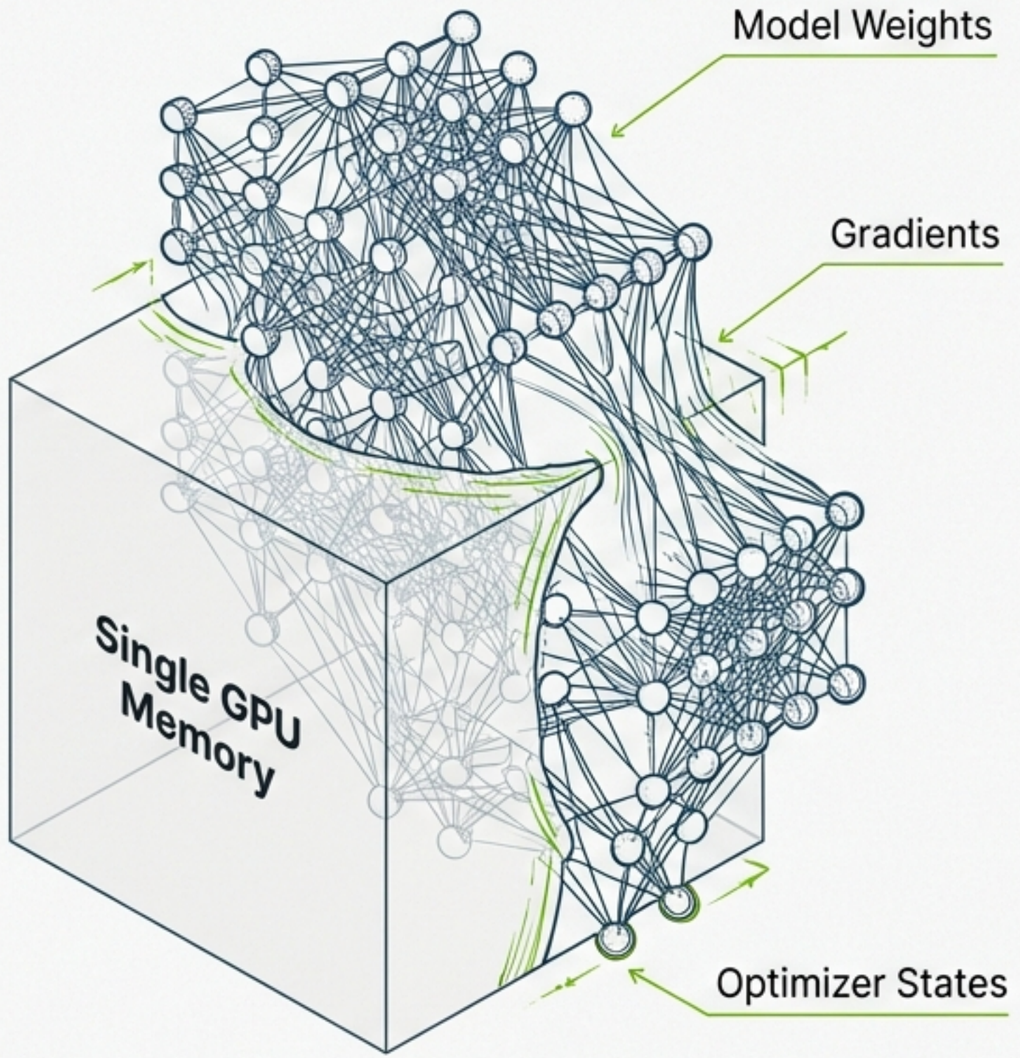
Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism

Mohammad Shoeybi, Mostofa Patwary, Raul Puri,
Patrick LeGresley, Jared Casper, Bryan Catanzaro

NVIDIA
ArXiv, 2019

State-of-the-art NLP models are growing faster than GPU memory.

- **The Trend:** Empirical evidence shows larger language models (BERT, GPT-2) are dramatically more powerful for tasks like question answering and article completion.
- **The Constraint:** Multi-billion parameter models physically exceed the memory capacity of a single GPU.
- **The Bottleneck:** It's impossible to store the model's weights, gradients, and optimizer states (e.g., for ADAM) on one device.



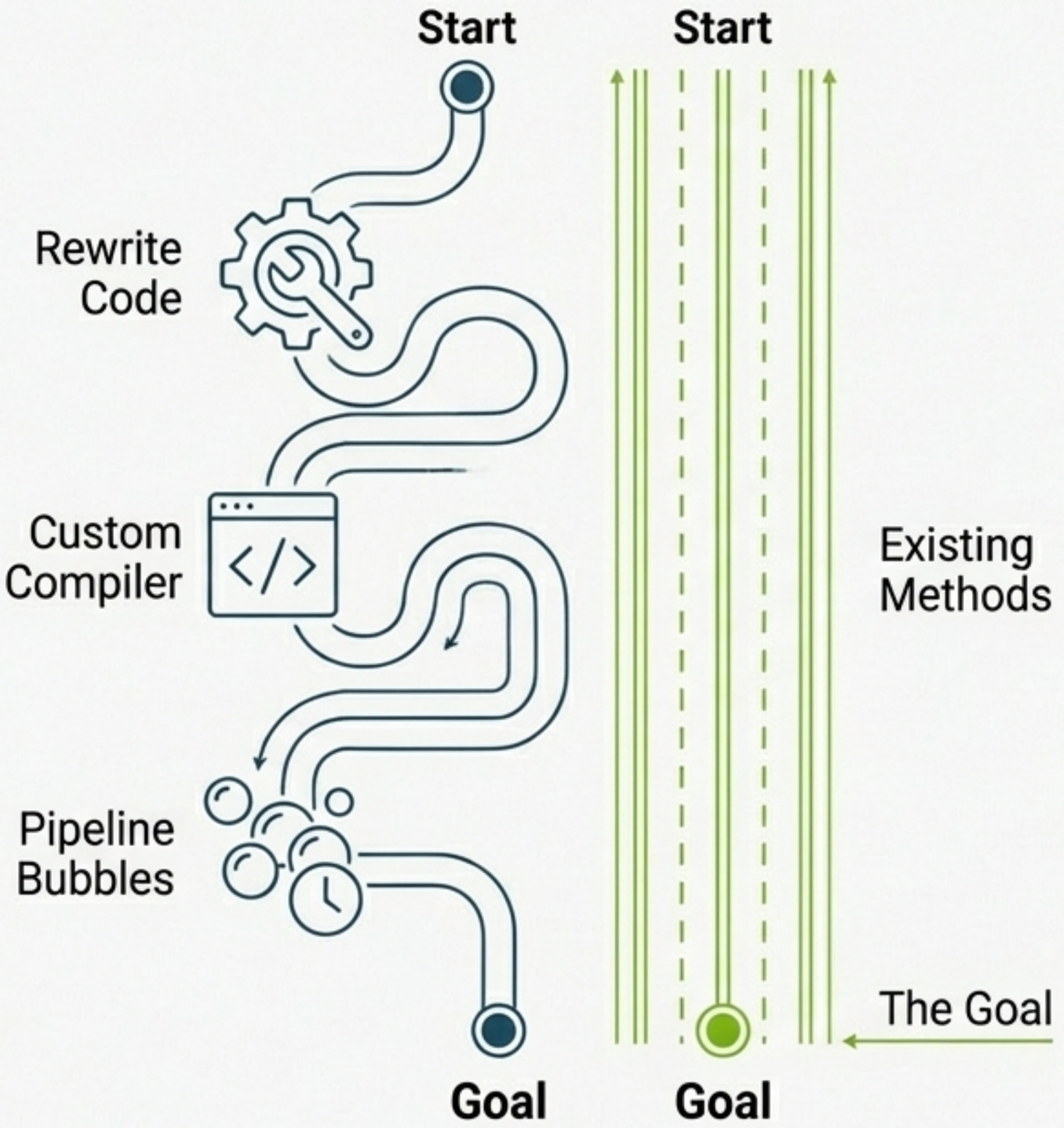
Existing model parallelism methods are often complex or inefficient.

Pipeline Parallelism (e.g., GPipe):

- Splits *layers* of the model across GPUs.
- Suffers from "pipeline bubbles" (idle GPU time), which reduces efficiency.

General Distributed Frameworks (e.g., Mesh-TensorFlow):

- Highly flexible, but often require rewriting models in a new language or using custom compilers.
- Represents a significant engineering investment and adoption barrier.



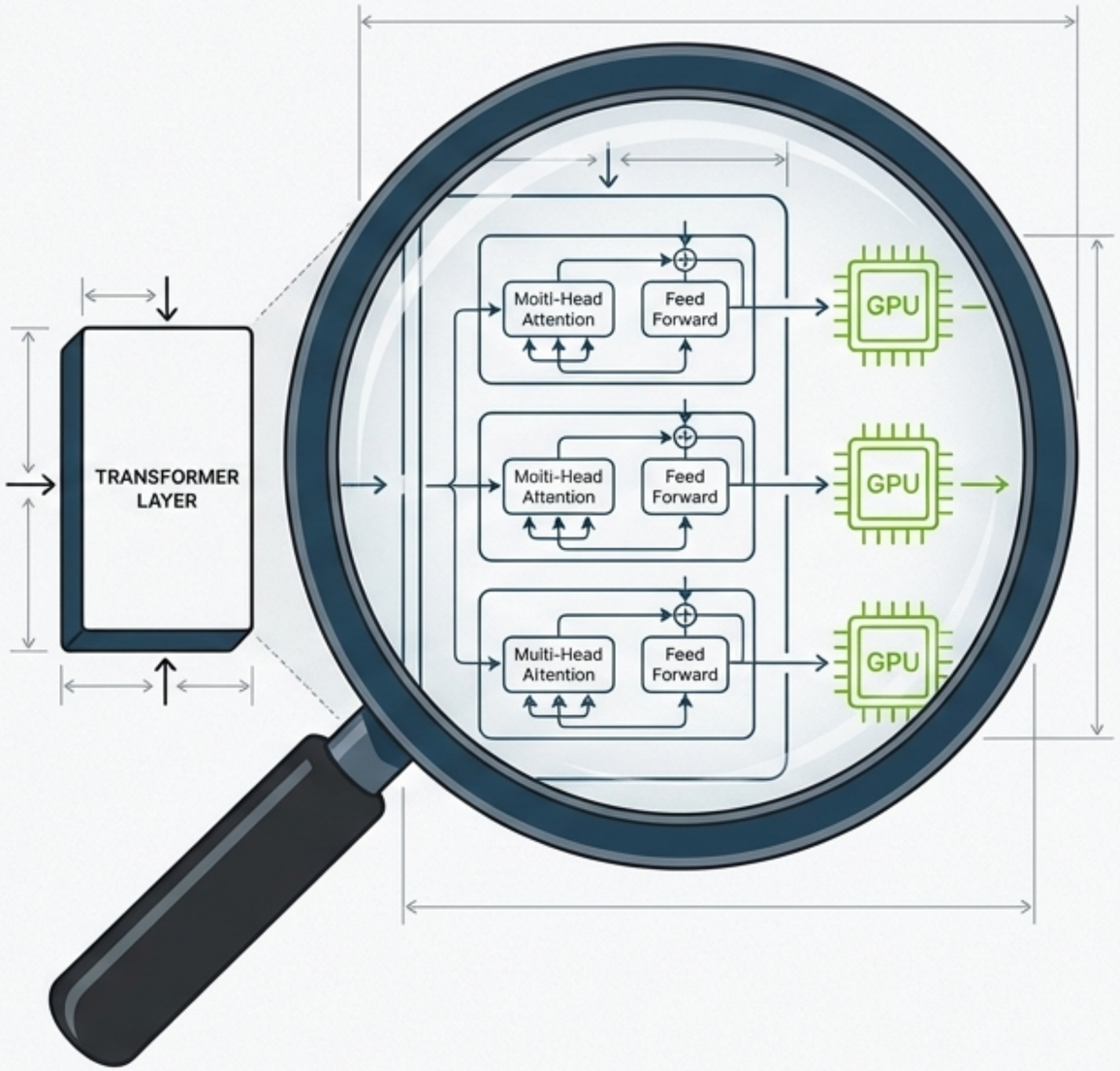
Our question: Can we parallelize *inside* layers, simply and efficiently?

The Goal: Create a simple, efficient **intra-layer** model parallel approach.

The Hypothesis: We can exploit the specific structure of Transformer layers to achieve efficient scaling with only a few targeted communication primitives.

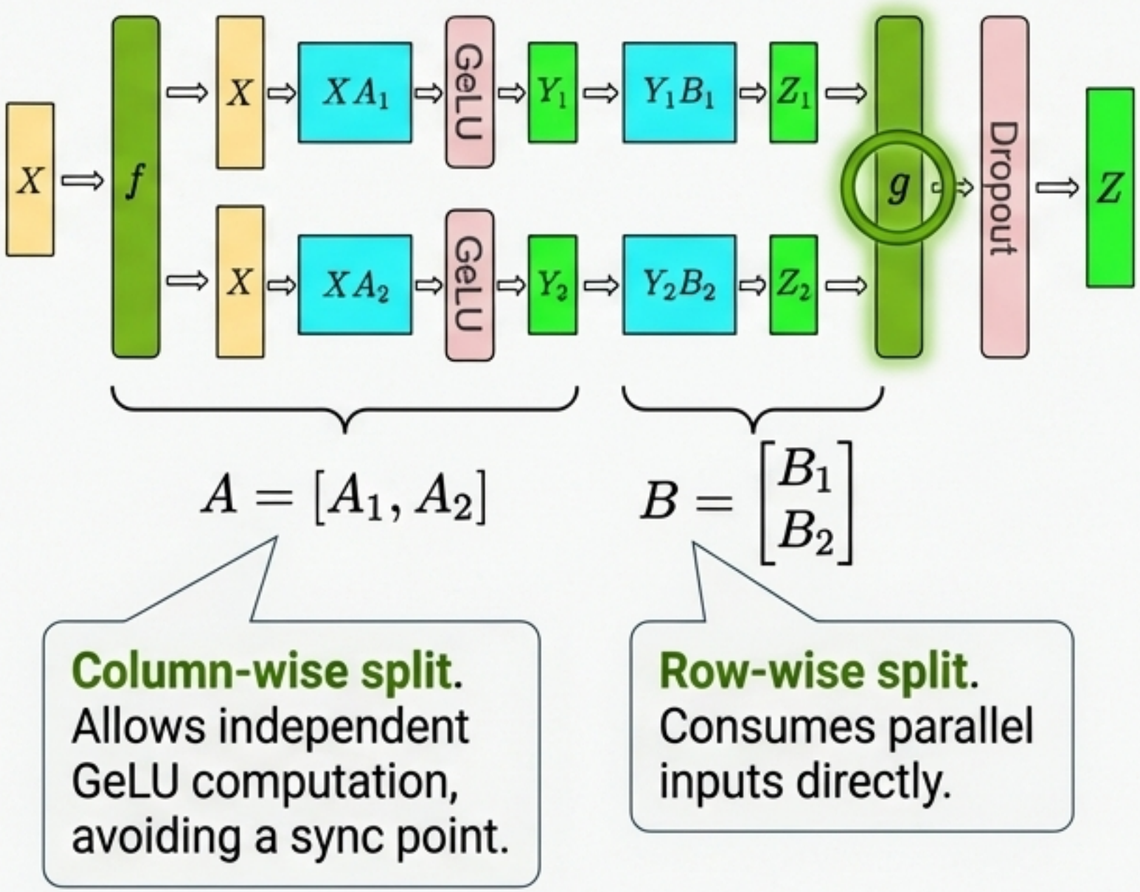
The Requirements:

- Minimal changes to existing PyTorch Transformer code.
- No new compiler or framework required.



We parallelize the MLP block by splitting matrix multiplies to avoid synchronization.

- An MLP block is two linear layers: $\mathbf{Y} = \text{GeLU}(\mathbf{X} * \mathbf{A}) * \mathbf{B}$.
- **Key Insight**: We split the first weight matrix **A** **column-wise**.
  - $[\mathbf{Y}_1, \mathbf{Y}_2] = [\text{GeLU}(\mathbf{X} * \mathbf{A}_1), \text{GeLU}(\mathbf{X} * \mathbf{A}_2)]$.
  - This allows the non-linear GeLU to be applied independently on each GPU with no communication.
- The second matrix **B** is split **row-wise**, and a single all-reduce syncs the final output.



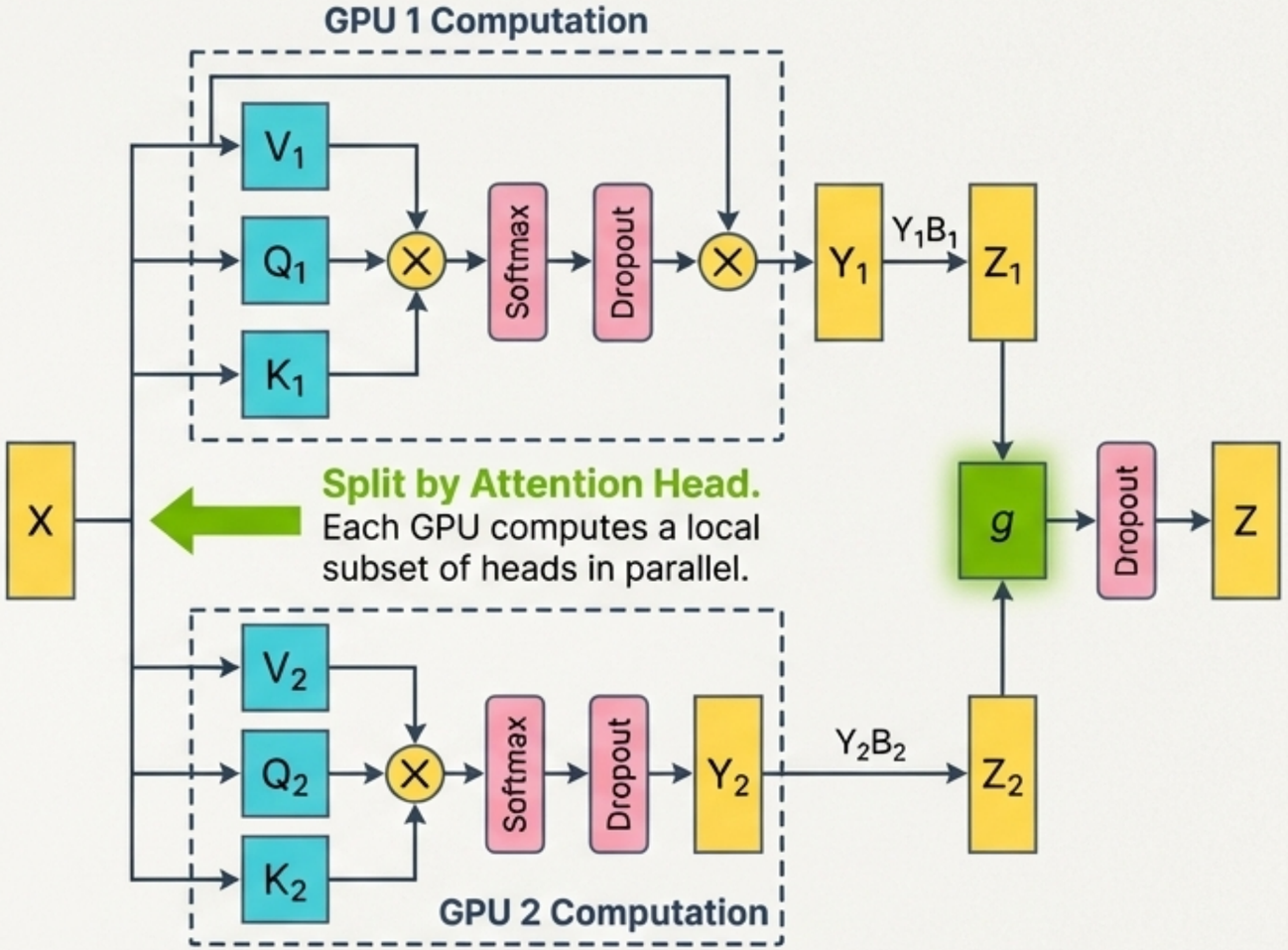
We parallelize the self-attention block by splitting across attention heads.

Multi-head attention is inherently parallel, as each head performs an independent calculation.

- **Our Approach:** We split the Query (Q), Key (K), and Value (V) weight matrices by head across GPUs.

Each GPU computes a subset of the attention heads locally with no communication required.

A single `all-reduce` after the final output projection gathers results, identical to the MLP pattern.



$$\mathbf{Y} = \text{Self-Attention}(\mathbf{X})$$
$$\mathbf{Z} = \text{Dropout}(\mathbf{YB})$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix}$$

$$\left.\begin{aligned} \mathbf{Q} &= [\mathbf{Q}_1, \mathbf{Q}_2] \\ \mathbf{K} &= [\mathbf{K}_1, \mathbf{K}_2] \\ \mathbf{V} &= [\mathbf{V}_1, \mathbf{V}_2] \end{aligned}\right\} \text{split attention heads}$$

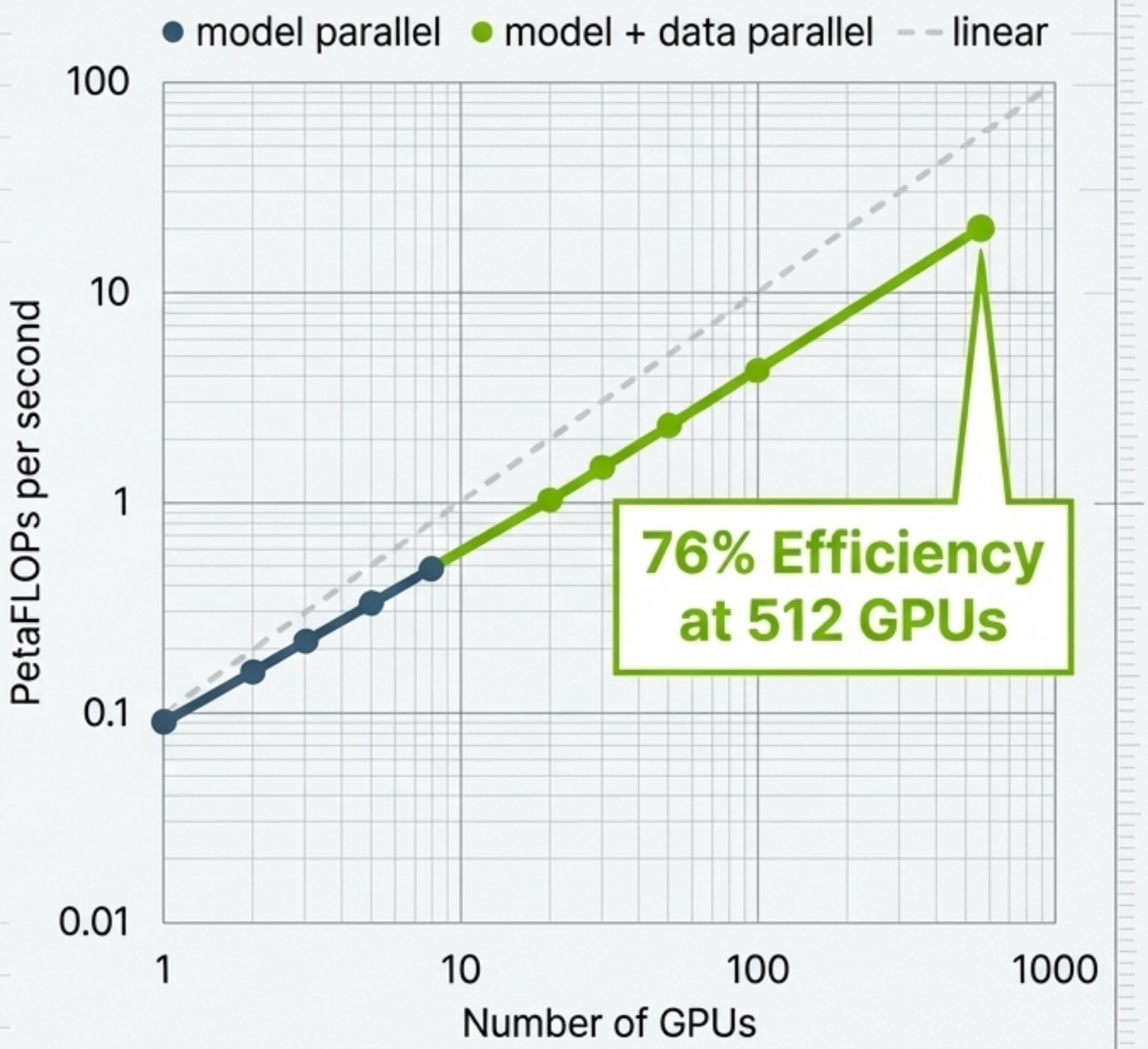
Our method achieves 76% scaling efficiency on 512 GPUs.

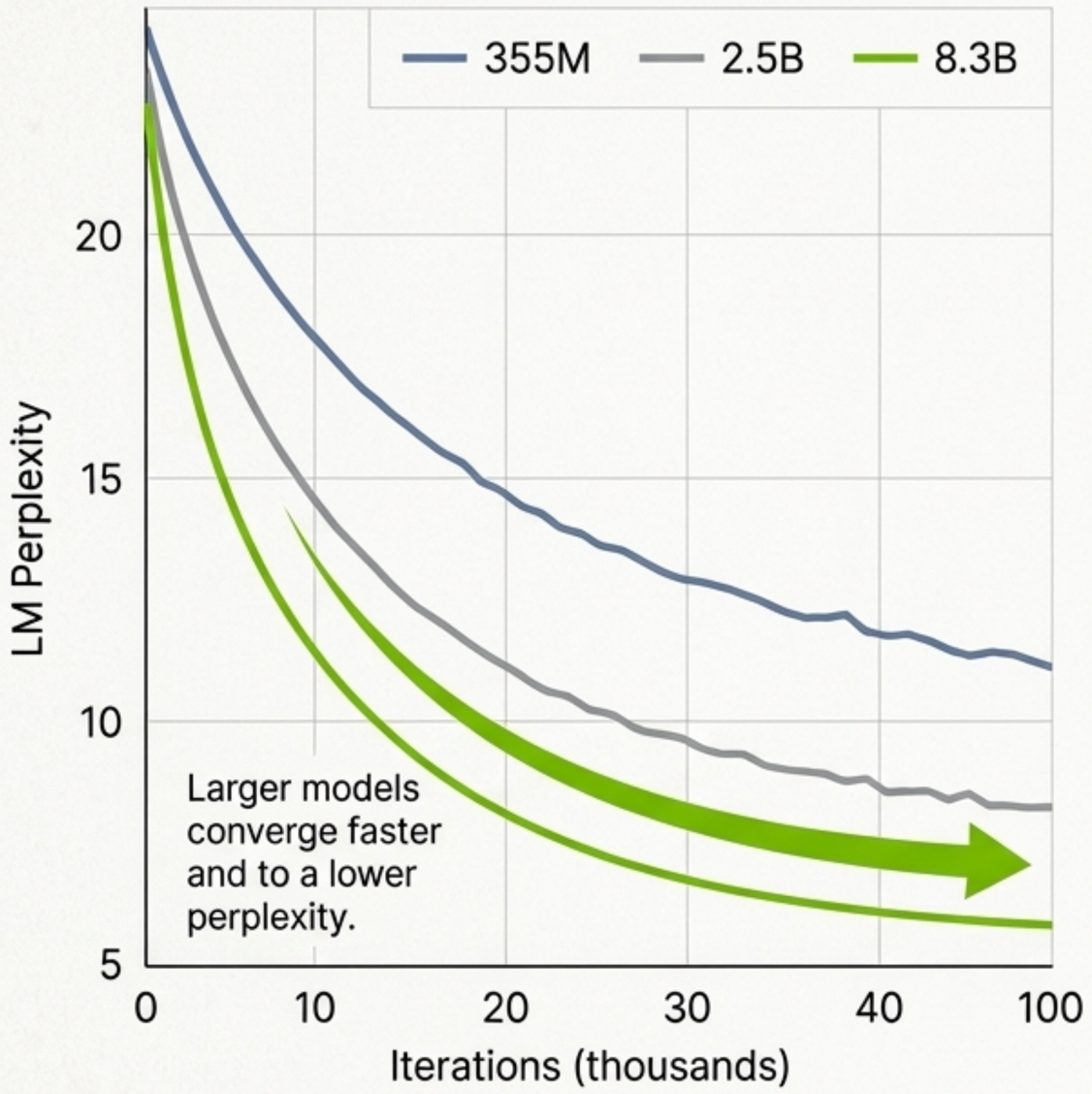
Baseline: A 1.2B parameter model on one V100 GPU sustains 39 TFLOPs (a strong 30% of peak).

Model Parallel Scaling: 8-way model parallelism achieves 77% of linear scaling.

Combined Scaling: Training an 8.3B parameter model on 512 GPUs (8-way model + 64-way data parallel) sustains **15.1 PetaFLOPs**.

This represents **76% weak scaling efficiency** compared to the highly optimized single-GPU baseline.





Scaling enables training of GPT-2 models that set new SOTA records.

We trained GPT-2 style models up to **8.3 billion parameters**.

Finding: As model size increases, validation perplexity consistently improves.

Results: The 8.3B model achieved new State-of-the-Art (SOTA) on key benchmarks:

WikiText103

10.8

perplexity

(vs. 15.8 prior SOTA)

LAMBADA

66.5%

accuracy

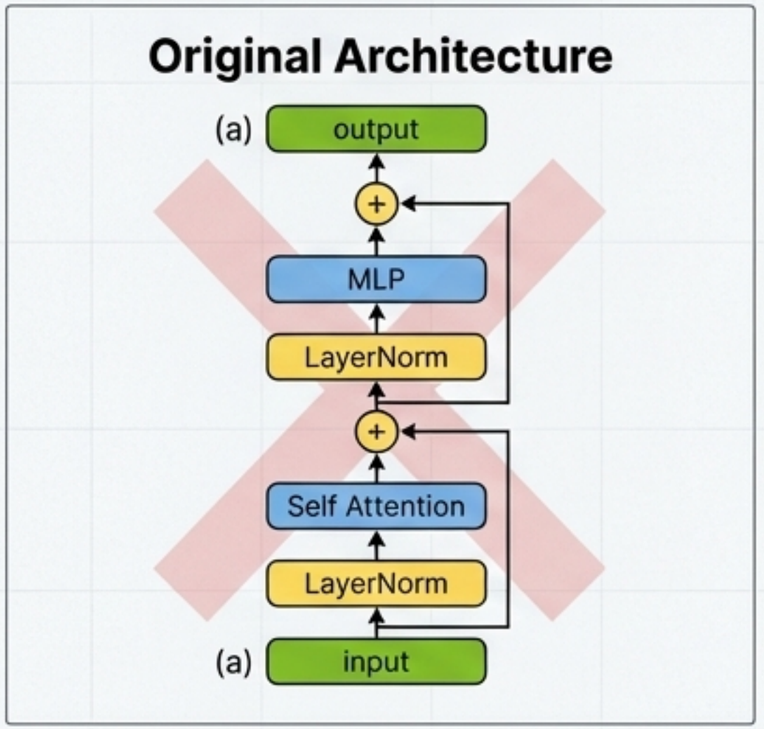
(vs. 63.2% prior SOTA)

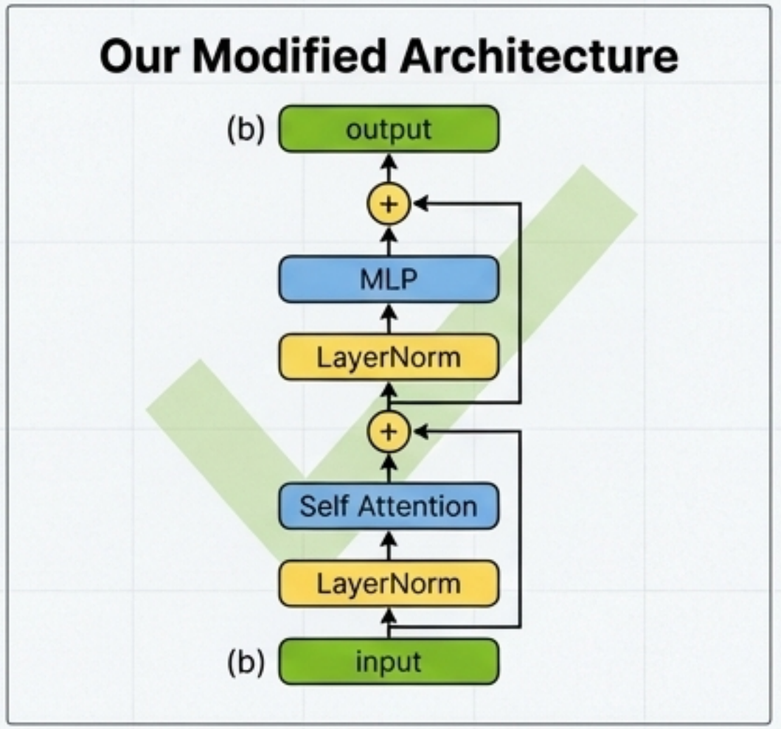
A key architectural fix was critical for training giant BERT models.

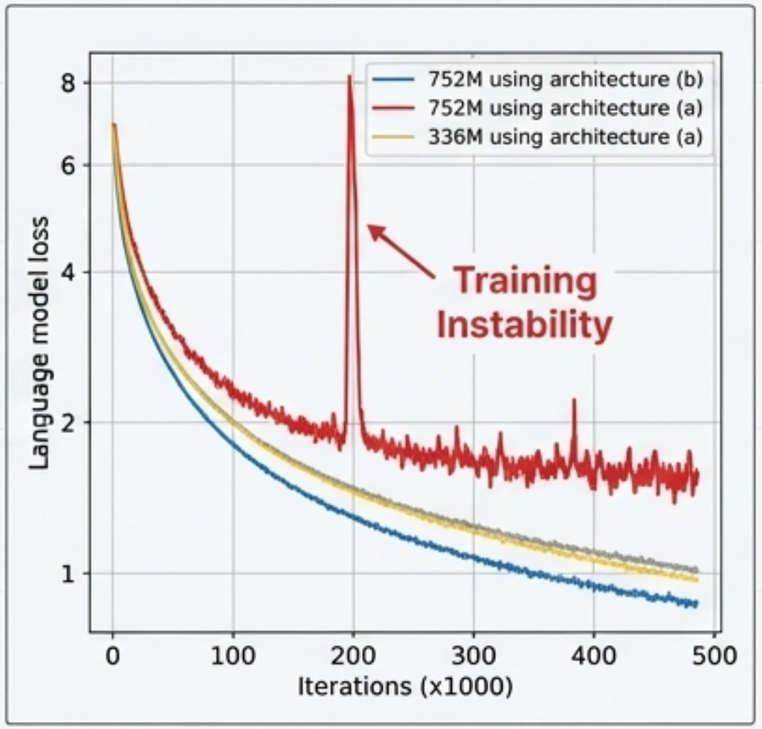
The Problem: The community found that BERT training becomes unstable and performance degrades when scaling beyond BERT-Large (336M parameters).

Our Discovery: Moving Layer Normalization to the **input** of each sub-layer (before the residual connection) completely stabilizes training for large models.

The Impact: This simple change enabled us to train a 3.9B parameter BERT model that improved monotonically with scale.



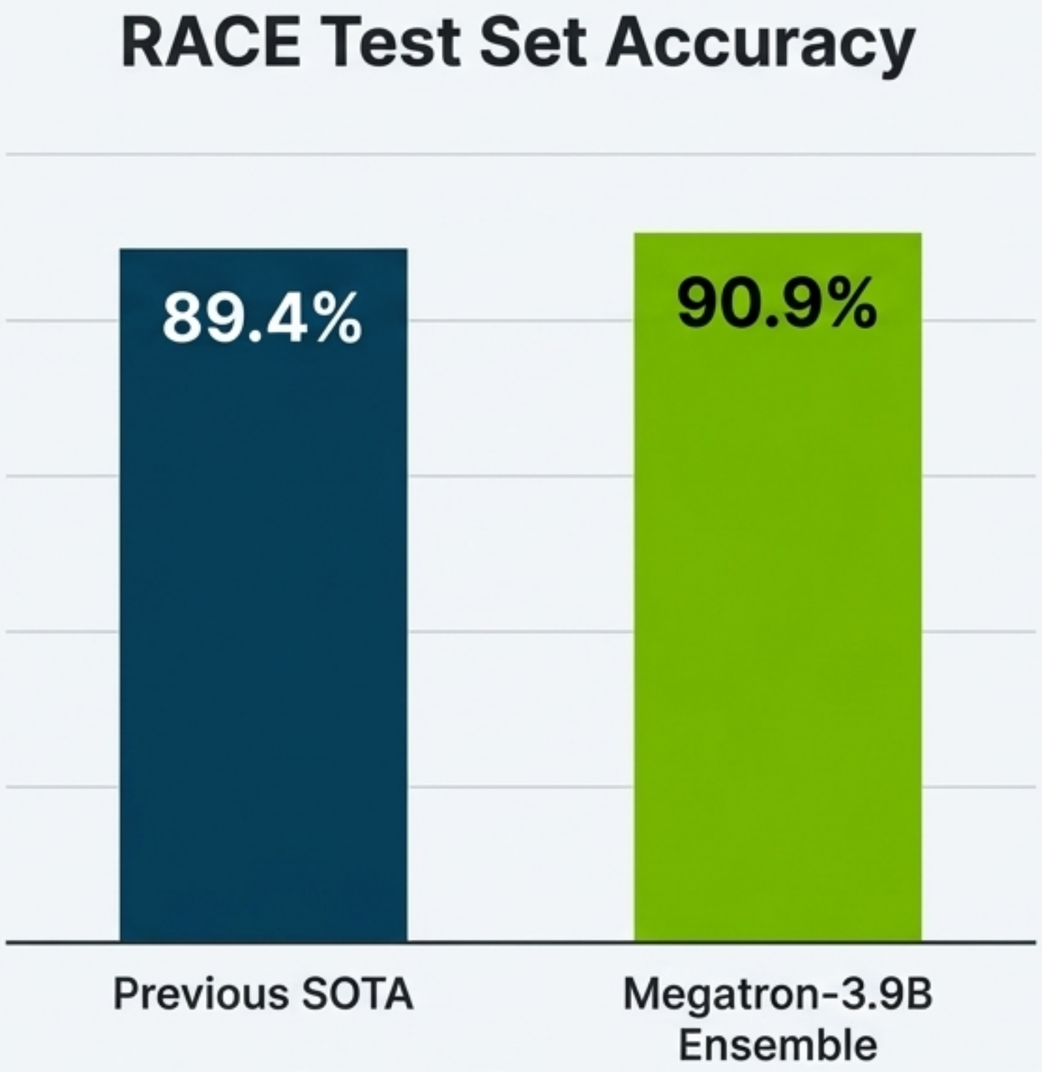




Our stabilized 3.9B parameter BERT sets a new record on reading comprehension.

With the modified architecture, we successfully trained a BERT model up to 3.9 billion parameters.

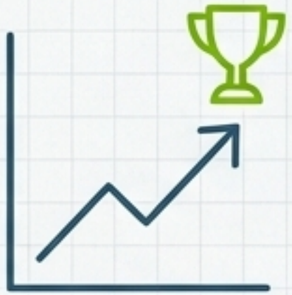
- **Finding:** Performance on downstream tasks consistently improved with model size.
- **Result:** Our 3.9B model achieved new **State-of-the-Art** on the RACE dataset, a challenging reading comprehension task.
  - Ensemble Accuracy: 90.9% (vs. previous SOTA of 89.4%).



Our work delivers a simple, efficient, and open-source path to giant language models.

A Simple & Efficient Method

Intra-layer model parallelism achieving 76% scaling efficiency with minimal code changes.



Proof that Scaling Works

Demonstrated SOTA results on GPT-2 and BERT models by enabling unprecedented scale.

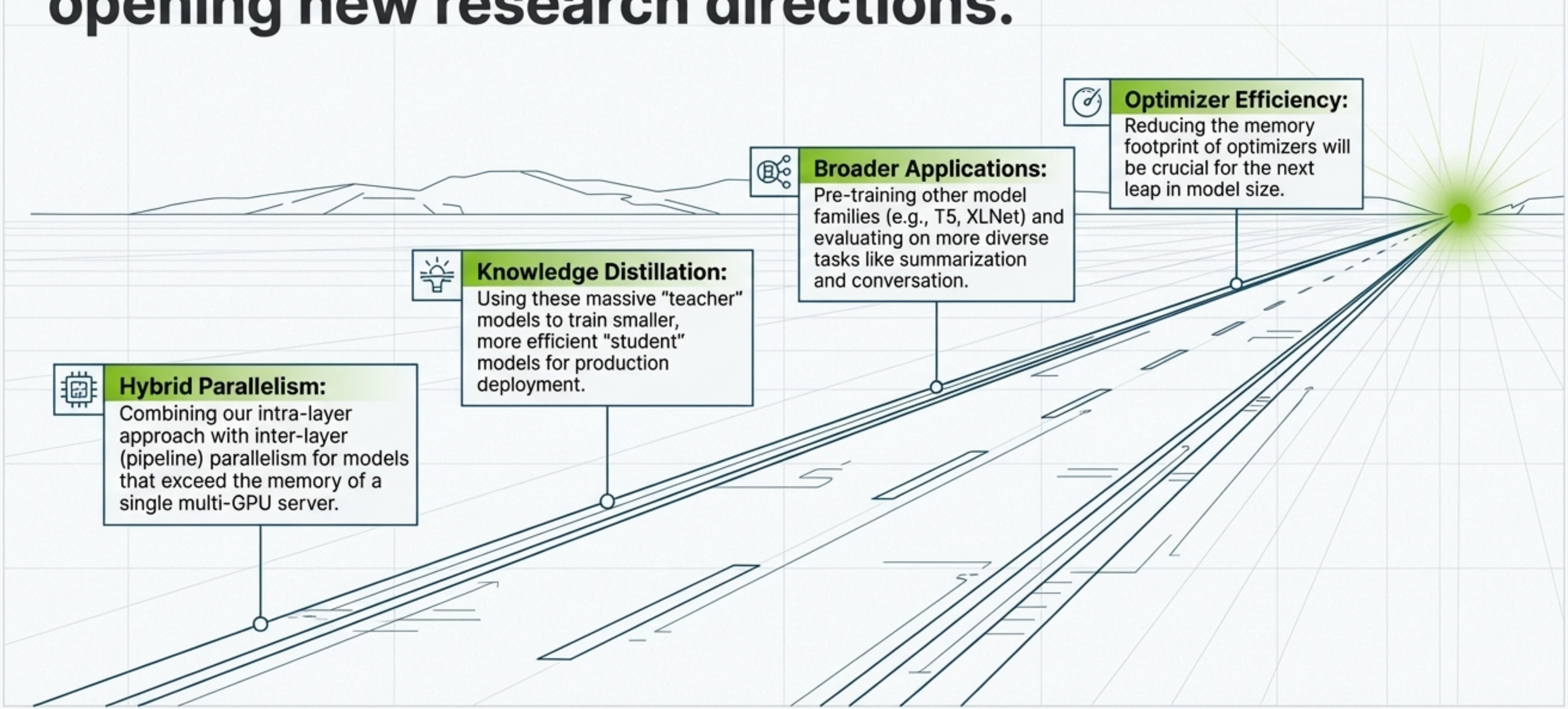
A Critical Architectural Insight

Uncovered the importance of LayerNorm placement for stable training of large BERT-style models.

Open-Source Code

Released the implementation (github.com/NVIDIA/Megatron-LM) to the community.

The quest for scale continues, opening new research directions.



Thank You

Paper: Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism

Code: https://github.com/NVIDIA/Megatron-LM

Questions?

NVIDIA