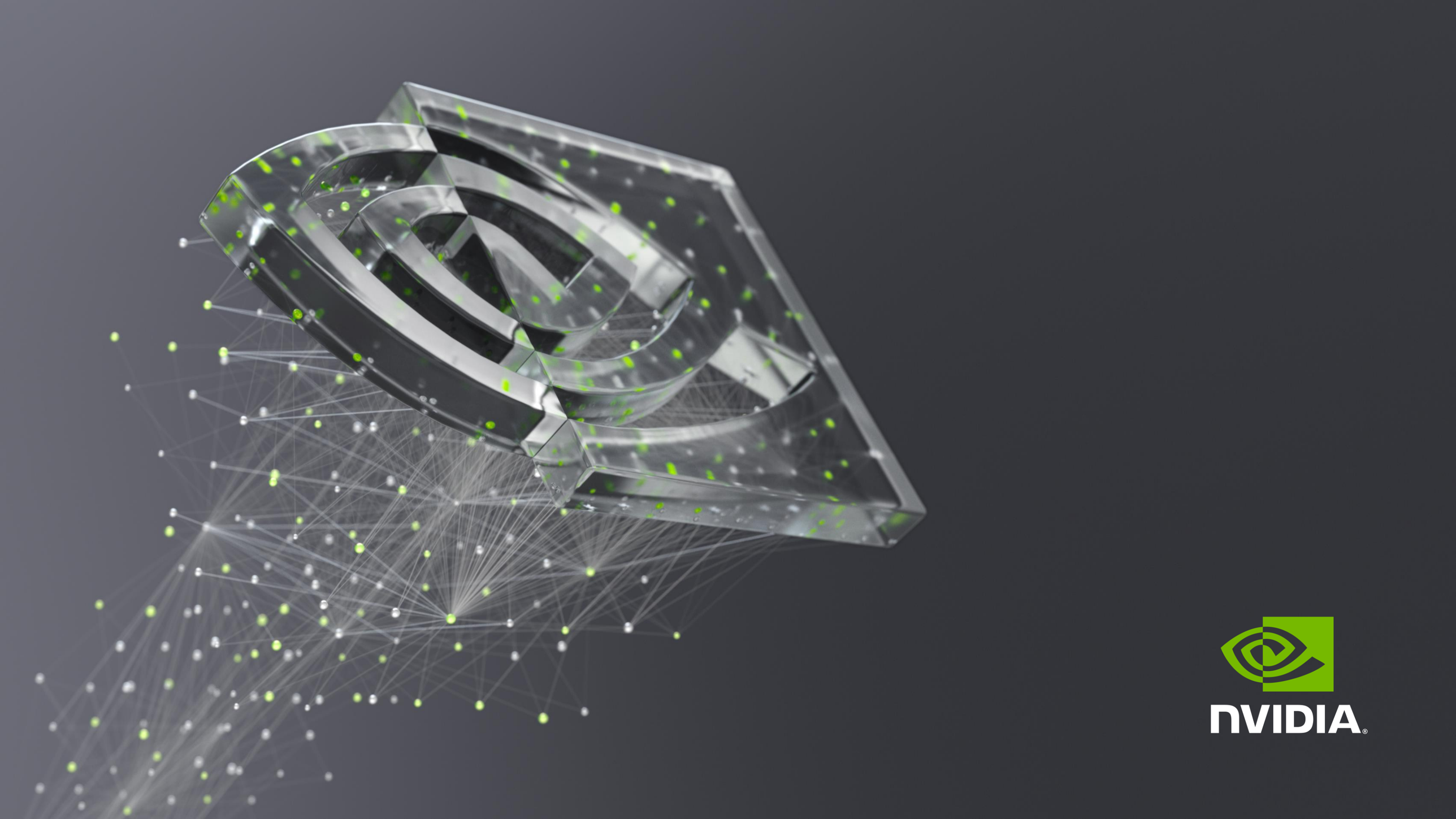


Multi-turn conversation modeling with the speakers' response conditioned on reference replies from their past conversations.

GENERATIVE CONVERSATION CONTROL (GCC)

Bigger Model Better Conversations

Model	Hidden Size (h)	# Layers (l)	# Attention Heads (A)	PPL
GCC - 117M	768	12	12	23.14
GCC - 355M	1024	24	16	18.92
GCC - 774M	1280	36	16	17.18
GCC – 1.2B	1536	40	16	16.08
GCC – 8.3B	**3072**	**72**	**24**	**13.24**


NVIDIA