

Guest Lecture - CSCI 544
A Few Projects on Pretraining LLMs

Willie Neiswanger

About me – some background context

About me – some background context

In PhD and postdoc: *AI-driven experimental design for science and engineering.*

About me – some background context

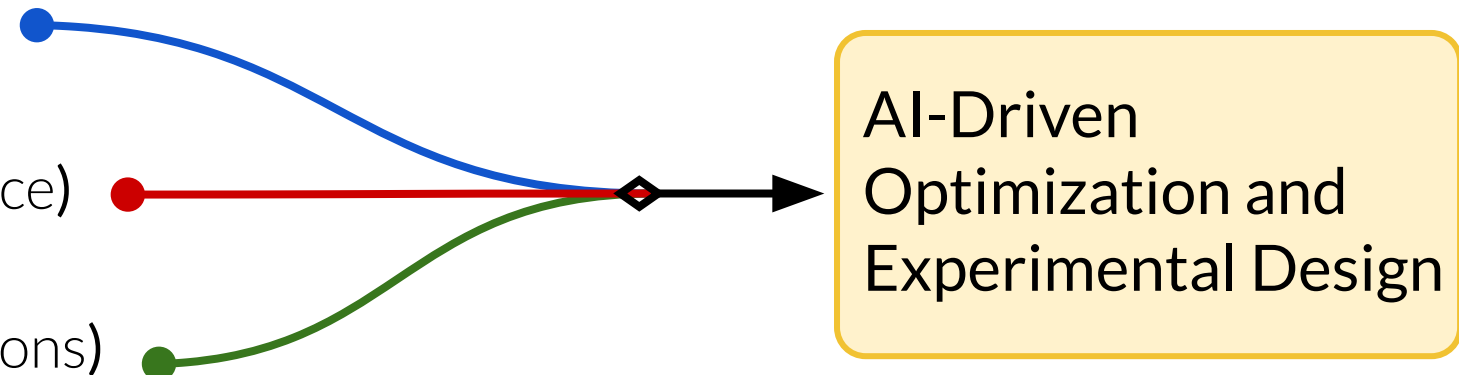
In PhD and postdoc: *AI-driven experimental design for science and engineering.*

Use techniques from:

Experimental Design (statistics)

Active Learning (computer science)

Bayesian Opt & Bandits (operations)



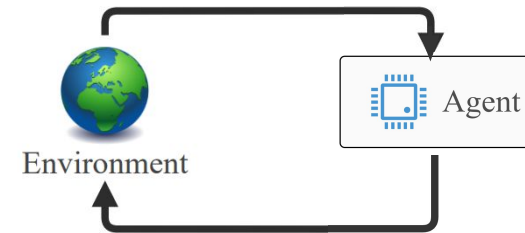
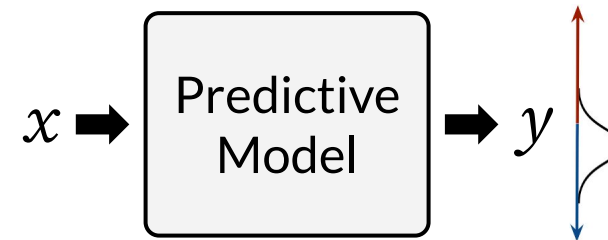
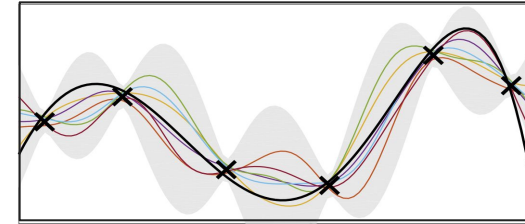
AI-Driven
Optimization and
Experimental Design

Focus on combining probabilistic machine learning with decision making.

About me – some background context

Probabilistic machine learning

- Classic probabilistic models
- Generative models



About me – some background context

Probabilistic machine learning

- Classic probabilistic models
- Generative models

With applications to science and engineering

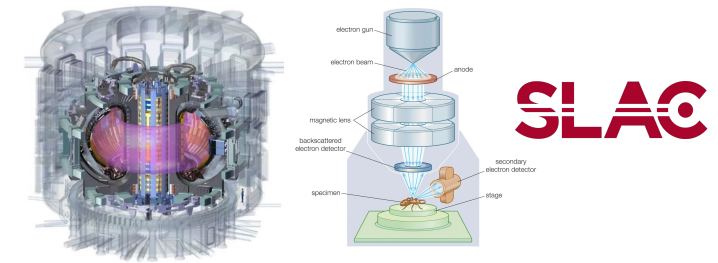
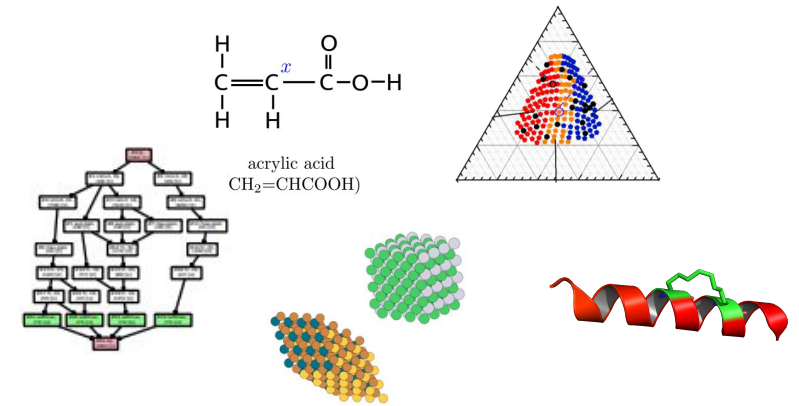
About me – some background context

Probabilistic machine learning

- Classic probabilistic models
- Generative models

With applications to science and engineering

- Black-box optimization & experimental design:
 - Materials science (high-throughput screening).
 - Scientific machines (particle accelerators, tokamaks).
 - Computer/ML systems (config. tuning, hyperparameter opt).



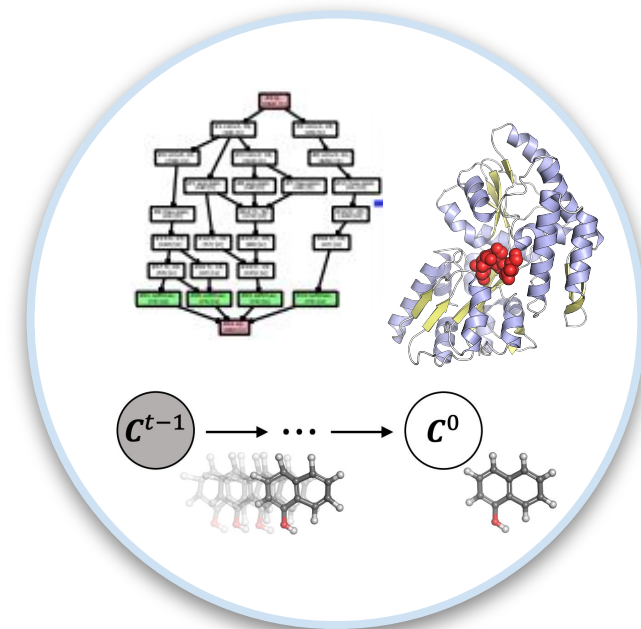
About me – some background context

Probabilistic machine learning

- Classic probabilistic models
- Generative models

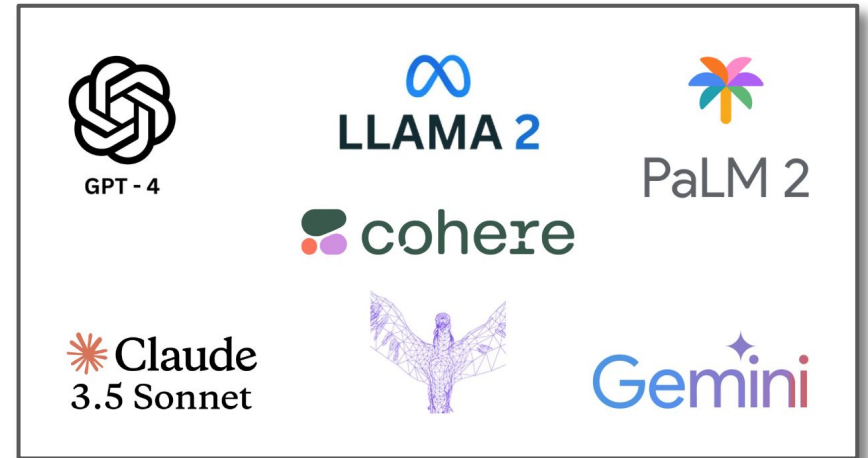
With applications to science and engineering

- Black-box optimization & experimental design:
 - Materials science (high-throughput screening).
 - Scientific machines (particle accelerators, tokamaks).
 - Computer/ML systems (config. tuning, hyperparameter opt).
- Generative + probabilistic modeling:
 - Chemical / molecular design
 - Neural architecture design



At the end of my postdoc (~mid 2023)

Some collaborators began a project on open source reproductions of LLMs ...



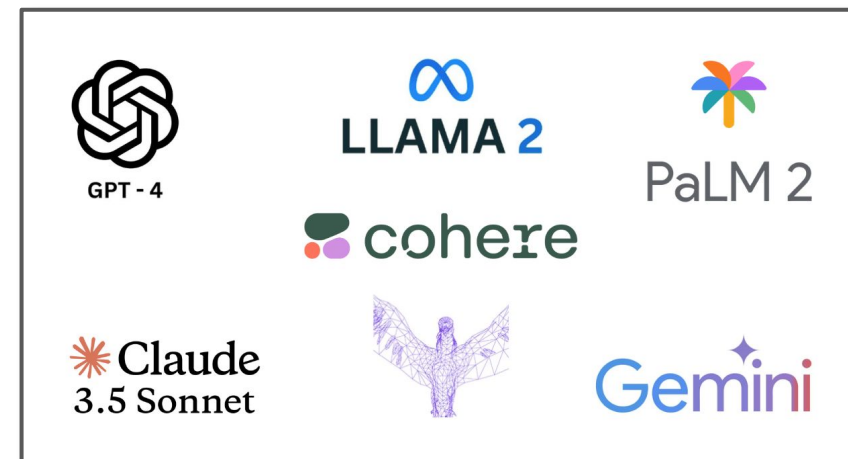
At the end of my postdoc (~mid 2023)

Some collaborators began a project on open source reproductions of LLMs ...

Through support from some universities/companies they got resources to do LLM pretraining.

⇒ I thought it would be a good opportunity to learn

⇒ Started working on LLMs



At the end of my postdoc (~mid 2023)

Some collaborators began a project on open source reproductions of LLMs ...

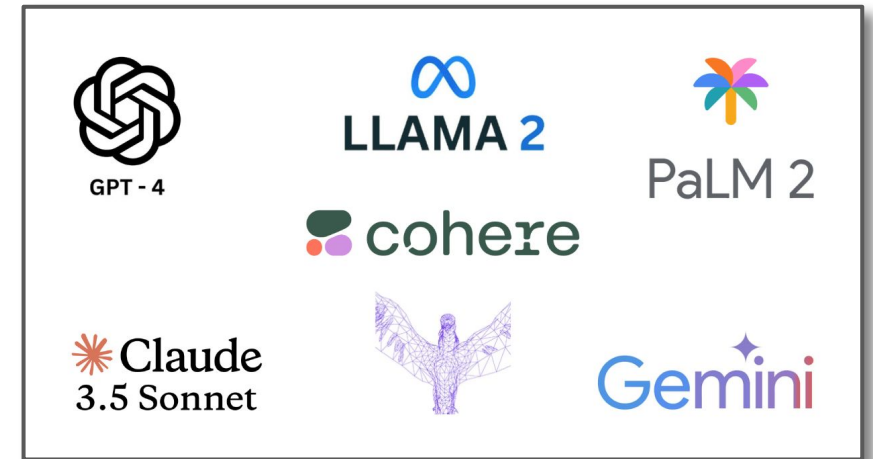
Through support from some universities/companies they got resources to do LLM pretraining.

⇒ I thought it would be a good opportunity to learn

⇒ Started working on LLMs

Increasingly working on LLM projects for past ~1.5 years.

- Including LLM pretraining, which I will describe today (as part of the **LLM360 project**).



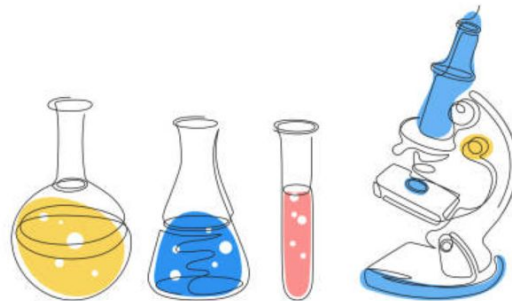
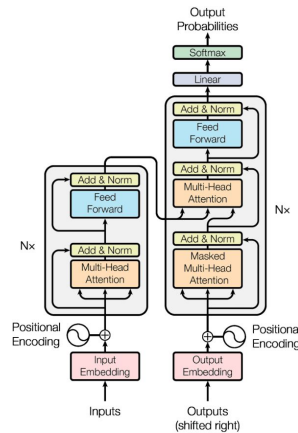
My current research

⇒ It influenced my research!

I still do probabilistic decision making and experimental design, but...

I also combine this with LLMs, e.g.

- LLMs (large sequence models) for scientific data (e.g., in biology, materials science).
- LLMs for (sequential) decision making, and it's variants.



Talk Outline

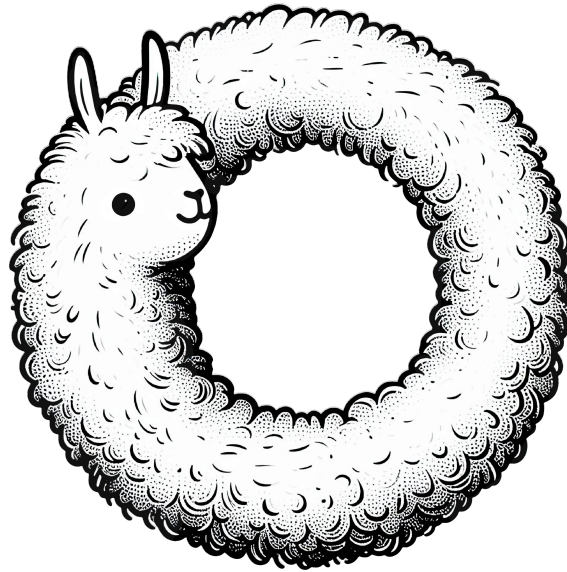
- **Open Source LLM Reproductions – and the LLM360 project**
- **LLMs in Biology: Metagenomic LLM for Pandemic Monitoring**

Talk Outline

- **Open Source LLM Reproductions – and the LLM360 project**
- **LLMs in Biology: Metagenomic LLM for Pandemic Monitoring**

Open Source LLM Reproductions – the LLM360 project

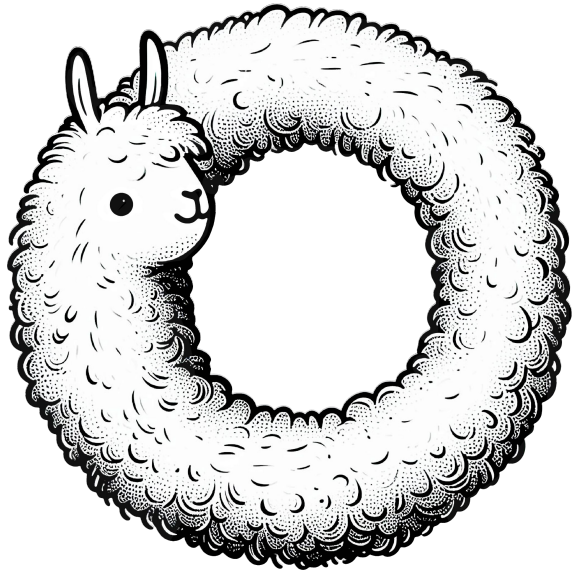
Previous collaboration, eventually transformed into: **the LLM360 project.**



LLM360

Open Source LLM Reproductions – the LLM360 project

Previous collaboration, eventually transformed into: **the LLM360 project.**



LLM360



Goal:

To develop and release **fully open source LLMs** to foster transparency, trust, and collaborative research.

The LLM360 Team (at COLM Conference)



What are “Fully Open Source” LLMs?

What are “Fully Open Source” LLMs?

We aim to produce models that are **completely reproducible!**

In contrast to “*open weight LLMs*”, where only the final model weights are released.

What are “Fully Open Source” LLMs?

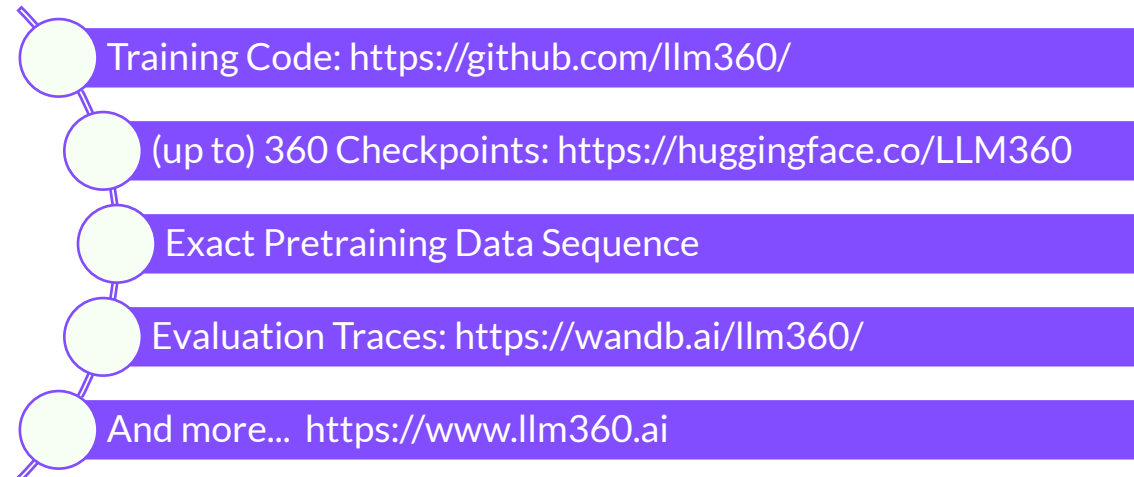
We aim to produce models that are **completely reproducible!**

In contrast to “*open weight LLMs*”, where only the final model weights are released.

We hope to:

- Provide artifacts for reproduction and collaboration.
- Level the level the playing field for LLM development via knowledge sharing.

Artifacts released for all LLM360 models:



Why Make “Fully Open Source” LLMs?

The problem: LLM pretraining is a complex and expensive engineering process, hence:

1. Many in the community lack understanding of the full LLM development process.
2. Open-source projects are developed under different conditions.
3. Research efforts on LLMs are harder to be stacked/composed together.

Why Make “Fully Open Source” LLMs?

The solution — Fully open source LLMs can help:

Debunk the Mystery of LLMs

- Share details of the full process to allow people to know what methods lead to what outcomes.

Promote Reproducibility and Auditability

- Full development trace including data, training log.

Collective and Collaborative Research

- Provide a standard playground, model, or environment for collaboration

Why Make “Fully Open Source” LLMs?

The solution — Fully open source LLMs can help:

Debunk the Mystery of LLMs

- Share details of the full process to allow people to know what methods lead to what outcomes.

Promote Reproducibility and Auditability

- Full development trace including data, training log.

Collective and Collaborative Research

- Provide a standard playground, model, or environment for collaboration

Why Make “Fully Open Source” LLMs?

The solution — Fully open source LLMs can help:

Debunk the Mystery of LLMs

- Share details of the full process to allow people to know what methods lead to what outcomes.

Promote Reproducibility and Auditability

- Full development trace including data, training log.

Collective and Collaborative Research

- Provide a standard playground, model, or environment for collaboration

Why Make “Fully Open Source” LLMs?

The solution — Fully open source LLMs can help:

Debunk the Mystery of LLMs

- Share details of the full process to allow people to know what methods lead to what outcomes.

Promote Reproducibility and Auditability

- Full development trace including data, training log.

Collective and Collaborative Research

- Provide a standard playground, model, or environment for collaboration

Why Make “Fully Open Source” LLMs?

The solution — Fully open source LLMs can help:

Debunk the Mystery of LLMs

- Share details of the full process to allow people to know what methods lead to what outcomes.

Promote Reproducibility and Auditability

- Full development trace including data, training log.

Collective and Collaborative Research

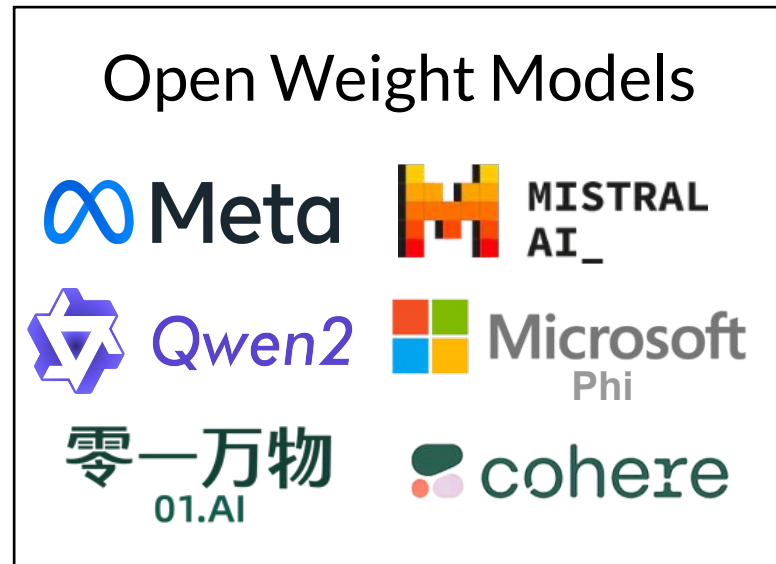
- Provide a standard playground, model, or environment for collaboration

- Want to promote knowledge transfer rather than competitive advantage.
- Want to be open about issues, difficult parts, rough edges, problems with our models, etc.
- Imagine if the future of AI is owned by the community — like great open source projects, e.g., linux, wikipedia, etc.

Open Weight vs Fully Open Source Models

Open Weight vs Fully Open Source Models

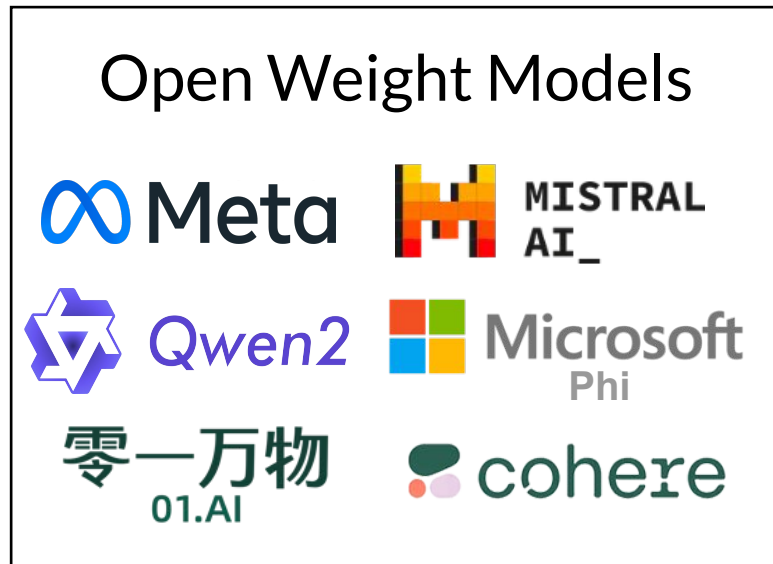
Open Weights = only the final weights are released



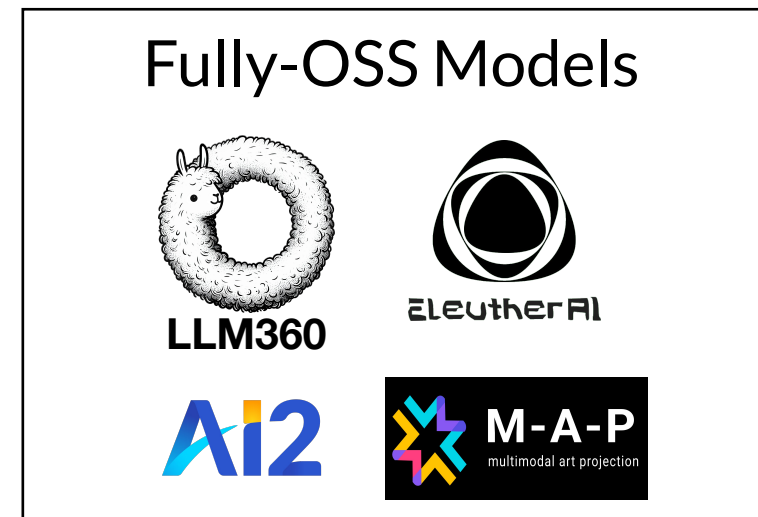
Open weight models \Rightarrow learnings are not transferred, code and data cannot be reused, and advanced models must be built from scratch.

Open Weight vs Fully Open Source Models

Open Weights = only the final weights are released



Fully Open Source = every step is fully reproducible (including all data, all methods, etc)



Open weight models \Rightarrow learnings are not transferred, code and data cannot be reused, and advanced models must be built from scratch.

Other Fully Open Source Projects

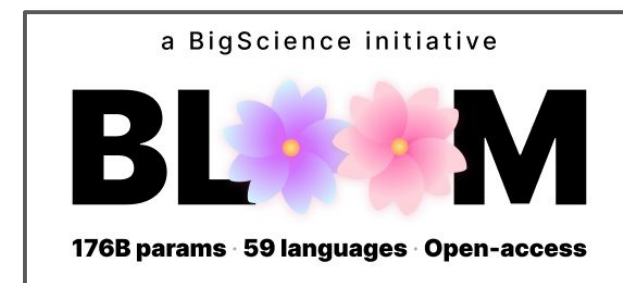
LLM360 ❤️'s other “fully open source” LLM projects — for example:



Pythia series,
GPT-J,
GPT-NeoX-20B
The Pile



OLMo, Tulu,
OLMoE, Molmo
Dolma



MAP-NEO,
OpenCodeInterpreter,

The field moves very fast...

Quick note on slides:

These slides were up to date (in terms of current/recent LLMs) earlier this year...

...but the field moves fast, and so you'll see a lot of references to slightly older models!

Model Releases — Some LLM360 History

Three main model release so far:

Model Releases — Some LLM360 History

Three main model release so far:



Amber: 7B Language Model

- The first model of the LLM360 project.

Model Releases — Some LLM360 History

Three main model release so far:



Amber: 7B Language Model

- The first model of the LLM360 project.



Crystal: 7B Language Model that also excels at Code

- More token efficient in training than some of the popular open-weight models.
- A better balance between coding and language

Model Releases — Some LLM360 History

Three main model release so far:



Fall 2023

Amber: 7B Language Model

- The first model of the LLM360 project.



End of 2023

Crystal: 7B Language Model that also excels at Code

- More token efficient in training than some of the popular open-weight models.
- A better balance between coding and language

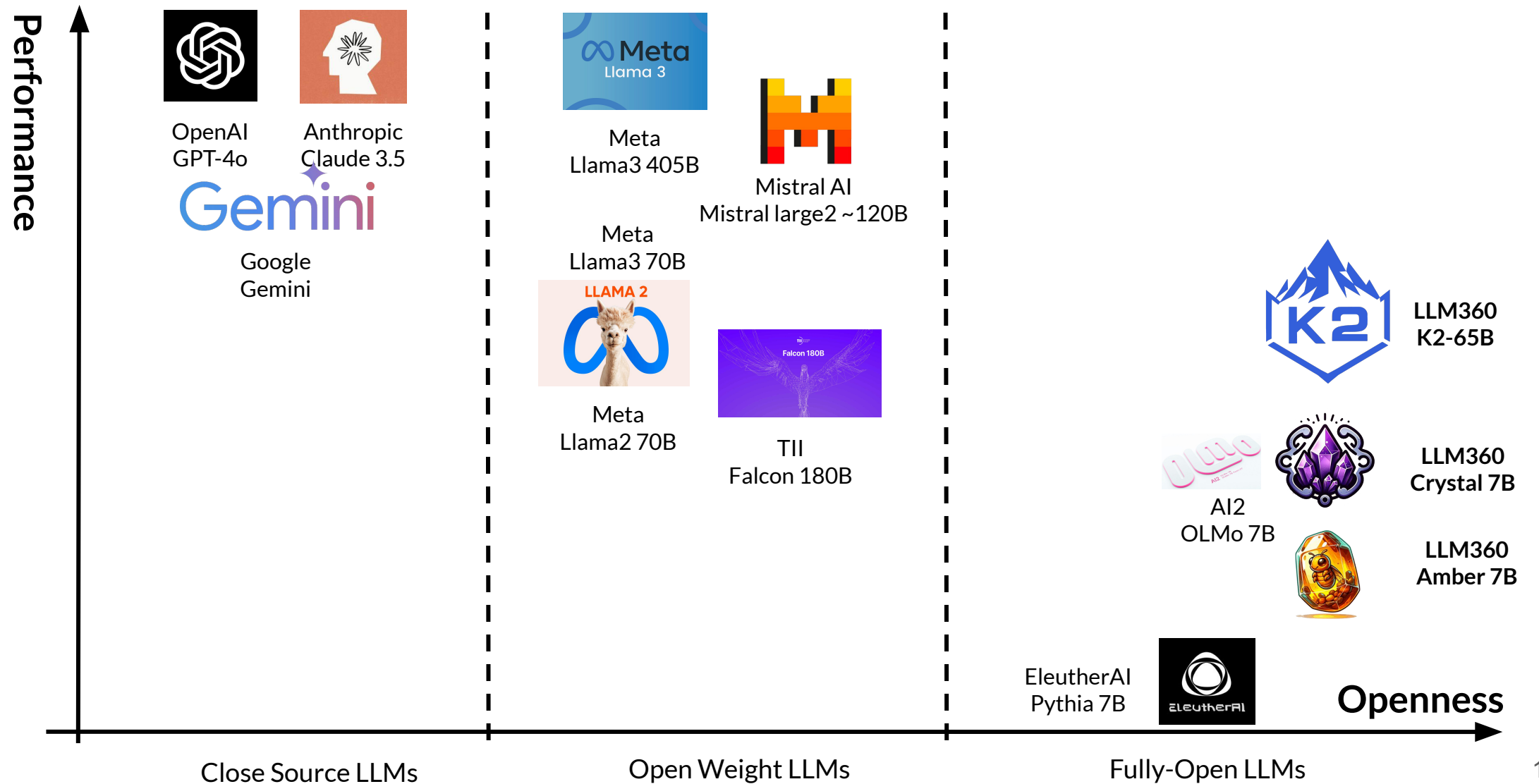


Summer 2024

K2-65B: a reproducible large-param model, at Llama 2-70B performance, trained with 35% less compute

- More FLOPs efficient in pretraining than the Llama series

The Landscape of LLMs



K2 Model

My plan is to go through all details of pretraining our models.

But to start, let's go through overview/results of the K2-65B model...

K2 Model – Goals

K2 Model – Goals

K2 was created to try and push the boundaries of sustainability, performance, and research benefit.

We set out to answer three questions:

K2 Model – Goals

K2 was created to try and push the boundaries of sustainability, performance, and research benefit.

We set out to answer three questions:

1. Can a leading model be created sustainably?
(Can we train a “good” model more efficiently?)



When we started in 2023, the main “open” comparison was Meta’s Llama 2.

(Though Llama 3 was released during our development)

K2 Model – Goals

K2 was created to try and push the boundaries of sustainability, performance, and research benefit.

We set out to answer three questions:

1. Can a leading model be created sustainably?
(Can we train a “good” model more efficiently?)
2. What are the key ingredients to achieve meaningful advances in a model's reasoning, math, and coding ability?



When we started in 2023, the main “open” comparison was Meta’s Llama 2.

(Though Llama 3 was released during our development)

K2 Model – Goals

K2 was created to try and push the boundaries of sustainability, performance, and research benefit.

We set out to answer three questions:

1. Can a leading model be created sustainably?
(Can we train a “good” model more efficiently?)
2. What are the key ingredients to achieve meaningful advances in a model's reasoning, math, and coding ability?
3. How can we empower the next phase of language model research through the release of open source artifacts?



When we started in 2023, the main “open” comparison was Meta’s Llama 2.

(Though Llama 3 was released during our development)

K2 Model – Comparison with Llama 2 70B

K2 Model – Comparison with Llama 2 70B

K2-65B outperforms Llama 2 70B using 35% less compute.



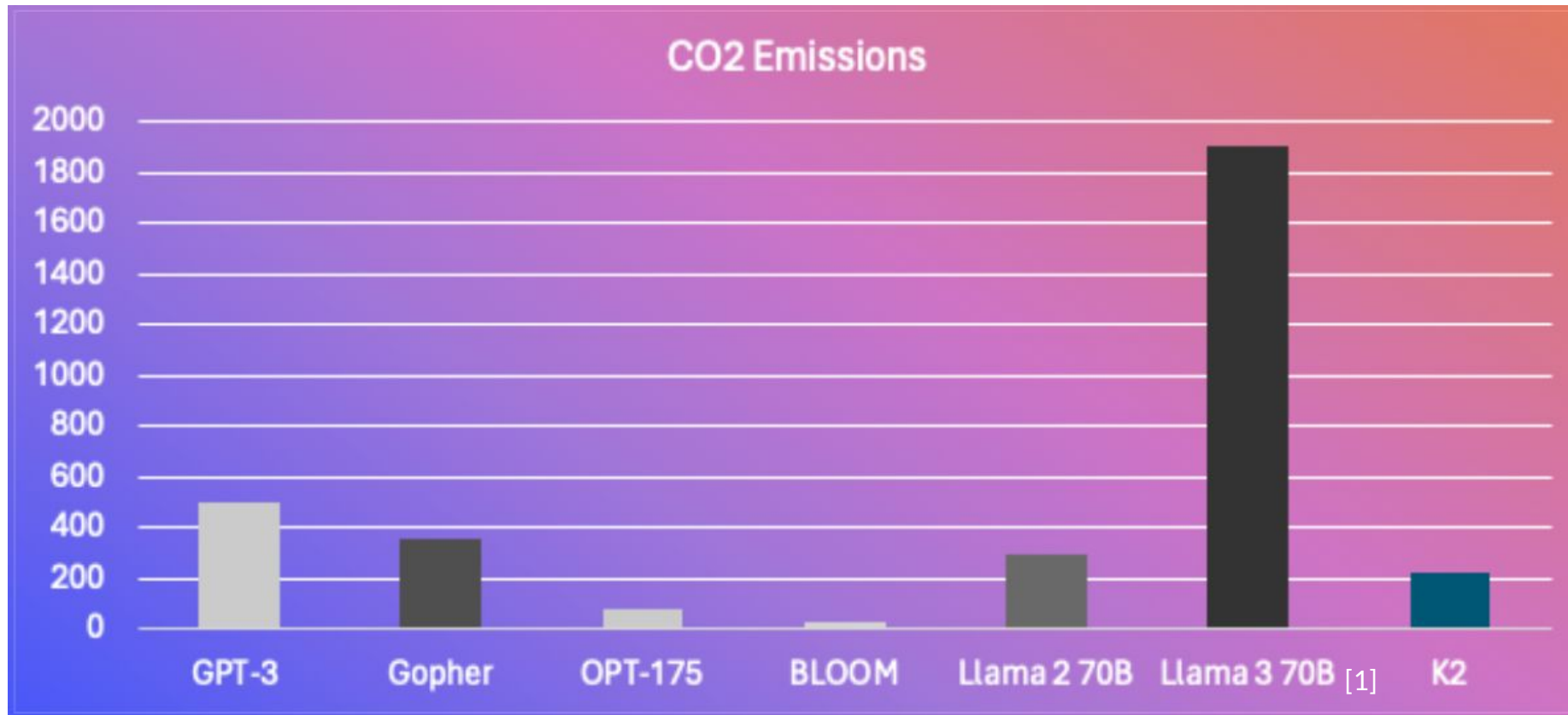
K2 Model – Comparison with Llama 2 70B

K2-65B outperforms Llama 2 70B using 35% less compute.



K2 Model – Comparison with Llama 2 70B

For comparison, see other related models (closed and open)



[1] *The Llama 3 Herd of Models*, Dubey et. al, 2024

K2 Model – Research and Development

We want to **enable future waves of LLM research and development.**

A cornerstone of research is producing reproducible artifacts for others to verify and advance your work.

“Fully open source” ⇒ **all** resources and learnings developed during K2's developments are made available to all, in particular...

K2 Model – Released artifacts

1.4T fully open training data tokens for advanced understanding into data mixtures, data efficiency, and to kickstart future training

+ 120 intermediate checkpoints of model weights to empower research into model training dynamics: <https://huggingface.co/LLM360/K2>

+ 100+ prompts and output showing how the model responses change over training lifetime: huggingface.co/spaces/LLM360/k2-gallery

+ 40+ metric curves collected through out training lifetime and made publicly available on W&B: <https://wandb.ai/llm360/K2-Diamond>

+ 21 evaluation metrics showing model holistic performance output over training lifetime: huggingface.co/spaces/LLM360/k2-eval-gallery

= 16TB+ collection size of model artifacts, the most complete set of ever released
(at least at the time)

K2 Model – Released artifacts

1.4T fully open training data tokens for advanced understanding into data mixtures, data efficiency, and to kickstart future training

+ **120** intermediate checkpoints of model weights to empower research into model training dynamics: <https://huggingface.co/LLM360/K2>

+ **100+** prompts and output showing how the model responses change over training lifetime: huggingface.co/spaces/LLM360/k2-gallery

+ **40+** metric curves collected through out training lifetime and made publicly available on W&B: <https://wandb.ai/llm360/K2-Diamond>

+ **21** evaluation metrics showing model holistic performance output over training lifetime: huggingface.co/spaces/LLM360/k2-eval-gallery

= **16TB+** collection size of model artifacts, the most complete set of ever released (*at least at the time*)

K2 Model – Released artifacts

1.4T fully open training data tokens for advanced understanding into data mixtures, data efficiency, and to kickstart future training

+ **120** intermediate checkpoints of model weights to empower research into model training dynamics: <https://huggingface.co/LLM360/K2>

+ **100+** prompts and output showing how the model responses change over training lifetime: huggingface.co/spaces/LLM360/k2-gallery

+ **40+** metric curves collected through out training lifetime and made publicly available on W&B: <https://wandb.ai/llm360/K2-Diamond>

+ **21** evaluation metrics showing model holistic performance output over training lifetime: huggingface.co/spaces/LLM360/k2-eval-gallery

= **16TB+** collection size of model artifacts, the most complete set of ever released
(at least at the time)

K2 Model – Released artifacts

1.4T fully open training data tokens for advanced understanding into data mixtures, data efficiency, and to kickstart future training

+ **120** intermediate checkpoints of model weights to empower research into model training dynamics: <https://huggingface.co/LLM360/K2>

+ **100+** prompts and output showing how the model responses change over training lifetime: huggingface.co/spaces/LLM360/k2-gallery

+ **40+** metric curves collected through out training lifetime and made publicly available on W&B: <https://wandb.ai/llm360/K2-Diamond>

+ **21** evaluation metrics showing model holistic performance output over training lifetime: huggingface.co/spaces/LLM360/k2-eval-gallery

= **16TB+** collection size of model artifacts, the most complete set of ever released
(at least at the time)

K2 Model – Released artifacts

1.4T fully open training data tokens for advanced understanding into data mixtures, data efficiency, and to kickstart future training

+ **120** intermediate checkpoints of model weights to empower research into model training dynamics: <https://huggingface.co/LLM360/K2>

+ **100+** prompts and output showing how the model responses change over training lifetime: huggingface.co/spaces/LLM360/k2-gallery

+ **40+** metric/log curves collected through out training lifetime and made publicly available on W&B: <https://wandb.ai/llm360/K2-Diamond>

+ **21** evaluation metrics showing model holistic performance output over training lifetime: huggingface.co/spaces/LLM360/k2-eval-gallery

= **16TB+** collection size of model artifacts, the most complete set of ever released
(at least at the time)

K2 Model – Released artifacts

1.4T fully open training data tokens for advanced understanding into data mixtures, data efficiency, and to kickstart future training

+ **120** intermediate checkpoints of model weights to empower research into model training dynamics: <https://huggingface.co/LLM360/K2>

+ **100+** prompts and output showing how the model responses change over training lifetime: huggingface.co/spaces/LLM360/k2-gallery

+ **40+** metric/log curves collected through out training lifetime and made publicly available on W&B: <https://wandb.ai/llm360/K2-Diamond>

+ **21** evaluation metrics showing model holistic performance output over training lifetime: huggingface.co/spaces/LLM360/k2-eval-gallery

= **16TB+** collection size of model artifacts, the most complete set of ever released
(at least at the time)

K2 Model – Released artifacts

1.4T fully open training data tokens for advanced understanding into data mixtures, data efficiency, and to kickstart future training

+ **120** intermediate checkpoints of model weights to empower research into model training dynamics: <https://huggingface.co/LLM360/K2>

+ **100+** prompts and output showing how the model responses change over training lifetime: huggingface.co/spaces/LLM360/k2-gallery

+ **40+** metric/log curves collected through out training lifetime and made publicly available on W&B: <https://wandb.ai/llm360/K2-Diamond>

+ **21** evaluation metrics showing model holistic performance output over training lifetime: huggingface.co/spaces/LLM360/k2-eval-gallery

= **16TB+** collection size of model artifacts, the most complete set of ever released (*at least at the time*)

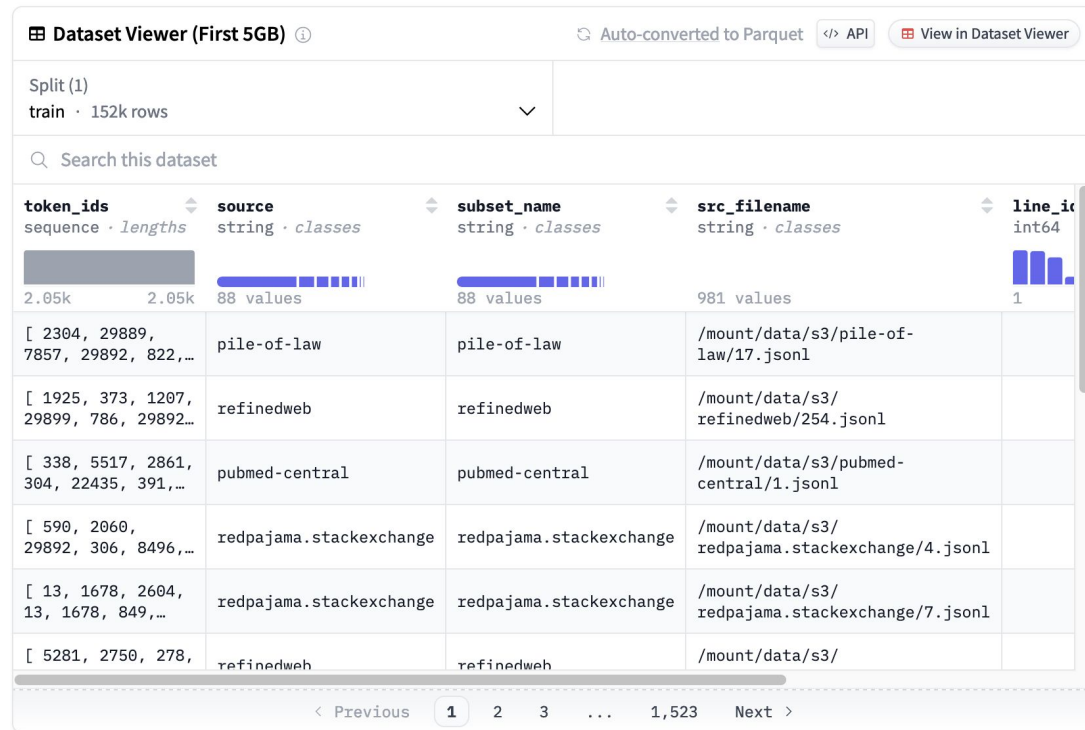
K2 Research Artifacts – Training Data

K2 Research Artifacts – Training Data

Training data is a closely held secret by enterprises such as OpenAI and Meta.

K2 openly shares all data to advance understanding into data **composition, mixtures, and stages** to help deliver the next generation of LLMs.

Check out the data here:
huggingface.co/datasets/LLM360/K2Datasets



Dataset Viewer (First 5GB) Auto-converted to Parquet API View in Dataset Viewer

Split (1)
train · 152k rows

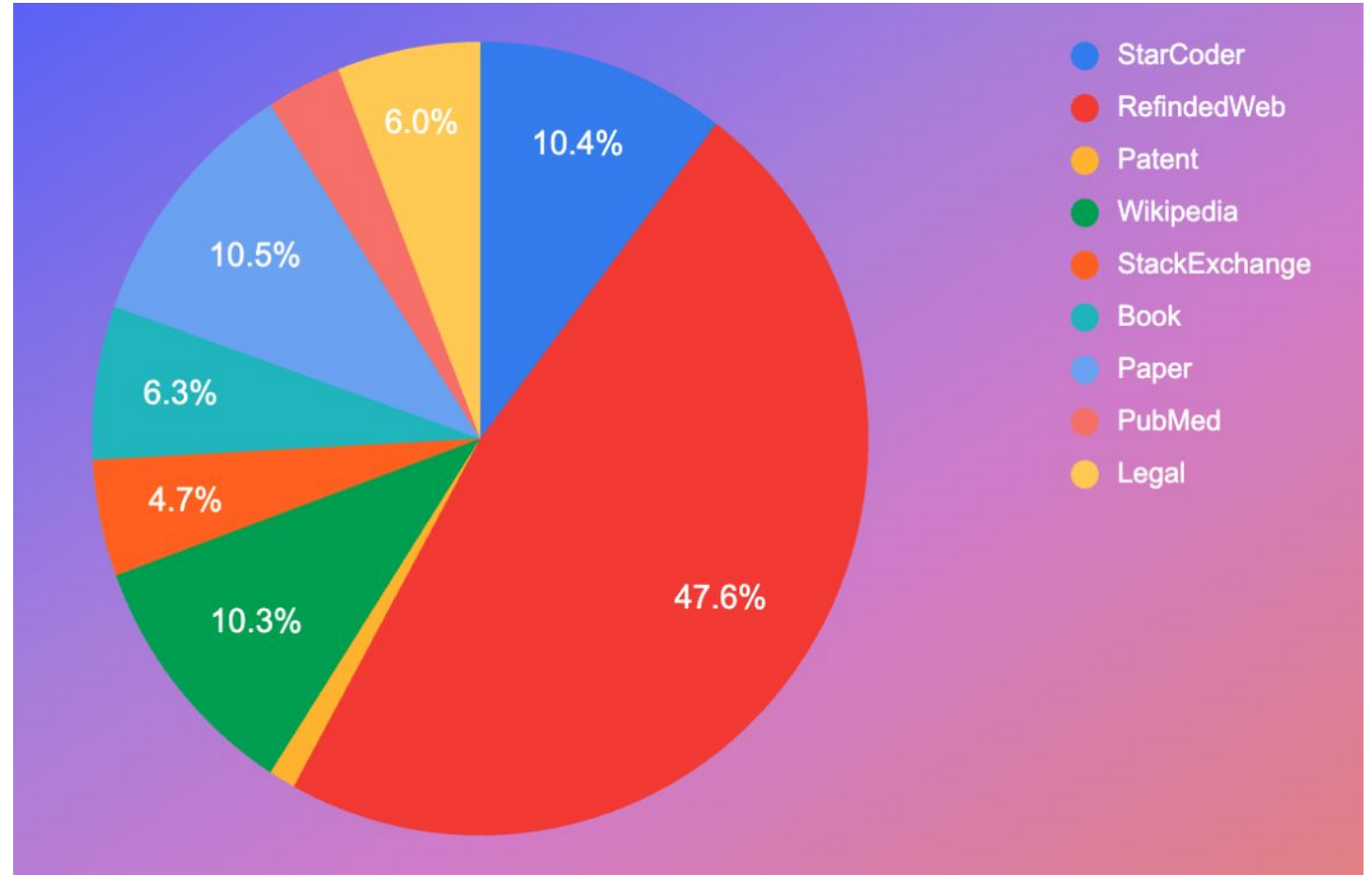
Search this dataset

token_ids	source	subset_name	src_filename	line_id
sequence · lengths 2.05k 2.05k	string · classes 88 values	string · classes 88 values	string · classes 981 values	int64 1
[2304, 29889, 7857, 29892, 822, ...]	pile-of-law	pile-of-law	/mount/data/s3/pile-of-law/17.jsonl	
[1925, 373, 1207, 29899, 786, 29892, ...]	refinedweb	refinedweb	/mount/data/s3/refinedweb/254.jsonl	
[338, 5517, 2861, 304, 22435, 391, ...]	pubmed-central	pubmed-central	/mount/data/s3/pubmed-central/1.jsonl	
[590, 2060, 29892, 306, 8496, ...]	redpajama.stackexchange	redpajama.stackexchange	/mount/data/s3/redpajama.stackexchange/4.jsonl	
[13, 1678, 2604, 13, 1678, 849, ...]	redpajama.stackexchange	redpajama.stackexchange	/mount/data/s3/redpajama.stackexchange/7.jsonl	
[5281, 2750, 278, ...]	refinedweb	refinedweb	/mount/data/s3/	

< Previous 1 2 3 ... 1,523 Next >

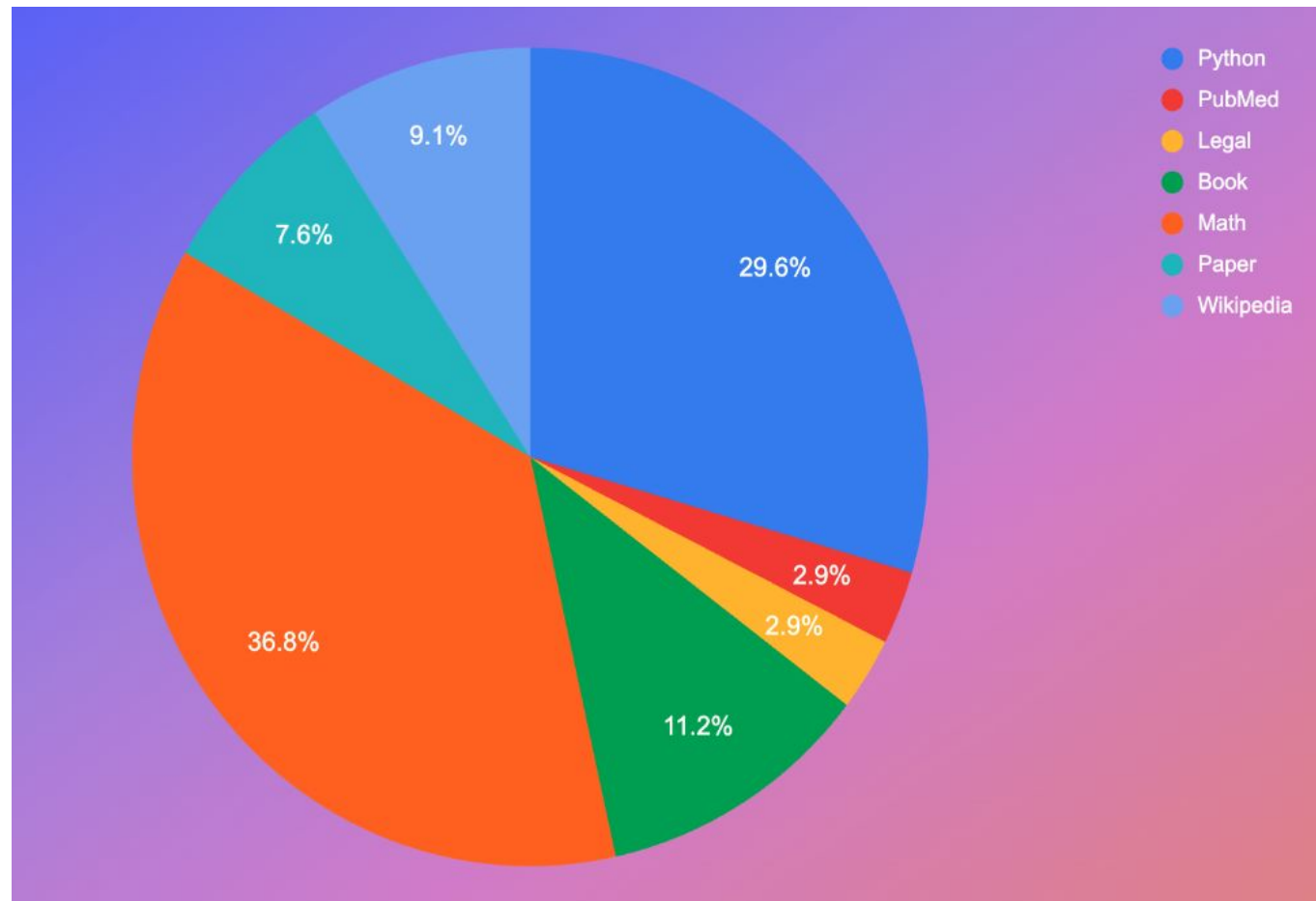
K2 Training Data – Stage 1

1.4 trillion tokens providing diverse knowledge, math, legal, and coding ability.



K2 Training Data – Stage 2

96.4 billion tokens heavily weighted with math and coding data.



K2 Research Artifacts – Code

K2 Research Artifacts – Code

All code is released so that others may reproduce K2 to expand or verify our findings.

- With permissible license (Apache 2.0)!

Includes: data preparation, training, fine-tuning, and evaluation.

Check out the code here:
<https://github.com/LLM360>

K2-65B

- [k2-data-prep](#) - Data preparation code for K2-65B model.
- [k2-train](#) - Training code for K2-65B model.

K2 Research Artifacts – Curves and Logs

K2 Research Artifacts – Curves and Logs

We show all training curves using raw logs from our Weights-&-Biases (experiment tracking) page.

Includes: evaluation metrics, throughput, system resources (disk, network, memory, gpu), and more.

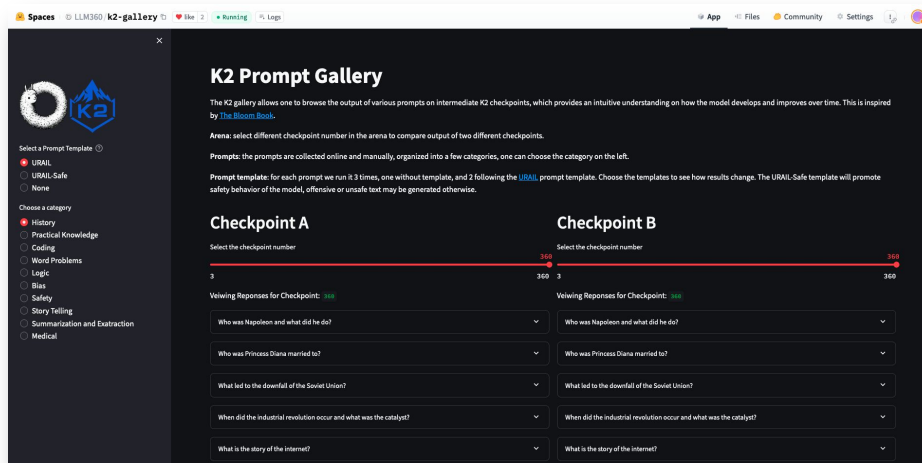
Check out the w&b page here:
<https://wandb.ai/llm360/projects>



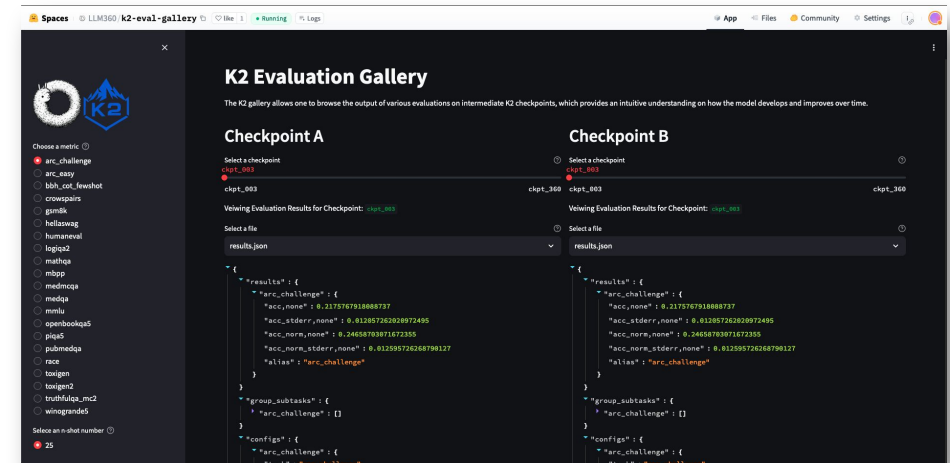
K2 Research Artifacts – Galleries

K2 Research Artifacts – Galleries

The **K2 Prompt Gallery** and **K2 Evaluation Gallery** provide insights into the model's behavior and abilities.



View model responses to curated prompts and compare across all model checkpoints



View output for 21 evaluations across full training run / all model checkpoints.

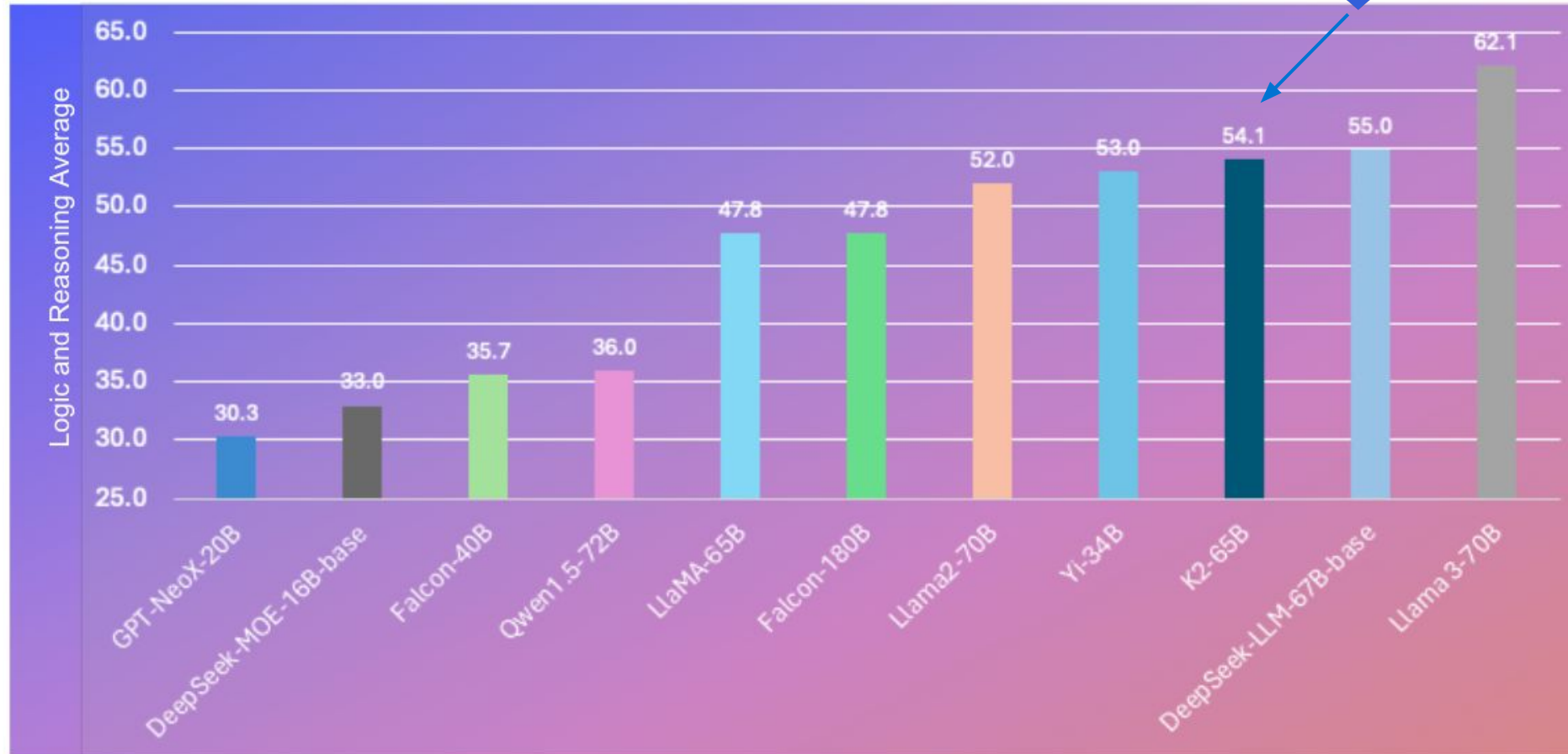
K2 Performance Results

K2 Performance Results

Based on our stages of training data (especially 2nd stage), the goal was to try and maximize performance on:

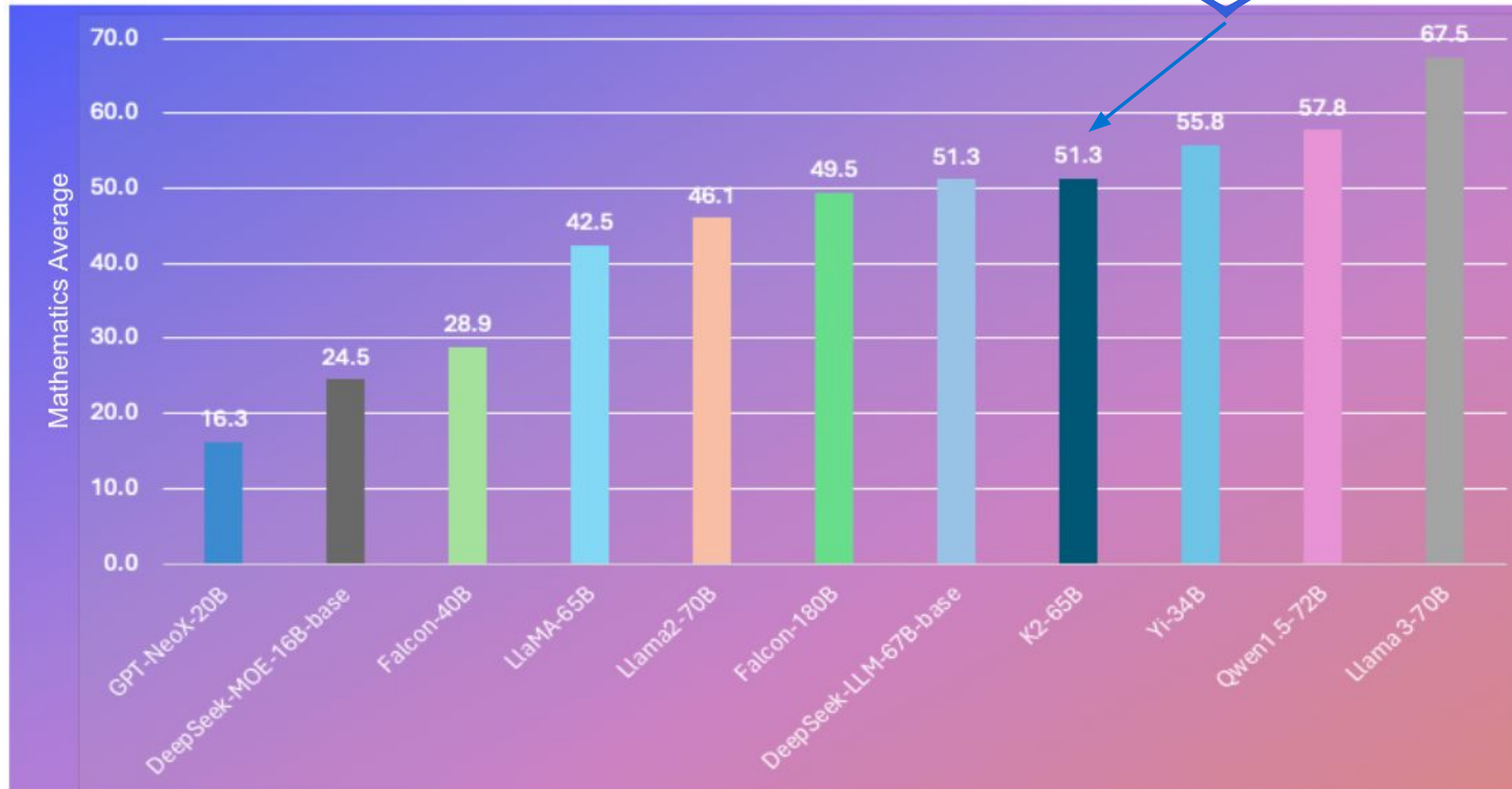
- Logic and reasoning
- Mathematics
- Coding

K2 Performance Results – Logic and Reasoning



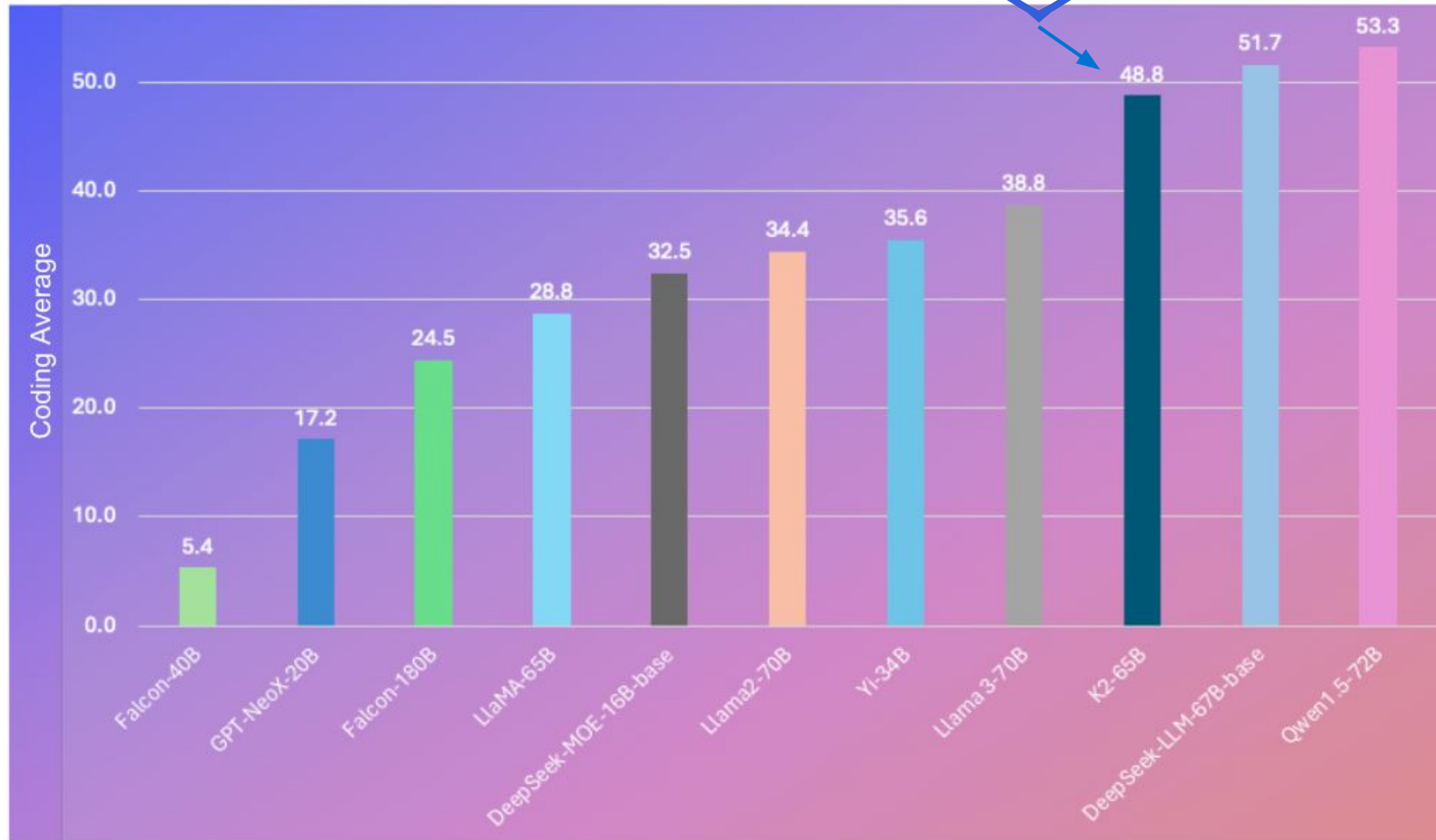
K2 outperforms the majority of open weight models of its size while performing similar to larger models trained on significantly more data.

K2 Performance Results – Mathematics



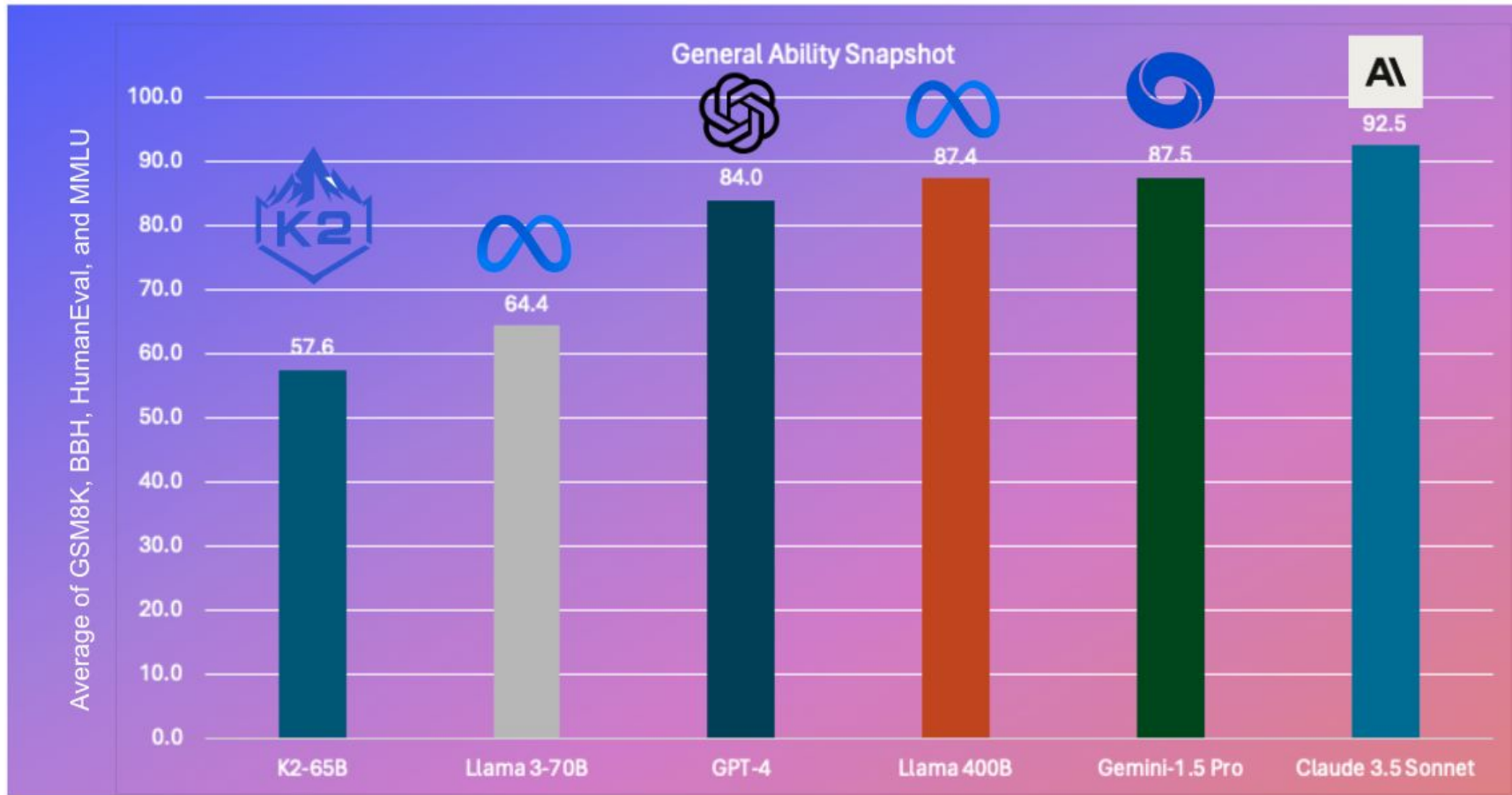
K2 outperforms the majority of open weight models of its size while performing similar to larger models trained on significantly more data.

K2 Performance Results – Coding



K2 outperforms the majority of open weight models of its size while performing similar to larger models trained on significantly more data.

K2 Performance Results – Closing the Gap to SOTA?



Still a ways to go to compete against the top models!

So how do we train these models?

Next:

A (Brief) Pretrainer's Guide to Large Language Models

A (Brief) Pretrainer's Guide to Large Language Models

A (Brief) Pretrainer's Guide to Large Language Models

- Today, your boss knocks on your door:
 - Boss: Can you to prepare an LLM training proposal by EOD? Let's beat GPT-4 next week.
 - You: ??
- What would you reply?



A (Brief) Pretrainer's Guide to Large Language Models

- Today, your ~~boss~~ ^{Advisor} knocks on your door:
 - Boss: Can you to prepare an LLM training proposal by EOD? Let's beat GPT-4 next week.
 - You: ??
- What would you reply?

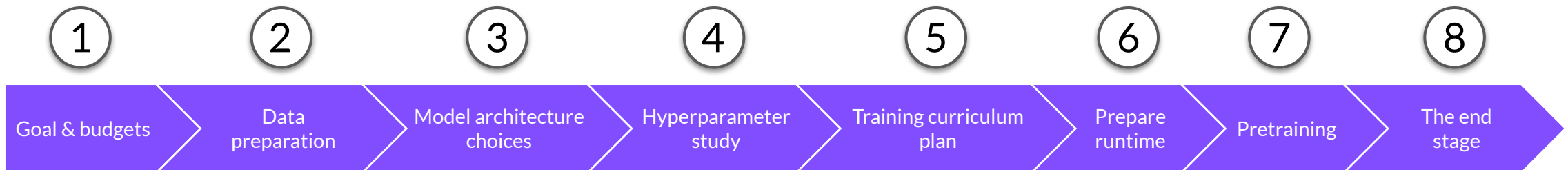


A (Brief) Pretrainer's Guide to Large Language Models

Goal: Go through the main considerations and details of all steps (*before, during, and after*) when pretraining a LLM.

A (Brief) Pretrainer's Guide to Large Language Models

Goal: Go through the main considerations and details of all steps (*before, during, and after*) when pretraining a LLM.



A (Brief) Pretrainer's Guide to Large Language Models



A (Brief) Pretrainer's Guide to Large Language Models

1



Large-scale training is about tradeoffs



Large-scale training is about tradeoffs

- Much of large scale training just comes down to engineering...
- Engineering requires making tradeoffs of resources.



Large-scale training is about tradeoffs

- Much of large scale training just comes down to engineering...
- Engineering requires making tradeoffs of resources.
- The science here is often about predicting the tradeoff.



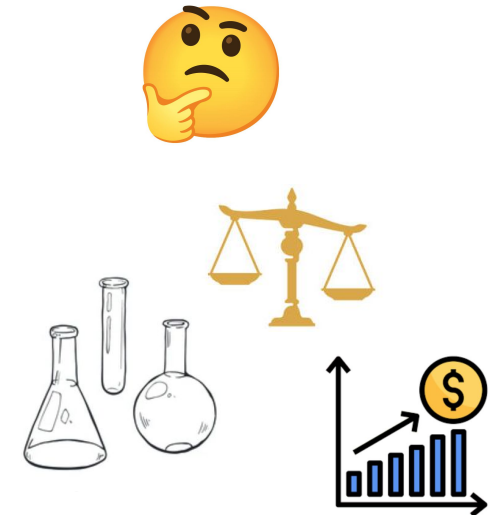
Determine the Goals and Budgets

As a training team lead, need to really **clarify the downstream goal** of your model.

Determine the Goals and Budgets

As a training team lead, need to really **clarify the downstream goal** of your model.

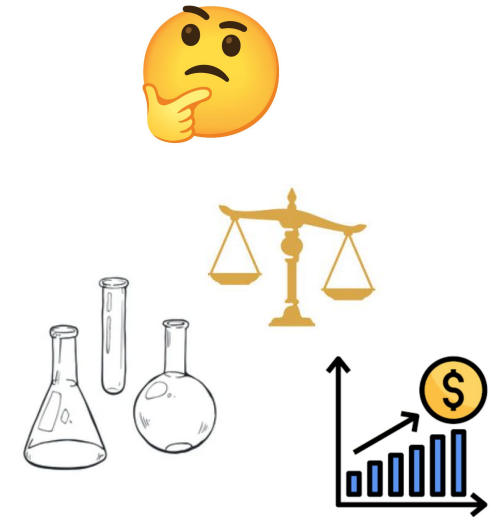
- What's the main use case of the model?
 - What are the major knowledge domains that need to be covered? (Finance, biomedical, legal, etc?)
 - What are the abilities that the model should have? Logical reasoning or programming?



Determine the Goals and Budgets

As a training team lead, need to really **clarify the downstream goal** of your model.

- What's the main use case of the model?
 - What are the major knowledge domains that need to be covered? (Finance, biomedical, legal, etc?)
 - What are the abilities that the model should have? Logical reasoning or programming?
- Useful to write down a *model use-case card*
 - Evaluation types and target scores.
 - Model capabilities.
 - Use-case restrictions (latency, model size, etc).



Determine the Goals and Budgets

As a training team lead, need to really **clarify the downstream goal** of your model.

- With the goals in mind, it's useful to determine precisely *what you are optimizing for*:
 - Optimize performance
 - Optimize performance per dollar
 - Optimize ROI per dollar

Example Optimization Goals

Performance (“raw score”)

- Highest performance achievable given some constraints (e.g., limited data)

Performance per dollar (PPD)

- Find a training setting to maximize performance per dollar spent
- Dollar can be spent on data, computation and other resources

Final ROI

- In practice, the optimal compute budget requires one to consider the tradeoff between cost of pretraining and cost of inference

Example Optimization Goals

Performance (“raw score”)

- Highest performance achievable given some constraints (e.g., limited data)

Performance per dollar (PPD)

- Find a training setting to maximize performance per dollar spent
- Dollar can be spent on data, computation and other resources

Final ROI

- In practice, the optimal compute budget requires one to consider the tradeoff between cost of pretraining and cost of inference

Example Optimization Goals

Performance (“raw score”)

- Highest performance achievable given some constraints (e.g., limited data)

Performance per dollar (PPD)

- Find a training setting to maximize performance per dollar spent
- Dollar can be spent on data, computation and other resources

Final ROI

- In practice, the optimal compute budget requires one to consider the tradeoff between cost of pretraining and cost of inference

Example Optimization Goals

Performance (“raw score”)

- Highest performance achievable given some constraints (e.g., limited data)

Performance per dollar (PPD)

- Find a training setting to maximize performance per dollar spent
- Dollar can be spent on data, computation and other resources

Final ROI

- In practice, the optimal compute budget requires one to consider the tradeoff between cost of pretraining and cost of inference

Key Decision: Model Scale

Models designed with high capacity/scale can potentially achieve high performance.

Key Decision: Model Scale

Models designed with high capacity/scale can potentially achieve high performance.

Main factors determining model scale:

- Model architecture (Transformer vs. RNN vs. State Space Model)
- Training FLOPs
 - # model parameters
 - # tokens of pretraining data

Key Decision: Model Scale

Models designed with high capacity/scale can potentially achieve high performance.

Main factors determining model scale:

- Model architecture (Transformer vs. RNN vs. State Space Model)
- Training FLOPs
 - # model parameters
 - # tokens of pretraining data

⇒ **Neural Scaling Laws**: the study of neural network behaviors that are predictable by scaling model size, dataset size, and training time across many orders of magnitude.

Scaling Laws

Try to develop a law (relationship) to predict loss as you scale #-model-parameters, #-tokens, etc.

Scaling Laws

Try to develop a law (relationship) to predict loss as you scale #-model-parameters, #-tokens, etc.

One example:

Chinchilla Scaling Law Formulation [1]

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}.$$

L: Loss

N: # Model Parameter

D: # Training Tokens

E: A constant capturing the Entropy of the text

E, A, B, α , and β are to be fit during experiments

Scaling Laws

Try to develop a law (relationship) to predict loss as you scale #-model-parameters, #-tokens, etc.

One example:

Chinchilla Scaling Law Formulation [1]

- Scaling law study allows one to estimate the model behaviors of high-capacity by experimenting on low-capacity ones.
- Model loss is one example; other predictions are also possible.

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}.$$

L: Loss

N: # Model Parameter

D: # Training Tokens

E: A constant capturing the Entropy of the text

E, A, B, α , and β are to be fit during experiments

Emergent Abilities

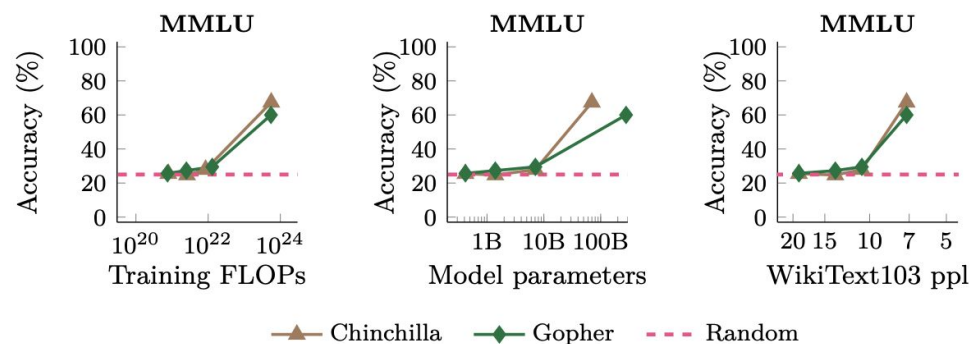
The phenomenon of “emergent abilities” makes the capacity decision more important.

- Want to choose the right budget that reaches desired ability.

Emergent Abilities

The phenomenon of “emergent abilities” makes the capacity decision more important.

- Want to choose the right budget that reaches desired ability.



Key metrics/abilities may only start to emerge (i.e., significantly improve) when model exceed certain capacity.

	Emergent scale		Model	Reference
	Train. FLOPs	Params.		
Few-shot prompting abilities				
• Addition/subtraction (3 digit)	2.3E+22	13B	GPT-3	Brown et al. (2020)
• Addition/subtraction (4-5 digit)	3.1E+23	175B		
• MMLU Benchmark (57 topic avg.)	3.1E+23	175B	GPT-3	Hendrycks et al. (2021a)
• Toxicity classification (CivilComments)	1.3E+22	7.1B	Gopher	Rae et al. (2021)
• Truthfulness (Truthful QA)	5.0E+23	280B		
• MMLU Benchmark (26 topics)	5.0E+23	280B		
• Grounded conceptual mappings	3.1E+23	175B	GPT-3	Patel & Pavlick (2022)
• MMLU Benchmark (30 topics)	5.0E+23	70B	Chinchilla	Hoffmann et al. (2022)
• Word in Context (WiC) benchmark	2.5E+24	540B	PaLM	Chowdhery et al. (2022)
• Many BIG-Bench tasks (see Appendix E)	Many	Many	Many	BIG-Bench (2022)
Augmented prompting abilities				
• Instruction following (finetuning)	1.3E+23	68B	FLAN	Wei et al. (2022a)
• Scratchpad: 8-digit addition (finetuning)	8.9E+19	40M	LaMDA	Nye et al. (2021)
• Using open-book knowledge for fact checking	1.3E+22	7.1B	Gopher	Rae et al. (2021)
• Chain-of-thought: Math word problems	1.3E+23	68B	LaMDA	Wei et al. (2022b)
• Chain-of-thought: StrategyQA	2.9E+23	62B	PaLM	Chowdhery et al. (2022)
• Differentiable search index	3.3E+22	11B	T5	Tay et al. (2022b)
• Self-consistency decoding	1.3E+23	68B	LaMDA	Wang et al. (2022b)
• Leveraging explanations in prompting	5.0E+23	280B	Gopher	Lampinen et al. (2022)
• Least-to-most prompting	3.1E+23	175B	GPT-3	Zhou et al. (2022)
• Zero-shot chain-of-thought reasoning	3.1E+23	175B	GPT-3	Kojima et al. (2022)
• Calibration via P(True)	2.6E+23	52B	Anthropic	Kadavath et al. (2022)
• Multilingual chain-of-thought reasoning	2.9E+23	62B	PaLM	Shi et al. (2022)
• Ask me anything prompting	1.4E+22	6B	EleutherAI	Arora et al. (2022)

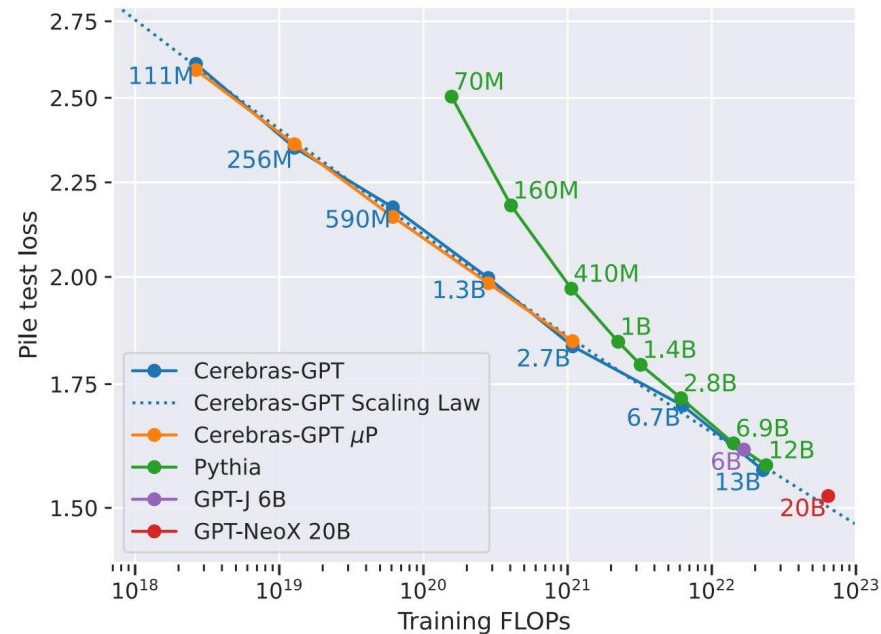
Table of emergent abilities at different model capacities.

Examples of Scaling Law Studies

Example scaling law studies: predicting loss vs other metrics.

Examples of Scaling Law Studies

Example scaling law studies: predicting loss vs other metrics.



Scaling law of different model families [1]

[1] Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster, arXiv:2304.03208

[2] Emergent and Predictable Memorization in Large Language Models, arXiv:2304.11158

Examples of Scaling Law Studies

Example scaling law studies: predicting loss vs other metrics.

- In [2], a 12B model's metric scores can be (somewhat) predicted by smaller models

Model	Precision	Recall
Pythia-70M	0.956	0.197
Pythia-160M	0.948	0.289
Pythia-410M	0.940	0.401
Pythia-1.0B	0.931	0.512
Pythia-1.4B	0.926	0.554
Pythia-2.8B	0.909	0.658
Pythia-6.9B	0.884	0.795
Pythia-12B	—	—

Example: Text memorization prediction with scaling laws [2]

[1] Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster, arXiv:2304.03208

[2] Emergent and Predictable Memorization in Large Language Models, arXiv:2304.11158

Extra tradeoffs

Also need to consider:

Extra tradeoffs

Also need to consider:

- Scaling Law Study Cost vs. Training Cost
 - The cost to fit a good scaling law curve is also significant, considering all the factors to be tried (e.g., hyperparameters, data selection).

Extra tradeoffs

Also need to consider:

- **Scaling Law Study Cost vs. Training Cost**
 - The cost to fit a good scaling law curve is also significant, considering all the factors to be tried (e.g., hyperparameters, data selection).
- **Training Resources vs. Supporting Resources**
 - Don't use all your GPUs for training, always reserve enough for evaluation, analysis.
 - There is a tradeoff between model evaluation frequency vs. rollback cost.

Extra tradeoffs

One more consideration:

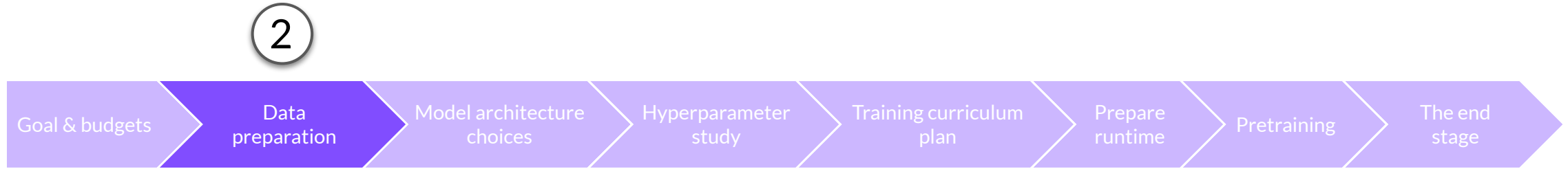
- Choose a model size that works well on the target inference hardware.
 - 7B works great on a single A100 GPU.
 - One may also consider whether a quantized (8-Bit, 4-Bit) version of the model fit into the memory of weaker hardware.
 - Sometimes other restrictions – e.g., in some parallel inference, make sure key layer sizes are divisible by power of 2.

A (Brief) Pretrainer's Guide to Large Language Models

2



A (Brief) Pretrainer's Guide to Large Language Models



One of the most important parts of LLM pretraining.

Data Collection

Data Collection

Recall in Scaling Laws, data plays a crucial role in final performance.

- Data size denoted by D
- Intuitively, data quality changes B and β

Chinchilla Scaling Law

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}.$$

Data Collection

Recall in Scaling Laws, data plays a crucial role in final performance.

- Data size denoted by D
- Intuitively, data quality changes B and β

Chinchilla Scaling Law

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}.$$

⇒ Collecting a high quality and large corpus is essential for producing in the final model.

Determine the data size based on budget and loss goal.

Special Domains

Sometimes one may want to collect data in special domains

Programming Code

- Enable model's programming ability.
- Helpful in function call.
- Could improve reasoning.

Multiple Languages

- Multi-lingual / non-English corpus.
- Note that filtering rules that rely on language statistics might need to be adapted.

Specialist Domains

- Professional areas such as legal, medical.
- Special document formats such as tables, forms, etc.

Special Domains

Sometimes one may want to collect data in special domains

Programming Code

- Enable model's programming ability.
- Helpful in function call.
- Could improve reasoning.

Multiple Languages

- Multi-lingual / non-English corpus.
- Note that filtering rules that rely on language statistics might need to be adapted.

Specialist Domains

- Professional areas such as legal, medical.
- Special document formats such as tables, forms, etc.

Special Domains

Sometimes one may want to collect data in special domains

Programming Code

- Enable model's programming ability.
- Helpful in function call.
- Could improve reasoning.

Multiple Languages

- Multi-lingual / non-English corpus.
- Note that filtering rules that rely on language statistics might need to be adapted.

Specialist Domains

- Professional areas such as legal, medical.
- Special document formats such as tables, forms, etc.

Special Domains

Sometimes one may want to collect data in special domains

Programming Code

- Enable model's programming ability.
- Helpful in function call.
- Could improve reasoning.

Multiple Languages

- Multi-lingual / non-English corpus.
- Note that filtering rules that rely on language statistics might need to be adapted.

Specialist Domains

- Professional areas such as legal, medical.
- Special document formats such as tables, forms, etc.

Data Preprocessing

When dealing with internet-scale data, it is typical to filter documents based on a pipeline of heuristic rules.

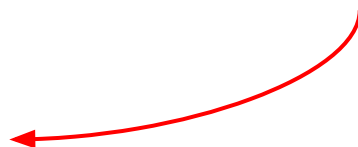
(Note that rules will be different in different domains)

Data Preprocessing

When dealing with internet-scale data, it is typical to filter documents based on a pipeline of heuristic rules.

(Note that rules will be different in different domains)

From TxT360 project...
will describe later on.



- 2.1.4 Duplicate Ngram
- 2.2 Line-wise Heuristics
- 2.4.1 Curly Bracket
- 1.3 Line-Level Removal from RefinedWeb
- 2.1.1+2.1.2 Fraction of Duplicate Lines
- 1.2 Special Word Java Script
- 2.3 Document Statistics
- 0.2.1 URL blacklist
- 3.1 Line-Level Deduplication
- 0. Text Extraction
- 0.1 Language Identification

LLM360 has released a dataset (TxT360) with careful data processing, with fully documented decision process

Data Deduplication

Data Deduplication

Empirically, many studies have confirmed that deduplication can improve model quality [1-3].

- Falcon uses RefinedWeb (*deduplicated Common Crawl*).
- LLM360/Crystal uses SlimPajama (*deduplicated RedPajama*).

[1] To Repeat or Not To Repeat: Insights from Scaling LLM under Token-Crisis, arXiv:2305.13230

[2] Deduplicating Training Data Makes Language Models Better, arXiv:2107.06499

[3] Scaling Laws and Interpretability of Learning from Repeated Data, arXiv:2205.10487

[4] Scaling Data-Constrained Language Models, arXiv:2305.16264

Data Deduplication

Empirically, many studies have confirmed that deduplication can improve model quality [1-3].

- Falcon uses RefinedWeb (*deduplicated Common Crawl*).
- LLM360/Crystal uses SlimPajama (*deduplicated RedPajama*).

OTOH, there is also evidence that it can be good to repeat certain training data

- [4] trains the model on multiple epochs.
- Llama and several of our models (LLM360/Amber, K2) also upweight certain high quality datasets (such as Wikipedia). K2 performs generally better after upweighting.

[1] To Repeat or Not To Repeat: Insights from Scaling LLM under Token-Crisis, arXiv:2305.13230

[2] Deduplicating Training Data Makes Language Models Better, arXiv:2107.06499

[3] Scaling Laws and Interpretability of Learning from Repeated Data, arXiv:2205.10487

[4] Scaling Data-Constrained Language Models, arXiv:2305.16264

Data Deduplication

In practice, we upsampled about 10% data in K2 training.

Some high quality data sources (such as Wikipedia) are repeated ~6 times.

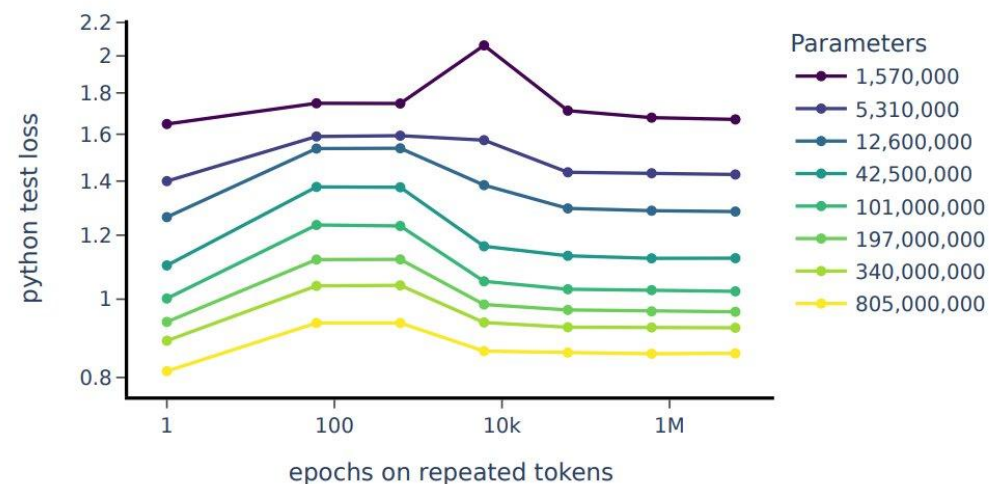
Dataset	Starting Tokens	Multiplier	Total Tokens	% of Total
dm-math	4.33B	3x	13B	1%
pubmed-abs tracts	4.77B	3x	14.3B	1.1%
uspto	4.77B	3x	14.3B	1.1%
pubmed-cen tral	26B	1x	26B	2%
redpajama.a rxiv	27.3B	1x	27.3B	2.1%
starcoder.sp m	67.6B	0.5x	33.8B	2.6%
starcoder.fim	67.6B	0.5x	33.8B	2.6%
redpajama.s tackexchang e	61.1B	1x	61.1B	4.7%
starcoder	132.6B	0.5x	66.3B	5.1%
pile-of-law	76.7B	1x	76.7B	5.9%
redpajama.b ook	80.6B	1x	80.6B	6.2%
s2orc	107.9B	1x	107.9B	8.3%
redpajama.w ikipedia	22.1B	6x	132.6B	10.2%
refinedweb	612.3B	1x	612.3B	47.1%
Totals	-	-	1.3T	100%

Data Deduplication

[1] hypothesize that *a lot* of duplicate data causes the model to replace generalization with memorization

And that duplicate data induces a double-descent [2] phenomenon

Repeated Python Data also causes Double Descent



[1] Scaling Laws and Interpretability of Learning from Repeated Data, arXiv:2205.10487

[2] Superposition, Memorization, and Double Descent, Transformer Circuits Thread

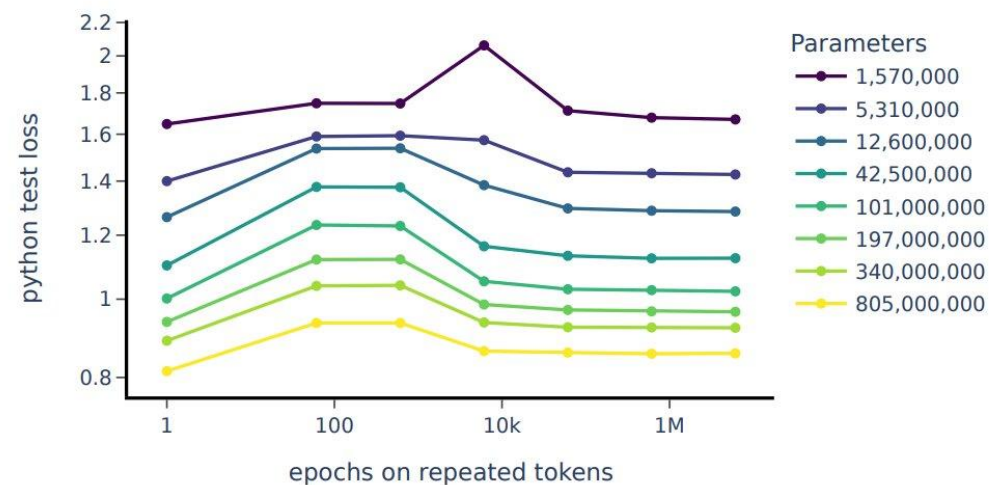
Data Deduplication

[1] hypothesize that *a lot* of duplicate data causes the model to replace generalization with memorization

And that duplicate data induces a double-descent [2] phenomenon

- Repeating a few times does not cause much damage
- Repeating very many times also does not cause much damage
- Eventually grokking happens, but hard to predict “when” in real-world scenarios

Repeated Python Data also causes Double Descent



[1] Scaling Laws and Interpretability of Learning from Repeated Data, arXiv:2205.10487

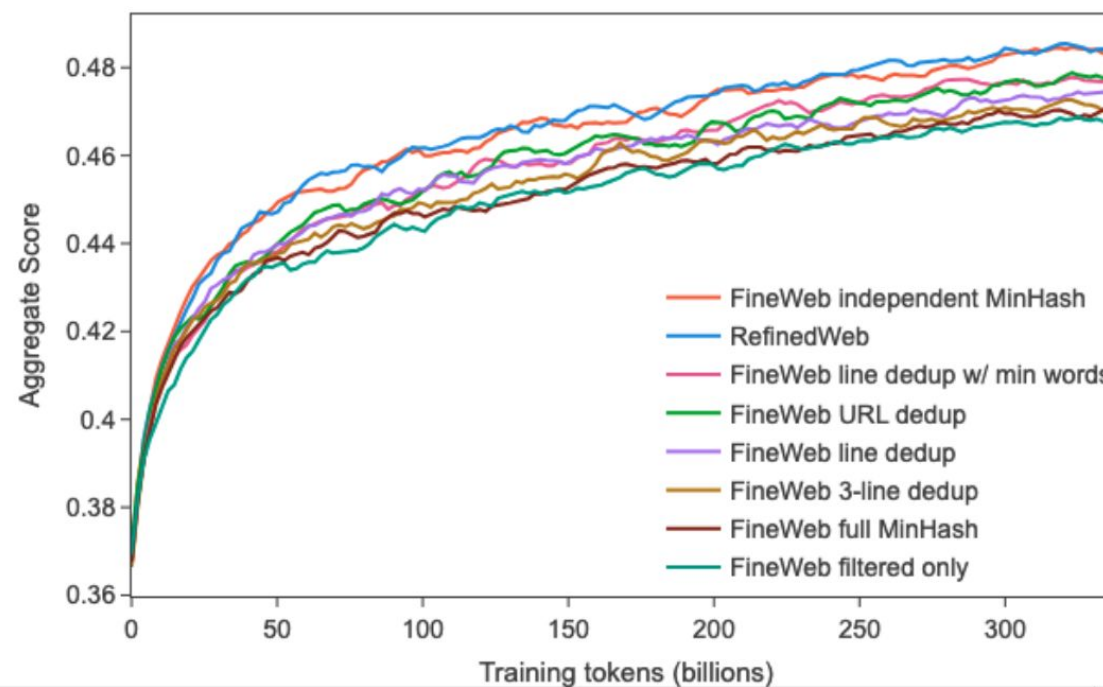
[2] Superposition, Memorization, and Double Descent, Transformer Circuits Thread

Data Deduplication

Some studies on deduplication are less clear/positive.

The Fineweb [1] report shows that deduplicating more greedily and globally actually hurts performance.

Deduplicating within groups (that are organized by year) is the best [1].



[1] <https://huggingface.co/spaces/HuggingFaceFW/blogpost-fineweb-v1>

Data Deduplication – Summary

Data Deduplication – Summary

The impact of data deduplication on performance is complex but important!

- Be careful when repeating the data, or training multiple epochs.
- Repeating high quality data is a bit safer.
- Conduct a scaling law study on memorization [1, 2] can be helpful.

[1] Emergent and Predictable Memorization in Large Language Models, arXiv:2304.11158

[2] Scaling Laws and Interpretability of Learning from Repeated Data, arXiv:2205.10487

[3] Scaling Data-Constrained Language Models, arXiv:2305.16264

Data Deduplication – Summary

The impact of data deduplication on performance is complex but important!

- Be careful when repeating the data, or training multiple epochs.
- Repeating high quality data is a bit safer.
- Conduct a scaling law study on memorization [1, 2] can be helpful.

Main takeaways

- If you deduplicate, you can control repetition precisely, and know the repeats of your data.
- Monitor when the training process reaches repeated data.
 - Check for key metrics at the start of epochs.
 - Pay attention to generalization results in addition to memorization evals.

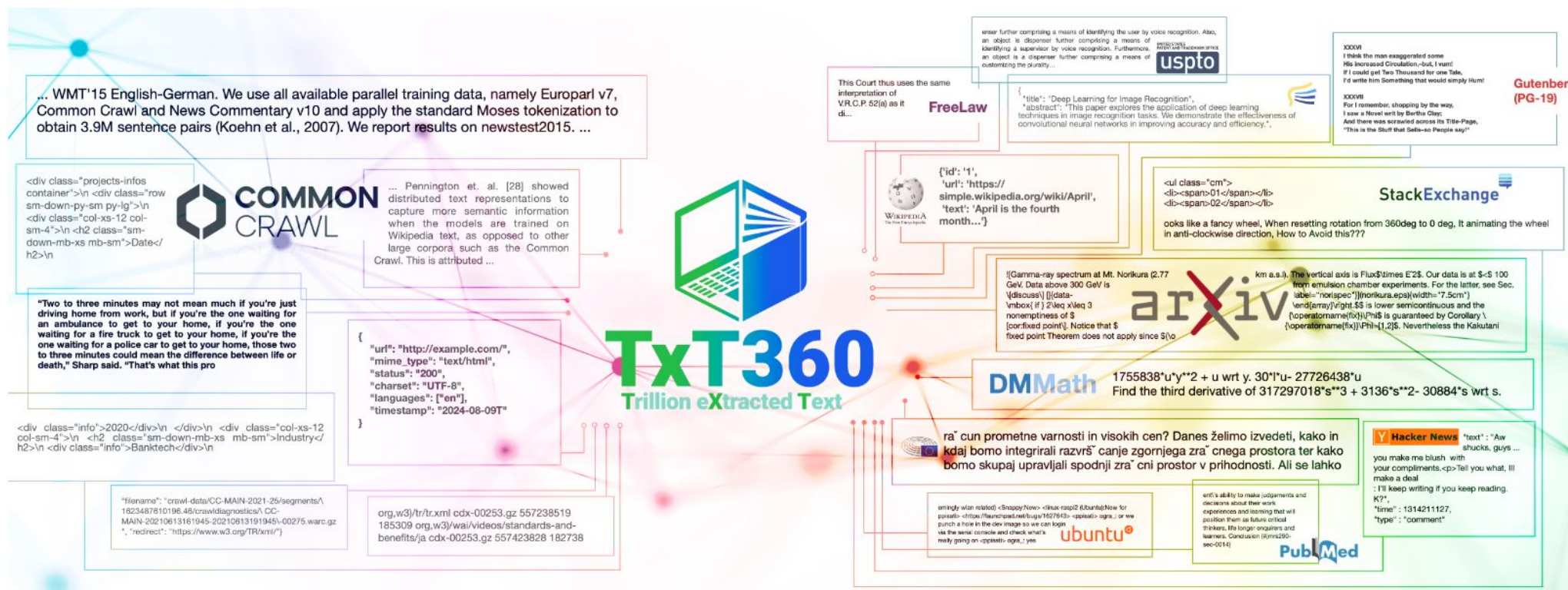
[1] Emergent and Predictable Memorization in Large Language Models, arXiv:2304.11158

[2] Scaling Laws and Interpretability of Learning from Repeated Data, arXiv:2205.10487

[3] Scaling Data-Constrained Language Models, arXiv:2305.16264

TxT360

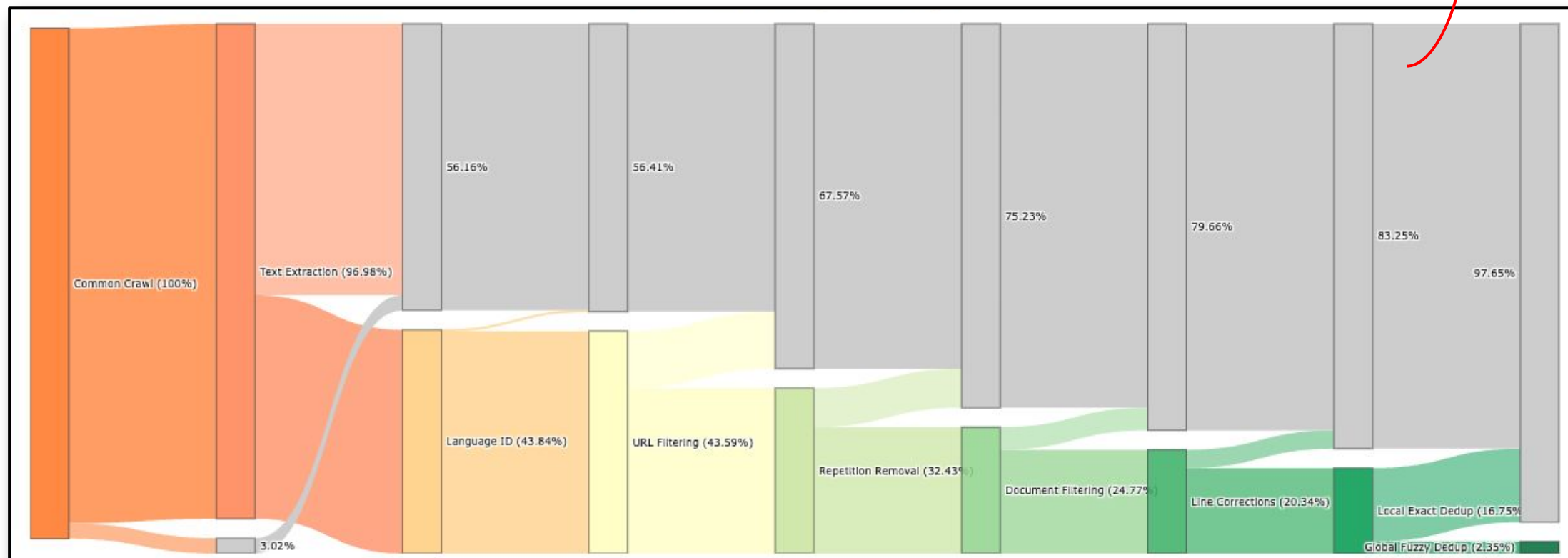
- TxT360 (*Trillion eXtracted Text*) is a new pretraining dataset released by LLM360.
- We deduplicate 99 CommonCrawl snapshots and 14 high-quality data sources from diverse domains (e.g., FreeLaw, PG-19, etc.).
- 15 trillion tokens in total.



TxT360

- Pipeline of filtering steps on web data – for details see the blog post: <https://huggingface.co/spaces/LLM360/TxT360>

Gray denotes the filtered data at each step of the pipeline.



Vocabulary

Vocabulary is typically determined by a subword tokenization algorithm over dataset.

[1] How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. arXiv:2012.15613

[2] StarCoder: may the source be with you! arXiv:2305.06161

Vocabulary

Vocabulary is typically determined by a subword tokenization algorithm over dataset.

A few important decisions:

- Are there special characters (such as control tokens in StarCoder [2])?
- If multilingual, take special care with tokenization [1], especially with the case of continuous pretraining over different languages.
- Vocabulary size: a hyperparameter to choose.

[1] How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. arXiv:2012.15613

[2] StarCoder: may the source be with you! arXiv:2305.06161

Vocabulary

Vocabulary is typically determined by a subword tokenization algorithm over dataset.

A few important decisions:

- Are there special characters (such as control tokens in StarCoder [2])?
- If multilingual, take special care with tokenization [1], especially with the case of continuous pretraining over different languages.
- Vocabulary size: a hyperparameter to choose.

A few metrics can be used to help decisions

- One good one: “Tokenizer fertility” → see next slide.

[1] How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. arXiv:2012.15613

[2] StarCoder: may the source be with you! arXiv:2305.06161

Vocabulary

Tokenizer Fertility: the average number of subwords produced per tokenized word.

- Fertility measures how aggressively a tokenizer splits.
- Low score is preferable as it indicates that the tokenizer is well suited to the target domain.

One should check the fertility score before reusing an existing tokenizer on another domain or language.

[1] How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. arXiv:2012.15613

[2] StarCoder: may the source be with you! arXiv:2305.06161

A (Brief) Pretrainer's Guide to Large Language Models

3



Model Architecture Choices

There are now quite a few popular architectures

- Transformer based: GPT series, Llama variants
- State Space Model: Mamba, Striped Hyena
- RNN-like models: RWKV

Model Architecture Choices

There are now quite a few popular architectures

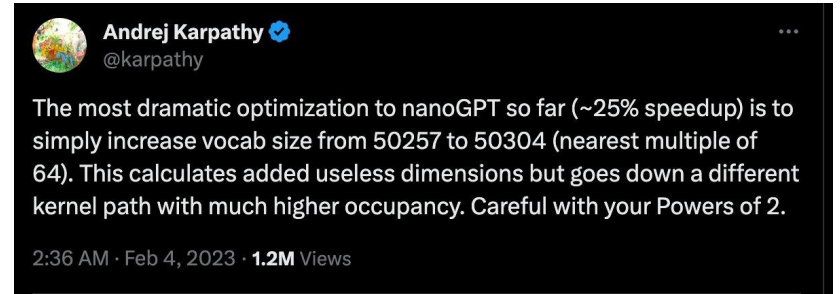
- Transformer based: GPT series, Llama variants
- State Space Model: Mamba, Striped Hyena
- RNN-like models: RWKV

In LLM360 experiments, transformer-based models still work consistently well.

- In our preliminary study, we find SSMs are hard to be trained for coding tasks
- There is also a large ecosystem around transformer-based models; good for community support.

Hardware-aware Decisions

The details of model architecture choice depend a lot on the underlying hardware.



Hardware-aware Decisions

The details of model architecture choice depend a lot on the underlying hardware.

- E.g., on Nvidia GPUs we typically want to control matrix dimensions to be multiples of 8 or 16 according to the official documentation [1].
- More parameter suggestions can be found in a study [2], key heuristic in the table



Parameter	Recommendation
Vocab size	divisible by 64
b	As large as possible
$b*s$, h/a , and h/t	divisible by a power of 2
$(b*a)/t$	should be an integer
t	As small as possible

a: # attention heads
s: seq length
t: tensor-parallel size
h: hidden dimension size
b: microbatch size

[1] <https://docs.nvidia.com/deeplearning/performance/dl-performance-matrix-multiplication/index.html#requirements-tc>

[2] The Case for Co-Designing Model Architectures with Hardware, arXiv:2401.14489

Model Architecture Efficiency

Bottlenecks in LLM models often happen with memory constraints – leads to various ways of improving memory usage.

- KV-Cache.
- Group Query Attention in place of regular Multi Query Attention.
- FlashAttention [1].

[1] FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness, Dao et.al. 2022

[2] Mamba: Linear-Time Sequence Modeling with Selective State Spaces, Dao et.al. 2023

Model Architecture Efficiency

Bottlenecks in LLM models often happen with memory constraints – leads to various ways of improving memory usage.

- KV-Cache.
- Group Query Attention in place of regular Multi Query Attention.
- FlashAttention [1].

When choosing architecture, often need to consider the performance on specific hardware.

- Mamba [2] on our hardware was found to be slower in terms of Wall Time.

[1] FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness, Dao et.al. 2022

[2] Mamba: Linear-Time Sequence Modeling with Selective State Spaces, Dao et.al. 2023

A (Brief) Pretrainer's Guide to Large Language Models

4



Hyperparameter Study

Hyperparameter Study

Need a way to select hyperparameters for our models!

- Learning rate, batch size, optimization algo params, architecture sizes, etc.

Hyperparameter Study

Need a way to select hyperparameters for our models!

- Learning rate, batch size, optimization algo params, architecture sizes, etc.

Ideally, want to conduct hyperparameter selection on smaller models.

- Often conducted during scaling law study.

Can scaling laws work here?

- How can optimal hyperparameters be “predicted” across model sizes?
- Some methods are available for this, particularly μP and $\mu Transfer$ [1].

μ Transfer

Algorithm 1 Tuning a Large Target Model via μ Transfer

- 1: Parametrize target model in Maximal Update Parametrization (μ P)
 - 2: Tune a smaller version (in width and/or depth) of target model
 - 3: Copy tuned hyperparameters to target model
-

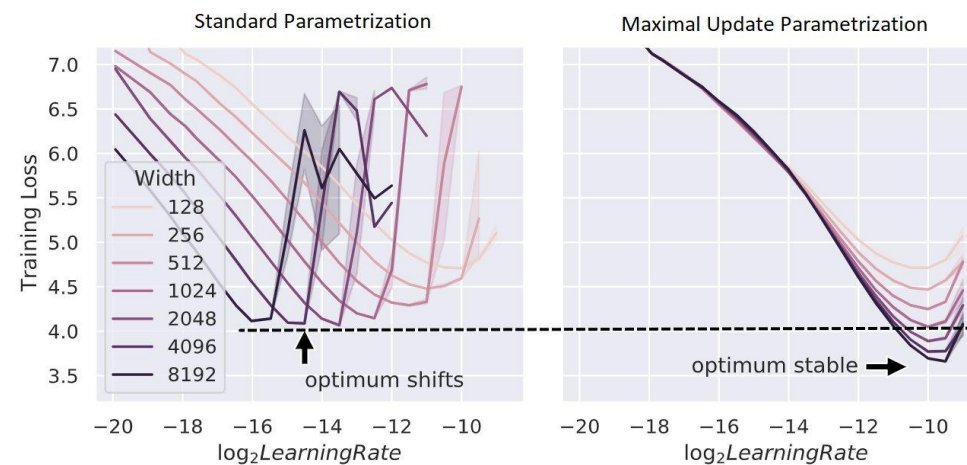
[1] Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer, arXiv:2203.03466

[2] Tensor Programs VI: Feature Learning in Infinite-Depth Neural Networks, arXiv:2310.02244

μ Transfer

Parameterizing model with μ P \Rightarrow optimal hyperparameters are same across scales.

- E.g., in right-hand, learning rate consistent across different model widths.



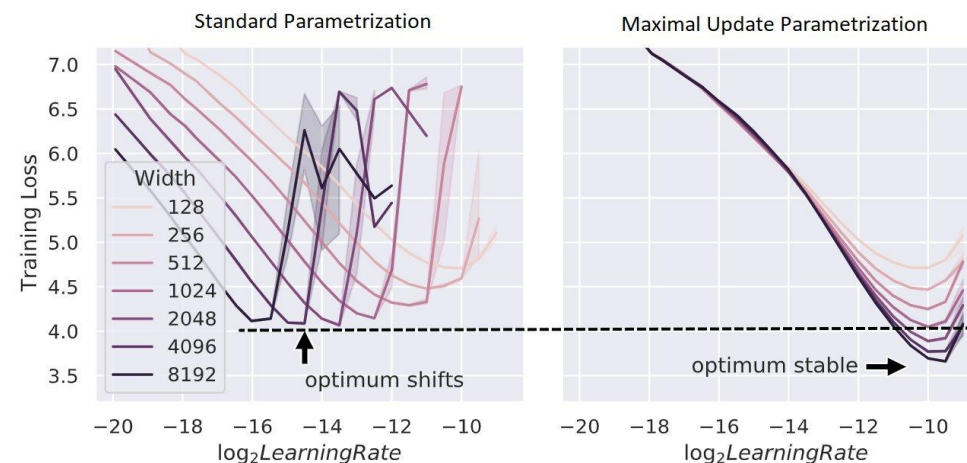
[1] Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer, arXiv:2203.03466

[2] Tensor Programs VI: Feature Learning in Infinite-Depth Neural Networks, arXiv:2310.02244

μ Transfer

Parameterizing model with μ P \Rightarrow optimal hyperparameters are same across scales.

- E.g., in right-hand, learning rate consistent across different model widths.



Zero-shot transfer hyperparameters via μ Transfer *

- Applies only to certain hyperparameters.

μ Transferable	Not μ Transferable	μ Transferred Across
optimization related, init, parameter multipliers, etc	regularization (dropout, weight decay, etc)	width, depth*, batch size*, training time*, seq length*

[1] proves the mechanism for “widthwise hyperparameter transfer”, though people use this method beyond the theoretical guarantee (applied on depth, different data sizes).

- Recently in [2], “depthwise transfer” is proposed.

[1] Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer, arXiv:2203.03466

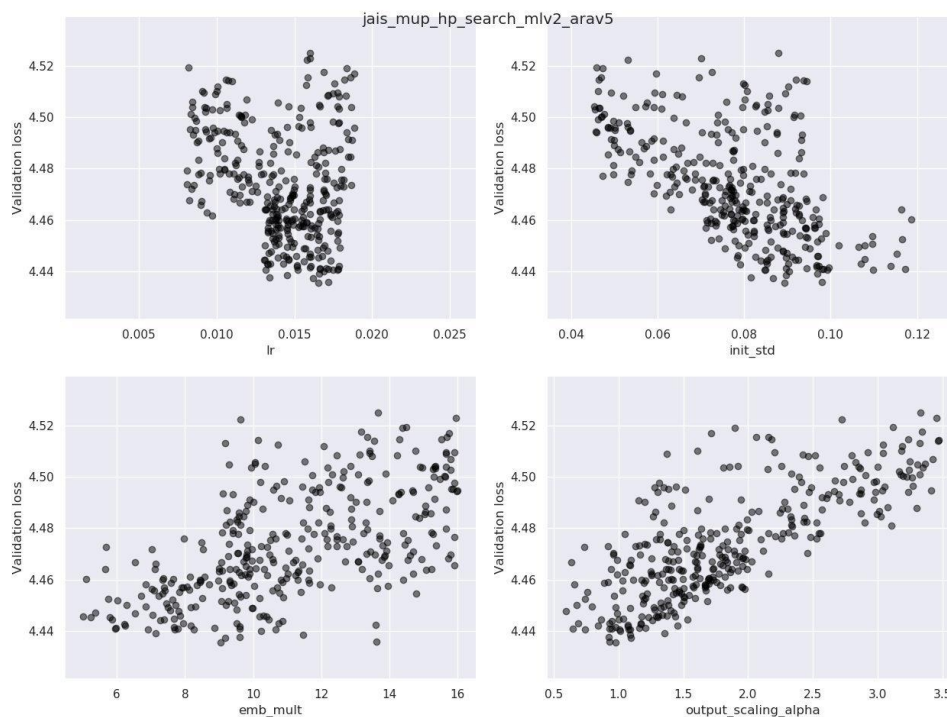
[2] Tensor Programs VI: Feature Learning in Infinite-Depth Neural Networks, arXiv:2310.02244

Hyperparameter study – tuning on a smaller model

We do this in practice: first quickly tune hyperparameters on a small (*proxy*) model.

Hyperparameter study – tuning on a smaller model

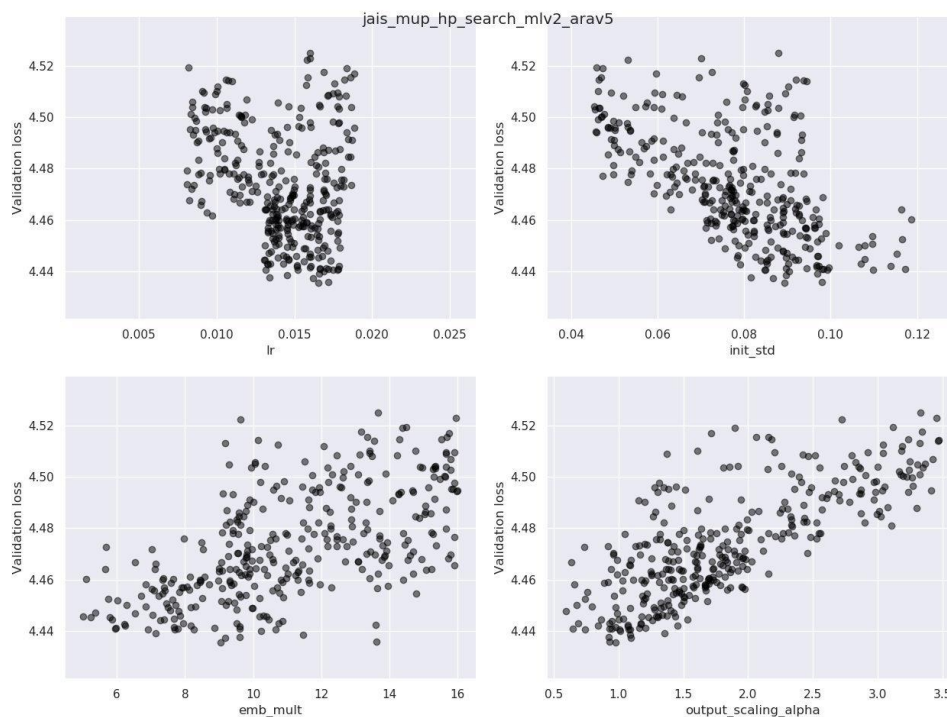
We do this in practice: first quickly tune hyperparameters on a small (*proxy*) model.



A hyperparameter search experiment on one of our models

Hyperparameter study – tuning on a smaller model

We do this in practice: first quickly tune hyperparameters on a small (*proxy*) model.



A red curved arrow points from the text below to the top-left plot. The text is: "A hyperparameter search experiment on one of our models".

Then use μ Transfer for zero-shot transfer from the proxy model to large model (small \rightarrow large)

...which works very well with scaling law study

A (Brief) Pretrainer's Guide to Large Language Models



Determine Training Curriculum

Model training is a very long and costly process...

- Before starting, we need to plan for a few key decisions involving the training data mixes.
- *I.e.*, determine what happens during training, or what's the “training curriculum” *.

* Not to be confused with curriculum learning, though some ideas are very similar

Determine Training Curriculum

Model training is a very long and costly process...

- Before starting, we need to plan for a few key decisions involving the training data mixes.
- *I.e.*, determine what happens during training, or what's the “training curriculum” *.

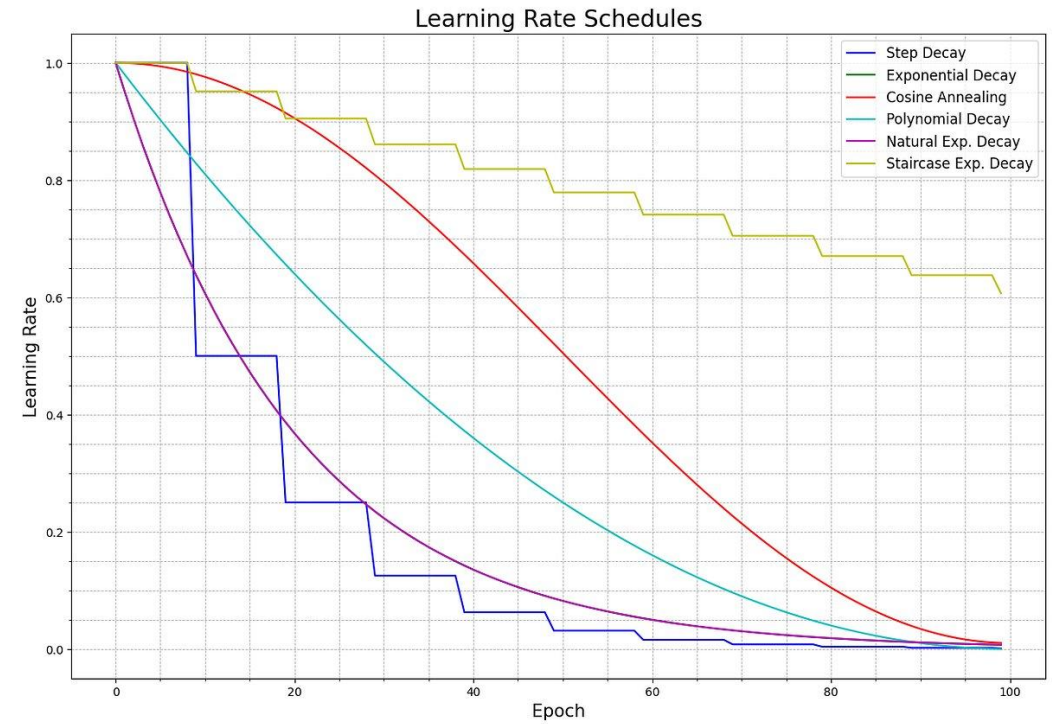
What should we consider in a training curriculum? For example:

- During early stage of training, we want the model to “warm-up” well to a new data distribution.
- Based on project needs, at the middle stages sometimes we have special training curriculums, leading to *multi-stage training*.
- At the end of training, we want to make sure the model converges well – *model ramp down*.

* Not to be confused with curriculum learning, though some ideas are very similar

Learning Rate Schedule

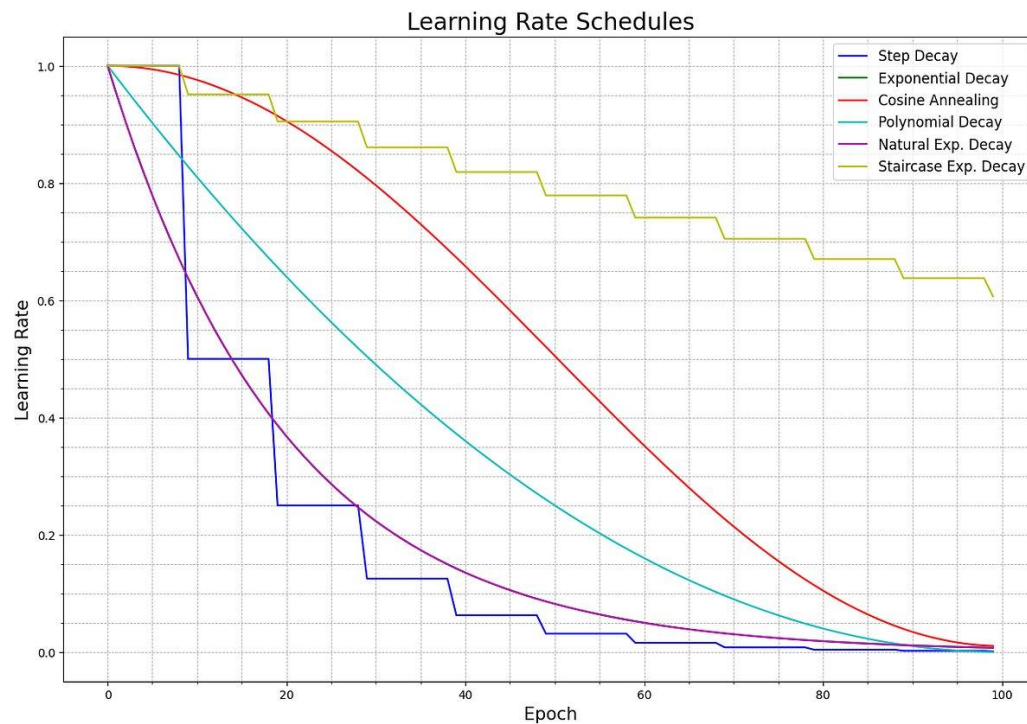
The *learning rate schedule* is strongly dictated by the training curriculum.



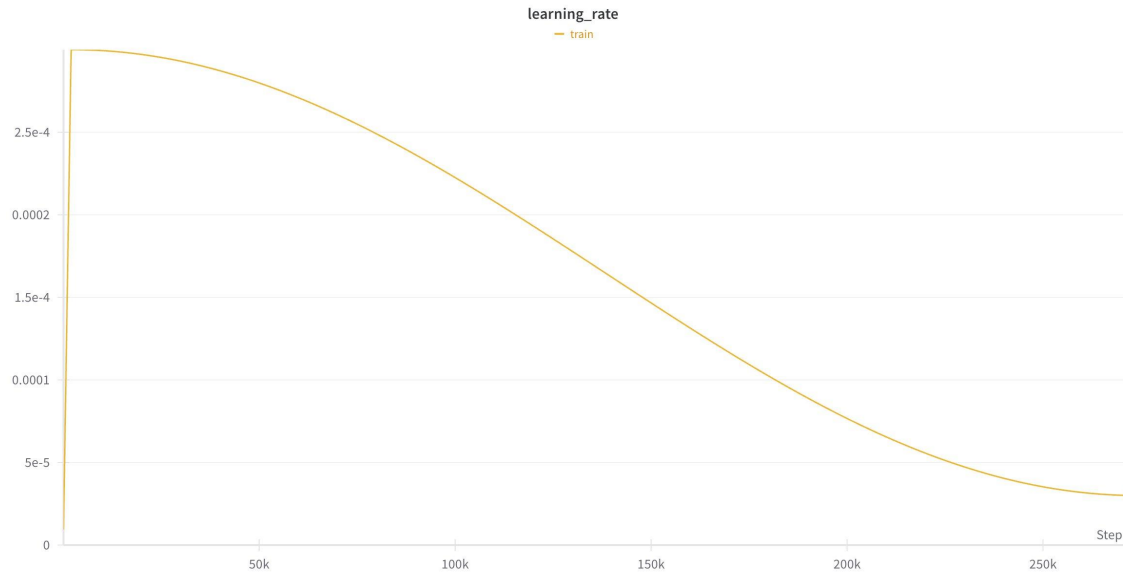
Learning Rate Schedule

The *learning rate schedule* is strongly dictated by the training curriculum.

- Need to plan the learning rate schedule in response to the curriculum details.
- Since the model will be trained for months, too large or small learning rate could cause instability or major performance slow down.

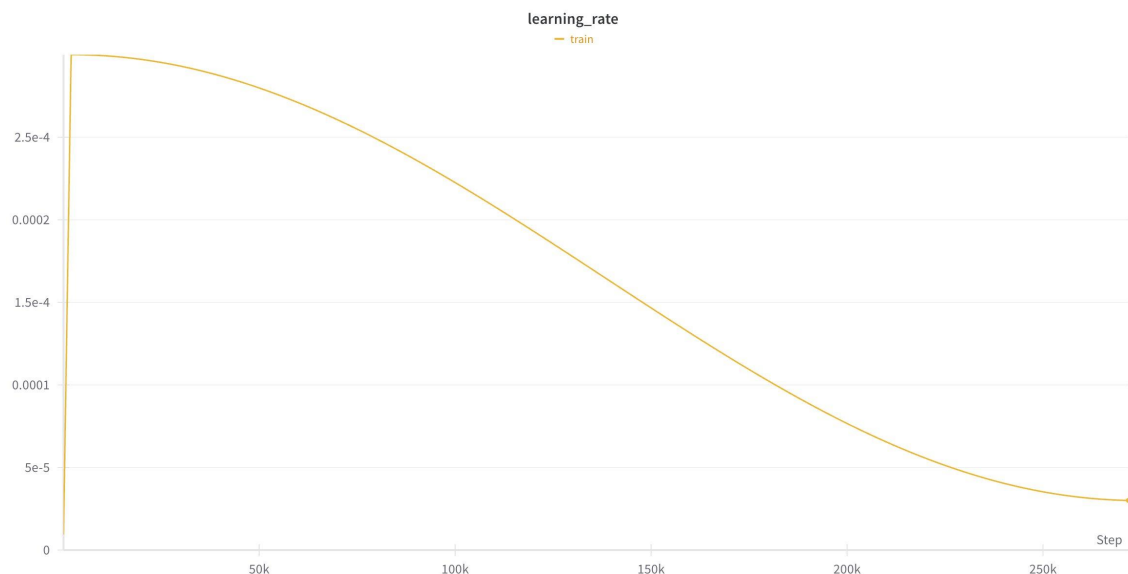


Learning Rate Schedule



LLM360/Amber uses cosine decay with an initial LR of 3×10^{-4} and a final rate of 3×10^{-5} . The learning rate is warmed up for 2,000 steps.

Learning Rate Schedule



LLM360/Amber uses cosine decay with an initial LR of $3e-4$ and a final rate of $3e-5$. The learning rate is warmed up for 2,000 steps.

	Phase 1	Phase 2	Phase 3
LR Warmup Steps	86	86	276
LR Initial Value	0.012	0.0087825	0.002
LR Final Value	0.00012408	0.00013679	0.0002
LR Decay	Linear	Linear	Linear

LLM360/Crystal's multi-phase schedule. (Note that only the base LR is shown).

Note that Amber uses standard parameterization while Crystal uses μP , hence Amber uses a shared learning rate while Crystal uses a per-layer learning rate.

Ramp-off and Warmups

Concretely, there are three factors that affect the learning rate schedule:

1. When your data distribution shifts over multiple stages.
2. When you want to get a final checkpoint for use (either at intermediate stage or at end) .
3. When you want to do continual training (*i.e.*, restart training on larger set of data).

Ramp-off and Warmups

Concretely, there are three factors that affect the learning rate schedule:

1. When your data distribution shifts over multiple stages.
2. When you want to get a final checkpoint for use (either at intermediate stage or at end) .
3. When you want to do continual training (*i.e.*, restart training on larger set of data).

Here are two possible strategies:

1. Prepare a learning rate schedule spanning all the stages, and ramp-off models at each stage.
 - Pro: Overall training is smoother, no risk of having an incorrect warmup.
 - Con: Hard to plan the whole learning rate schedule without knowing the full data size in advance.

Ramp-off and Warmups

Concretely, there are three factors that affect the learning rate schedule:

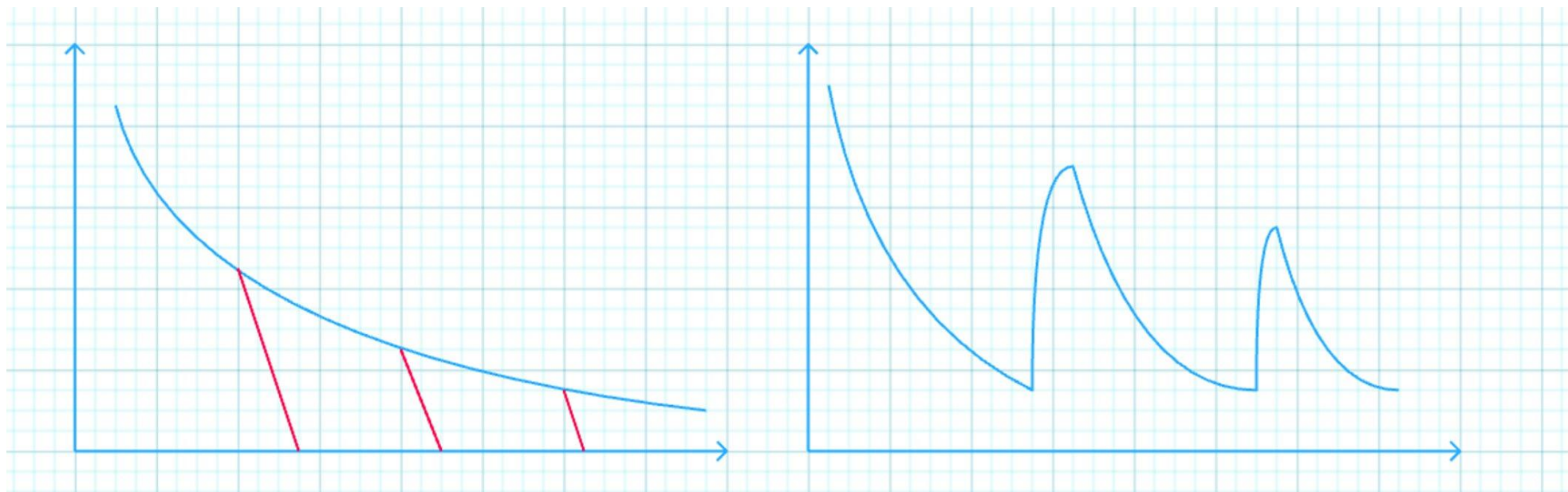
1. When your data distribution shifts over multiple stages.
2. When you want to get a final checkpoint for use (either at intermediate stage or at end) .
3. When you want to do continual training (*i.e.*, restart training on larger set of data).

Here are two possible strategies:

1. Prepare a learning rate schedule spanning all the stages, and ramp-off models at each stage.
 - Pro: Overall training is smoother, no risk of having an incorrect warmup.
 - Con: Hard to plan the whole learning rate schedule without knowing the full data size in advance.
2. Have a learning rate schedule for each stage, and re-warmup at each stage.
 - Pro: Easy to plan and implement
 - Con: Warm-up and hyperparameter choices can be tricky

Ramp-off and Warmups

Visualizing both strategies:



Ramp-off approach, used in some LLM360 models tests. Prior work such as [1] suggests linearly decaying to zero.

Multi-stage warm-up approach, as in LLM360/Crystal and K2. Prior work such as [2] empirically confirms that warmup is necessary in different settings

[1] <https://github.com/Stability-AI/StableLM/?tab=readme-ov-file#stablelm-3b-4e1t>

[2] Continual Pre-Training of Large Language Models: How to (re)warm your model? arXiv:2308.04014

Data Mix

How do we determine our data weighting?

Data Mix

How do we determine our data weighting?

- Methods such as DoReMi [1] aim to determine the best data weights.
 - Difficult to find scaling law: we empirically find DoReMi predictions are different when using different proxy sizes.
- A more common approach is to find data weighting empirically by performing sweeps

Data Mix

How do we determine our data weighting?

- Methods such as DoReMi [1] aim to determine the best data weights.
 - Difficult to find scaling law: we empirically find DoReMi predictions are different when using different proxy sizes.
- A more common approach is to find data weighting empirically by performing sweeps

Can we find a scaling law for multi-stage data schedules?

- Note that most scaling law studies assume the dataset is sampled uniformly from the same distribution across training.
- Hence it is possible to estimate data mix for a large model using a small proxy model.
- It is uncertain whether we can estimate *data schedules* (for multi-stage training) with a proxy.
 - Intuitively: larger model may learn a certain “ability” with less data, allowing them want “next step” earlier.
 - In LLM360, we attempt to estimate this phenomenon but don’t have enough compute to draw conclusions.

Data Readiness

In reality, not everything is ready at the start of training.

Data Readiness

In reality, not everything is ready at the start of training.

LLM training usually spans over weeks if not months

- New requirement, new data may be available during training

Data Readiness

In reality, not everything is ready at the start of training.

LLM training usually spans over weeks if not months

- New requirement, new data may be available during training

Practically, we suggest model ramp-off and warm-up strategies as described earlier, when things change.

A (Brief) Pretrainer's Guide to Large Language Models



Preparation of runtime before pretraining

This section is about determining:

- Pretraining implementation frameworks.
- Parallel/distributed training strategies.
- Fault tolerance and handling hardware (or other) failures.
- Various problems we encountered in practice.

Pretraining Framework

There are various requirements we need from the training framework:

- Code stability and implementation correctness.
- Parallelization support.

Pretraining Framework

There are various requirements we need from the training framework:




- Code stability and implementation correctness.
- Parallelization support.

See code:
<https://github.com/LLM360/k2-train>

Need to choose a framework that fits your needs!

- We found many existing open source frameworks were not perfect for our needs:

Our customized and integrated solution

	Megatron-LM		Alpa 	Megatron-LM + 
Dataset & model supports	✗	✓	✓	✓
All parallelism support	✓	✗	✓	✓
FlashAttention-2	✓	✓	✗	✓

Background on Parallel Training

Two cases to consider – large data vs large model:

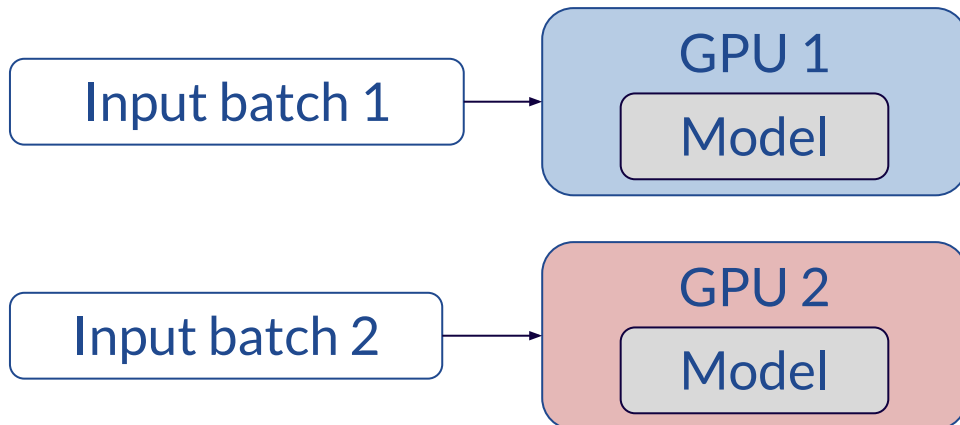
Background on Parallel Training

Two cases to consider – large data vs large model:

1. The input dataset is very large.

😊 Easy.

Data parallelism: partition input data and replicate the model



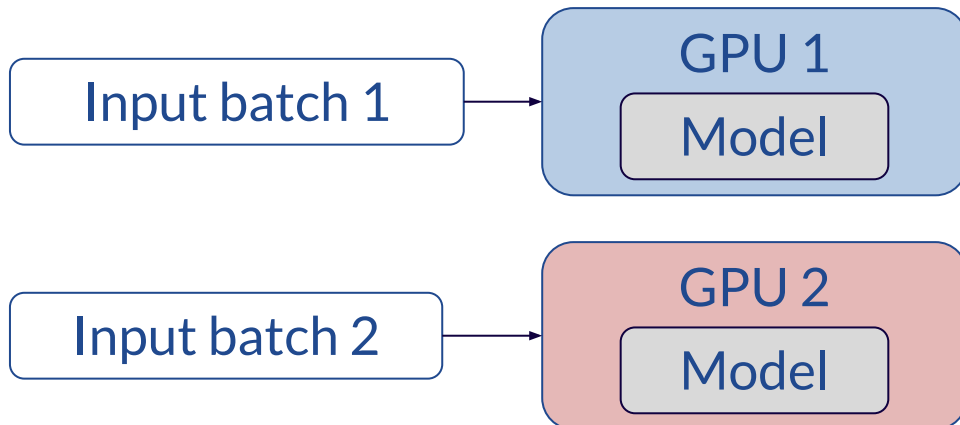
Background on Parallel Training

Two cases to consider – large data vs large model:

1. The input dataset is very large.

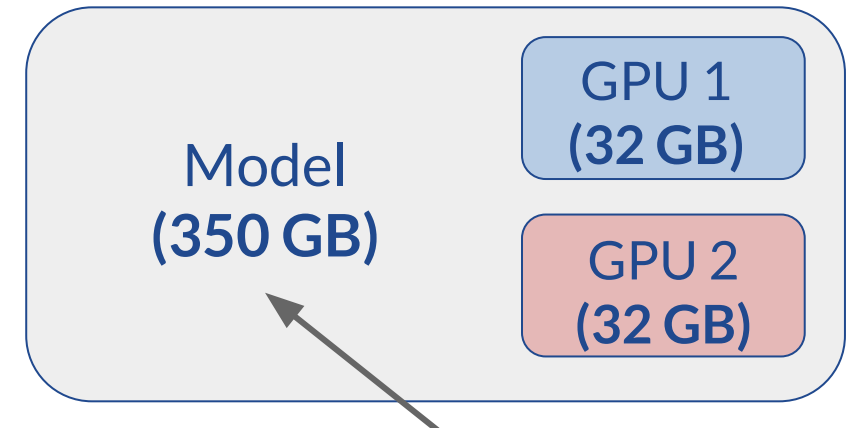
😊 Easy.

Data parallelism: partition input data and replicate the model



2. The model is very large.

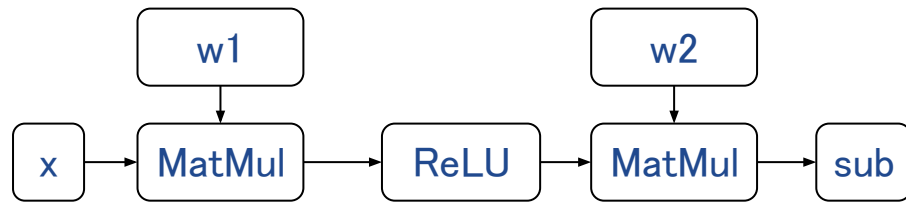
😞 Harder !!



Challenge: How to partition a computational graph?

Background on Parallel Training

Different strategies for model parallelism:

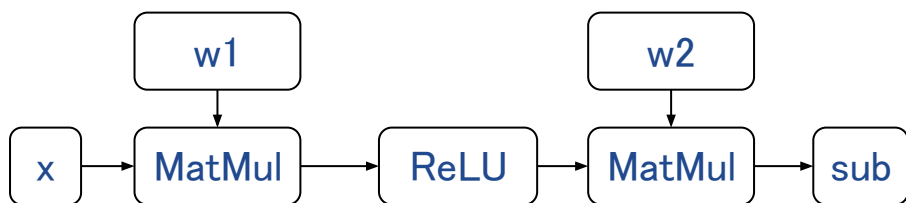


Consider this
compute graph.

Background on Parallel Training

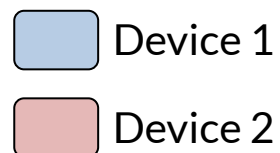
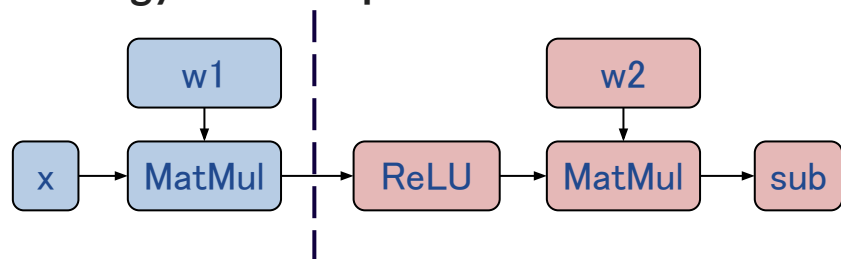
[1] *Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning*, 2022, arXiv: 2201.12023

Different strategies for model parallelism:

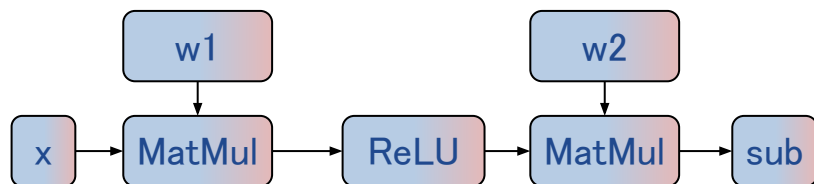


Consider this compute graph.

Strategy 1: Inter-operator Parallelism



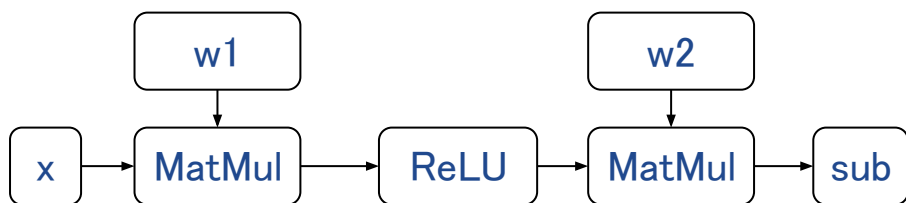
Strategy 2: Intra-operator Parallelism



Background on Parallel Training

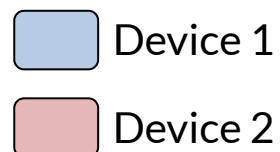
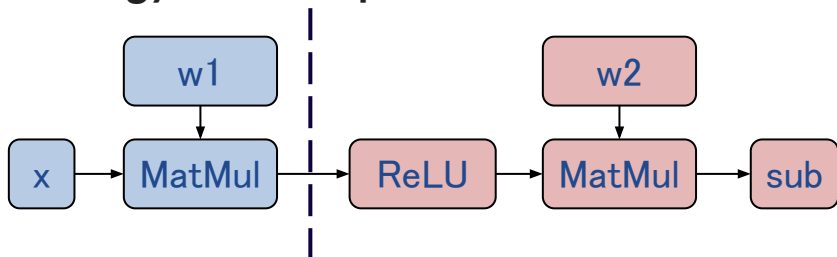
[1] *Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning*, 2022, arXiv: 2201.12023

Different strategies for model parallelism:

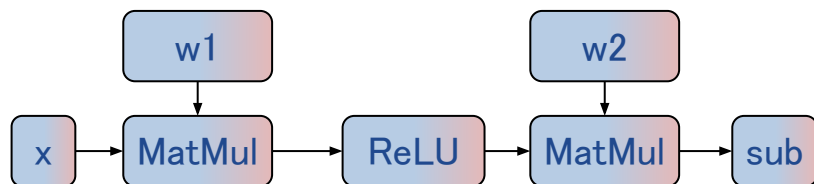


Consider this compute graph.

Strategy 1: Inter-operator Parallelism



Strategy 2: Intra-operator Parallelism



Trade-off

	Inter-operator Parallelism	Intra-operator Parallelism
Communication	Less	More
Device Idle Time	More	Less

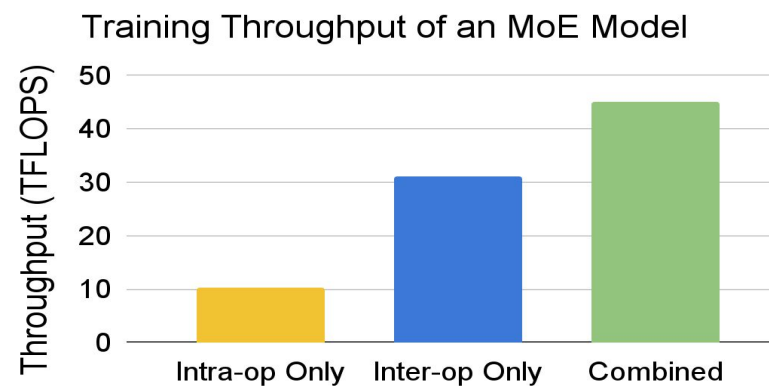
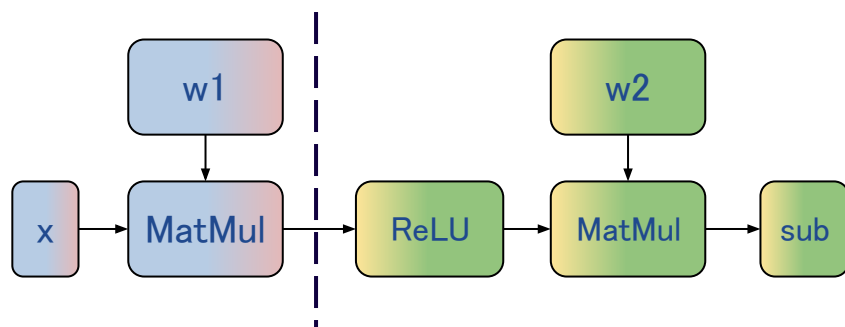
Background on Parallel Training

A couple of additional variations:

Background on Parallel Training

A couple of additional variations:

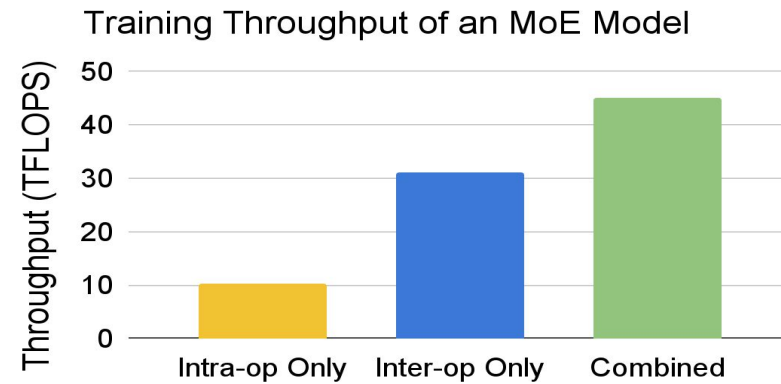
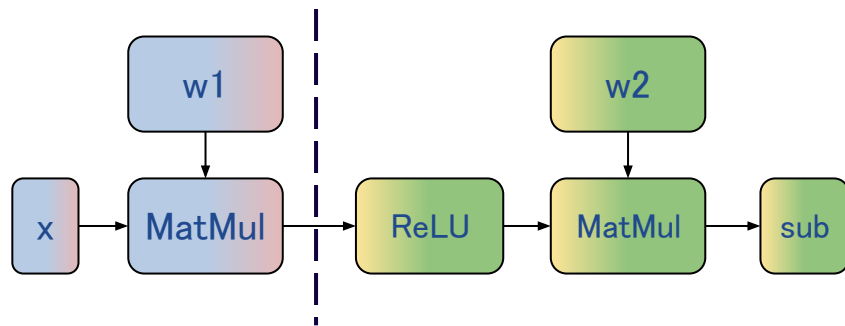
Combine Intra-op and Inter-op



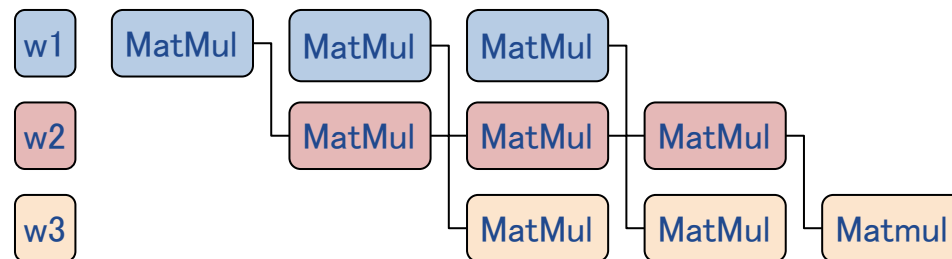
Background on Parallel Training

A couple of additional variations:

Combine Intra-op and Inter-op



Pipeline the execution for inter-op parallelism



Background on Parallel Training

Can automatically choose distributed strategy based on network topology!

Background on Parallel Training

Can automatically choose distributed strategy based on network topology!

Checkout more on this topic in MLSys work, such as Alpa [1] and RedCoast [2].

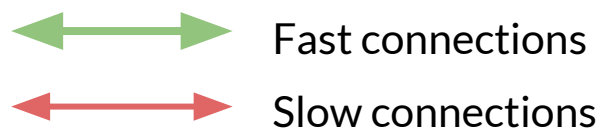
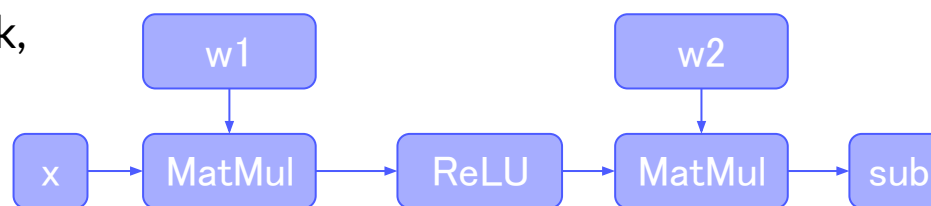
[1] Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning, arXiv:2201.12023

[2] RedCoast: A Lightweight Tool to Automate Distributed Training of LLMs on Any GPU/TPUs, arXiv:2310.16355

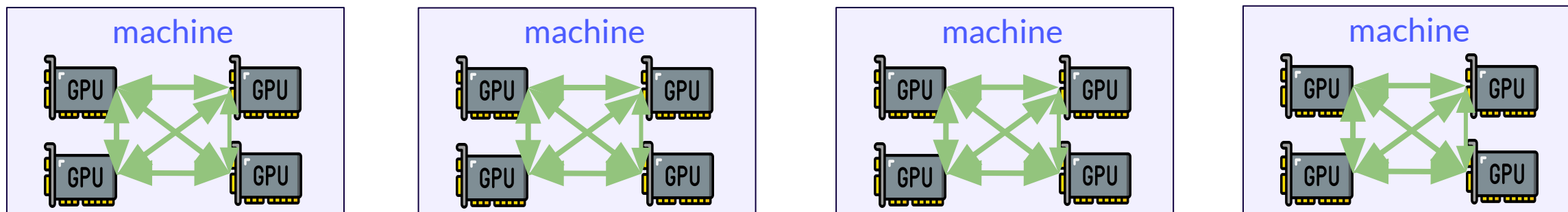
Background on Parallel Training

Can automatically choose distributed strategy based on network topology!

Checkout more on this topic in MLSys work, such as Alpa [1] and RedCoast [2].



Determine Method to Assign Compute Graph to According Network Topology

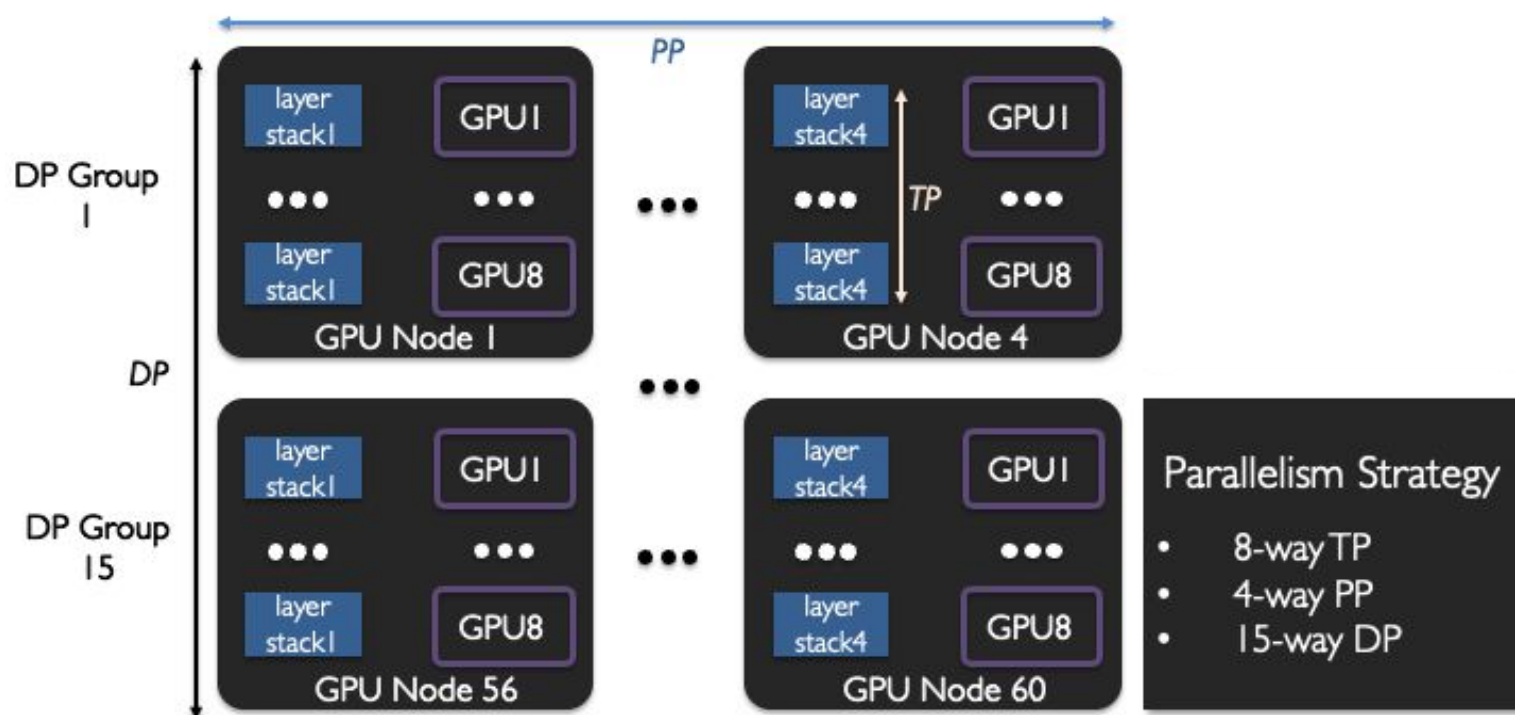


[1] Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning, arXiv:2201.12023

[2] RedCoast: A Lightweight Tool to Automate Distributed Training of LLMs on Any GPU/TPUs, arXiv:2310.16355

Background on Parallel Training

Altogether, here is the training parallelism strategy of our K2 model [\(on 60 nodes\)](#)



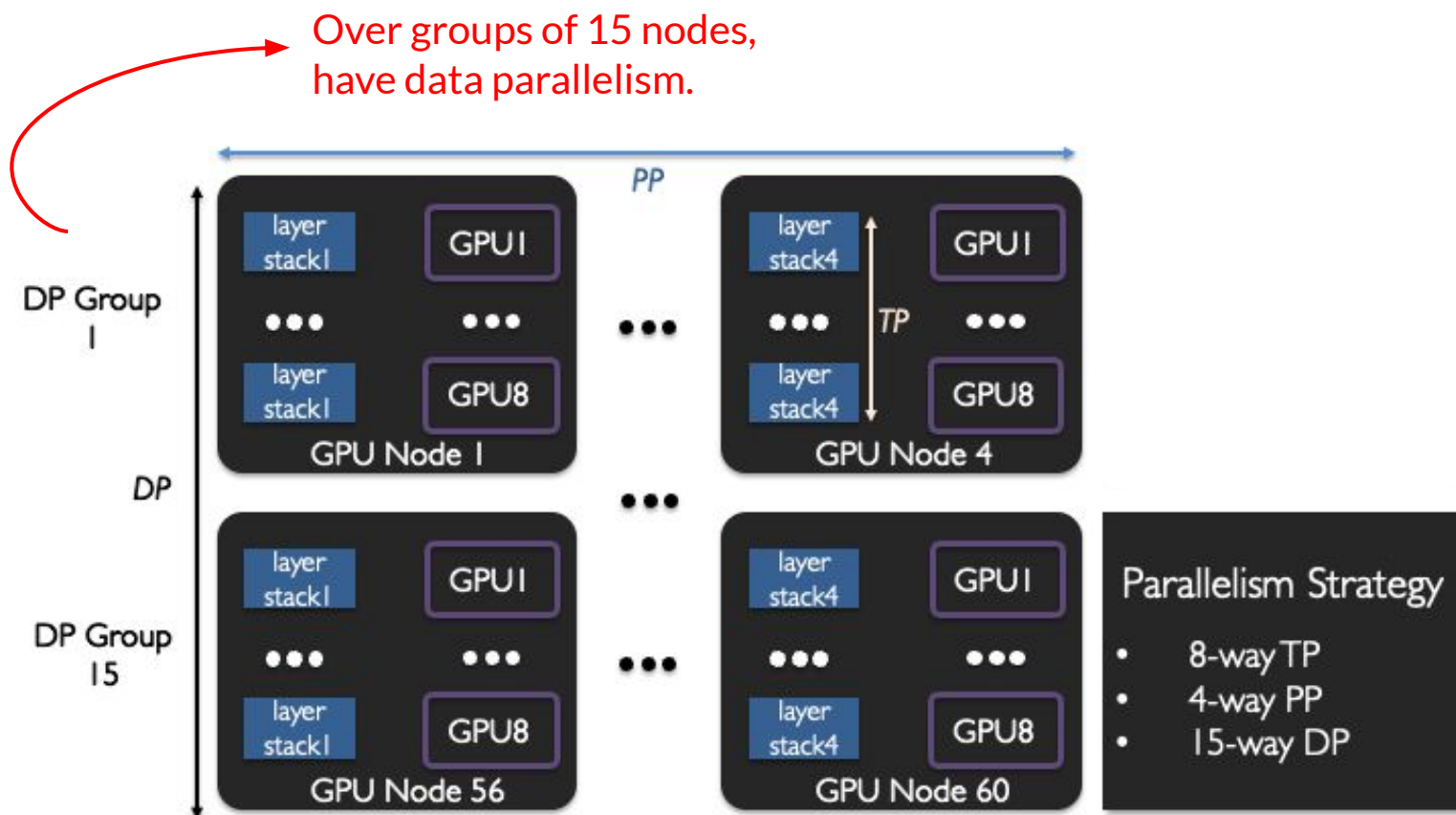
DP: Data Parallelism

TP: (Tensor) Model Parallelism

PP: Pipeline Parallelism

Background on Parallel Training

Altogether, here is the training parallelism strategy of our K2 model [\(on 60 nodes\)](#)



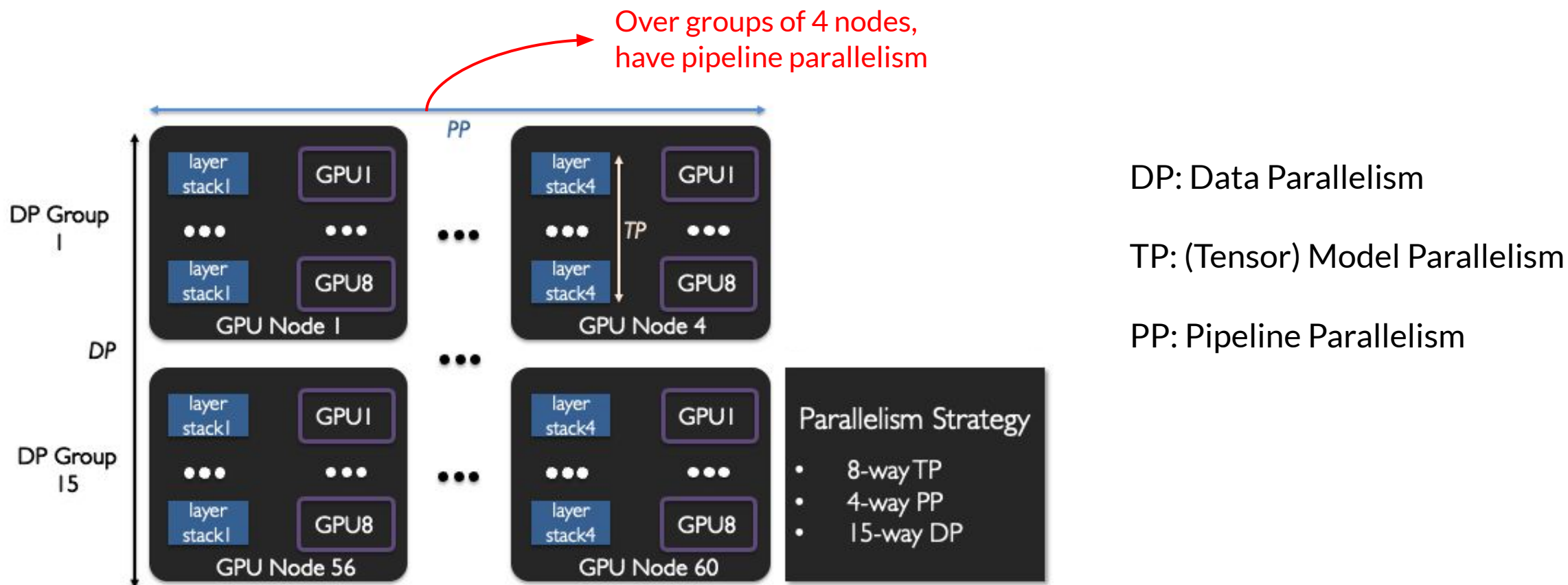
DP: Data Parallelism

TP: (Tensor) Model Parallelism

PP: Pipeline Parallelism

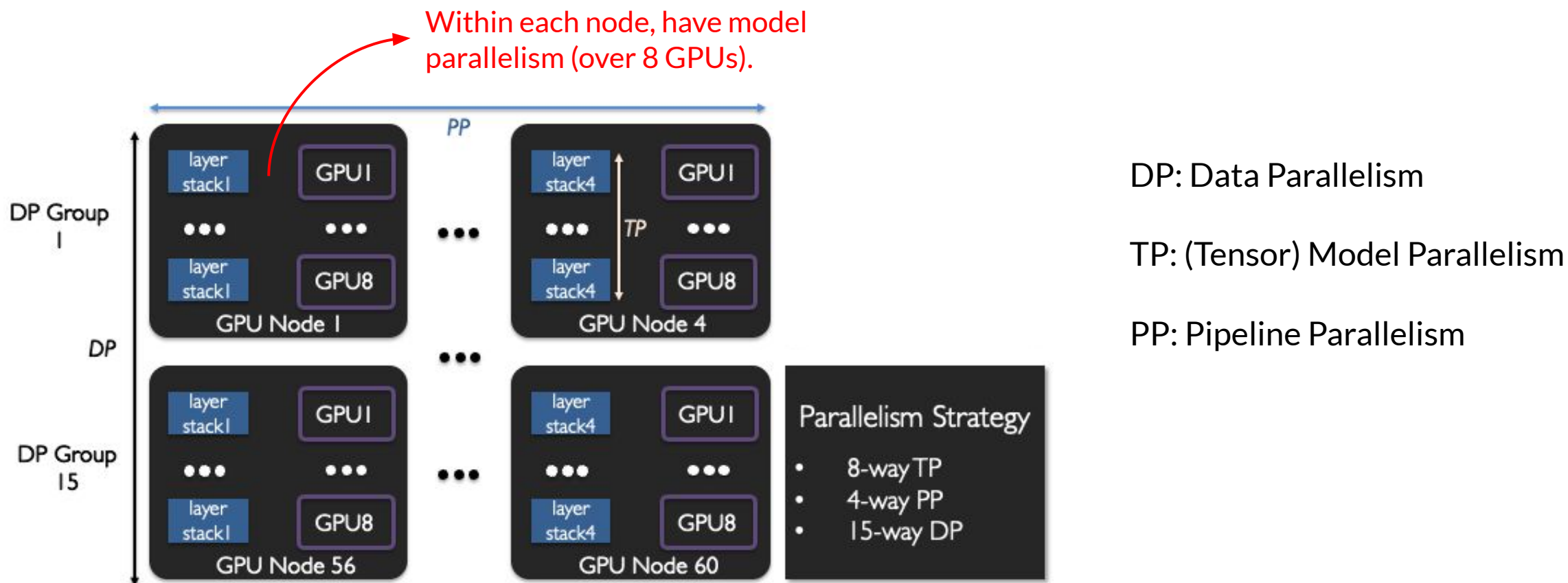
Background on Parallel Training

Altogether, here is the training parallelism strategy of our K2 model [\(on 60 nodes\)](#)



Background on Parallel Training

Altogether, here is the training parallelism strategy of our K2 model [\(on 60 nodes\)](#)



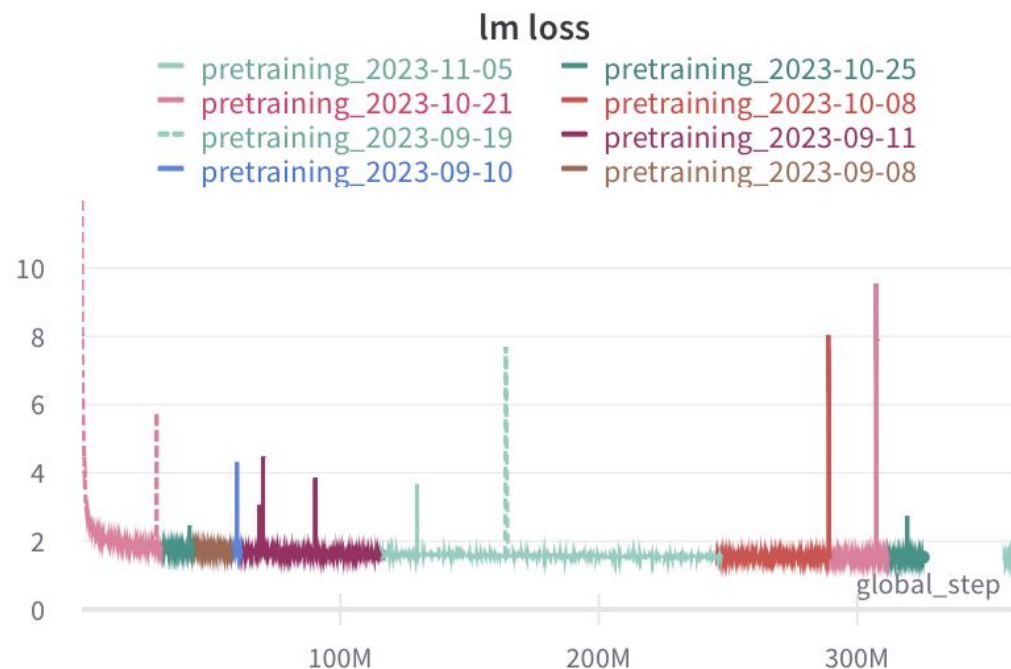
Hardware Failures and Fault Tolerance

LLM pre-training still can have many issues:

Hardware Failures and Fault Tolerance

LLM pre-training still can have many issues:

- Hardware failure, e.g., CUDA NCCL error.
- Unknown hardware/network slowing down.
- Loss spikes during training.
- NaN loss and training divergence.



Types of Hardware Failure

Many possible types of hardware failures, most of which we encountered

Types of Hardware Failure

Many possible types of hardware failures, most of which we encountered

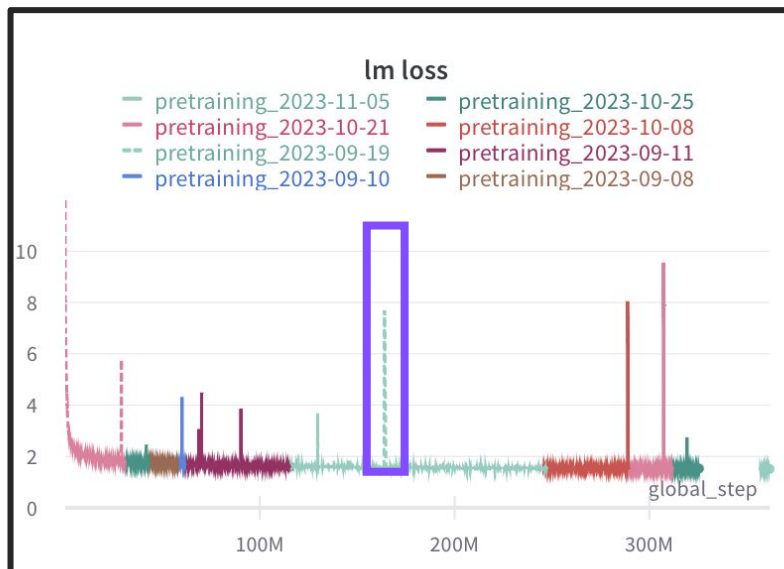
Type of failure	Description
NCCL timeout	Timeout duration has been exceeded by init container.
Dead GPU	A GPU is down.
Unhealthy GPU nodes	GPU node-level hardware failure.
OS input/output error	Operating systems issue/failure.
Lustre file system error	GPU nodes reboot due to lustre error.
Mount failure	GPU nodes mount failures because the user job is stuck in pending state.
Lack of storage	Running out of disk space on the cluster.

Fault Tolerance

How did we handle these failures in practice?

Fault Tolerance

How did we handle these failures in practice?

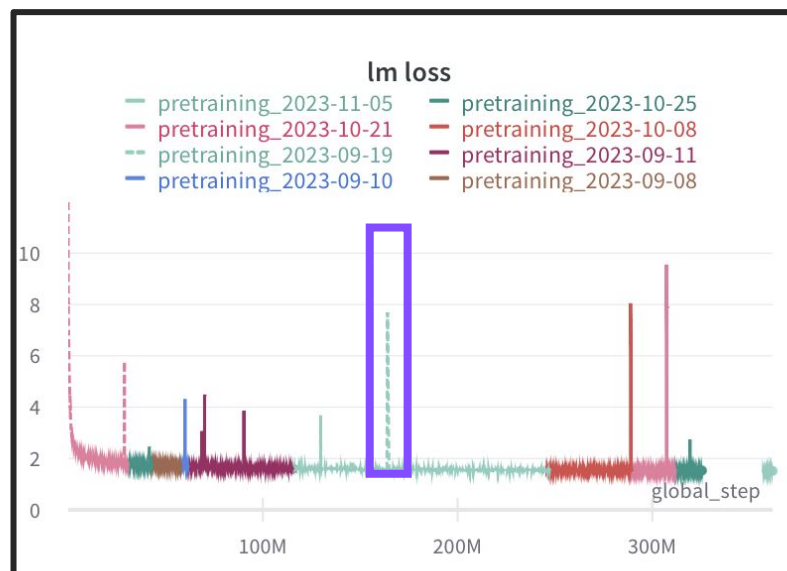


e.g., × 4

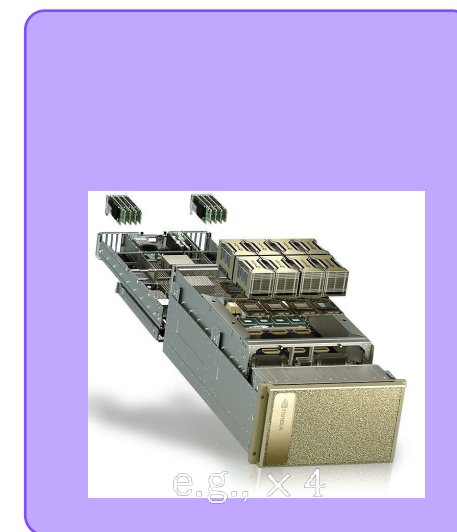
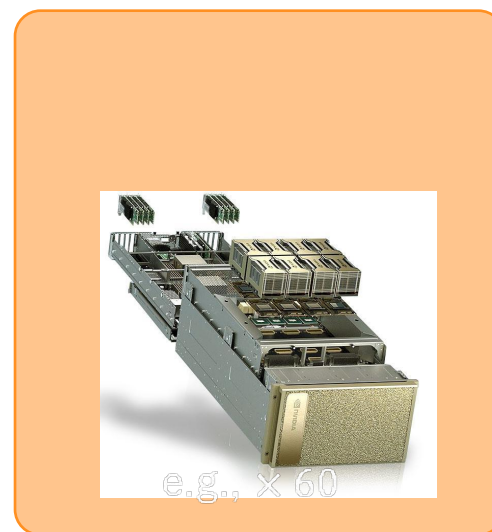
Skip the current data batch when loss spike or NaN loss are observed.

Fault Tolerance

How did we handle these failures in practice?



Skip the current data batch when loss spike or NaN loss are observed.



Replacing "failed" node with a new one from the backup GPU pool.

Still, problems were encountered!

During LLM360 training, we still encountered a few problems during runtime:

Still, problems were encountered!

During LLM360 training, we still encountered a few problems during runtime:

- Configuration bug in Lit-llama repository
 - Models get stored at incorrect precision in certain environments.
- Incorrect precision details in Megatron-LLM repository
- Precision changes midway due to Cerebras hardware upgrade.

Still, problems were encountered!

During LLM360 training, we still encountered a few problems during runtime:

- Configuration bug in Lit-llama repository
 - Models get stored at incorrect precision in certain environments.
- Incorrect precision details in Megatron-LLM repository
- Precision changes midway due to Cerebras hardware upgrade.

⇒ All issues detailed in our reports! [1, 2]

[1] LLM360: Towards Fully Transparent Open-Source LLMs, arxiv.org/abs/2312.06550

[2] LLM360 K2-65B: Scaling Up Fully Transparent Open-Source LLMs, www.llm360.ai/blog/several-new-releases-to-further-our-mission.html

A (Brief) Pretrainer's Guide to Large Language Models



Evaluation and Logging

Remember to plan enough resources for evaluation

- One mistake during K2 training was that we didn't have enough machines to eval the 65B model!
- Frequent evaluation will ensure we don't waste time on an ill-behaved model for too long.
 - This is again a tradeoff: evaluation resources vs. training resource

Evaluation and Logging

Remember to plan enough resources for evaluation

- One mistake during K2 training was that we didn't have enough machines to eval the 65B model!
- Frequent evaluation will ensure we don't waste time on an ill-behaved model for too long.
 - This is again a tradeoff: evaluation resources vs. training resource

Which evaluation measures to run during training?

- Held-out perplexity: one of the most direct measures.
 - Practically, training loss is reasonably correlated / works reasonably well (if the data is dedup).
 - Held-out set is still important to ensure no accidental data repetition.
- Benchmark: task-based benchmarks to directly measure desired metrics (but make sure you don't have data leaks!)
- Test generation — you should frequently sample output from the model.

Benchmarks – Multiple Choice vs. Generation

Two examples “*types*” of popular benchmarks: multiple choice and generation-based.

Benchmarks – Multiple Choice vs. Generation

Two examples “types” of popular benchmarks: multiple choice and generation-based.

Multiple Choice

- Example: MMLU, ARC (AI2 Reasoning Challenge)
- Easy to implement and evaluate
- Some metrics can be misleading and cannot detect degenerated models

Generation

- Example: BigBench, GSM8K, MBPP
- Often involve CoT, or complex generation
- More reliable benchmarks for generation tasks
- Difficult and high variance for small-scale models

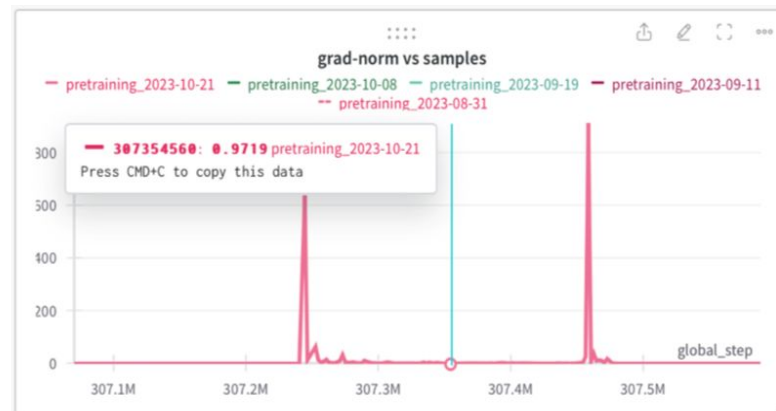
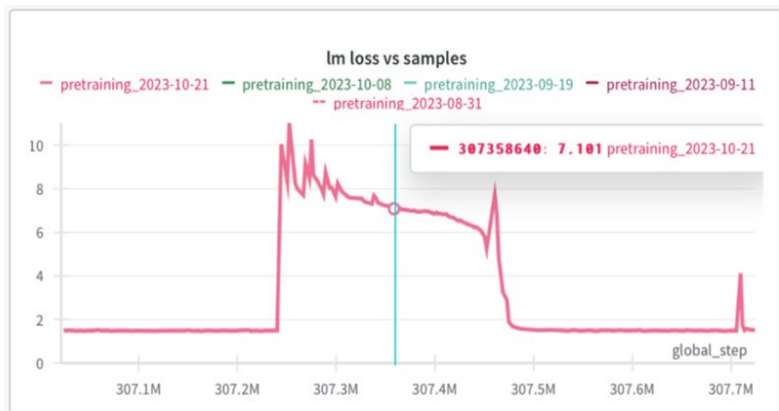
Dealing with Problems that Arise

Example: Loss Spikes

Dealing with Problems that Arise

Example: Loss Spikes

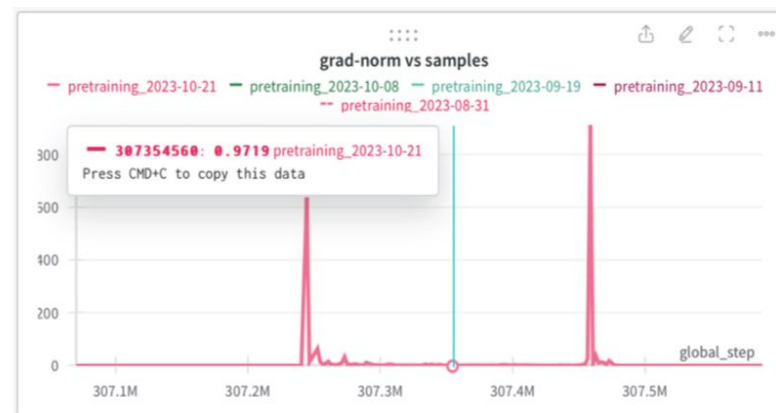
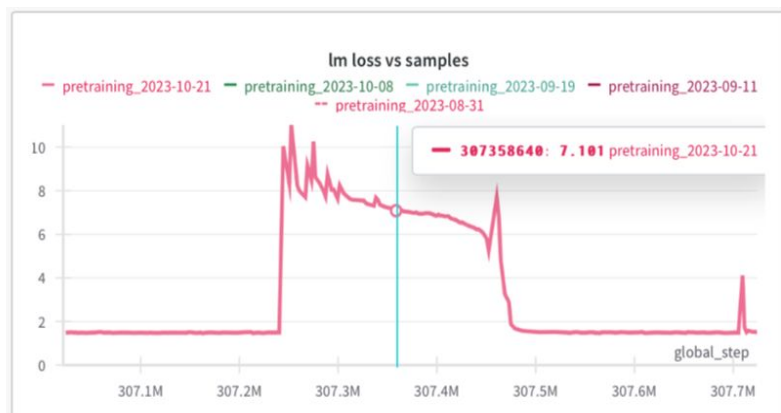
- Loss spikes are common problems during large scale pretraining
 - Empirically, we only observe spikes during our 65B model training but not in the 7B training



Dealing with Problems that Arise

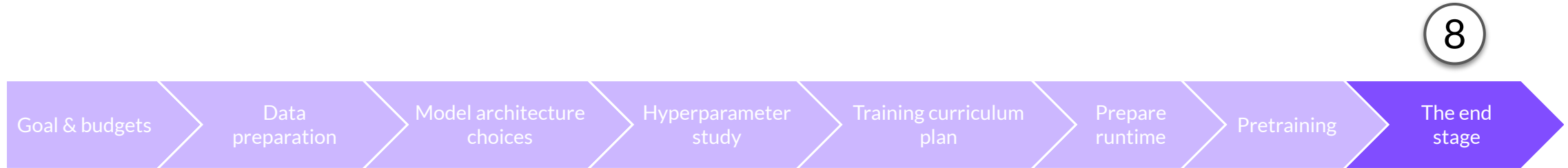
Example: Loss Spikes

- Loss spikes are common problems during large scale pretraining
 - Empirically, we only observe spikes during our 65B model training but not in the 7B training

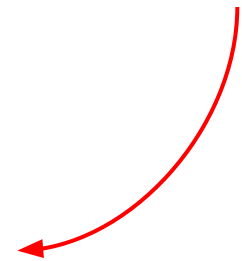


- There are many solutions to loss spikes
 - Simple ones: can simply skip a data instance and restart the training.
 - Some other methods are also reported to alleviate spikes, such as model averaging [1], alternative losses.

A (Brief) Pretrainer's Guide to Large Language Models

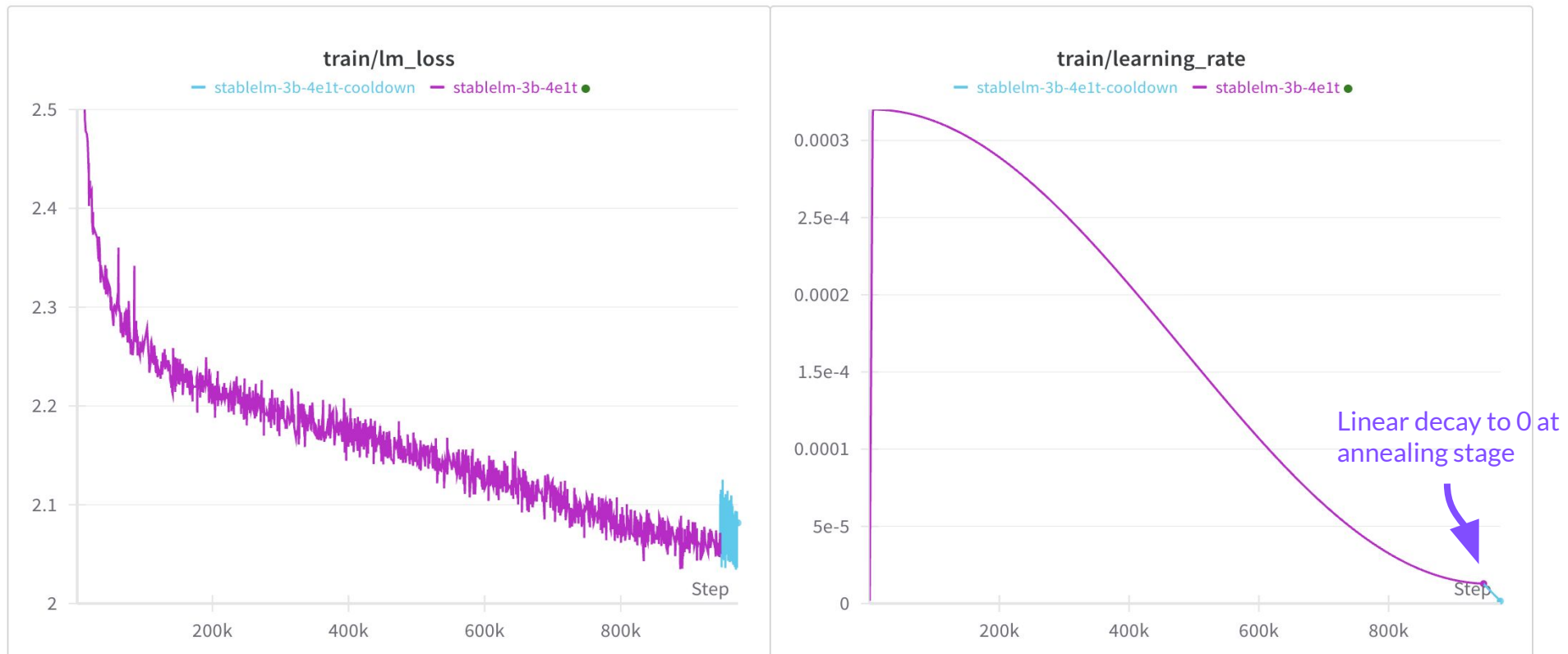


After pretraining,
there's still some
work to be done



Annealing Stage

At end of pretraining, anneal learning rate down to zero.



Model Averaging

Averaging the model weights of subsequent training steps sometimes can improve performance [1].

- In some training tests, we averaged model weights over last 10 epochs and obtained a better model.
- Though we did not find this effect with Amber and Crystal (7B) models.

[1] Averaging Weights Leads to Wider Optima and Better Generalization arXiv:1803.0540

[2] Early Weight Averaging meets High Learning Rates for LLM Pre-training arXiv:2306.03241

Model Averaging

Averaging the model weights of subsequent training steps sometimes can improve performance [1].

- In some training tests, we averaged model weights over last 10 epochs and obtained a better model.
- Though we did not find this effect with Amber and Crystal (7B) models.

More recent work [2] shows that

- High learning rate (LR) can better performance when performing model averaging.
- The training trajectory of averaging with high LR mimics a low LR scenario.
 - \Rightarrow if we use model averaging, we may be able to keep high LR for longer time (less LR decay).

[1] Averaging Weights Leads to Wider Optima and Better Generalization arXiv:1803.0540

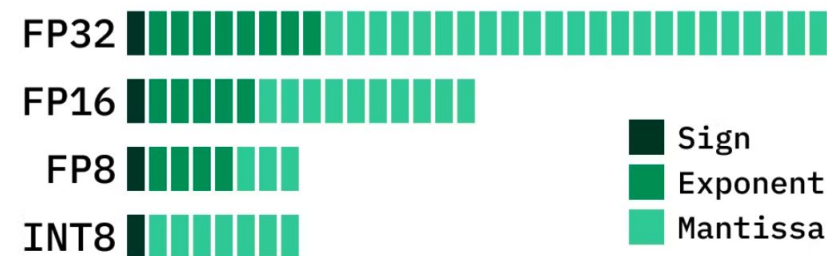
[2] Early Weight Averaging meets High Learning Rates for LLM Pre-training arXiv:2306.03241

Quantization

Quantization converts the final model into much lower precision (e.g., BF16 \rightarrow Int8).

- Some work [1] shows that some parts of a model may be less suitable for quantization.

Comparing number formats



"FP8: Efficient model inference with 8-bit floating point numbers", baseten blog post, <https://www.baseten.co/blog/fp8-efficient-model-inference-with-8-bit-floating-point-numbers/>

[1] LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale

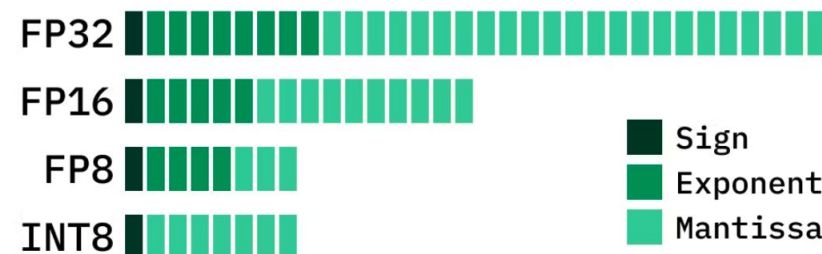
[2] <https://pytorch.org/docs/stable/quantization.html>

Quantization

Quantization converts the final model into much lower precision (e.g., BF16 → Int8).

- Some work [1] shows that some parts of a model may be less suitable for quantization.

Comparing number formats



"FP8: Efficient model inference with 8-bit floating point numbers", baseten blog post, <https://www.baseten.co/blog/fp8-efficient-model-inference-with-8-bit-floating-point-numbers/>

Another strategy: Quantization Aware Training (QAT)

- QAT models the *effects of quantization* during training allowing for higher accuracy compared to other quantization methods
- These techniques are supported by popular learning frameworks such as PyTorch [2]

[1] LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale

[2] <https://pytorch.org/docs/stable/quantization.html>

Finetuning and Alignment

Finetuning and Alignment

Can finetune an instruction-following, chat, or other model variants.

- Or for enhance specific abilities: arithmetic, coding

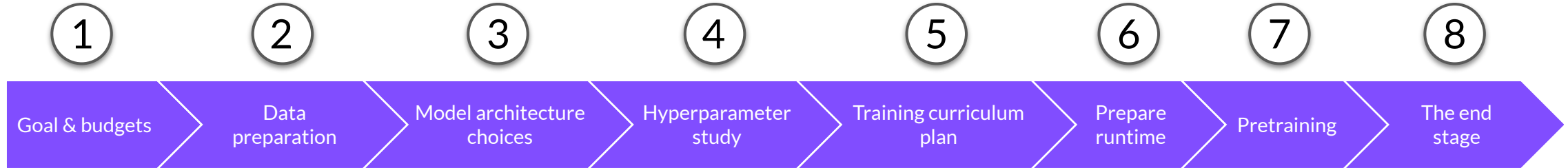
Tune with safety and culture alignment

- Larger models tend to overfit on finetuning data a lot more (e.g., harder to tune K2 than Crystal)

Finetune with vision ability – using multimodal architectures

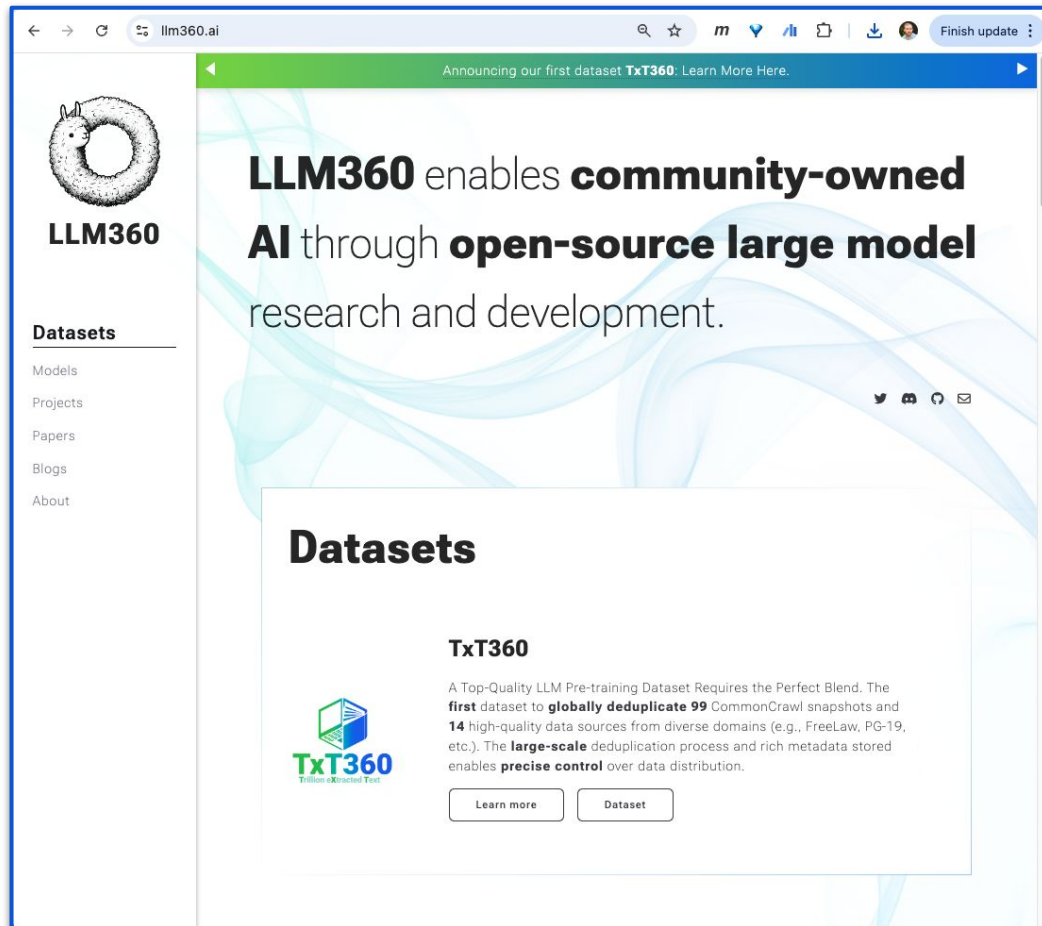
- E.g., Crystal-Vision and K2-Vision (coming soon)

A (Brief) Pretrainer's Guide to Large Language Models



Check Out Our Website

LLM360 – enables community-owned AI through open-source large model R&D.



Check out:

<https://www.llm360.ai>

Links to all of our artifacts mentioned above!

- Multiple pre-trained models (7B+).
- 360 model checkpoints.
- All training code.
- All datasets, data processing.
- Analysis of checkpoints, logs, metrics.
- And more...