[COMSE6998-015] Fall 2024 Introduction to Deep Learning and LLM based Generative AI Systems

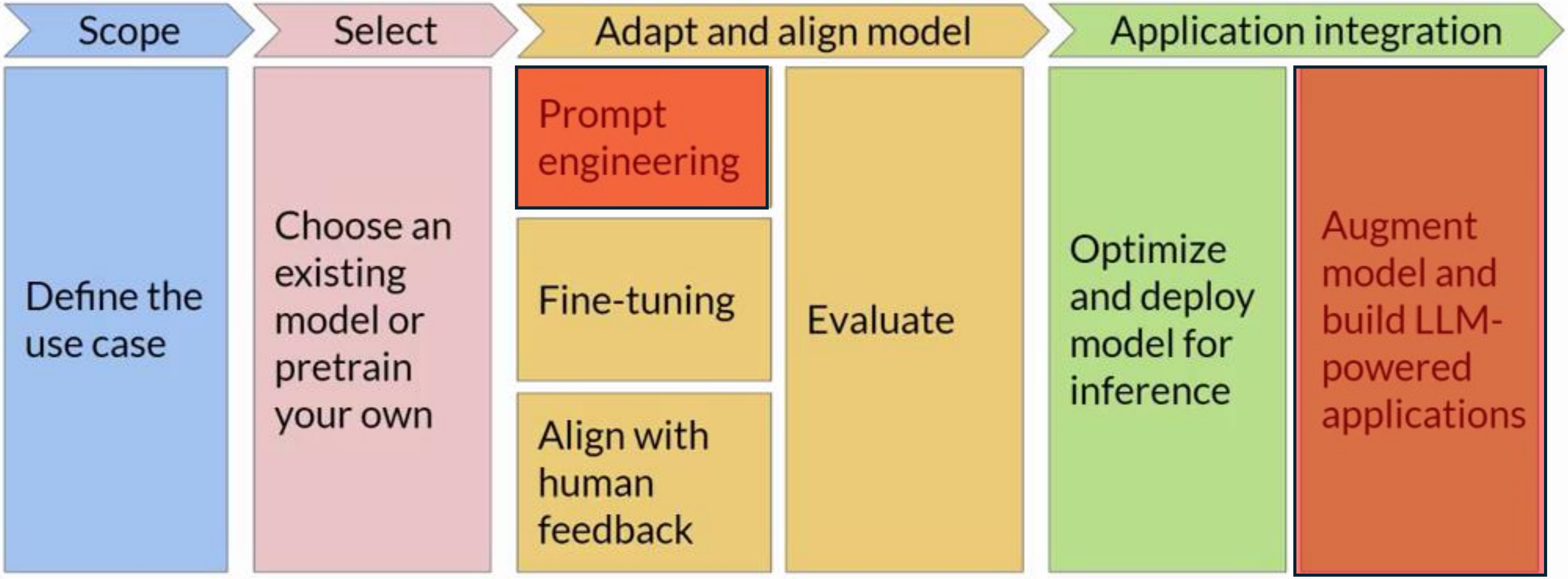
Lecture 9

Parijat Dube, Chen Wang

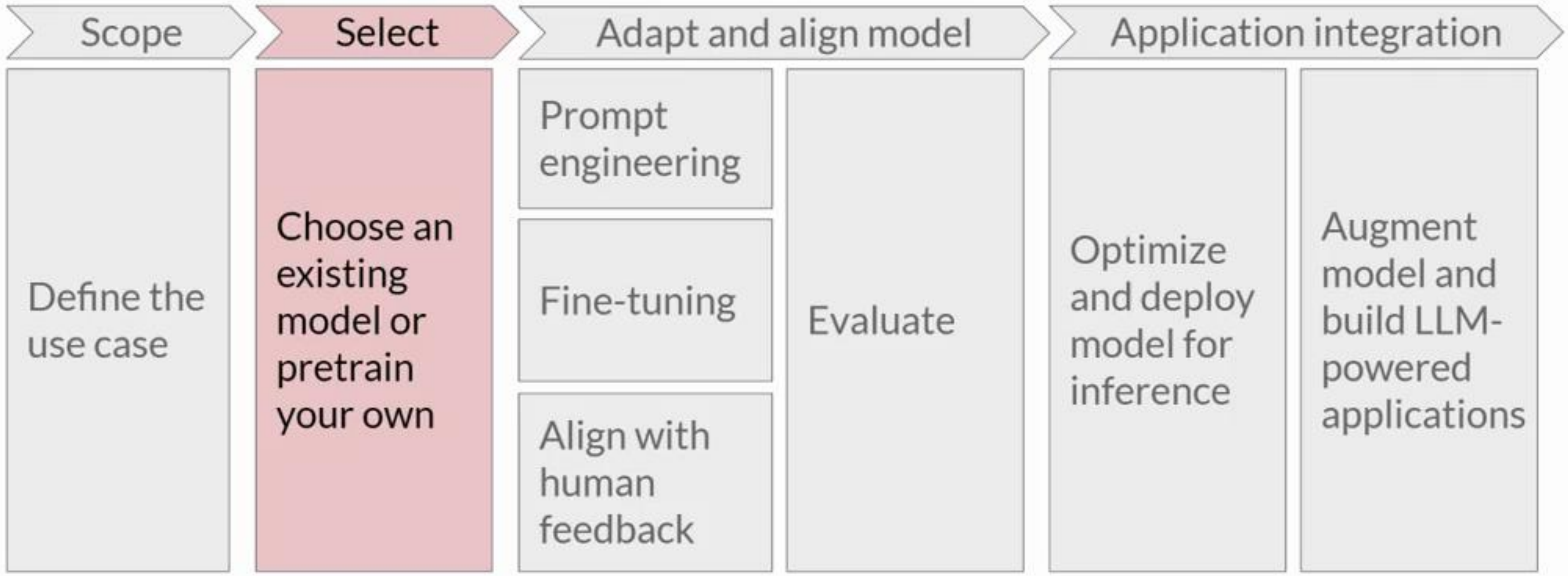
Agenda

- Pretrained Model Selection and Model Pretraining
- Quantization
- Efficient Mult-GPU Compute Strategies
- Scaling Law
- BloombergGPT

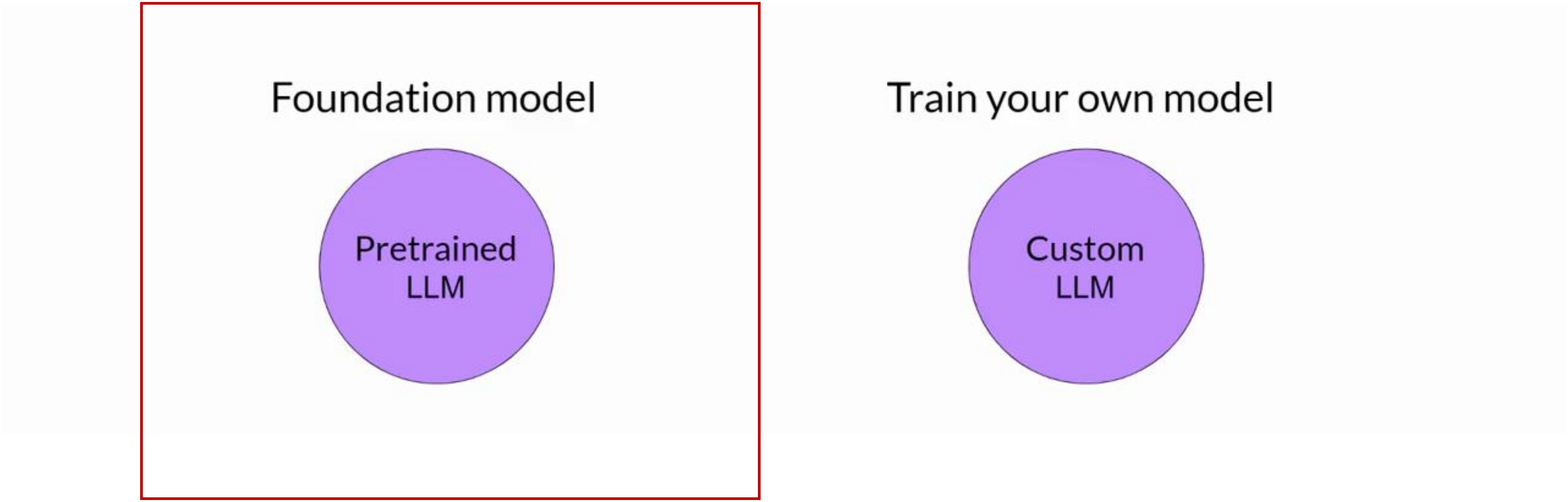
Generative AI project lifecycle



Generative AI project lifecycle



Considerations for choosing a model



Model Hubs

Model Card for T5 Large

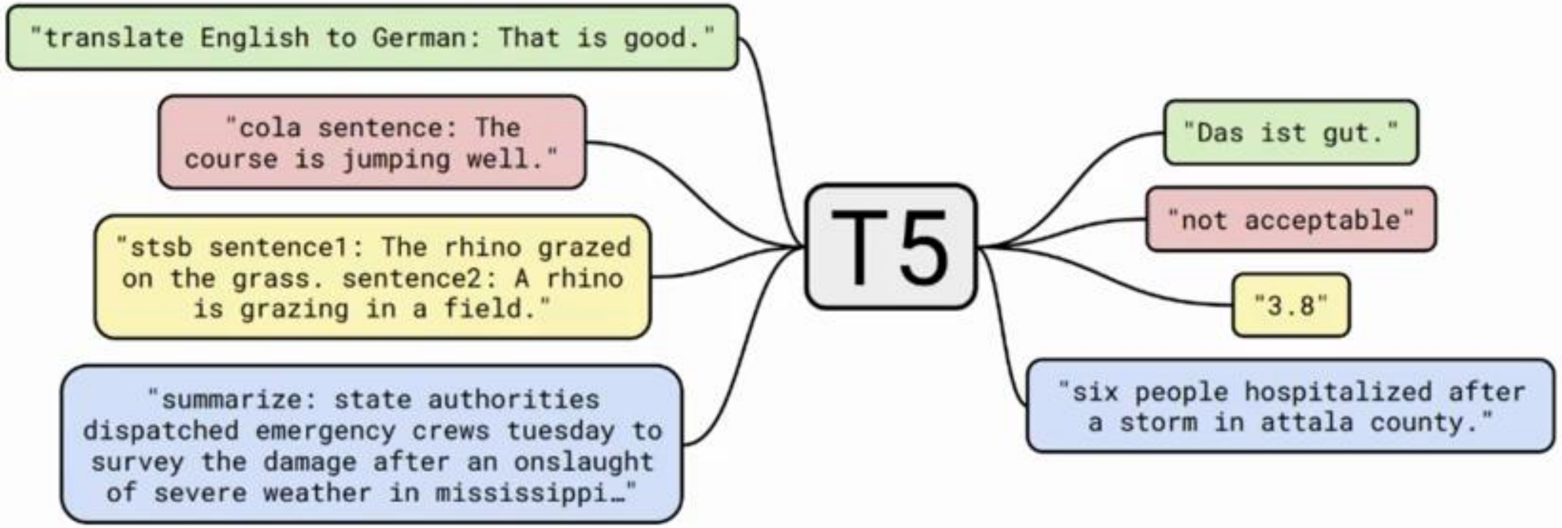
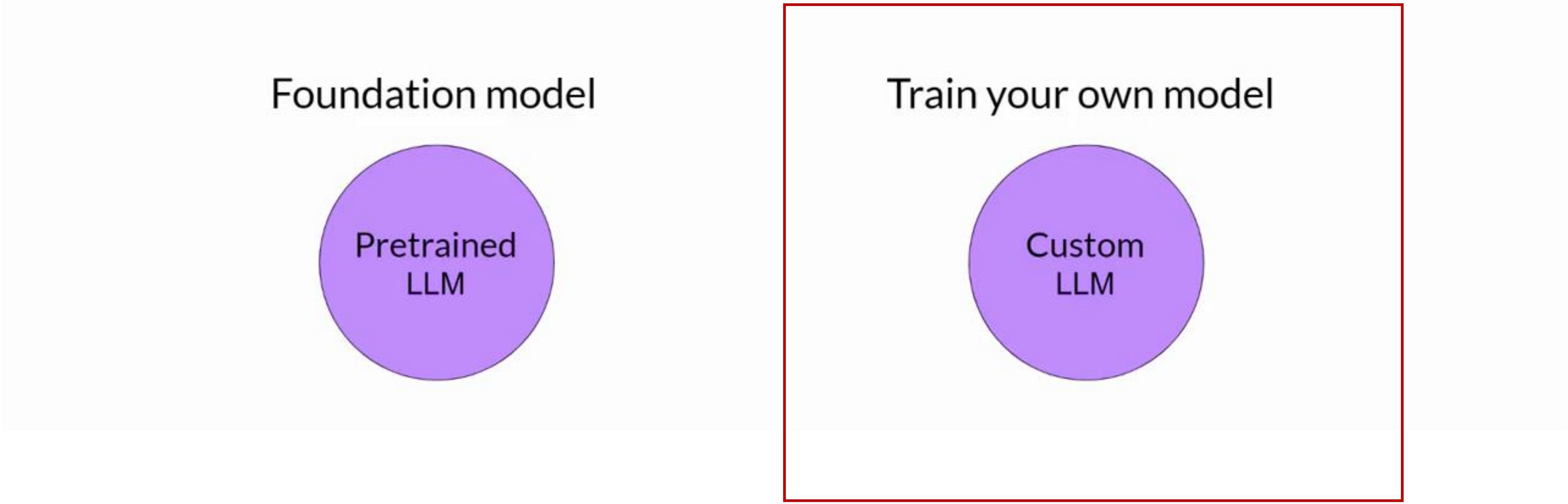


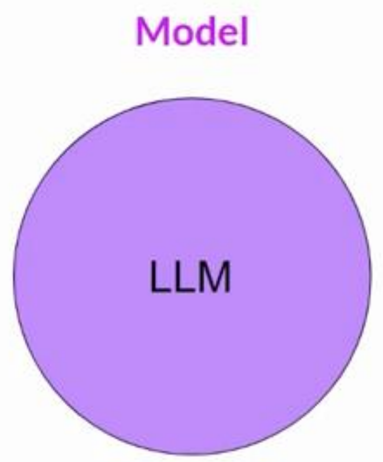
Table of Contents

How Large Language Models are trained?

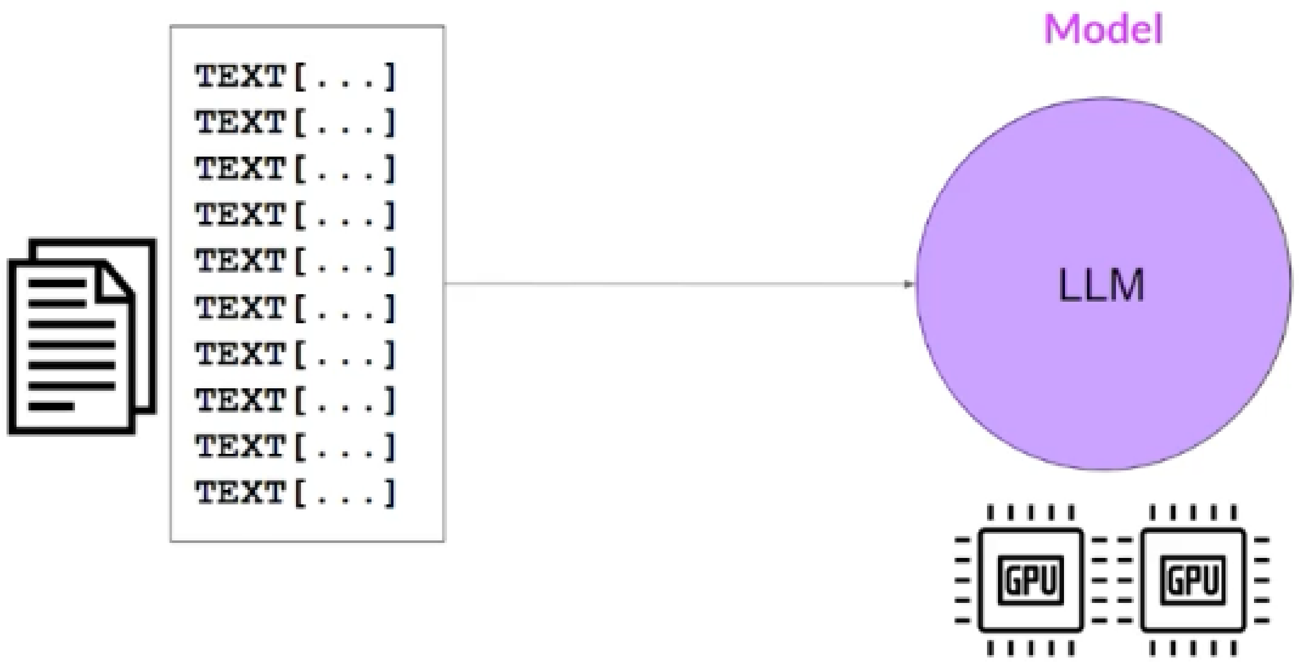


Model architectures and pre-training objectives

LLM pre-training at a high level



LLM pre-training at a high level



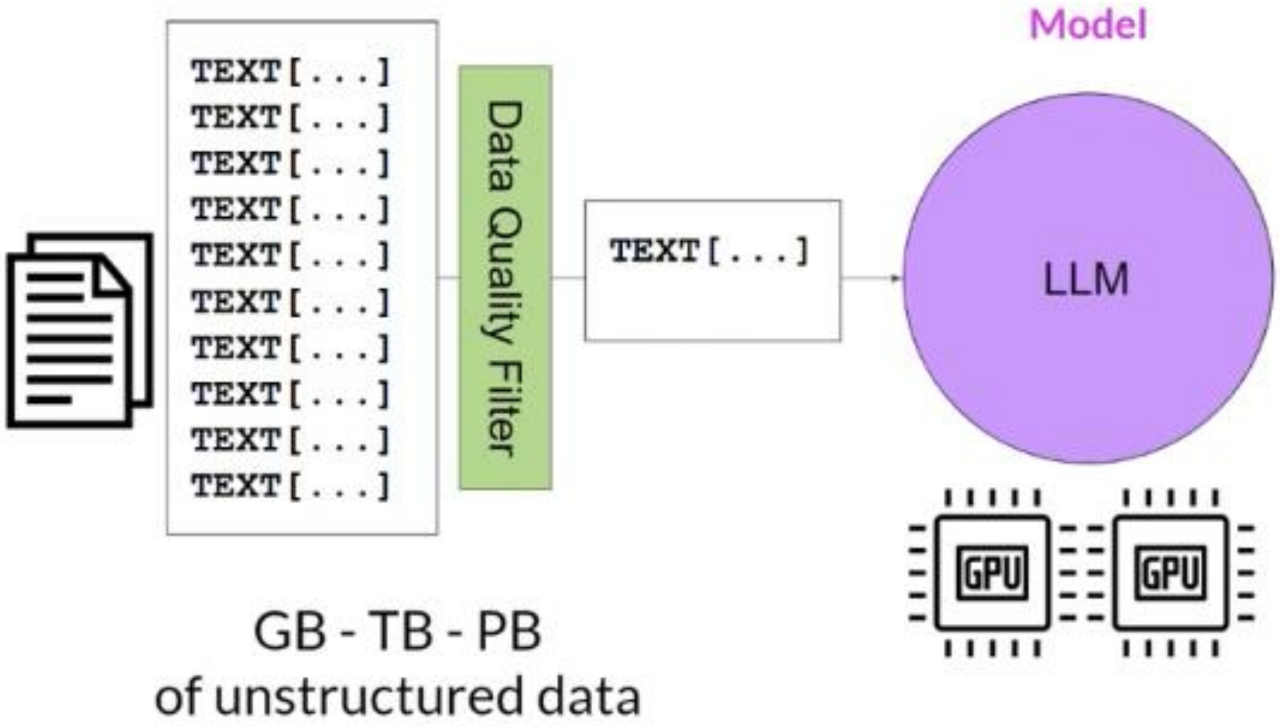
GB - TB - PB
of unstructured data

The model internalizes the patterns and structures present in the language.

Token String	Token ID	Embedding / Vector Representation
'_The'	37	[-0.0513, -0.0584, 0.0230, ...]
'_teacher'	3145	[-0.0335, 0.0167, 0.0484, ...]
'_teaches'	11749	[-0.0151, -0.0516, 0.0309, ...]
'_the'	8	[-0.0498, -0.0428, 0.0275, ...]
'_student'	1236	[-0.0460, 0.0031, 0.0545, ...]
...	...	...

Vocabulary

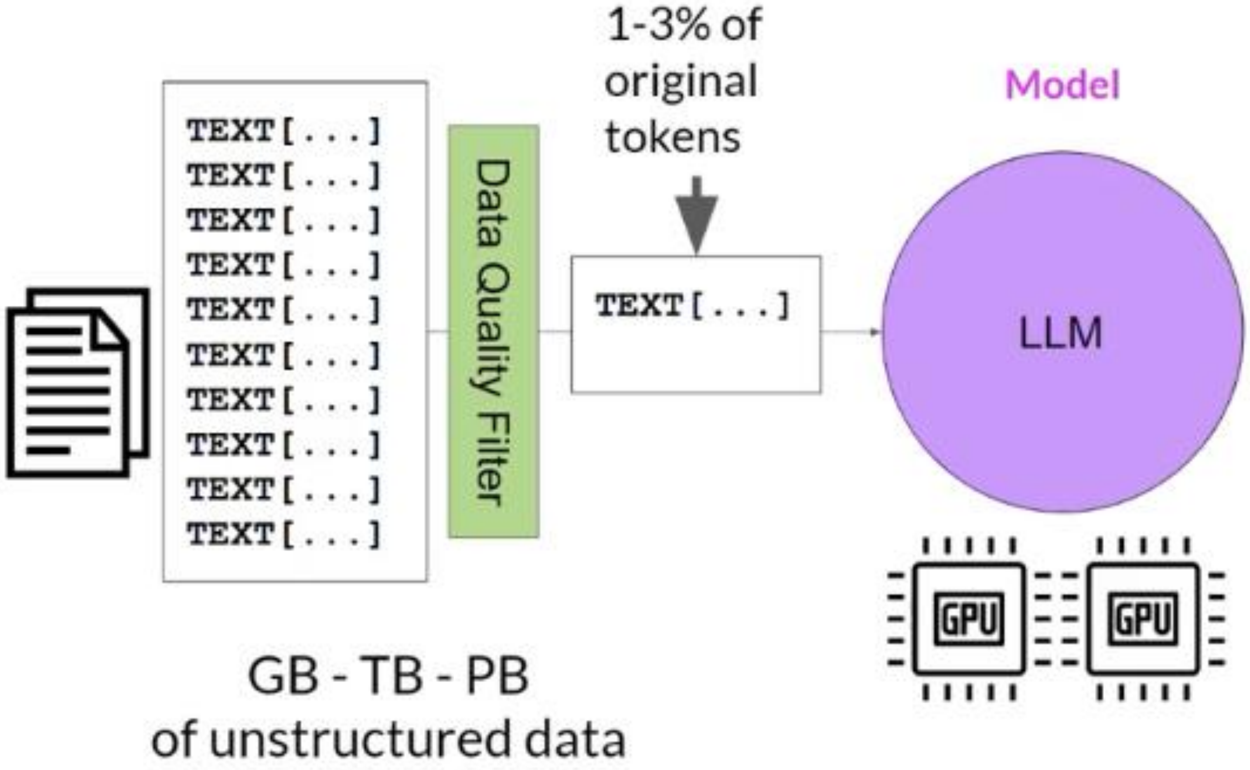
LLM pre-training at high level



Token String	Token ID	Embedding / Vector Representation
'_The'	37	[-0.0513, -0.0584, 0.0230, ...]
'_teacher'	3145	[-0.0335, 0.0167, 0.0484, ...]
'_teaches'	11749	[-0.0151, -0.0516, 0.0309, ...]
'_the'	8	[-0.0498, -0.0428, 0.0275, ...]
'_student'	1236	[-0.0460, 0.0031, 0.0545, ...]
...	...	...

Vocabulary

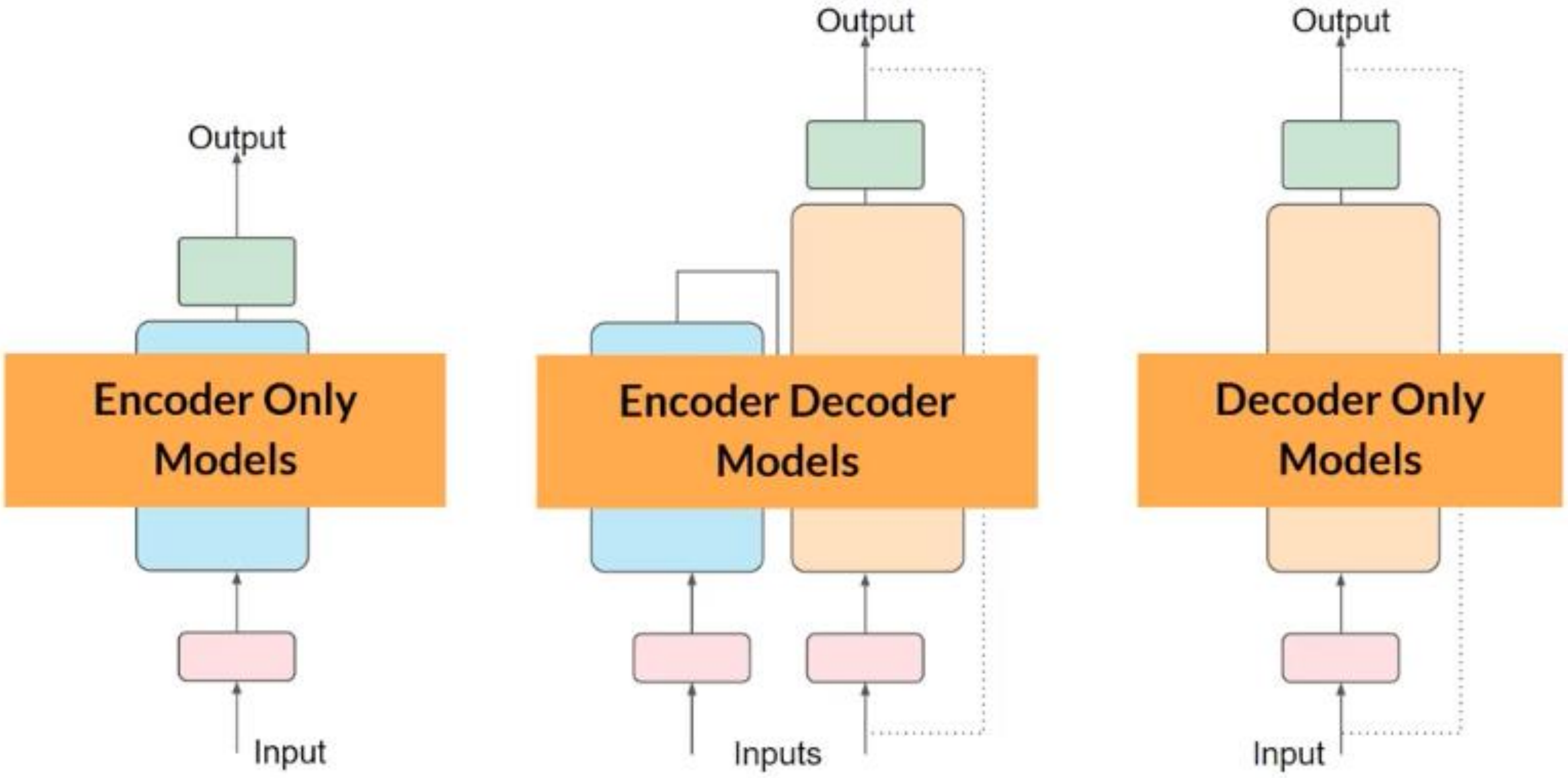
LLM pre-training at high level



Token String	Token ID	Embedding / Vector Representation
'_The'	37	[-0.0513, -0.0584, 0.0230, ...]
'_teacher'	3145	[-0.0335, 0.0167, 0.0484, ...]
'_teaches'	11749	[-0.0151, -0.0516, 0.0309, ...]
'_the'	8	[-0.0498, -0.0428, 0.0275, ...]
'_student'	1236	[-0.0460, 0.0031, 0.0545, ...]
...	...	...

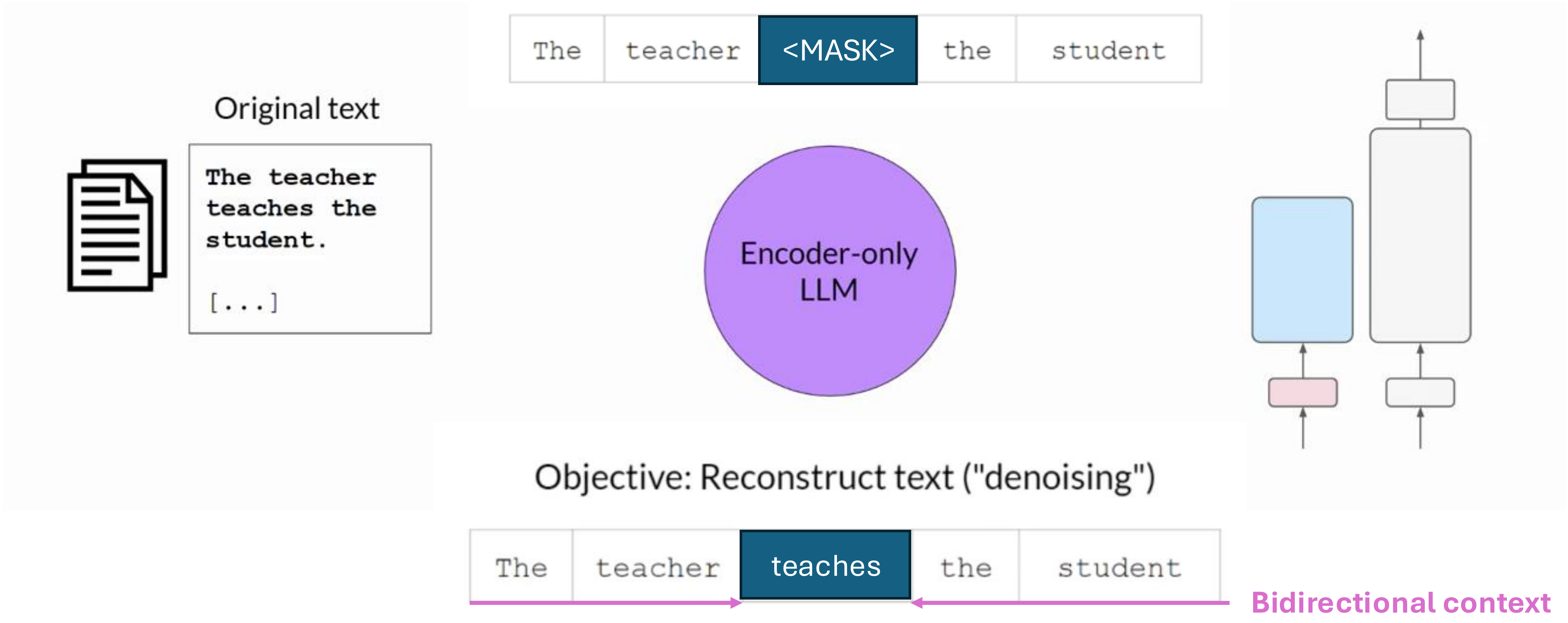
Vocabulary

Transformers



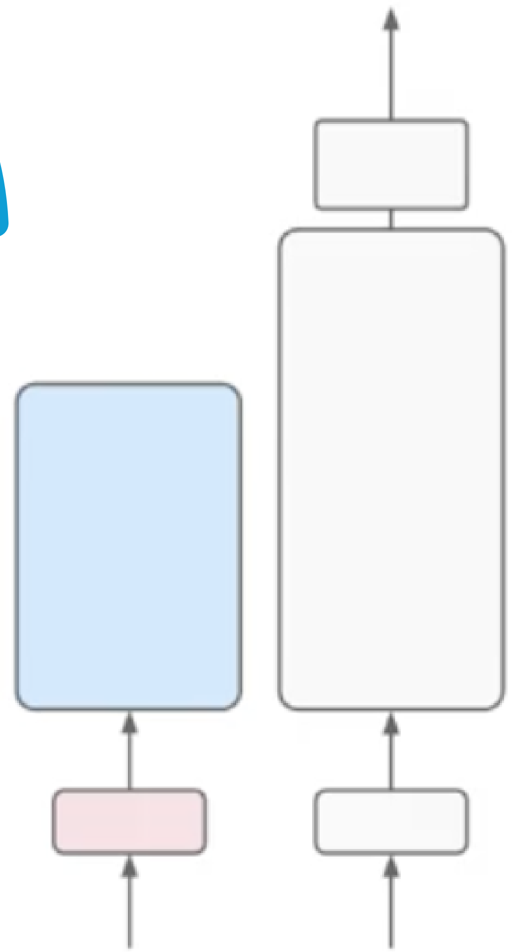
Autoencoding models

Masked Language Modeling (MLM)



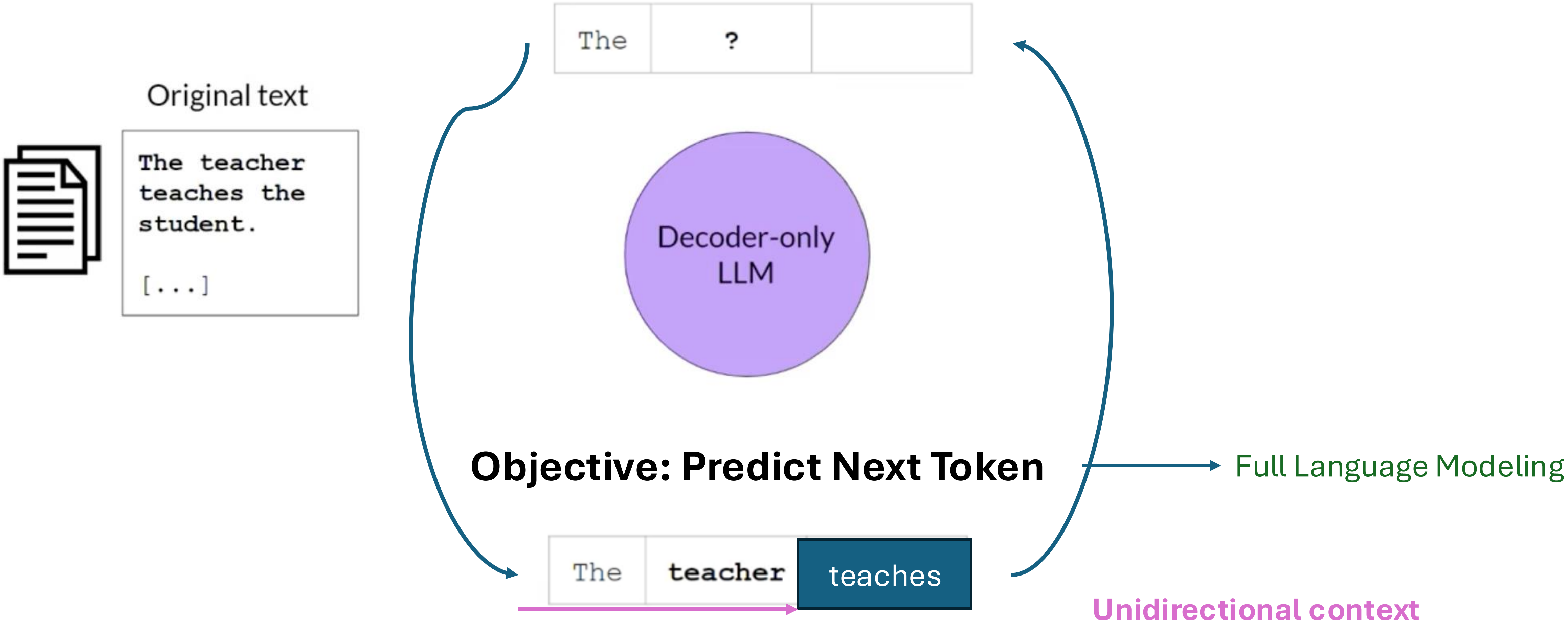
Autoencoding models

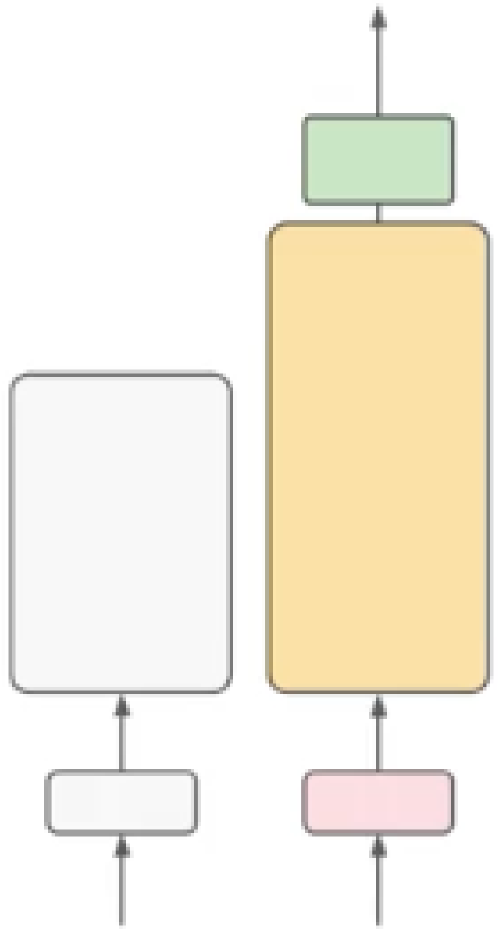
- Good use cases
  - Sentiment Analysis
  - Named entity recognition
  - Word classification
- Example models:
  - BERT
  - ROBERTA



Autoregressive Models

Causal Language Modeling (CLM)



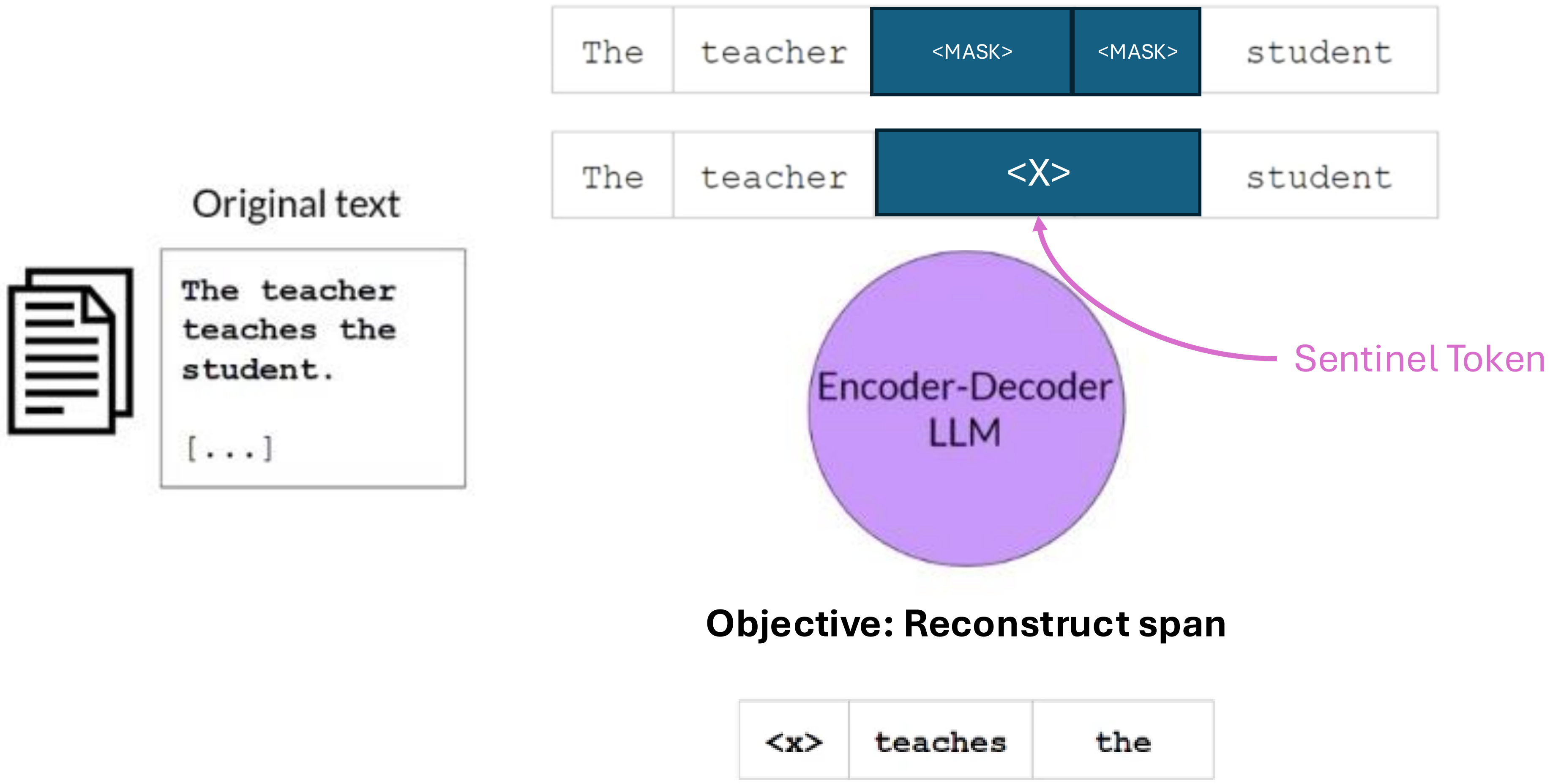


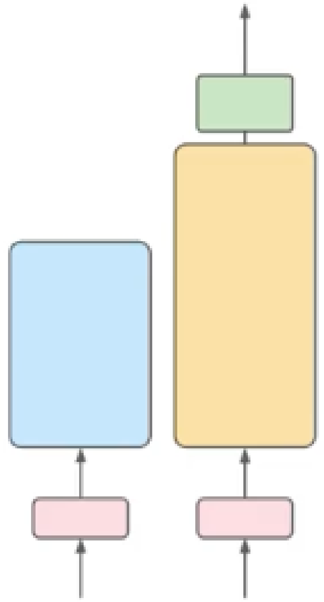
Autoregressive models

- Good use cases:
  - Text generation
  - Other emergent behavior
    - Depends on model size
- Example models:
  - GPT
  - BLOOM

Sequence-to-sequence models

Span Corruption

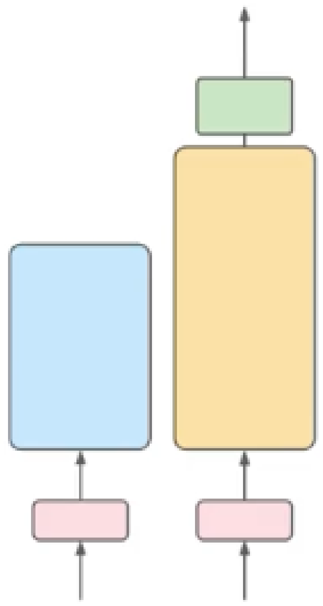




Sequence-to-sequence models

- Good use cases:
  - Translation
  - Text Summarization
  - Question answering

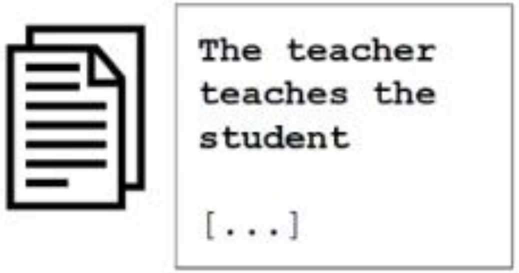
- Example models
  - T5
  - BART



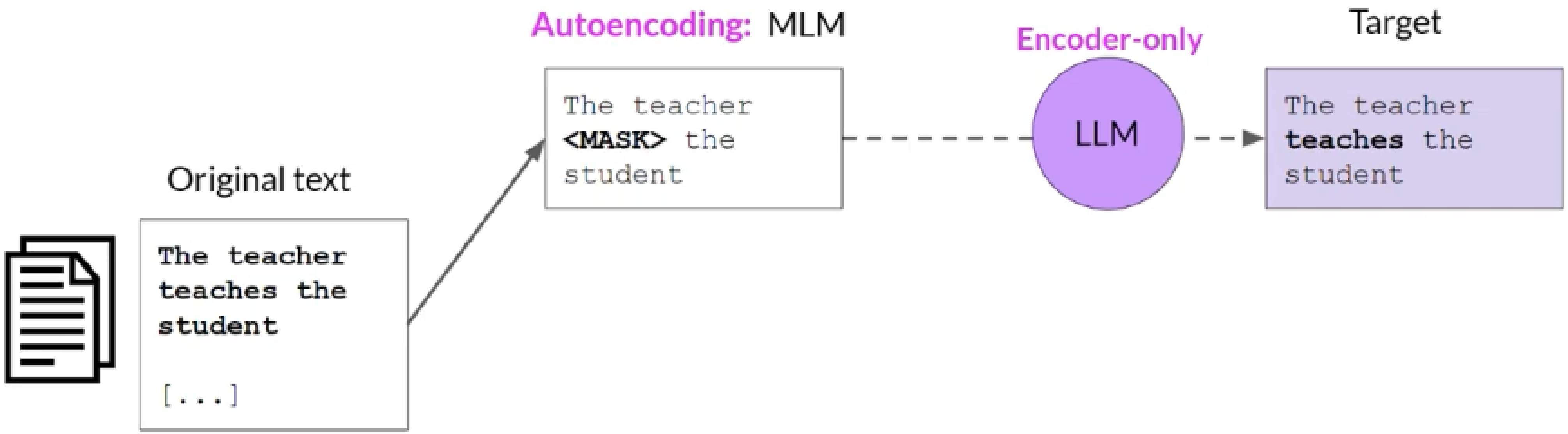
Model architectures and pre-training objectives

Target

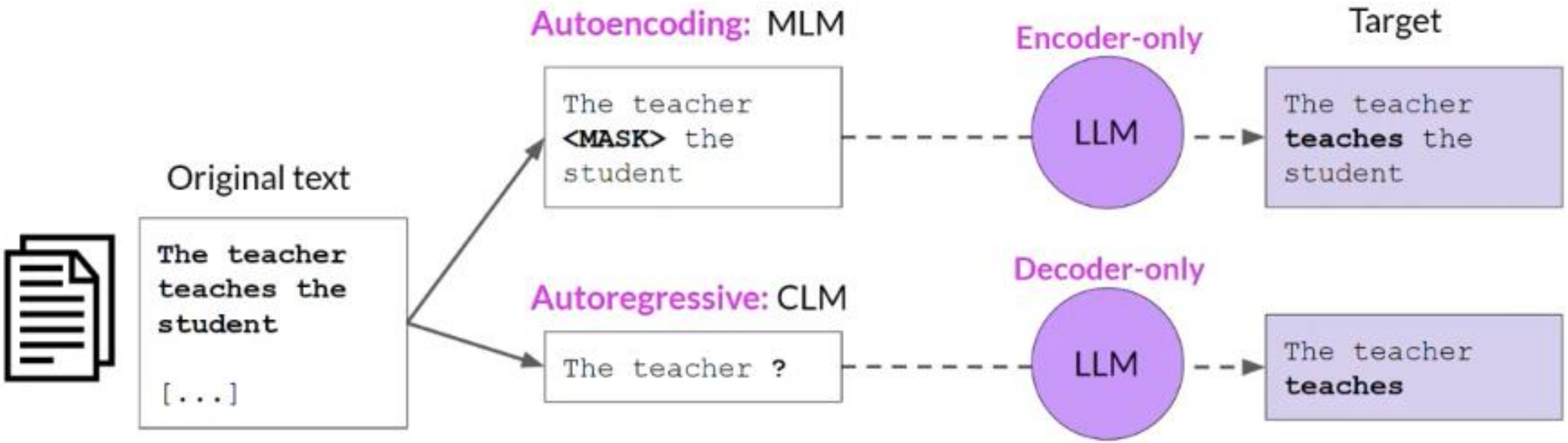
Original text



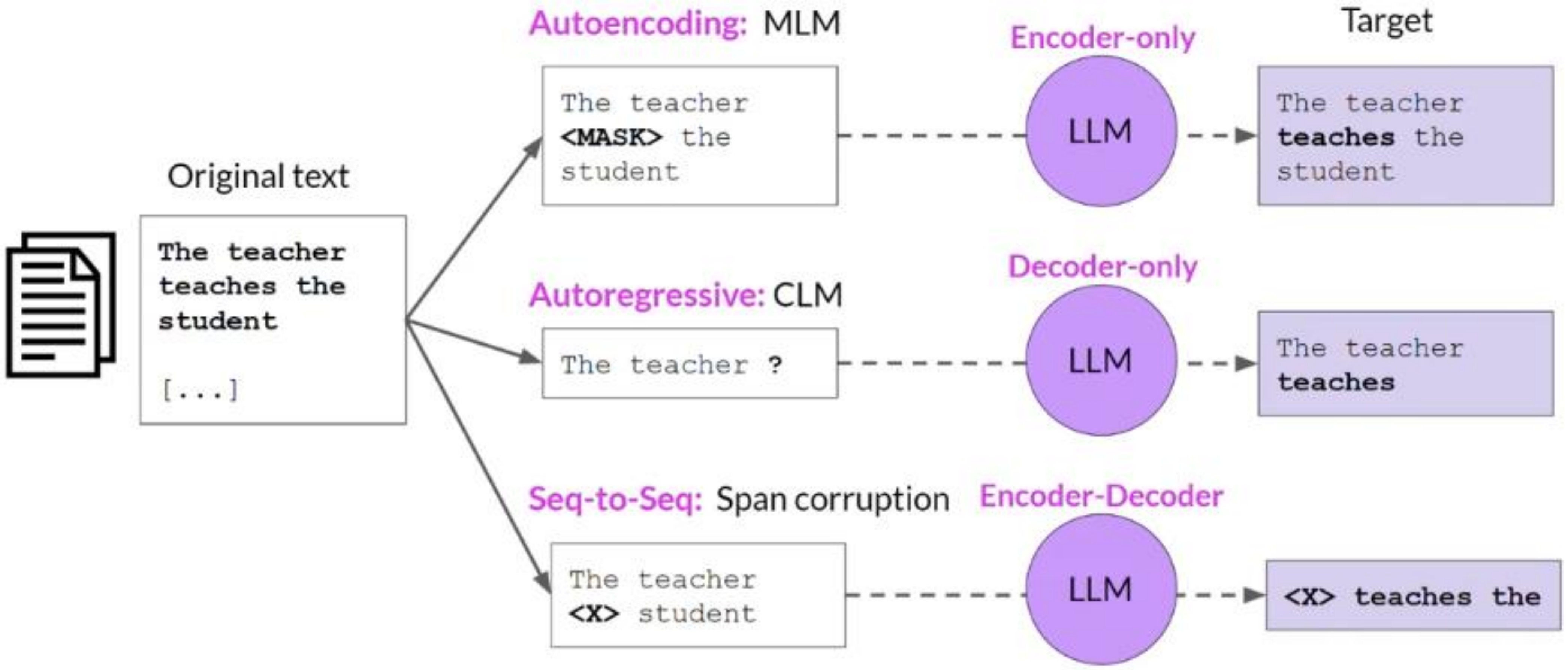
Model architectures and pre-training objectives



Model architectures and pre-training objectives



Model architectures and pre-training objectives



Question?

Large Language Models (LLMs) are capable of performing multiple tasks supporting a variety of use cases. Which of the following tasks supports the use case of converting code comments into executable code?

- o Translation
- o Invoke actions from text
- o Information Retrieval
- o Text summarization

Question?

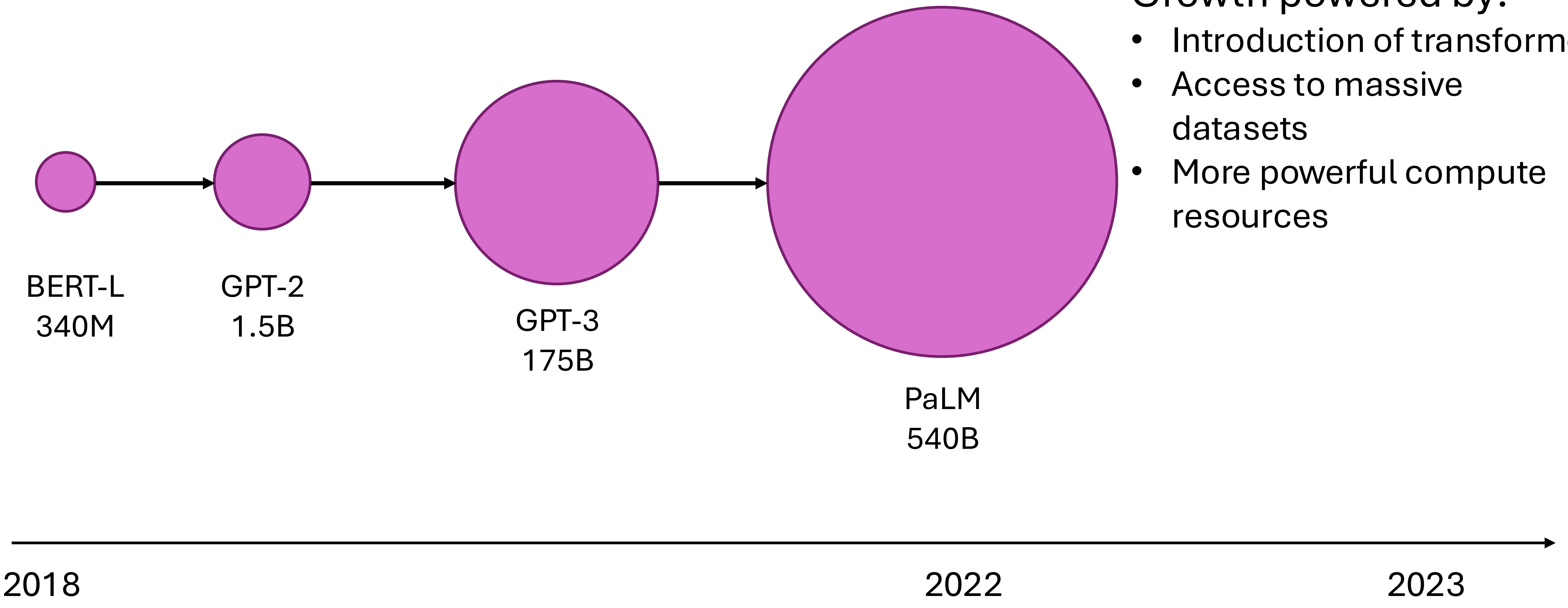
Which transformer-based model architecture is well-suited to the task of text translation?

- Autoencoder
- Sequence-to-sequence
- Autoregressive

The significance of scale: task ability

*Bert-base

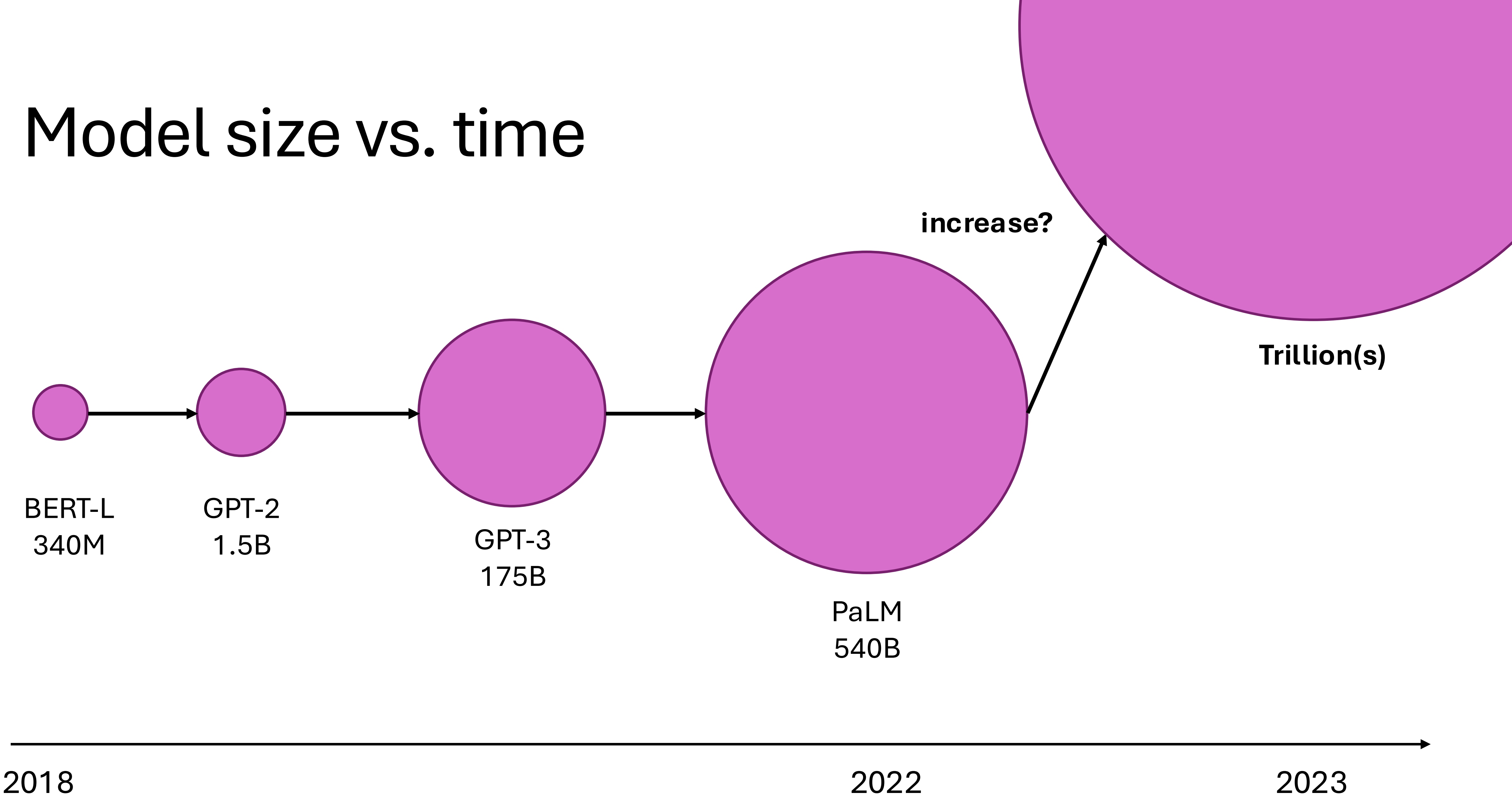
Model size vs. time



Growth powered by:

- Introduction of transformer
- Access to massive datasets
- More powerful compute resources

Model size vs. time



Computational Challenges

```
OutOfMemoryError: CUDA out of memory.
```

Approximate GPU RAM needed to store 1B parameters

1 parameter = 4 bytes (32-bit float)

1B parameters = 4×10^9 bytes =4 GB

4GB @ 32-bit
Full precision

References: https://github.com/bitsandbytes-foundation/bitsandbytes
https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory

Additional GPU RAM needed to train 1B parameters

~ 20 extra bytes per parameter

References: https://github.com/bitsandbytes-foundation/bitsandbytes
https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory

Approximate GPU RAM needed to train 1B-params

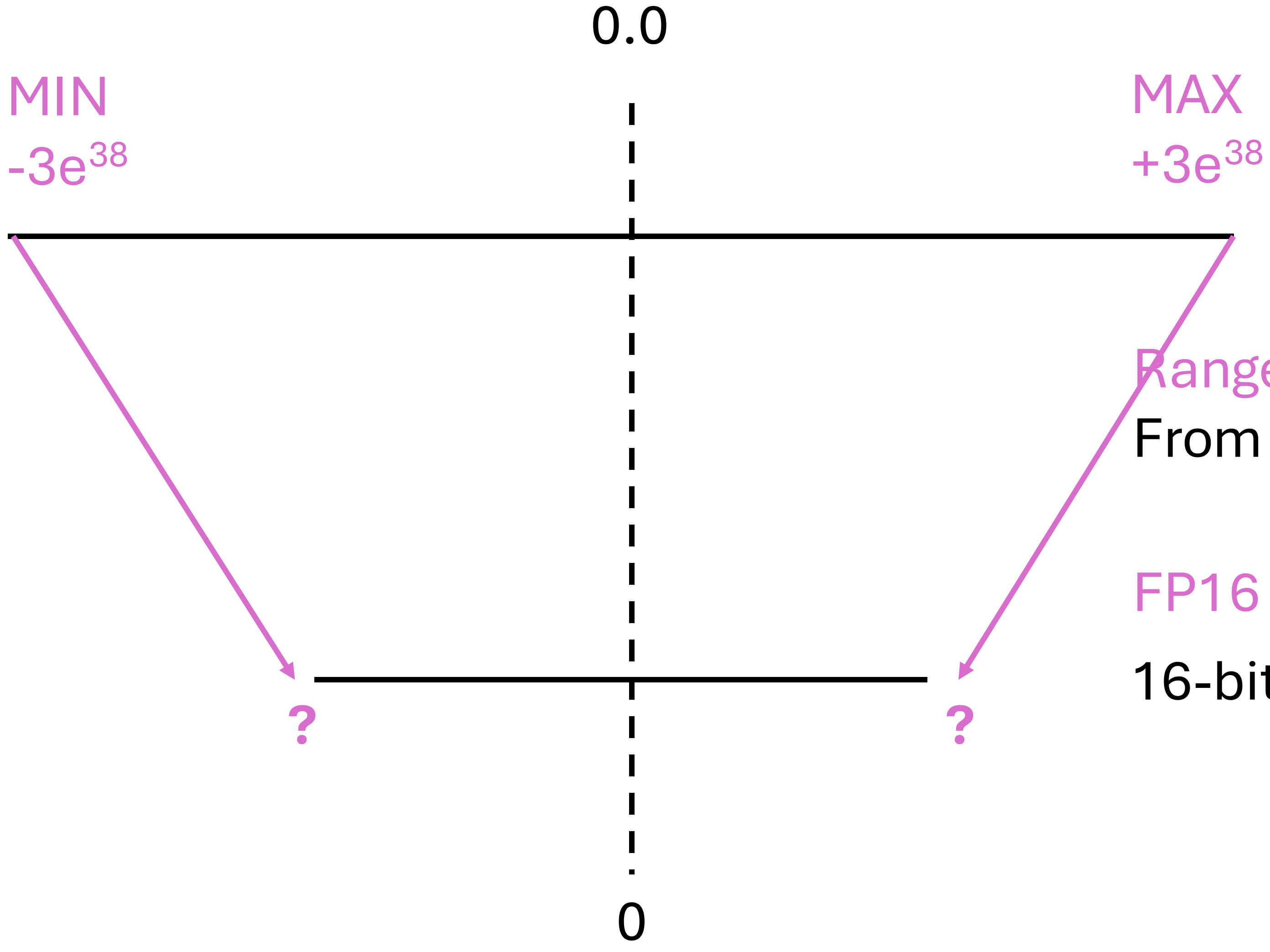
Memory needed to store model

4GB@32-bit
full precision

Memory needed to train model

24 GB @ 32-bit
full precision

Quantization



FP32

32-bit floating point

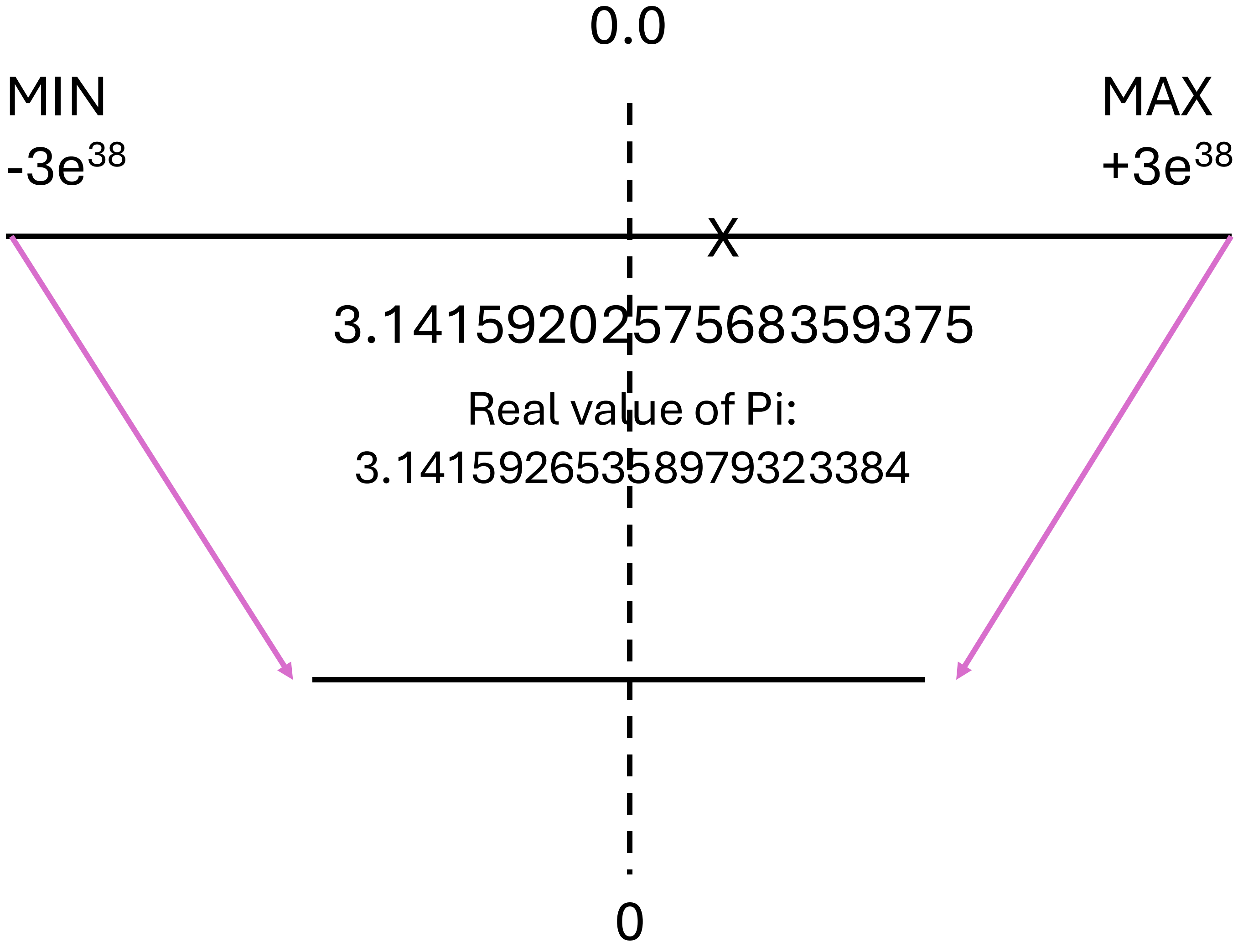
Range:

From $-3e^{38}$ to $+3e^{38}$

FP16 | BFLOAT16 | INT8

16-bit floating point | 8-bit integer

Quantization: FP32



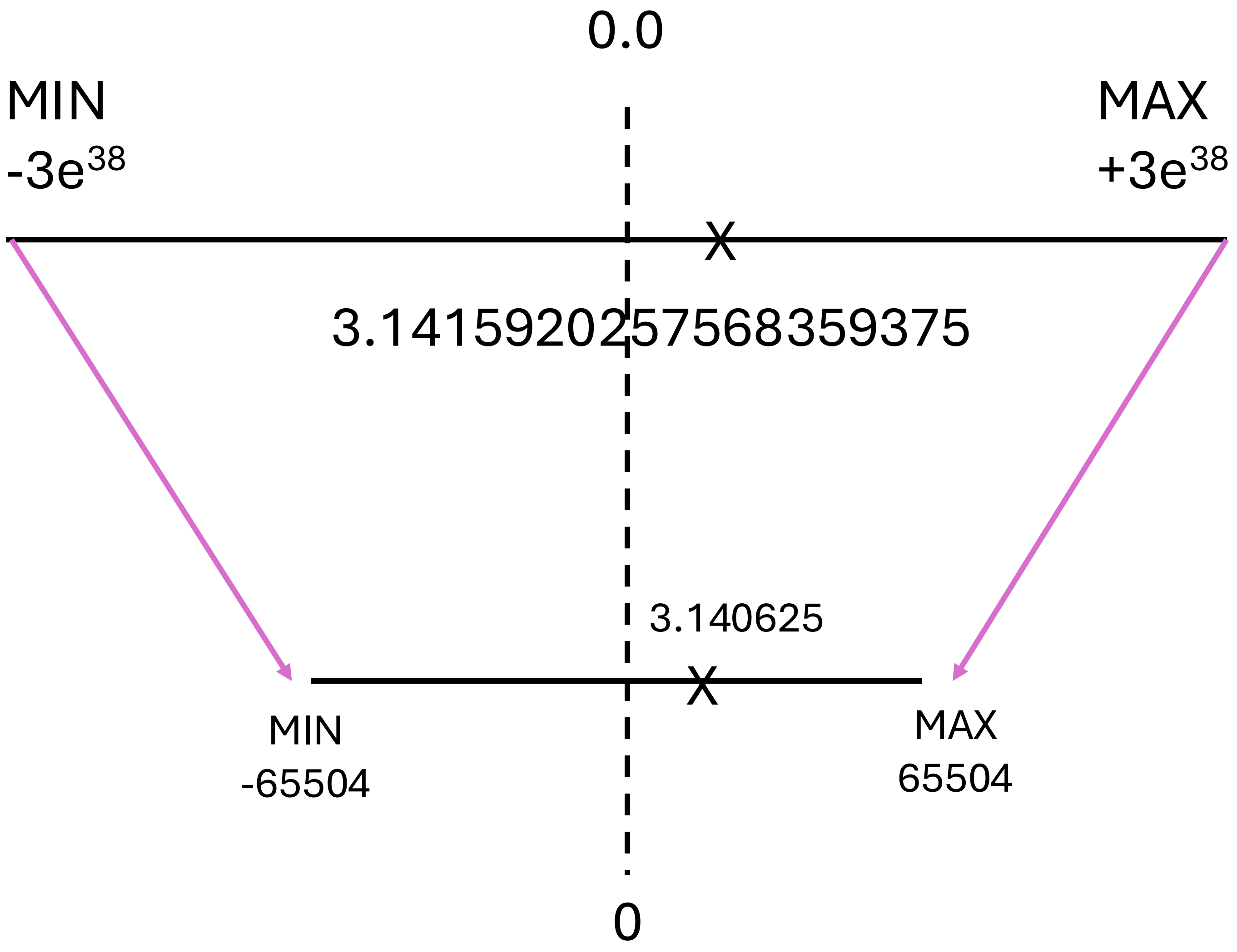
Let's store Pi: 3.141592

FP32 4 bytes memory

$$\underbrace{0}_{\text{Sign 1 bit}} \quad \underbrace{10000000}_{\text{Exponent 8 bits}} \quad \underbrace{10010010000111111011000}_{\text{Fraction 23 bits}}$$

Mantissa/Significand
= Precision

Quantization: FP16



Let's store Pi: 3.141592

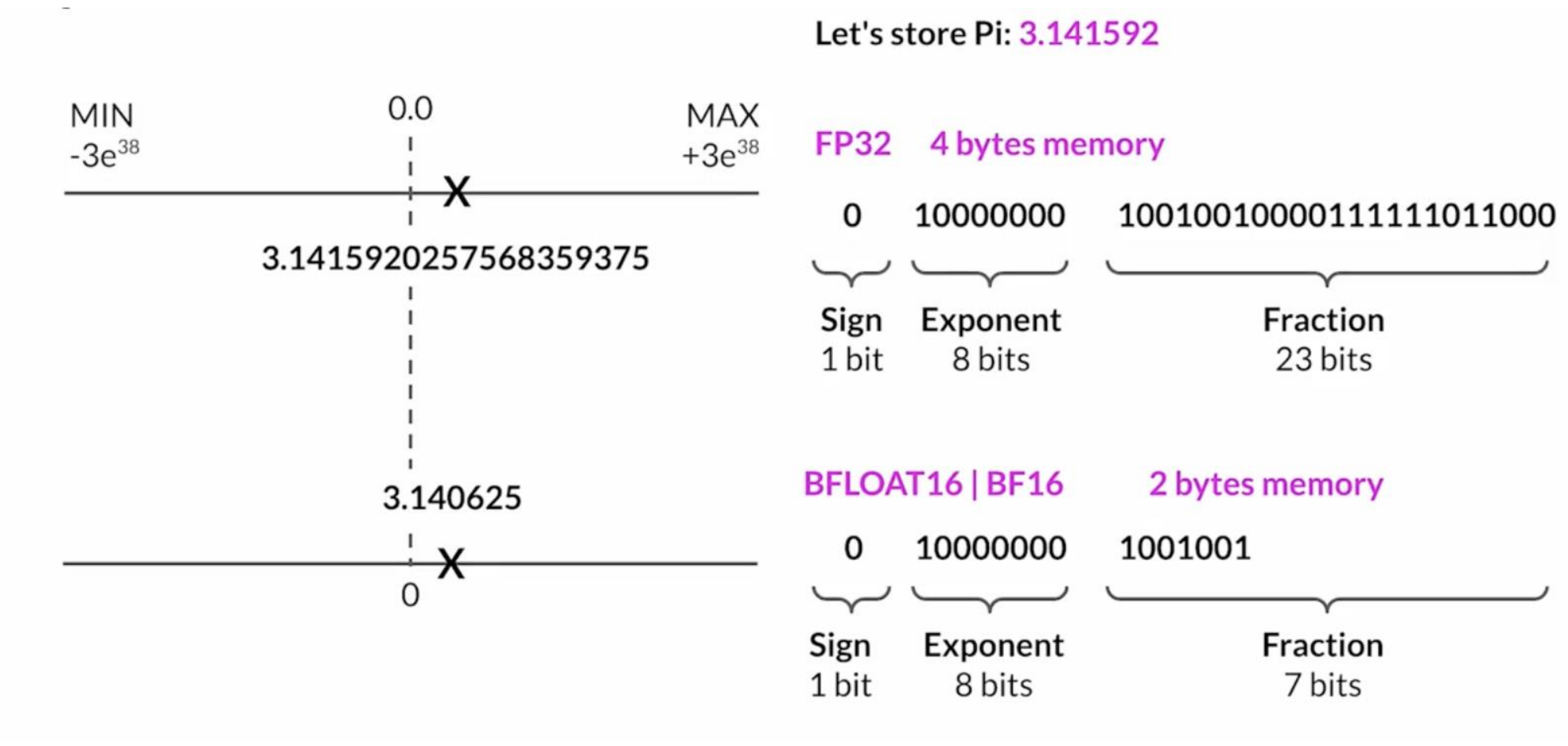
FP32 4 bytes memory

$$\underbrace{0}_{\substack{\textbf{Sign}\\ \textbf{1 bit}}} \quad \underbrace{10000000}_{\substack{\textbf{Exponent}\\ \textbf{8 bits}}} \quad \underbrace{10010010000111111011000}_{\substack{\textbf{Fraction}\\ \textbf{23 bits}}}$$

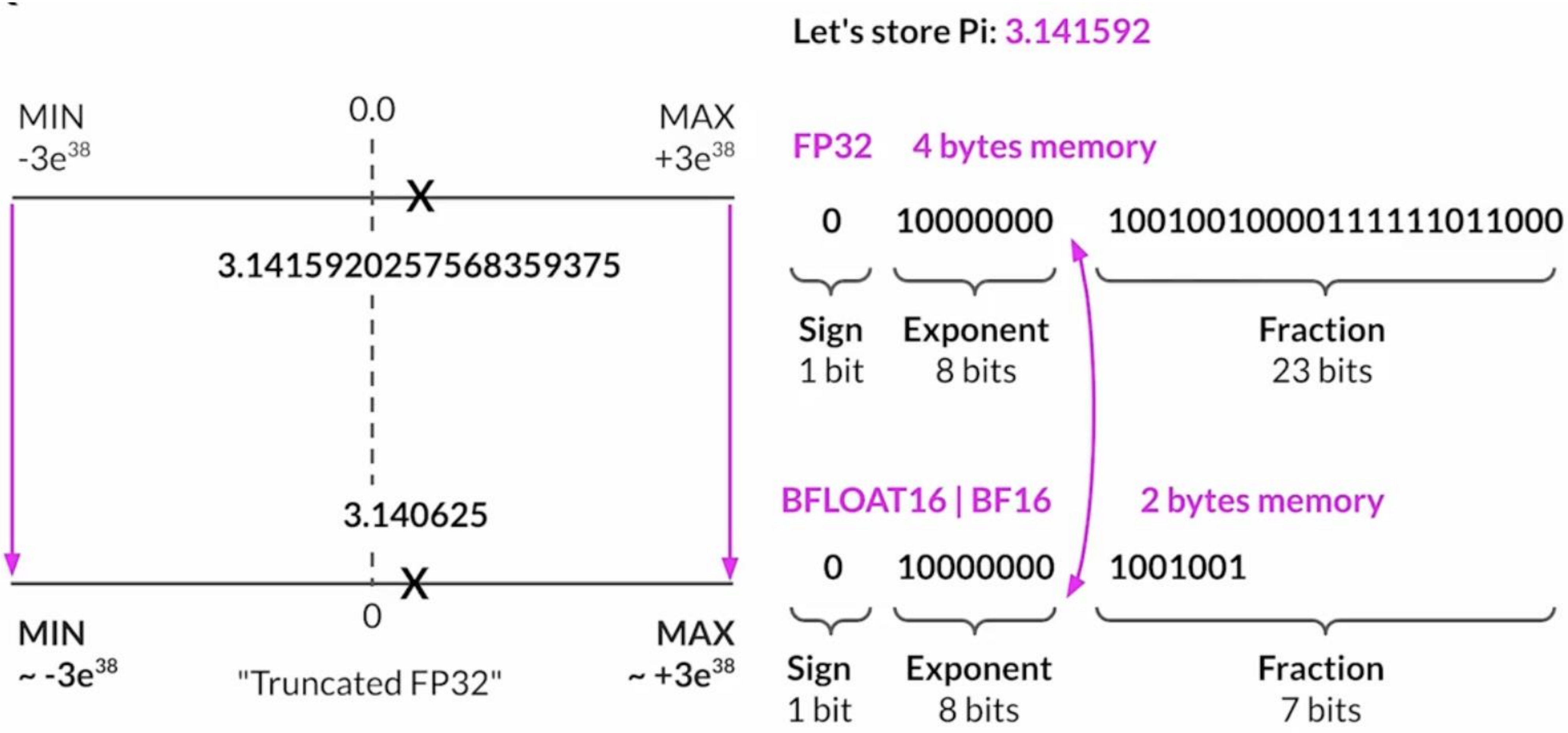
FP16 2 bytes memory

$$\underbrace{0}_{\substack{\textbf{Sign}\\ \textbf{1 bit}}} \quad \underbrace{10000}_{\substack{\textbf{Exponent}\\ \textbf{5 bits}}} \quad \underbrace{1001001000}_{\substack{\textbf{Fraction}\\ \textbf{10 bits}}}$$

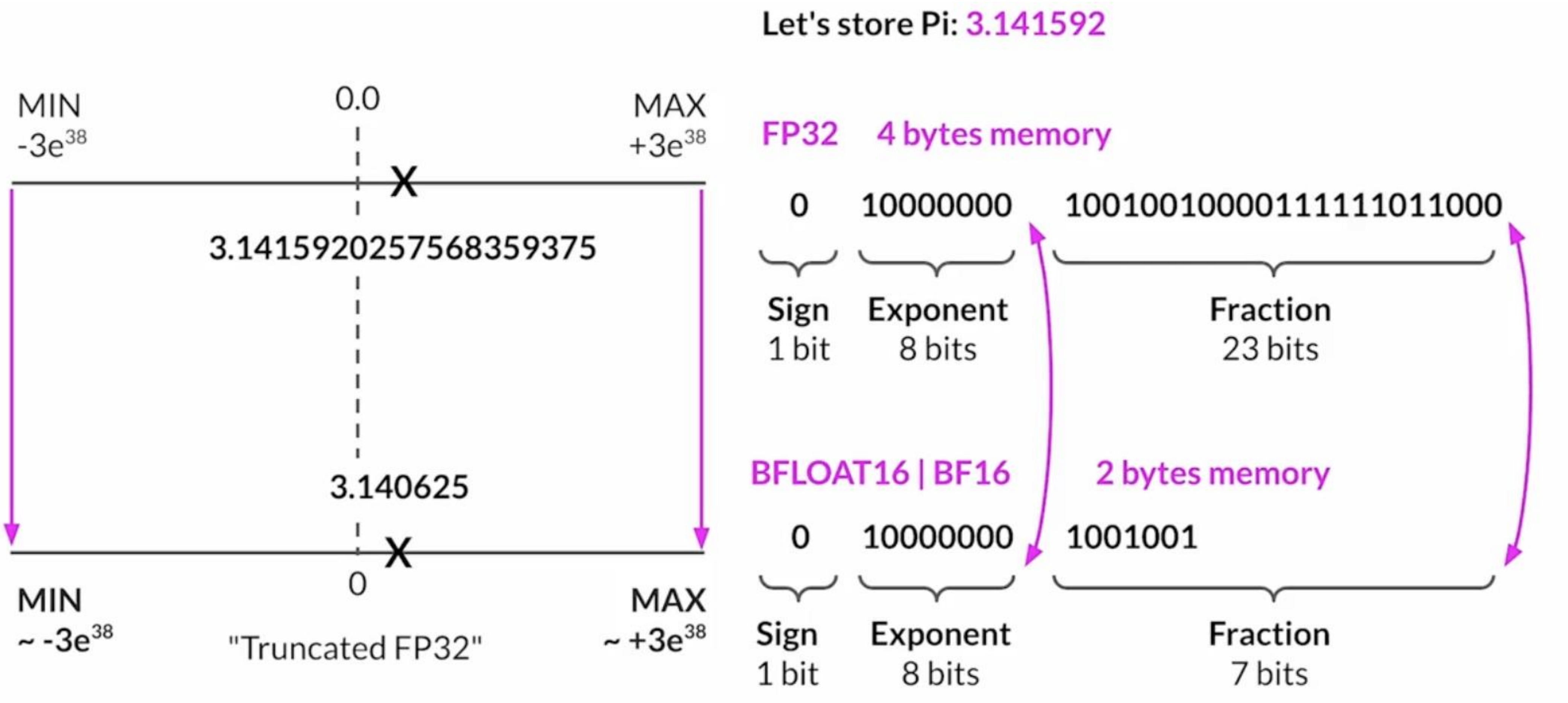
Quantization: BFLOAT16



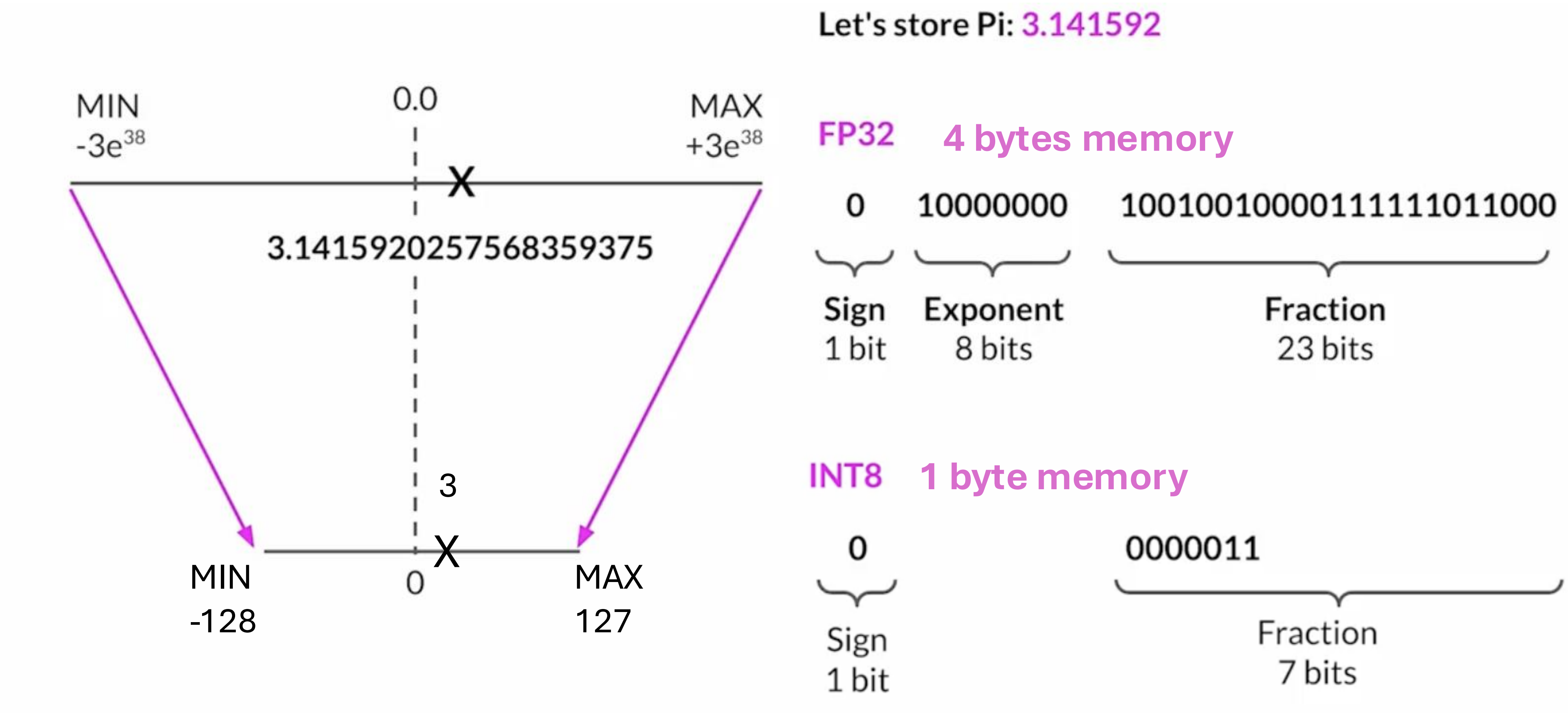
Quantization: BFLOAT16



Quantization: BFLOAT16



Quantization: INT8



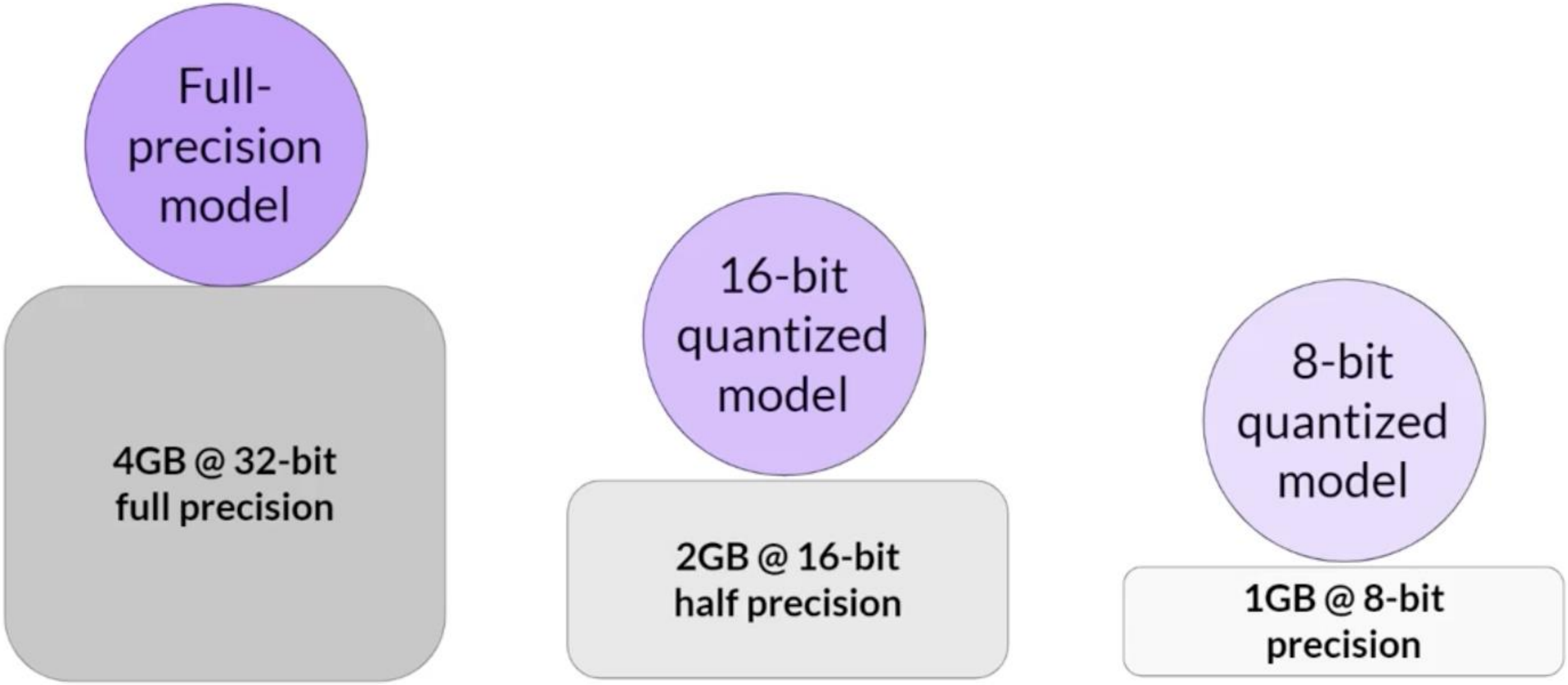
Quantization: Summary

	Bits	Exponent	Fraction	Memory needed to store one value
FP32	32	8	23	4 bytes
FP16	16	5	10	2 bytes
BFLOAT16	16	8	7	2 bytes
INT8	8	-/-	7	1 byte

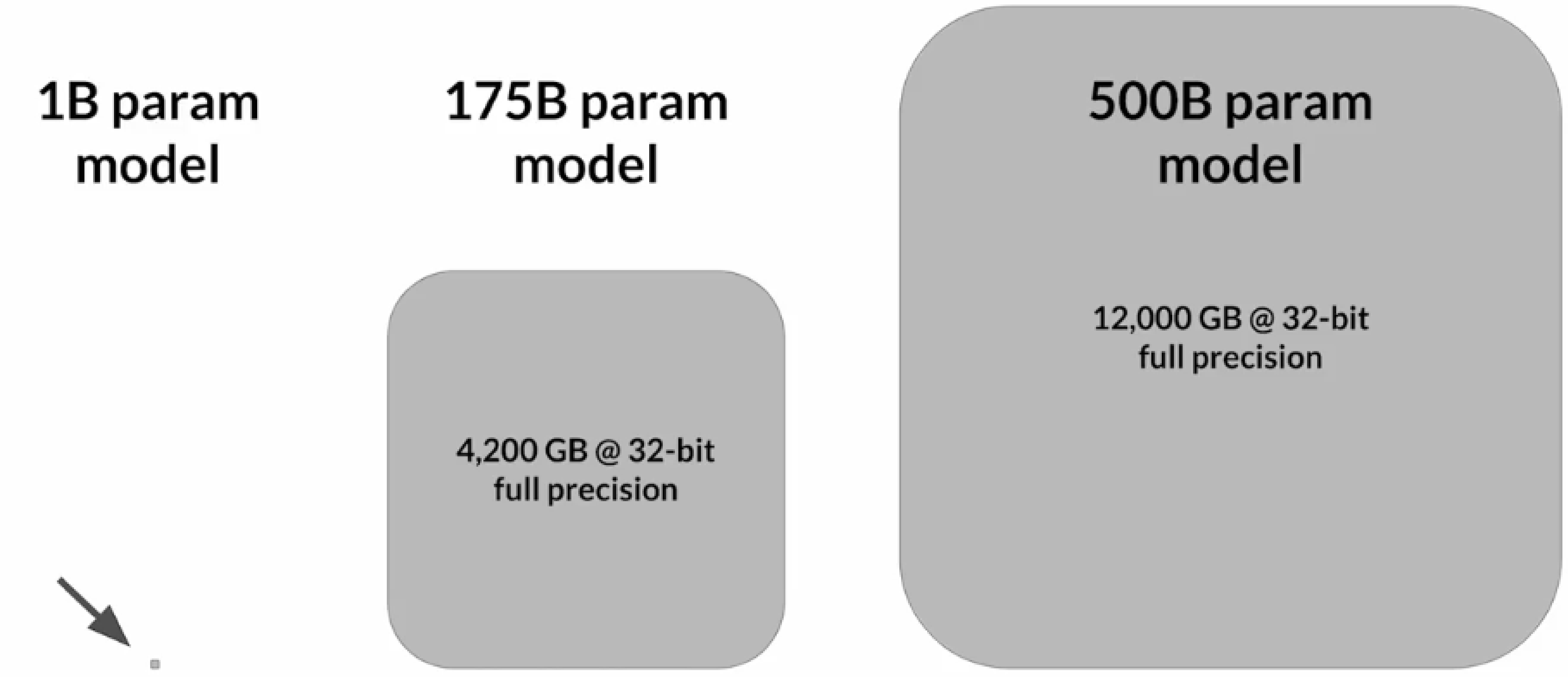
FLAN T5

- Reduce required memory to store and train models.
- Statistically projects 32-bit floating point numbers into lower precision spaces .
- Quantization-aware training (QAT) learns the quantization scaling factors during training.
- BFLOAT16 is a popular choice.

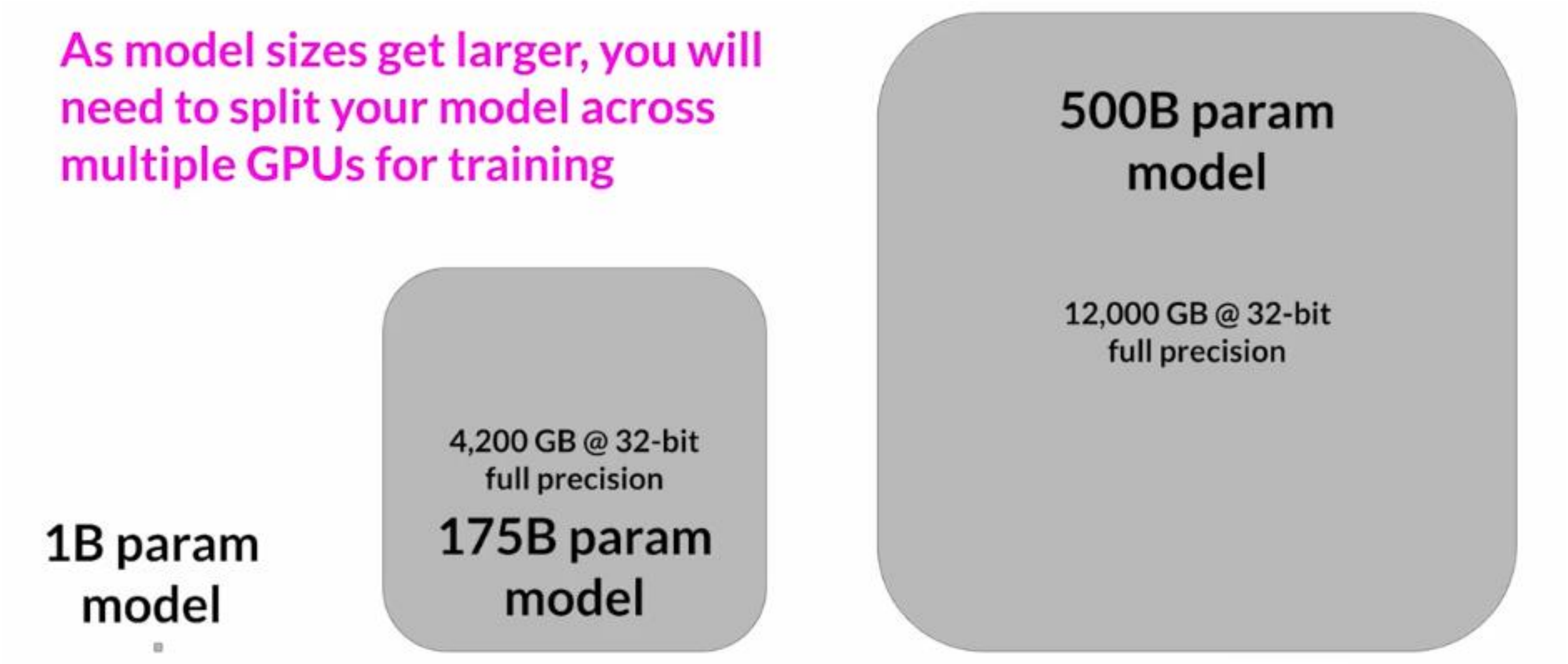
Approximate GPU RAM needed to store 1B parameters



GPU RAM needed to train larger models

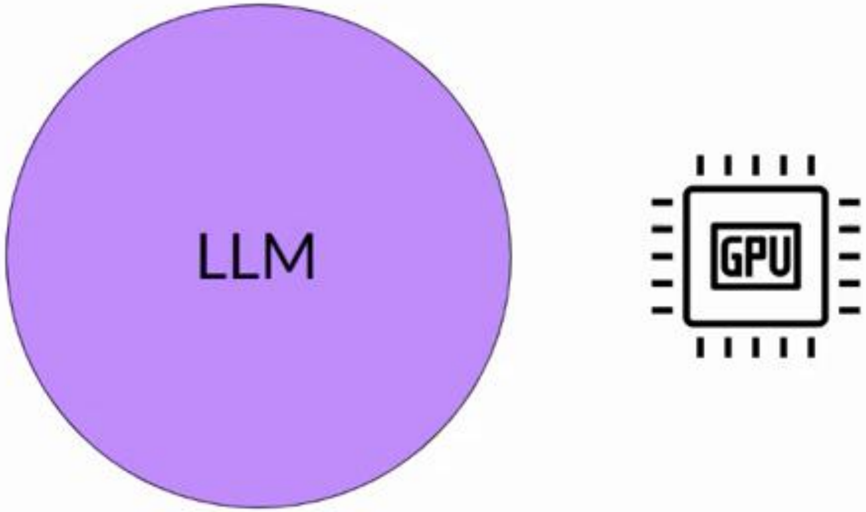


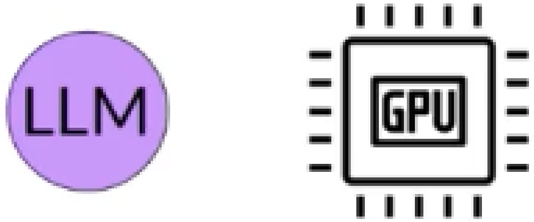
GPU RAM needed to train larger models



Efficient Multi-GPU Compute Strategies

When to use distributed compute



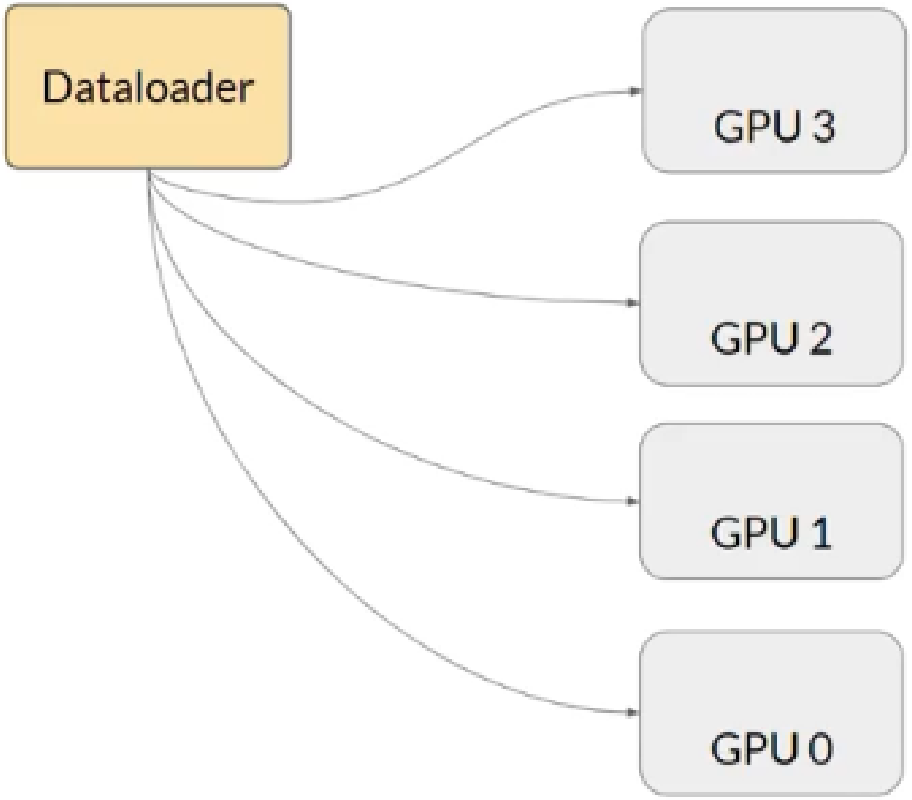


Model too big for single GPU

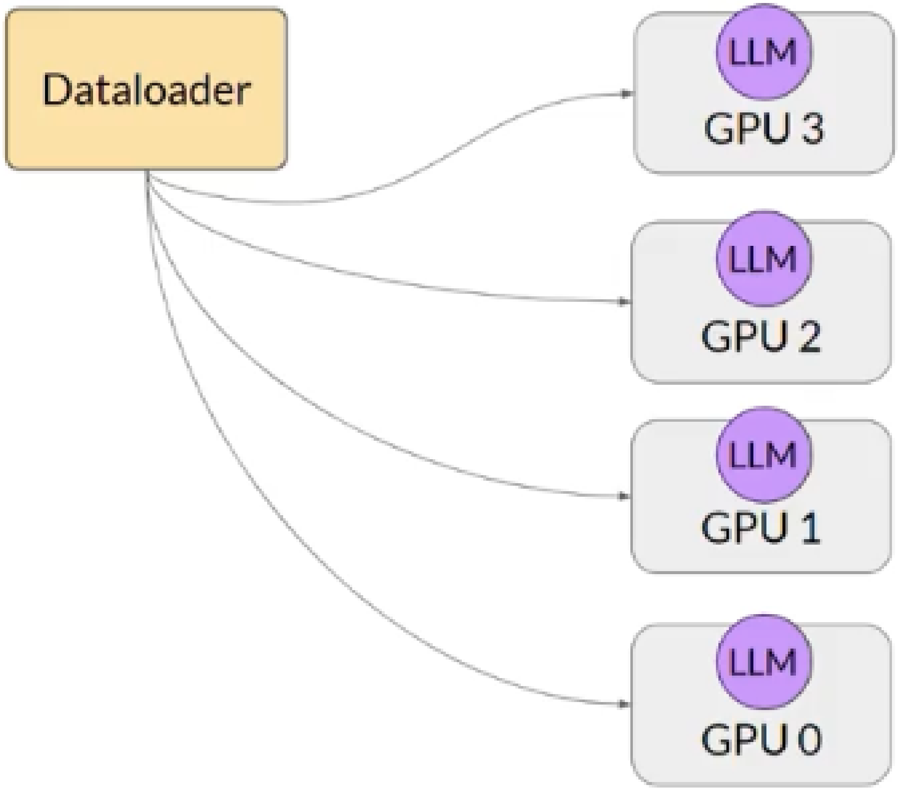
Distributed Data Parallel (DDP)

Dataloader

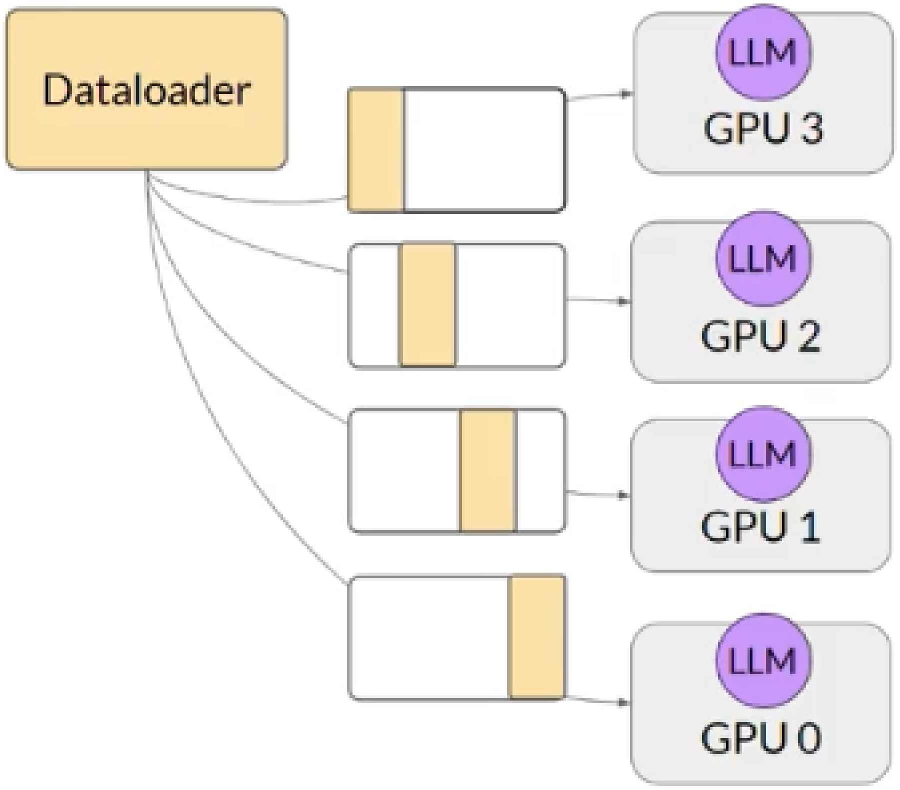
Distributed Data Parallel (DDP)



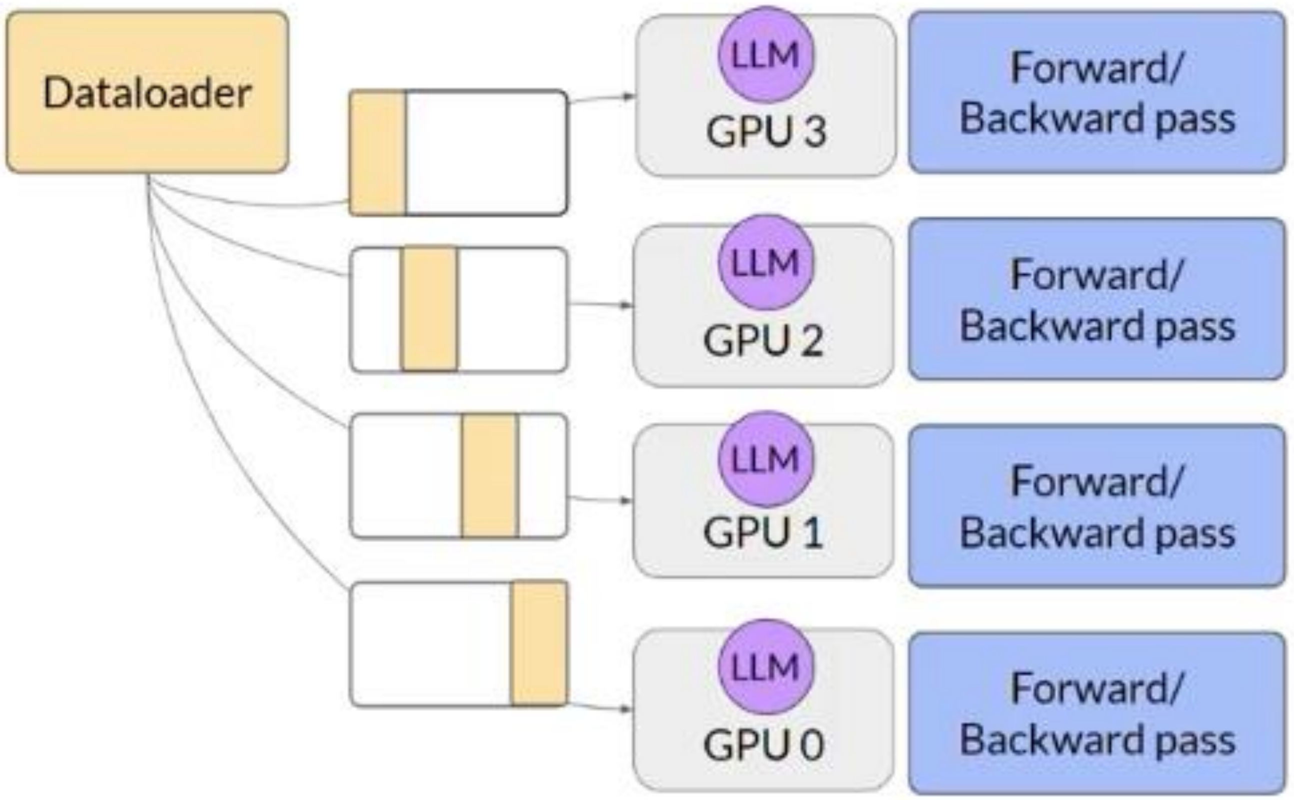
Distributed Data Parallel (DDP)



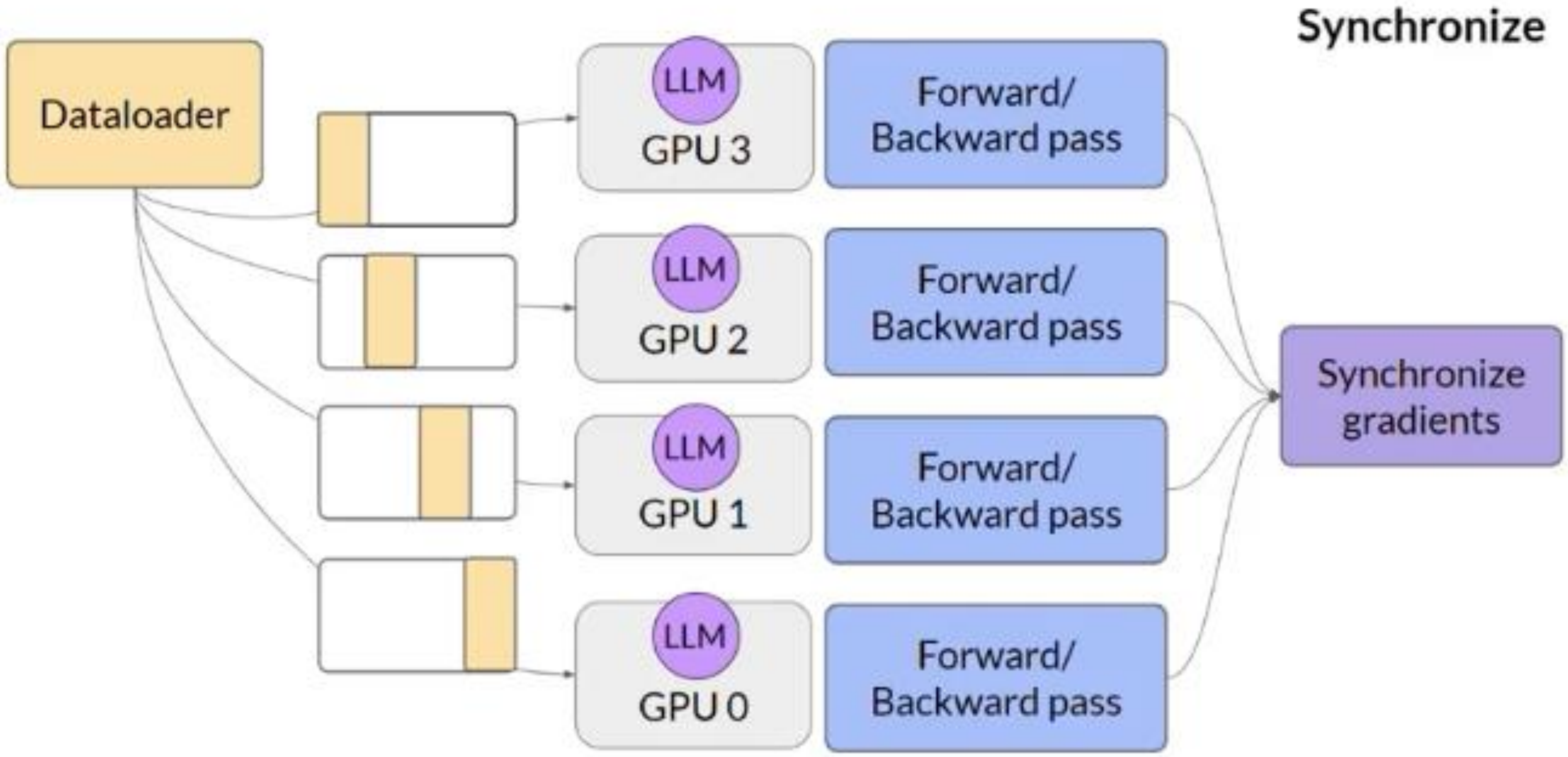
Distributed Data Parallel (DDP)



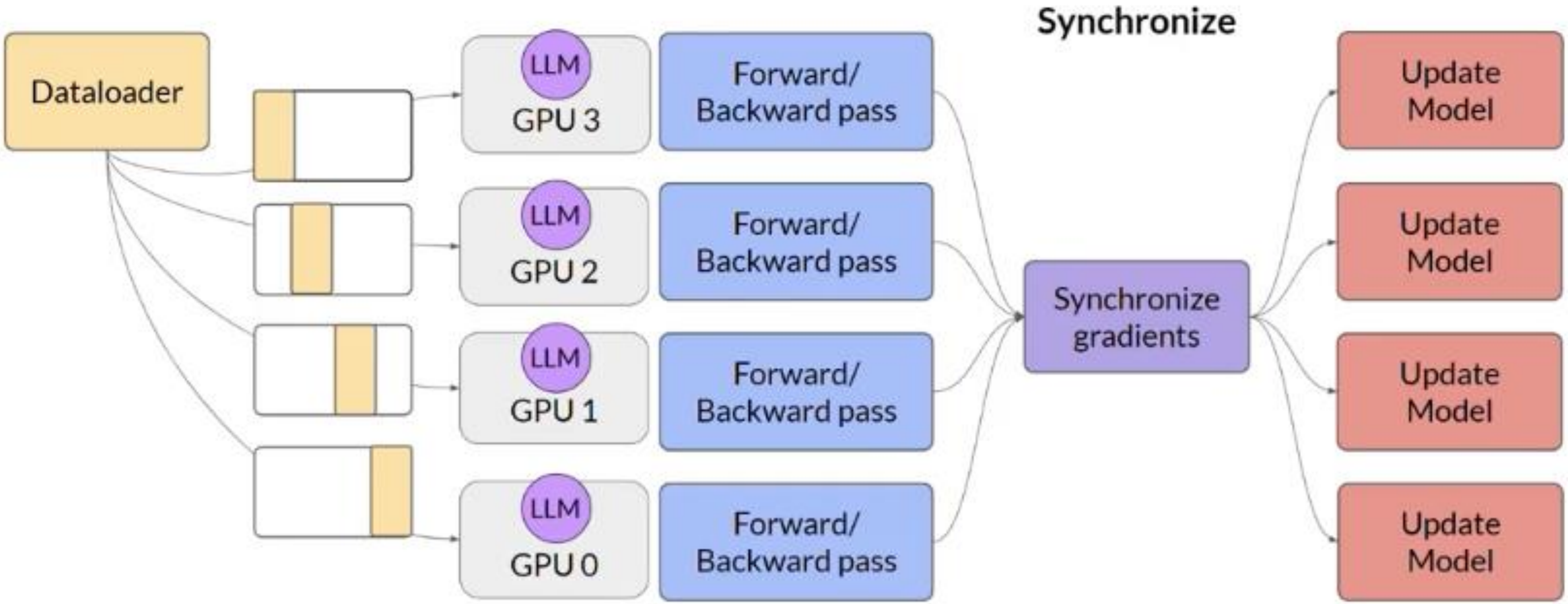
Distributed Data Parallel (DDP)



Distributed Data Parallel (DDP)



Distributed Data Parallel (DDP)



Fully Sharded Data Parallel (FSDP)

- Motivated by the “ZeRO” paper – zero data overlap between GPUs

ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

Samyam Rajbhandari*, Jeff Rasley*, Olatunji Ruwase, Yuxiong He
{samyamr, jerasley, olruwase, yuxhe}@microsoft.com

ABSTRACT

Large deep learning models offer significant accuracy gains, but training billions to trillions of parameters is challenging. Existing solutions such as data and model parallelisms exhibit fundamental limitations to fit these models into limited device memory, while obtaining computation, communication and development efficiency. We develop a novel solution, Zero Redundancy Optimizer (*ZeRO*), to optimize memory, vastly improving training speed while increasing the model size that

common settings like mixed precision and ADAM optimizer [6]. Other existing solutions such as Pipeline Parallelism (PP), Model Parallelism (MP), CPU-Offloading, etc, make trade-offs between functionality, usability, as well as memory and compute/communication efficiency, all of which are crucial to training with speed and scale.

Among different existing solution for training large models, MP is perhaps the most promising one. The largest models in the current literature, the 11B T5 model [5], and Megatron-

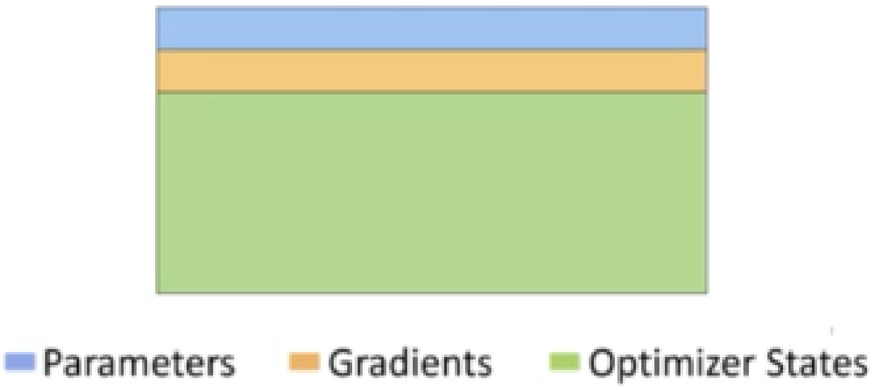
S. Rajbhandari, J. Rasley, O. Ruwase and Y. He, "ZeRO: Memory optimizations Toward Training Trillion Parameter Models," *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, Atlanta, GA, USA, 2020, pp. 1-16, doi: 10.1109/SC41405.2020.00024.

Recap: Additional GPU RAM needed for training

	Bytes per parameter
Model Parameters (Weights)	4 bytes per parameter
Adam optimizer (2 states)	+8 bytes per parameter
Gradients	+4 bytes per parameter
Activations and temp memory (variable size)	+8 bytes per parameter (high-end estimate)
TOTAL	**=4 bytes per parameter +20 extra bytes per parameter**

Memory usage in DDP

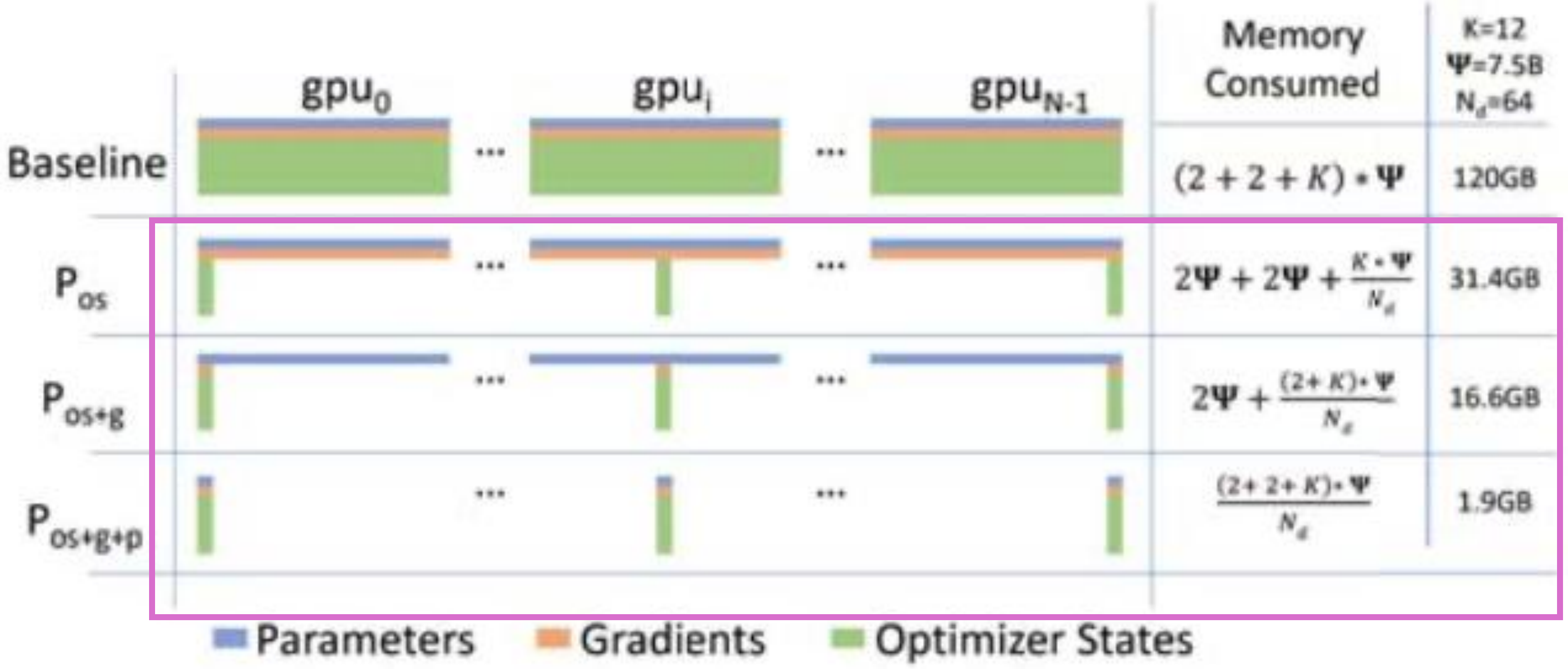
- One full copy of model and training parameters on each GPU

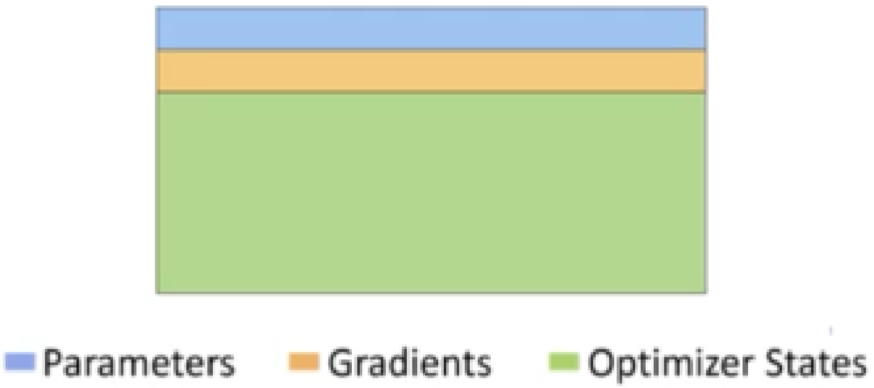


Sources:
Rajbhandari et al. 2019: "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models"
Zhao et al. 2023: "PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel"

Zero Redundancy Optimizer (ZeRO)

- Reduces memory by distributing (sharding) the model

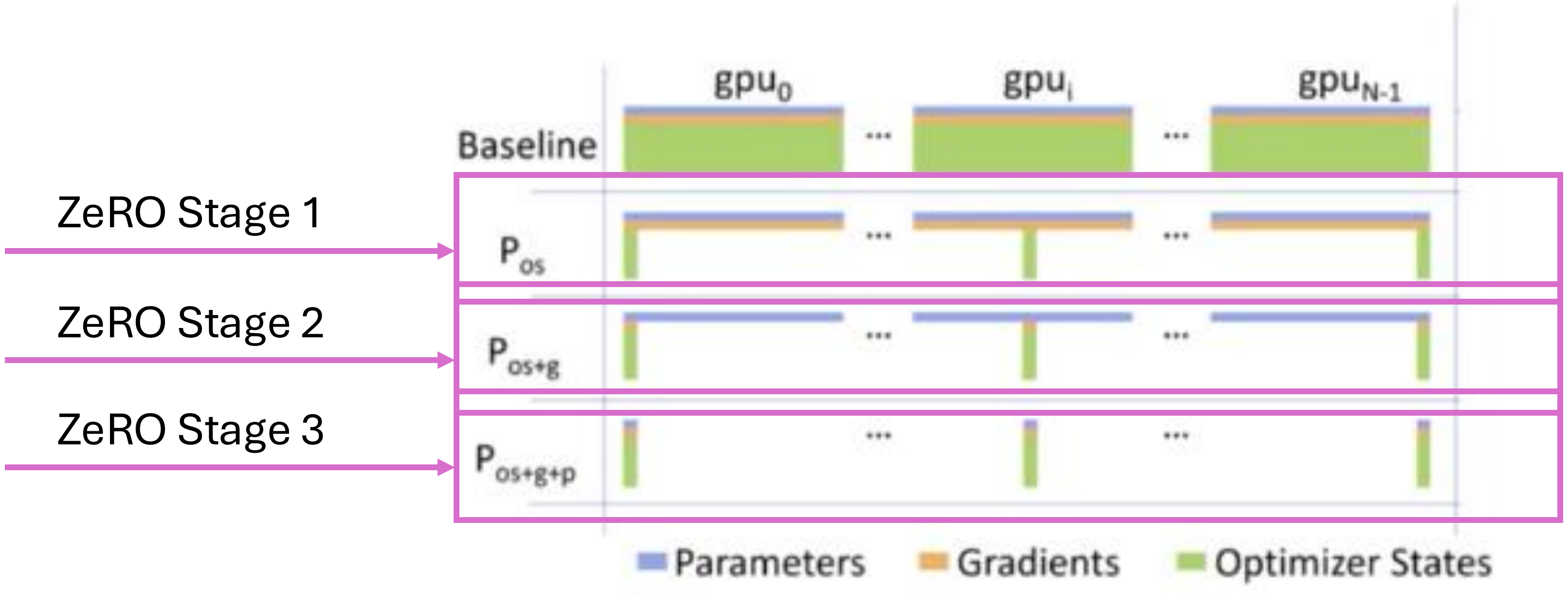




Sources:
Rajbhandari et al. 2019: "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models"
Zhao et al. 2023: "PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel"

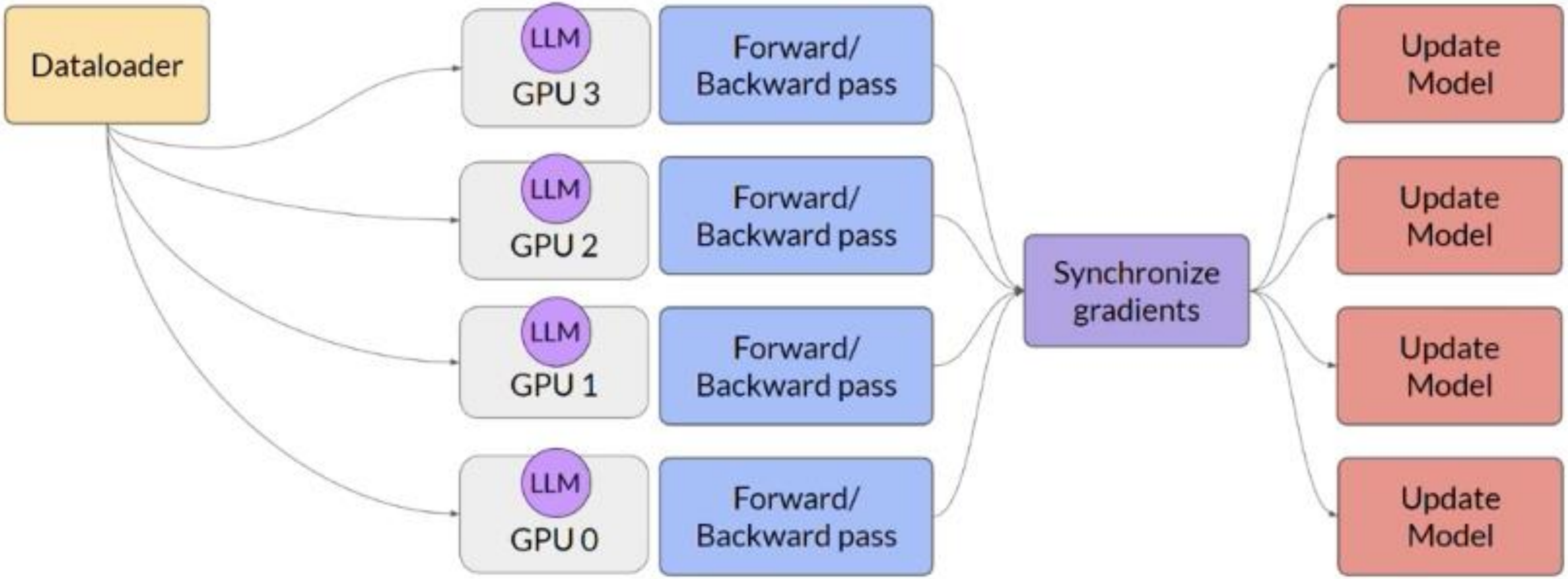
Zero Redundancy Optimizer (ZeRO)

- Reduces memory by distributing (sharding) the model parameters, gradients

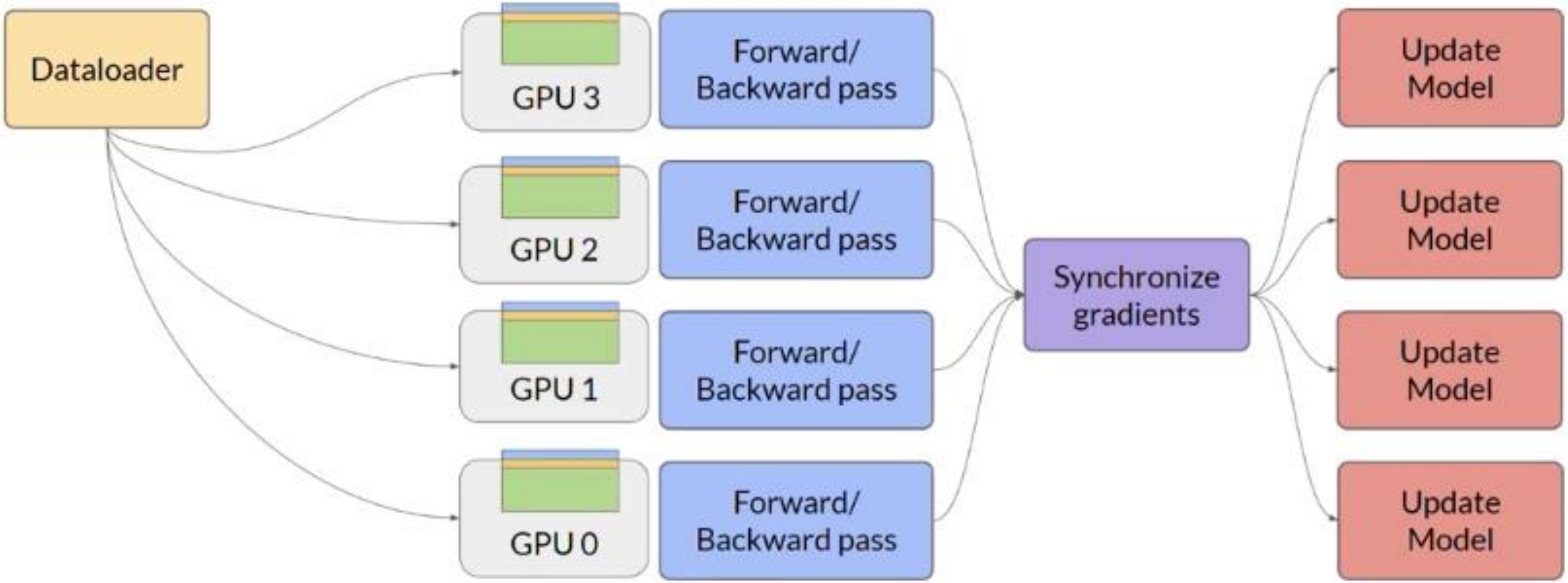


Sources:
Rajbhandari et al. 2019: "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models"
Zhao et al. 2023: "PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel"

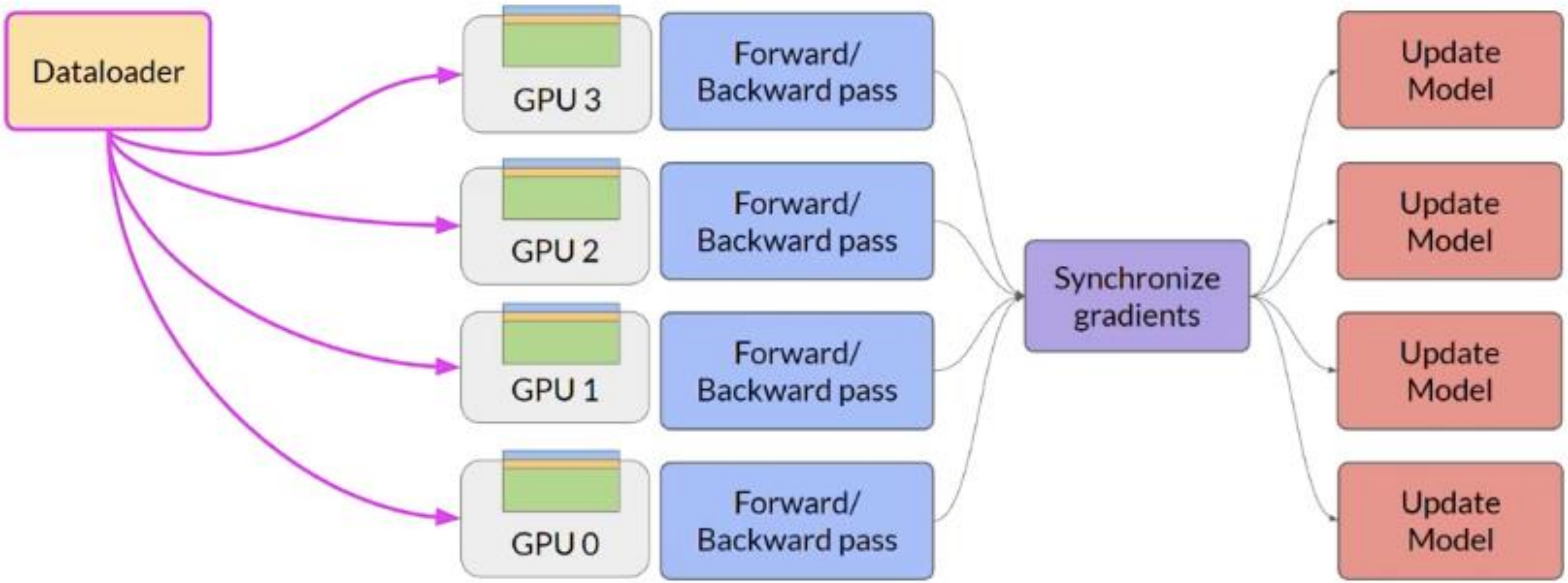
Distributed Data Parallel (DDP)



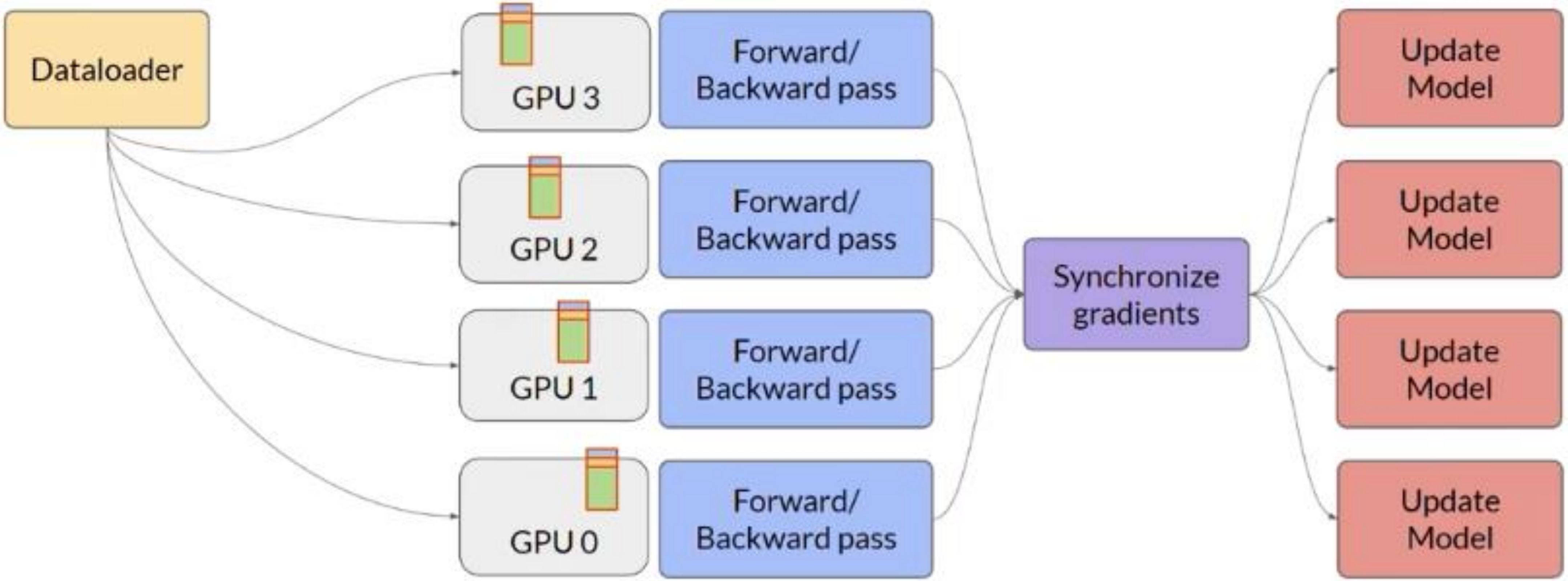
Distributed Data Parallel (DDP)



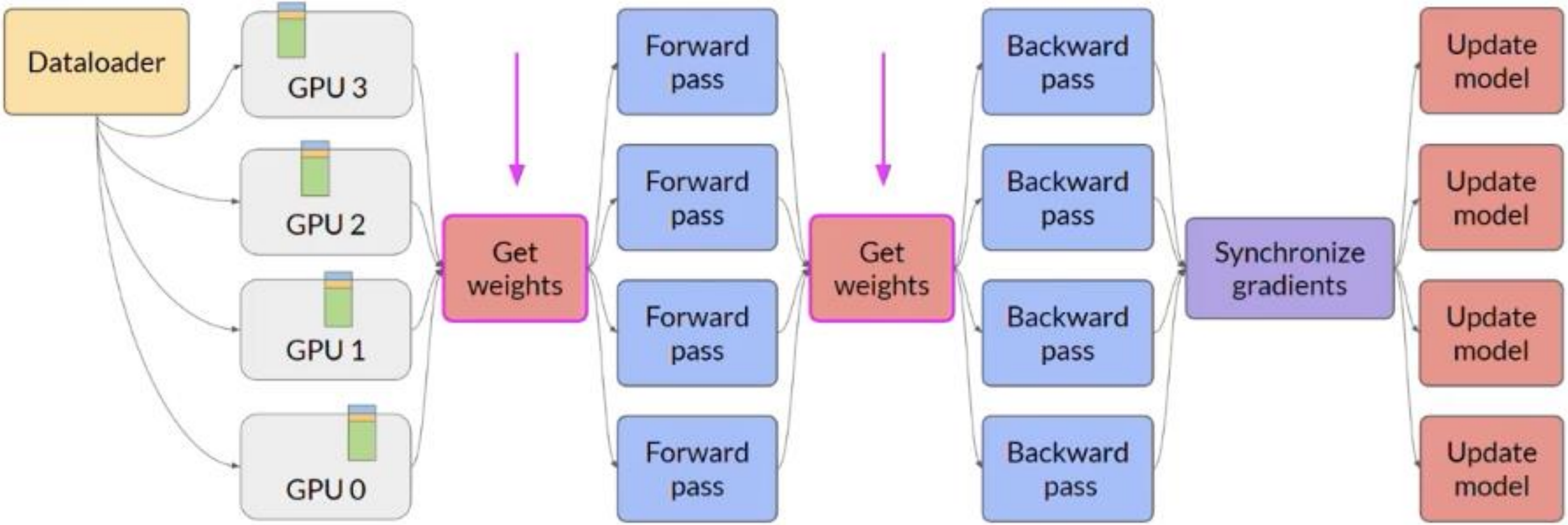
Fully Sharded Data Parallel (FSDP)



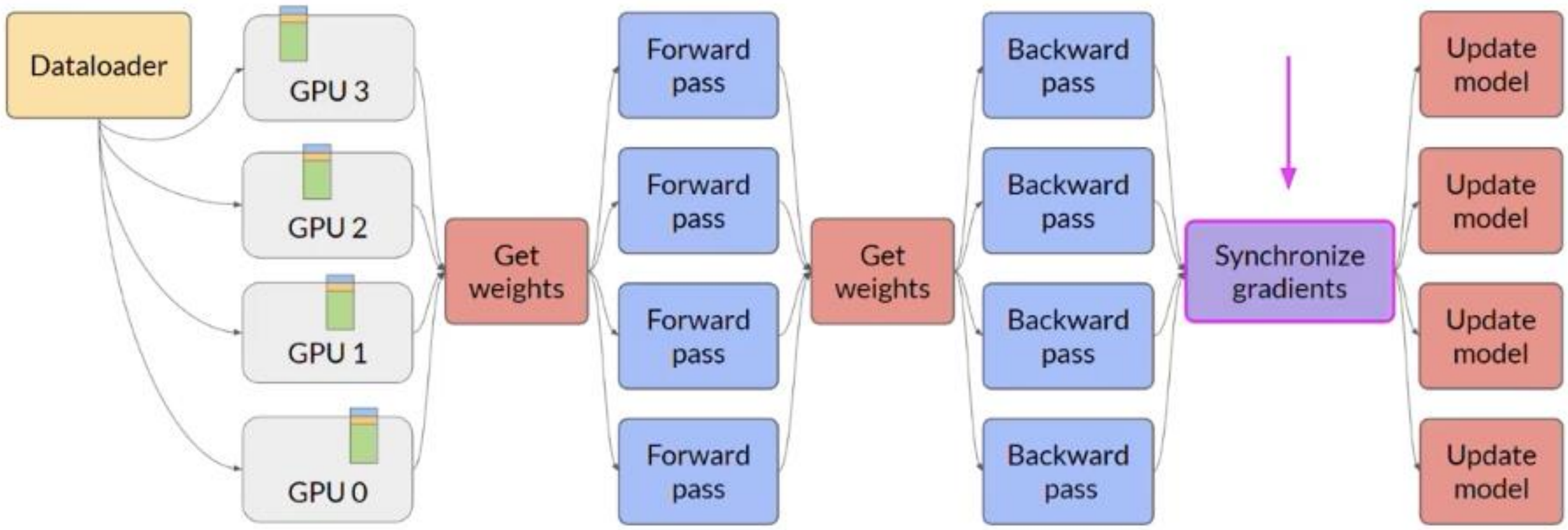
Fully Sharded Data Parallel (FSDP)



Fully Sharded Data Parallel (FSDP)

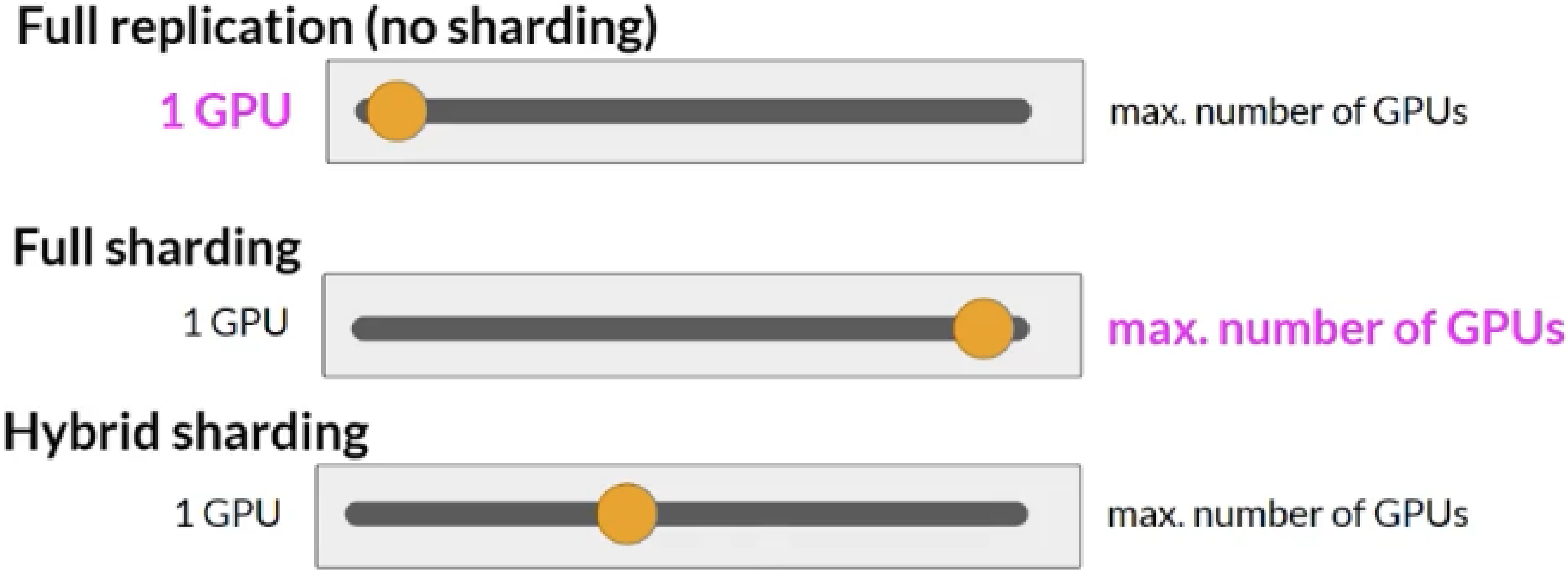


Fully Sharded Data Parallel (FSDP)



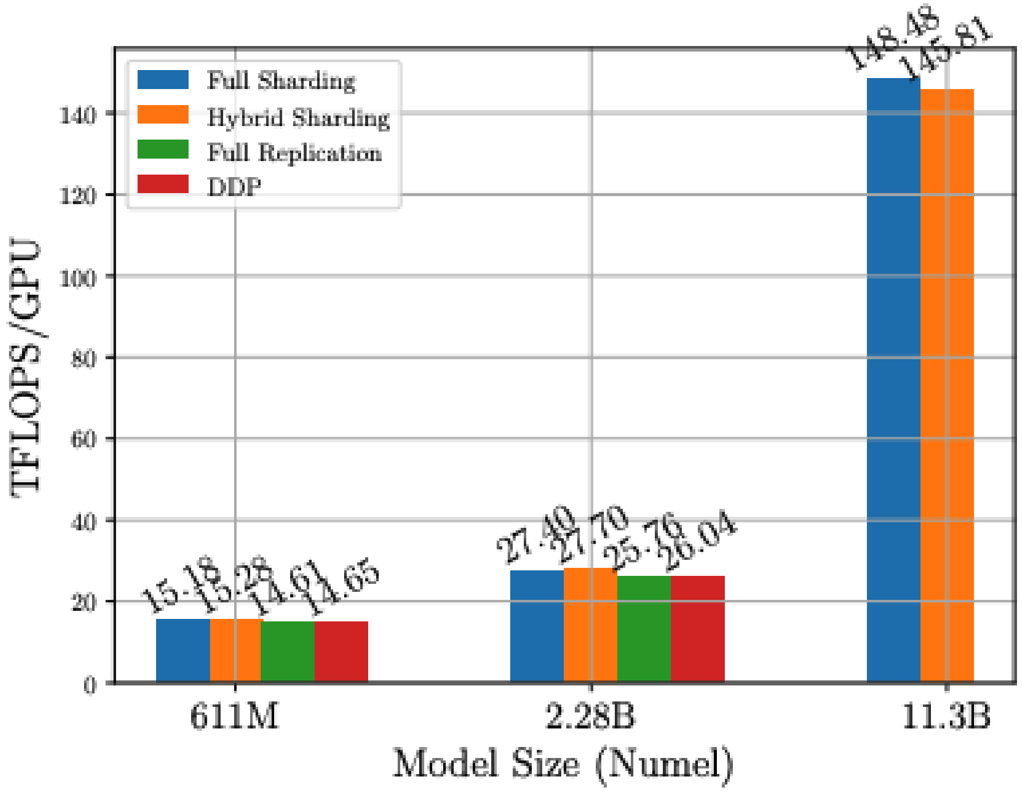
Fully Sharded Data Parallel (FSDP)

- Helps to reduce overall GPU memory utilization
- Supports offloading to CPU if needed
- Configure level of sharding via sharding factor

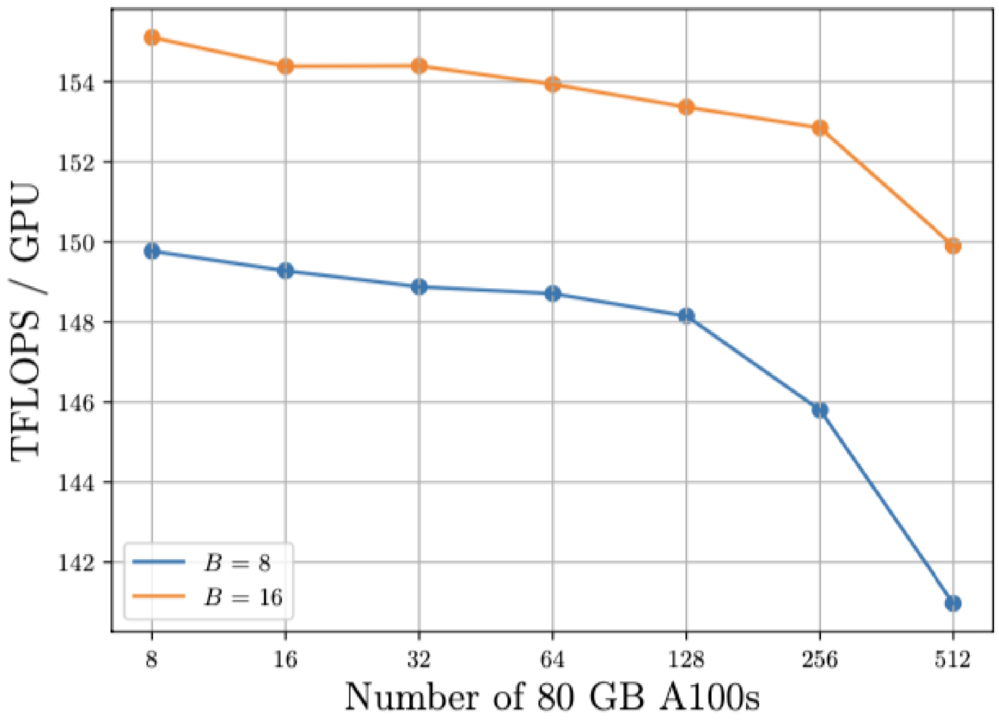


Impact of using FSDP

Note: 1 teraflop/s = 1,000,000,000,000 (One trillion) floating point operations per second



(a) Model Scale



(c) T5-11B TFLOPS

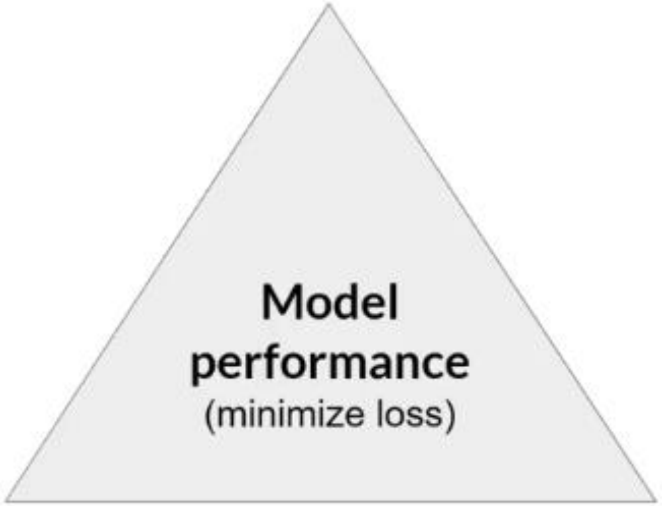
Zhao et al. 2023: "PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel"

Scaling laws and compute-optimal models

Scaling choices for pre-training

Goal: **maximize model performance**

CONSTRAINT:
Compute budget
(GPUs, training time, cost)



SCALING CHOICE:
Dataset size
(number of tokens)

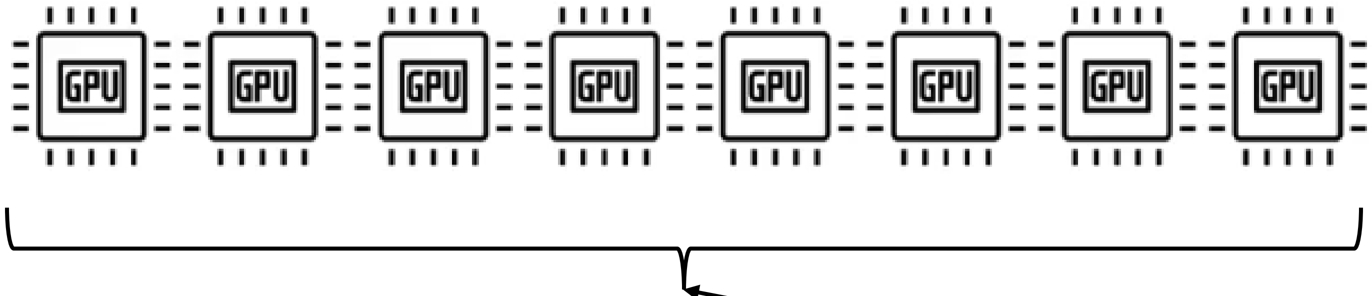
SCALING CHOICE:
Model size
(number of parameters)

Compute budget for training LLMs

1 “petaflop/s-day” =

floating point operations performed at rate of 1 petaFLOP per second for one day

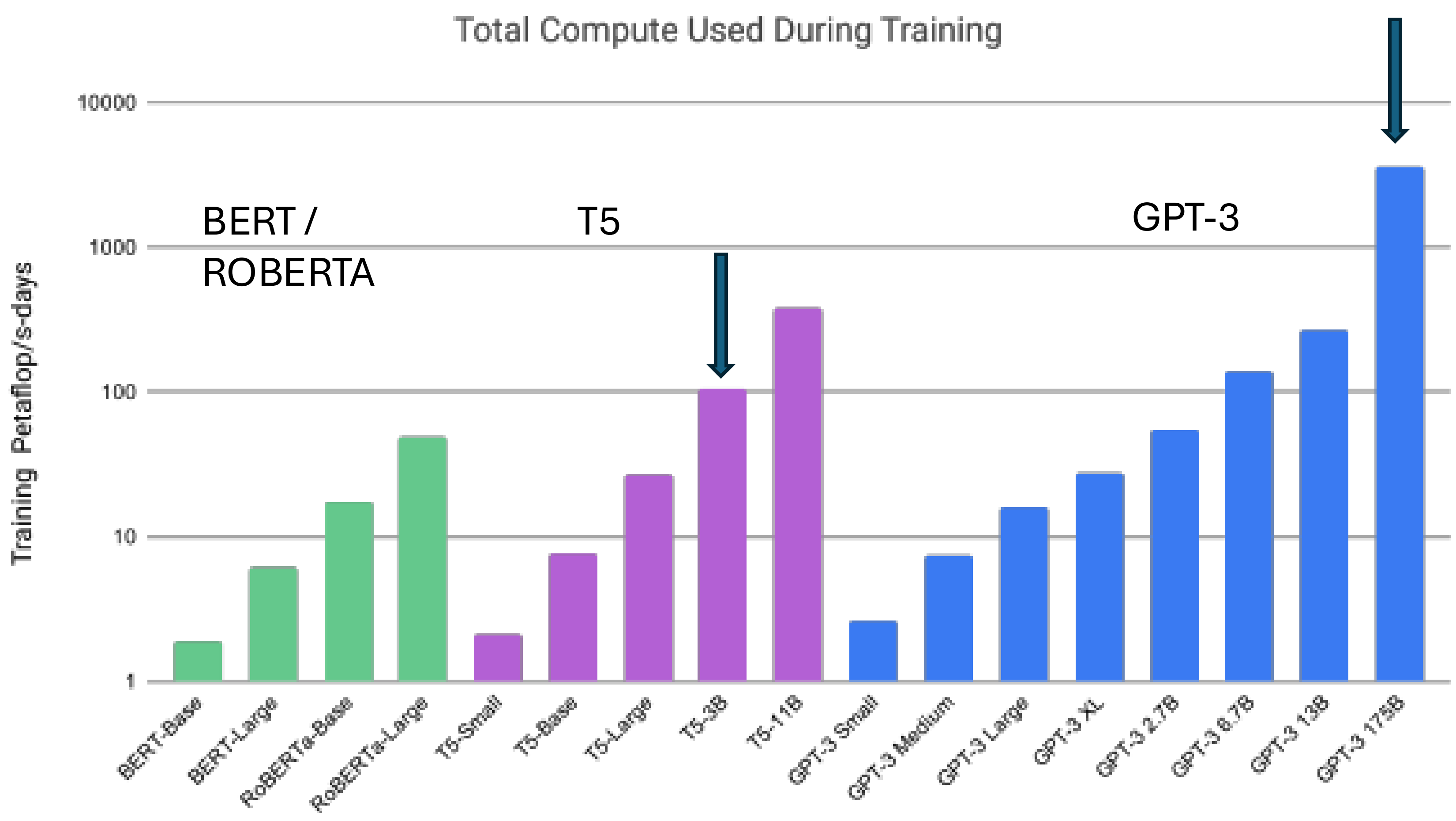
NVIDIA V100s



1 petaflops/s-day is these chips running at full efficiency for 24 hours

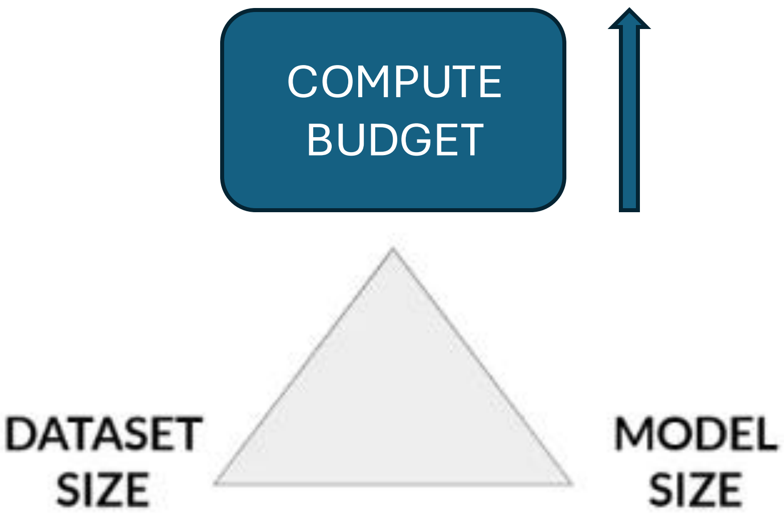
Note: 1 petaFLOP/s = 1,000,000,000,000,000 (one quadrillion) floating point operations per second

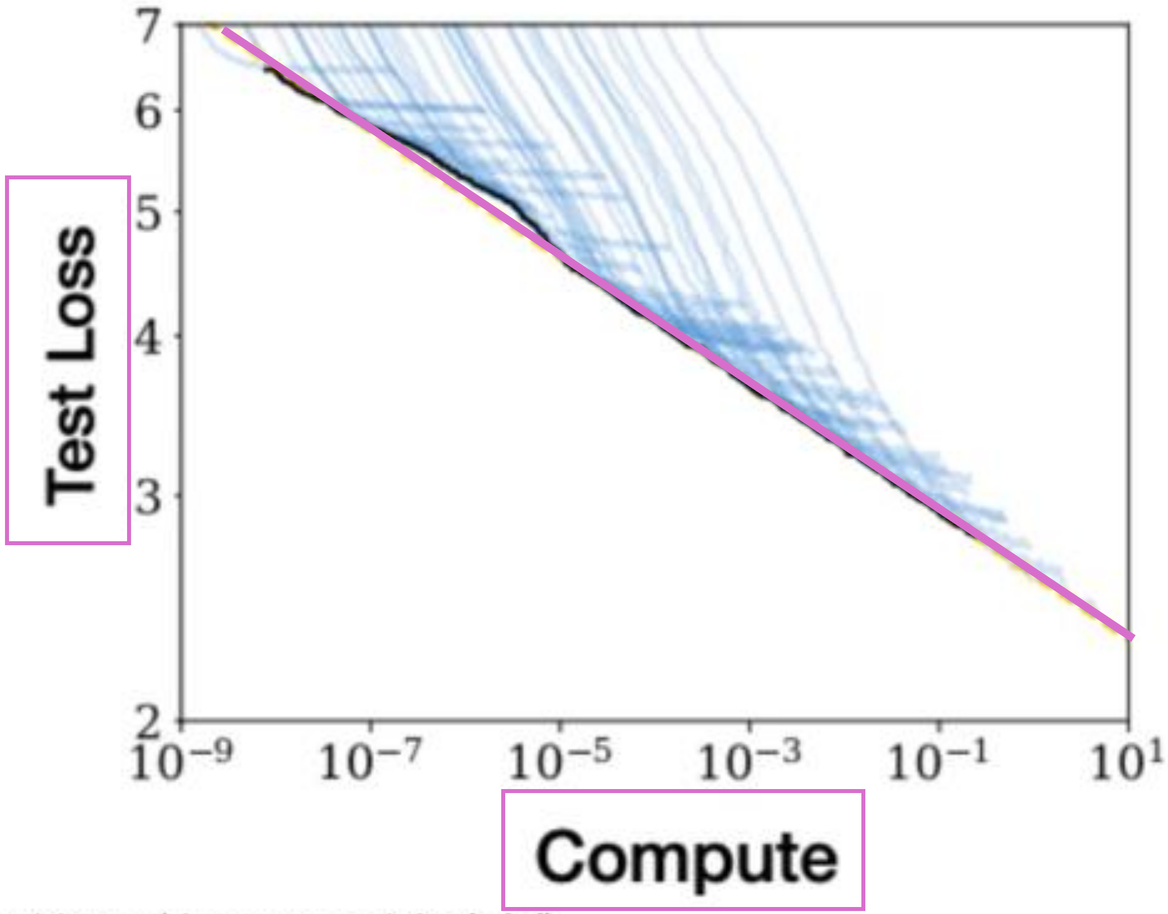
Number of petaflop/s-days to pretrain various LLMs



Brown et al. 2020, "Language Models are Few-Shot Learners", https://arxiv.org/pdf/2005.14165

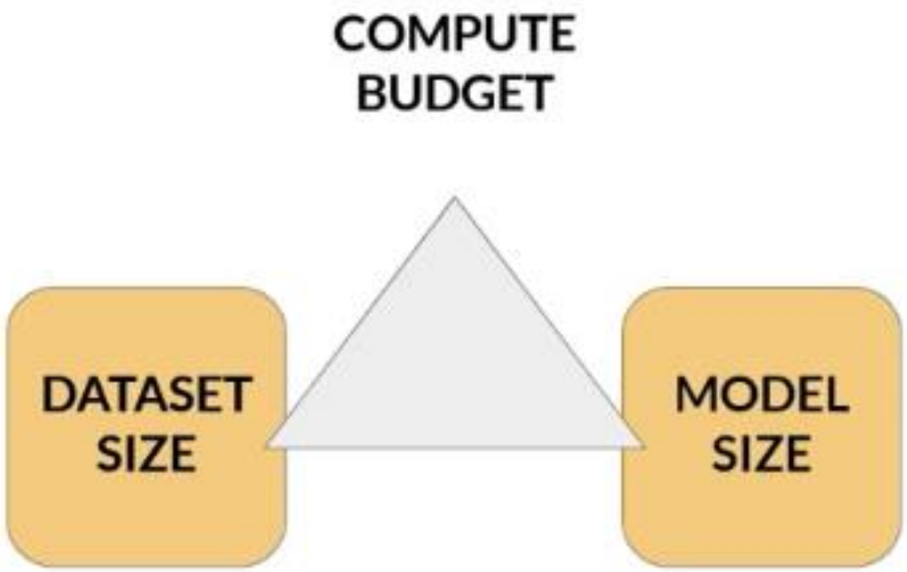
Compute budget vs. model performance





Source: Kaplan et al. 2020, "Scaling Laws for Neural Language Models"

Dataset size and model size vs. performance

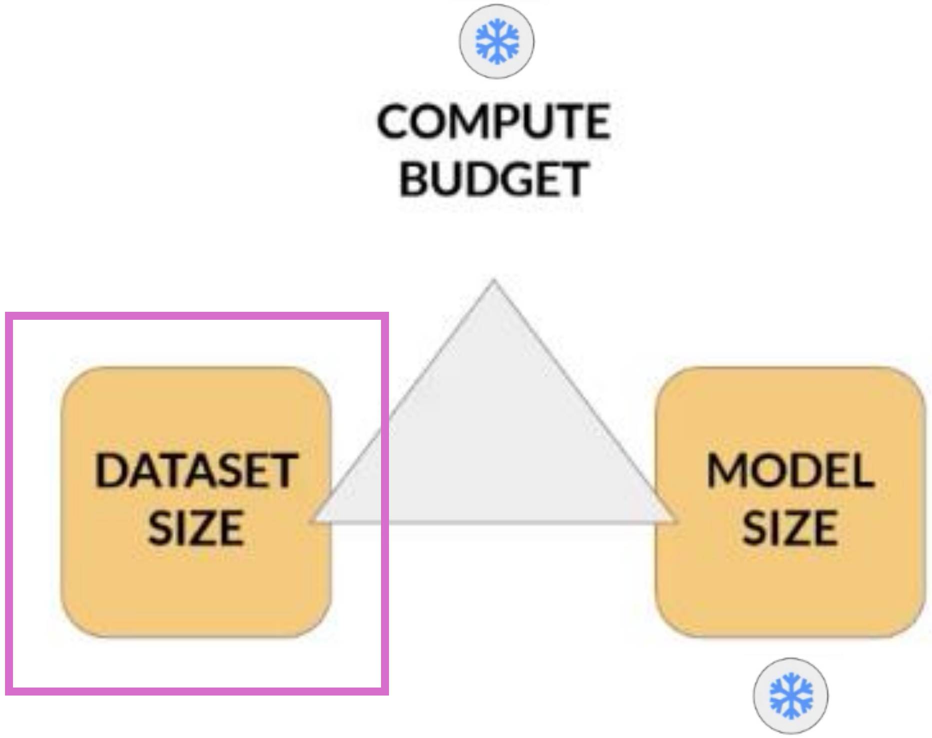


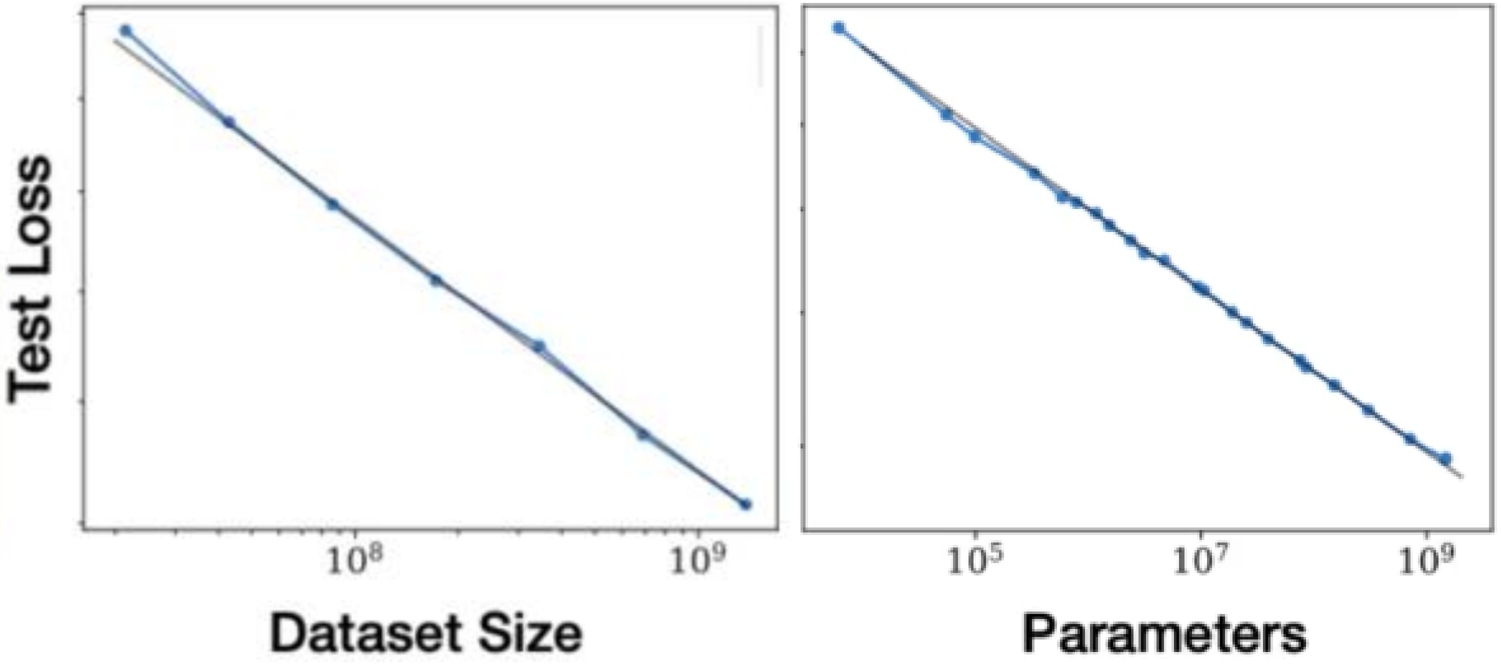
Compute resource constraints

- Hardware
- Project timeline
- Financial budget

Source: Kaplan et al. 2020, "Scaling Laws for Neural Language Models"

Data size and model size vs. performance





Source: Kaplan et al. 2020, "Scaling Laws for Neural Language Models"

Chinchilla paper

DeepMind

Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

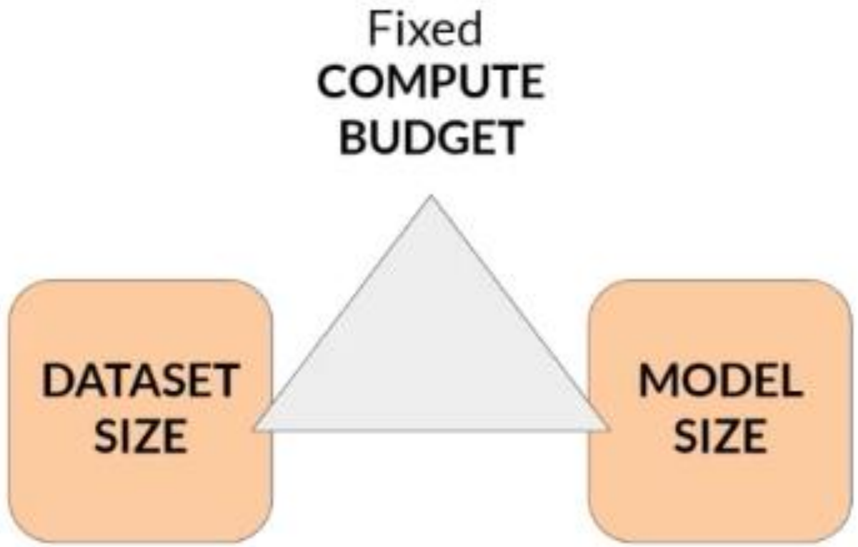
*Equal contributions

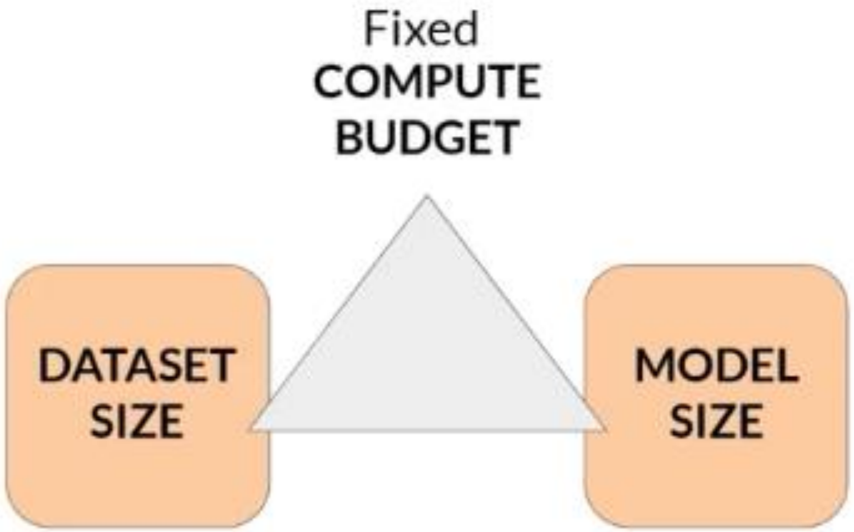
We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly undertrained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

Jordan et al. 2022

Compute optimal models

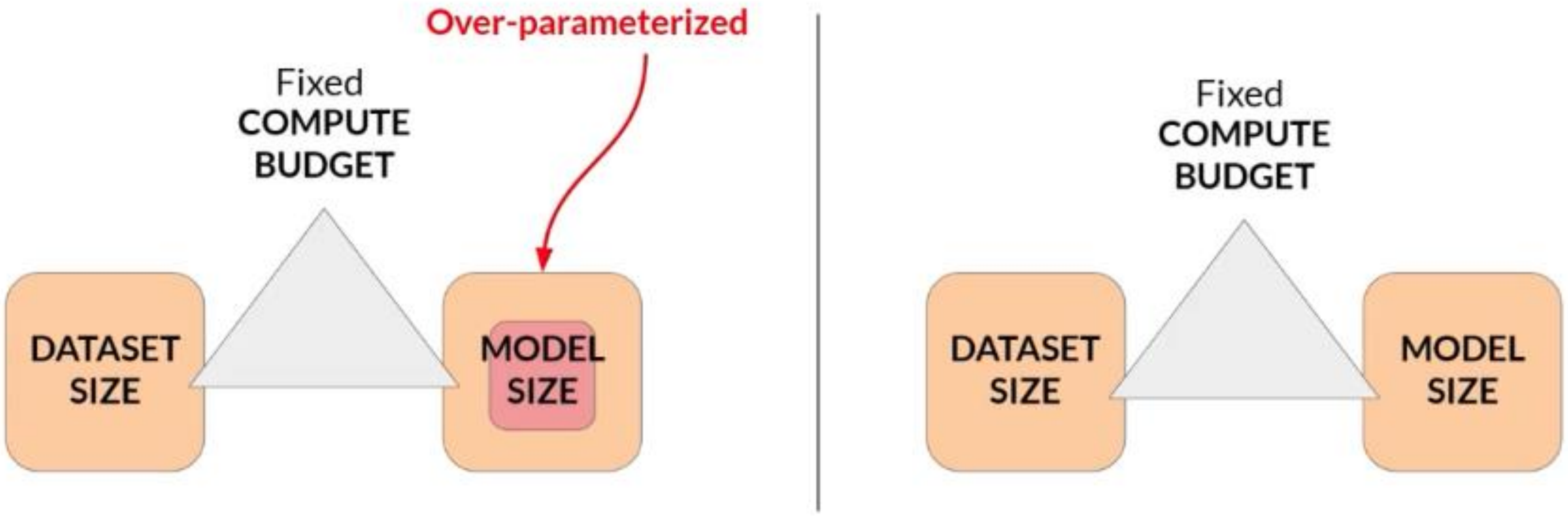
- Very large models may be **over-parameterized** and **under-trained**





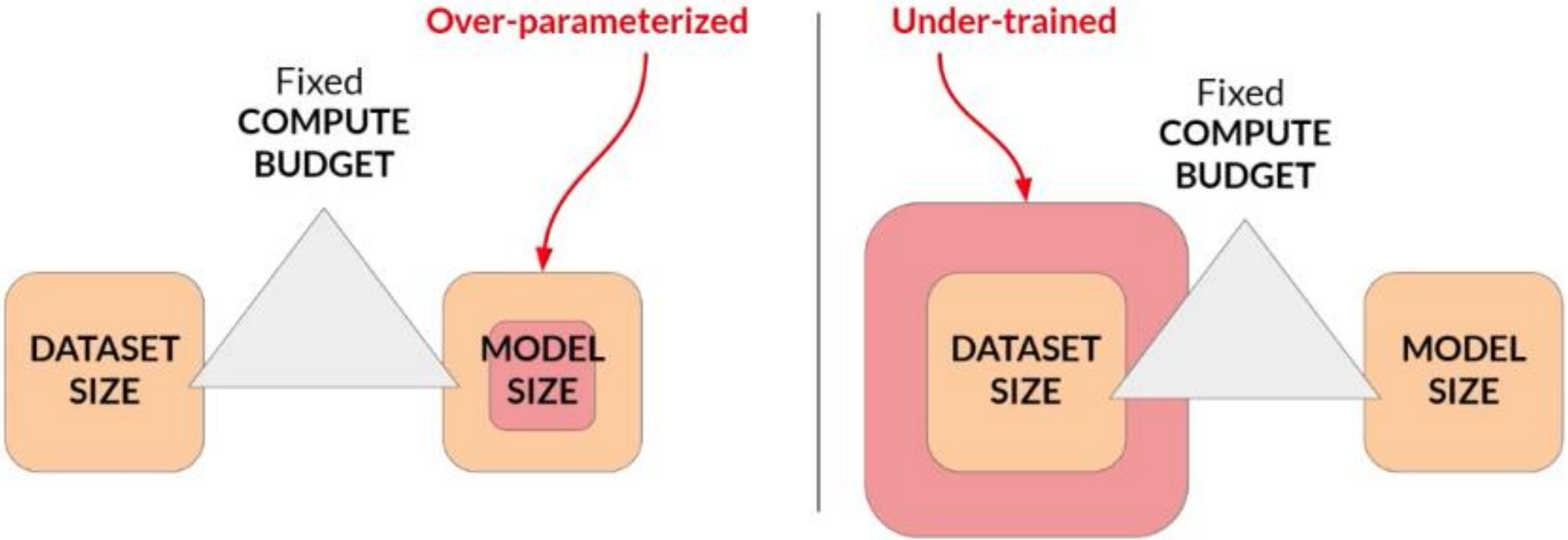
Compute optimal models

- Very large models may be **over-parameterized** and **under-trained**



Compute optimal models

- Very large models may be **over-parameterized** and **under-trained**
- Smaller models trained on more data could perform as well as large models



Chinchilla scaling laws for model and dataset size

Model	# of parameters	Compute-optimal* # of tokens (~20x)	Actual # tokens
Chinchilla	70B	~1.4T	1.4T
LLaMA-65B	65B	~1.3T	1.4T
GPT-3	175B	~3.5T	300B
OPT-175B	175B	~3.5T	180B
BLOOM	176B	~3.5T	350B

Compute optimal training datasize
is ~**20x** number of parameters

* assuming models are trained to be compute-optimal per Chinchilla paper

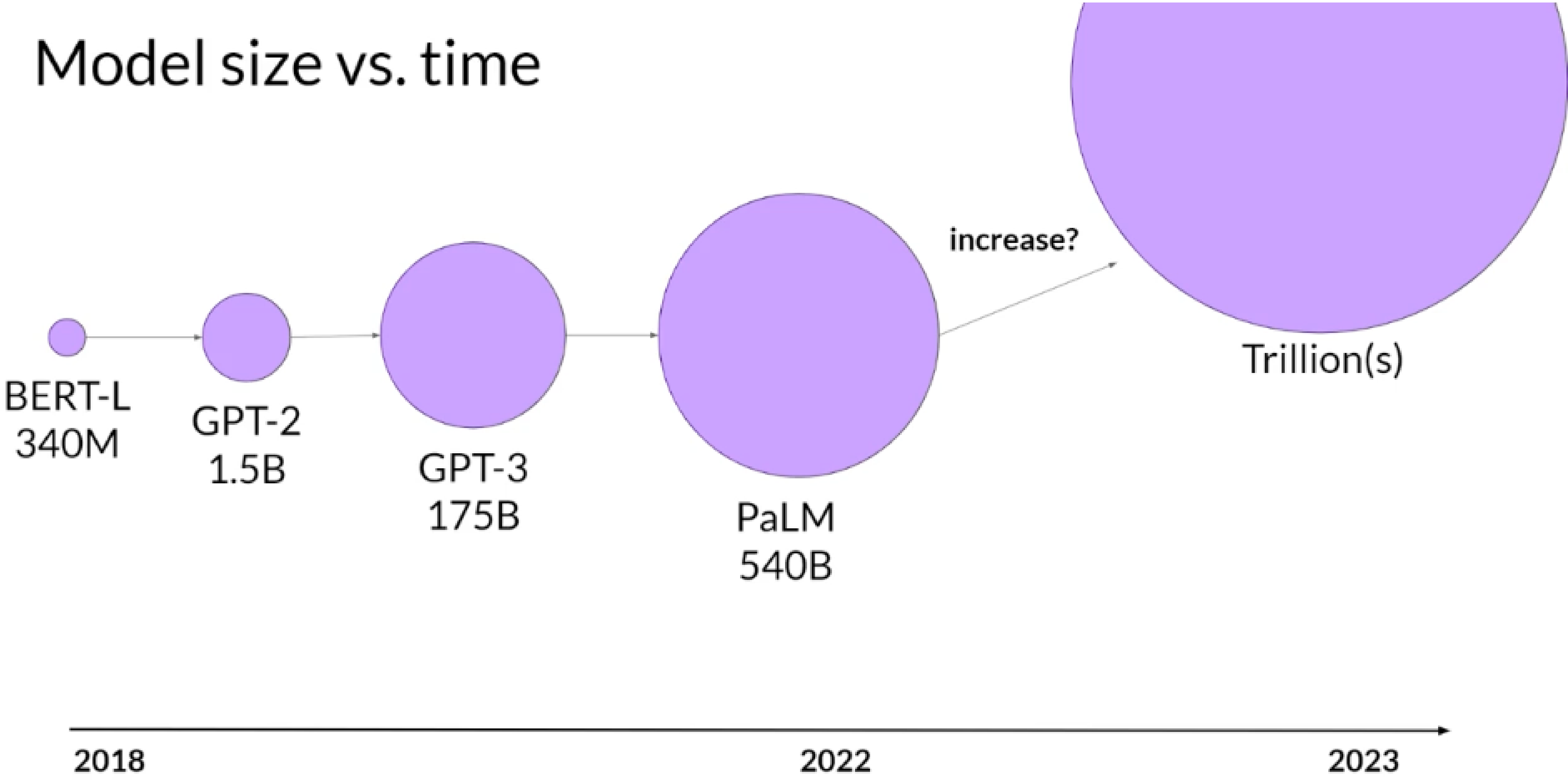
Sources: Hoffmann et al. 2022, "Training Compute-Optimal Large Language Models"
Touvron et al. 2023, "LLaMA: Open and Efficient Foundation Language Models"

Question

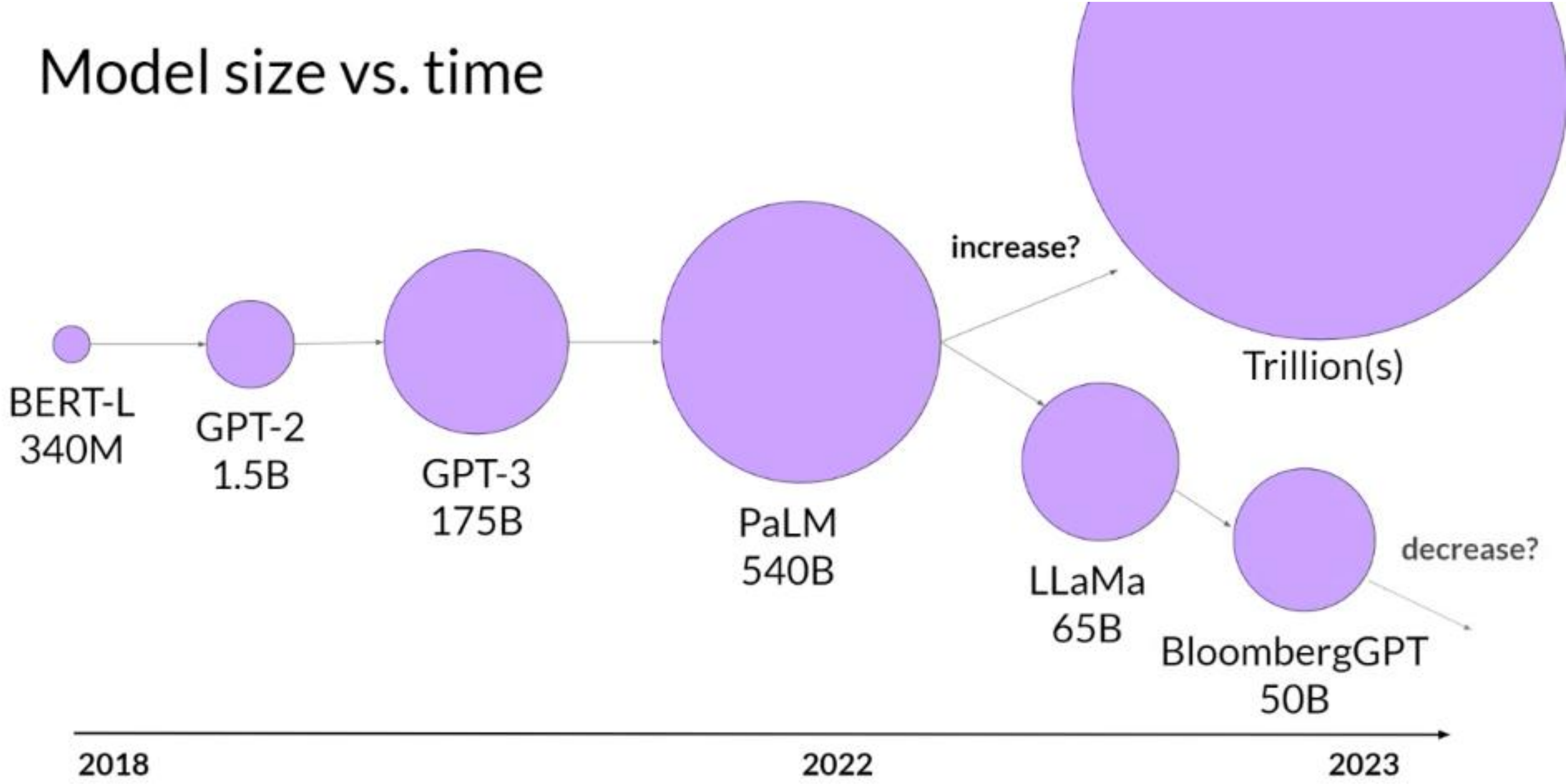
Scaling laws for pre-training large language models consider several aspects to maximize performance of a model within a set of constraints and available scaling choices. Select all alternatives that should be considered for scaling when performing model pre-training?

A. Batch size: Number of samples per iteration
B. Model size: Number of parameters
C. Compute budget: Compute constraints
D. Dataset size: Number of tokens

Model size vs. time



Model size vs. time



Pre-training for domain adaptation

Pre-training for domain adaptation

Legal language

The prosecutor had difficulty proving mens rea, as the defendant seemed unaware that his actions were illegal.

The judge dismissed the case, citing the principle of res judicata as the issue had already been decided in a previous trial.

Despite the signed agreement, the contract was invalid as there was no consideration exchanged between the parties.

Medical language

After a strenuous workout, the patient experienced severe myalgia that lasted for several days.

After the biopsy, the doctor confirmed that the tumor was malignant and recommended immediate treatment.

Sig: 1 tab po qid pc & hs

Take one tablet by mouth four times a day, after meals, and at bedtime.

BloombergGPT: domain adaptation for finance

BloombergGPT: A Large Language Model for Finance

Shijie Wu[1,*], Ozan İrsoy[1,*], Steven Lu[1,*], Vadim Dabravolski[1], Mark Dredze[1,3], Sebastian Gehrmann[1], Prabhanjan Kambadur[1], David Rosenberg[2], Gideon Mann[1]
[1] **Bloomberg, New York, NY USA**
[2] **Bloomberg, Toronto, ON Canada**
[3] **Computer Science, Johns Hopkins University, Baltimore, MD USA**

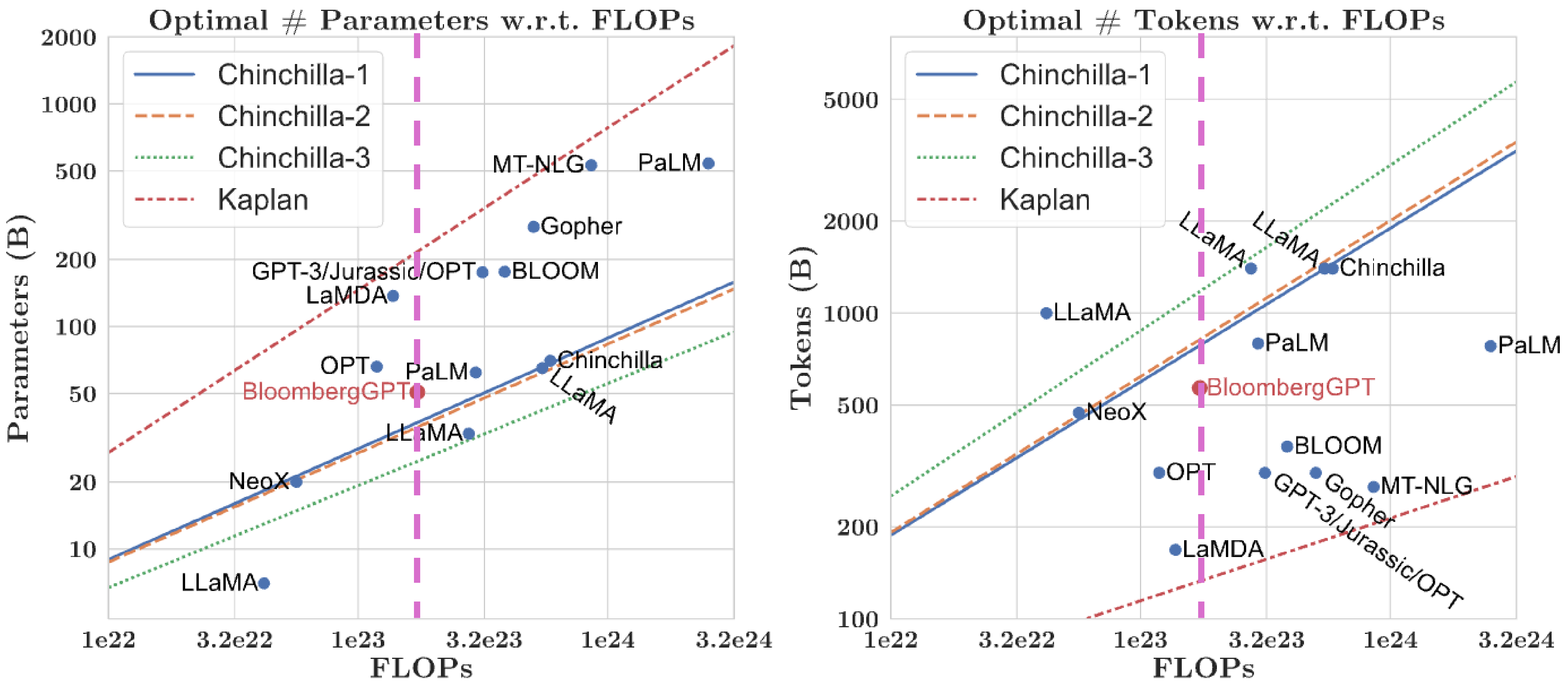
Abstract

The use of NLP in the realm of financial technology is broad and complex, with applications ranging from sentiment analysis and named entity recognition to question answering. Large Language Models (LLMs) have been shown to be effective on a variety of tasks; however, no LLM specialized for the financial domain has been reported in literature. In this work, we present BloombergGPT, a 50 billion parameter language model that is trained on a wide range of financial data. We construct a 363 billion token dataset based on Bloomberg's extensive data sources, perhaps the largest domain-specific dataset yet, augmented with 345 billion tokens from general purpose datasets. We validate BloombergGPT on standard LLM benchmarks, open financial benchmarks, and a suite of internal benchmarks that most accurately reflect our intended usage. Our mixed dataset training leads to a model that outperforms existing models on financial tasks by significant margins without sacrificing performance on general LLM benchmarks. Additionally, we explain our modeling choices, training process, and evaluation methodology. We release Training Chronicles (Appendix C) detailing our experience in training BloombergGPT.

~51% **Financial (Public & Private)**

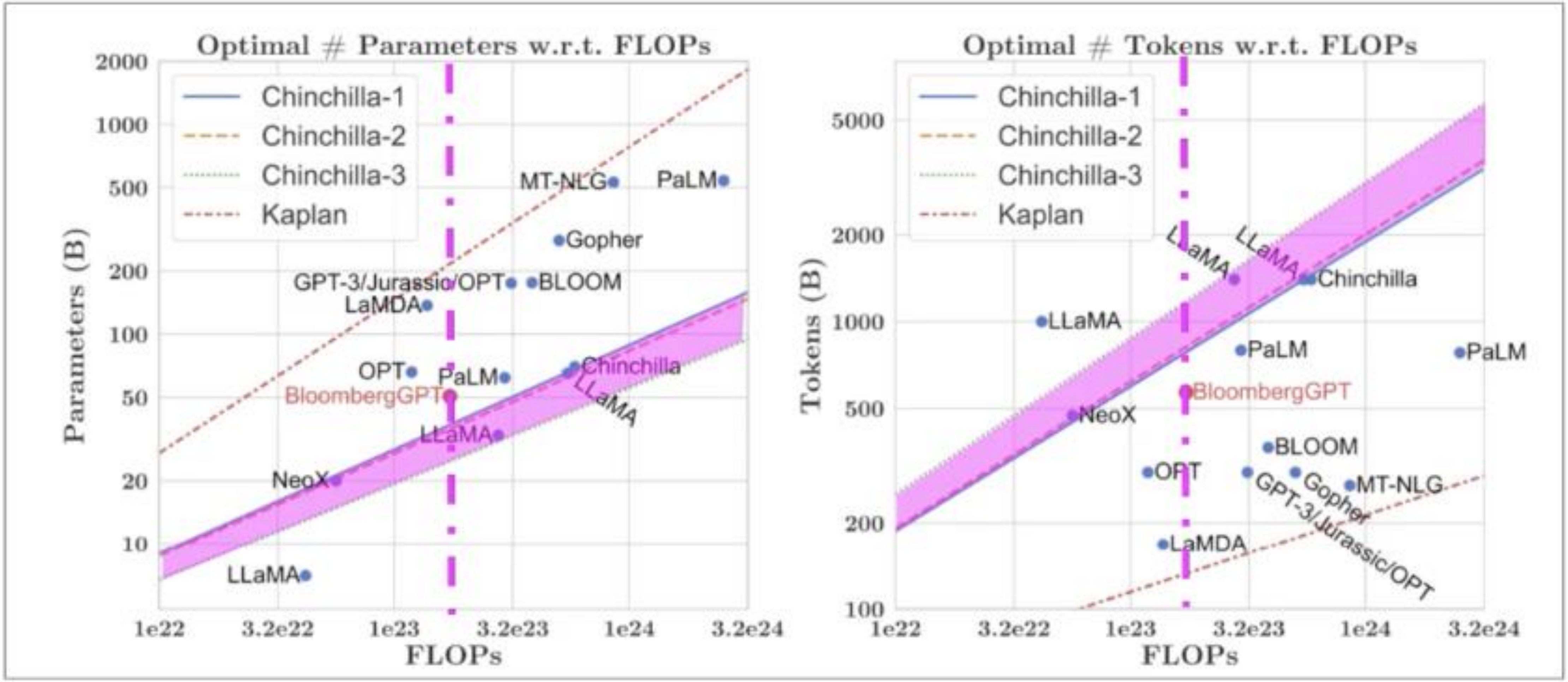
~49% **Other (Public)**

BloombergGPT relative to other LLMs



Source: Wu et al. 2023, “BloombergGPT: A Large Language Model for Finance.”

BloombergGPT relative to other LLMs



Source: Wu et al. 2023, "BloombergGPT: A Large Language Model for Finance"