A little guide to building Large Language Models in 2024

thomas@huggingface.co

Hi!

CSO and Co-founder of Hugging Face

Created HuggingFace Transformers and Datasets libraries.

Exploring Open-science in research in AI/ML, trying to lower the gap between academia and industrial labs through projects like the BLOOM/BigScience Workshop

Co-wrote "Natural Language Processing with Transformers" (O'Reilly)

Push for lowering the access barrier in AI for researchers and practitioners

Website: https://thomwolf.io
E-Mail: thomas@huggingface.co
Twitter: @thom_wolf
LinkedIn: @thom-wolf

Plan of today

- Quick overview of Hugging Face
- LLMs in 2024 at Hugging Face
  - data
  - training
  - evaluation
  - alignment
  - inference
- Closing notes

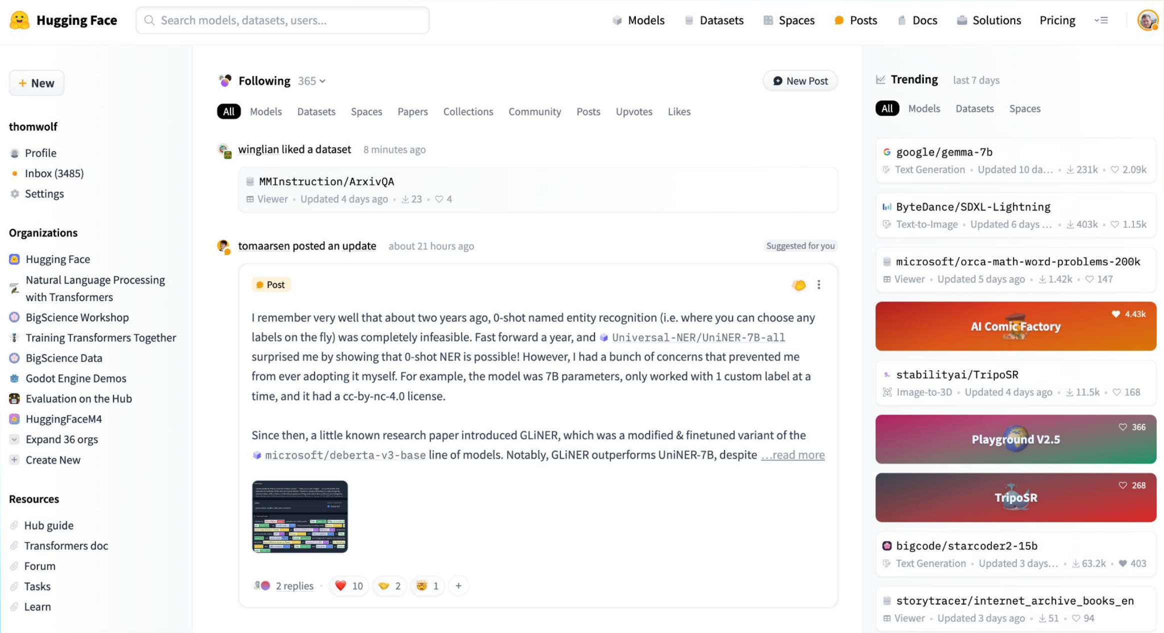
WTF is Hugging Face?

Open vs Closed Models

Open-source and closed models have different benefits that should be considered carefully for each use-case

	Open-Source	Closed / Proprietary
Security	Models can be **self-hosted**, data stays in your environment	Models cannot be self-hosted. Data is **sent outside** your environment to vendor
Control	The timing and nature of updates are **controlled** by you	Updates and changes to performance can happen **without notice**
Customization	**Full source code access** to customize the model for your needs	**Limited ability to customize** for your needs
Transparency	**Inspect code and data** provides better auditability and understandability	**No ability to audit** or understand performance
Cost	Typical **lower long term cost** due to smaller model size	Typically higher long term cost due to larger model size and proprietary **premium**
Latency	**Lower latency** due to on premise and smaller model sizes	Typically **greater latency** due to larger model sizes
Quality	No single approach is best. Each use case will vary. Proprietary is typically **closer to the frontier of performance**.	
Examples	OpenAI, Meta, salesforce, FLAN-T5, MISTRAL AI_, Microsoft	OpenAI, ANTHROPIC

Hugging Face: The home of open ML



Founded In
2016

170
Employees

300K+
stars on Github

500K+
open source models

100K+
public data sets

1M+
daily downloads

700K+
daily visitors

30+
Libraries

Used everywhere in the AI world

15,000+ startups and enterprises

grammarly, HUAWEI, Dropbox, intuit, Spotify, orange, databricks, Apple, servicenow, Meta, Microsoft, ORACLE, NVIDIA, salesforce, Bloomberg, HubSpot, Klarna., Uber, SAMSUNG

Hugging Face

Open source contributors

Meta, MISTRAL AI_, stability.ai, Microsoft, Ai2, OpenAI

Cloud partners

aws, Azure, NVIDIA, CLOUDFLARE

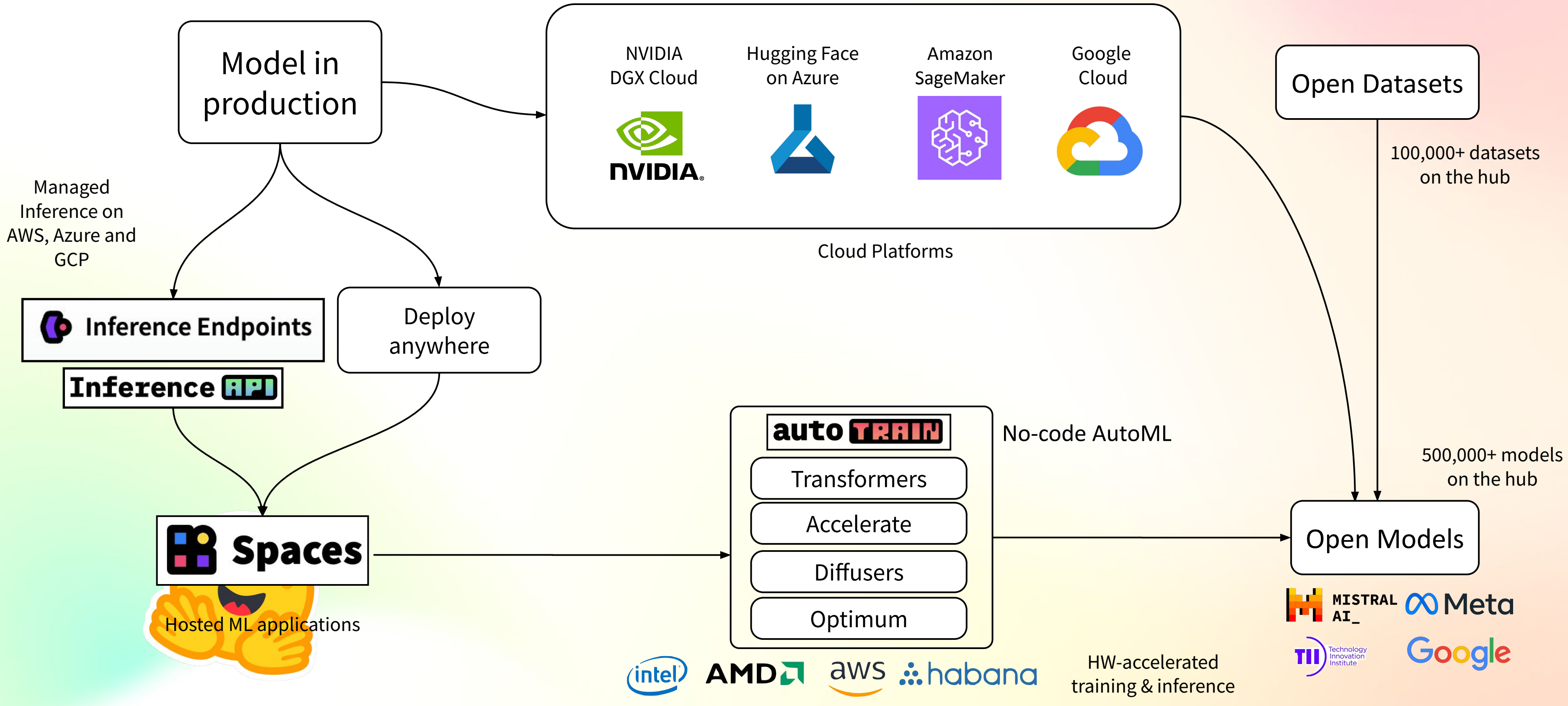
Hardware partners

NVIDIA, AMD, aws, intel, habana, GRAPHCORE, Qualcomm

On-prem partners

DELL, IBM

Integrated with the ML ecosystem



Open-source Ecosystem

- **Transformers**

  State-of-the-art ML for Pytorch, TensorFlow, and JAX.

- **Datasets**

  Access and share datasets for computer vision, audio, and NLP tasks.

- **Gradio**

  Build machine learning demos and other web apps, in just a few lines of Python.

- **Safetensors**

  Simple, safe way to store and distribute neural networks weights safely and quickly.

- **Transformers.js**

  Community library to run pretrained models from Transformers in your browser.

- **Hub Python Library**

  Client library for the HF Hub: manage repositories from your Python runtime.

- **Diffusers**

  State-of-the-art diffusion models for image and audio generation in PyTorch.

- **Accelerate**

  Easily train and use PyTorch models with multi-GPU, TPU, mixed-precision.

- **TRL**

  Train transformer language models with reinforcement learning.

- **timm**

  State-of-the-art computer vision models, layers, optimizers, training/evaluation, and utilities.

- **PEFT**

  Parameter efficient finetuning methods for large models

- **Text Generation Inference**

  Toolkit to serve Large Language Models.

LLMs in 2024
at Hugging Face

The workflow for LLMs

I – Training:

- data preparation – **datatrove**
- efficient training techniques – **nanotron**
- evaluation – **lighteval**

II – Fine-tuning:

- RLHF – **TRL**

III – Inference:

- quantization – **bitsandbytes**
- deployment – **Text Generation Inference**

Training:
1. Data preparation

I – Training

Often a lot of focus on models architecture by participants

The "it" in AI models is the dataset.

Posted on June 10, 2023 by jbetker

I've been at OpenAI for almost a year now. In that time, I've trained a **lot** of generative models. More than anyone really has any right to train. As I've spent these hours observing the effects of tweaking various model configurations and hyperparameters, one thing that has struck me is the similarities in between all the training runs.

It's becoming awfully clear to me that these models are truly approximating their datasets to an incredible degree. What that means is not only that they learn what it means to be a dog or a cat, but the interstitial frequencies between distributions that don't matter, like what photos humans are likely to take or words humans commonly write down.

What this manifests as is – trained on the same dataset for long enough, pretty much every model with enough weights and training time converges to the same point. Sufficiently large diffusion conv-unets produce the same images as ViT generators. AR sampling produces the same images as diffusion.

This is a surprising observation! It implies that model behavior is not determined by architecture, hyperparameters, or optimizer choices. It's determined by your dataset, nothing else. Everything else is a means to an end in efficiently delivery compute to approximating that dataset.

Then, when you refer to "Lambda", "ChatGPT", "Bard", or "Claude" then, it's not the model weights that you are referring to. It's the dataset.

https://nonint.com/2023/06/10/the-it-in-ai-models-is-the-dataset/

I – Training

2 Pretraining

Our approach to pretraining is to train a standard dense transformer architecture on a heavily engineered large pretraining corpora, where our underlying assumption is that when trained on extensive data of high-enough quality, a standard architecture can exhibit advanced capability. This is to say, we may not need much architectural modification, although we have indeed conducted extensive preliminary architectural experiments. In the following subsections, we first detail our data engineering pipeline, then briefly discuss the model architecture.

Yi: Open Foundation Models by 01.AI

https://arxiv.org/abs/2403.04652

I – Training

I – Data preparation:

- ★ Latest resources
  - A Survey on Data Selection for Language Models
    https://arxiv.org/abs/2402.16827
  - Yi: Open Foundation Models by 01.AI
    https://arxiv.org/abs/2403.04652

- ★ Recent dataset reports
  - Dolma (AllenAI): an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research
    https://arxiv.org/abs/2402.00159
  - RefinedWeb (Falcon): Outperforming Curated Corpora with Web Data, and Web Data Only
    https://arxiv.org/abs/2306.01116

Data preparation

- LM training requires multiple stages:
  - pretraining
  - instruction-tuning
  - alignment
  - in-context learning
  - task-specific fine-tuning
- each training stage has different goals
- data selection methods will use different mechanisms

Data preparation

Pretraining stage:

★ Goal: train a general-purpose model – maximal coverage

★ Requires: train on massive quantities of text, at least 1 trillion of tokens nowadays

★ Challenges:

- maximizing diversity and coverage
- maximising quality, robustness
- data quality evaluation: how to measure data quality at the billion tokens scale

Data preparation

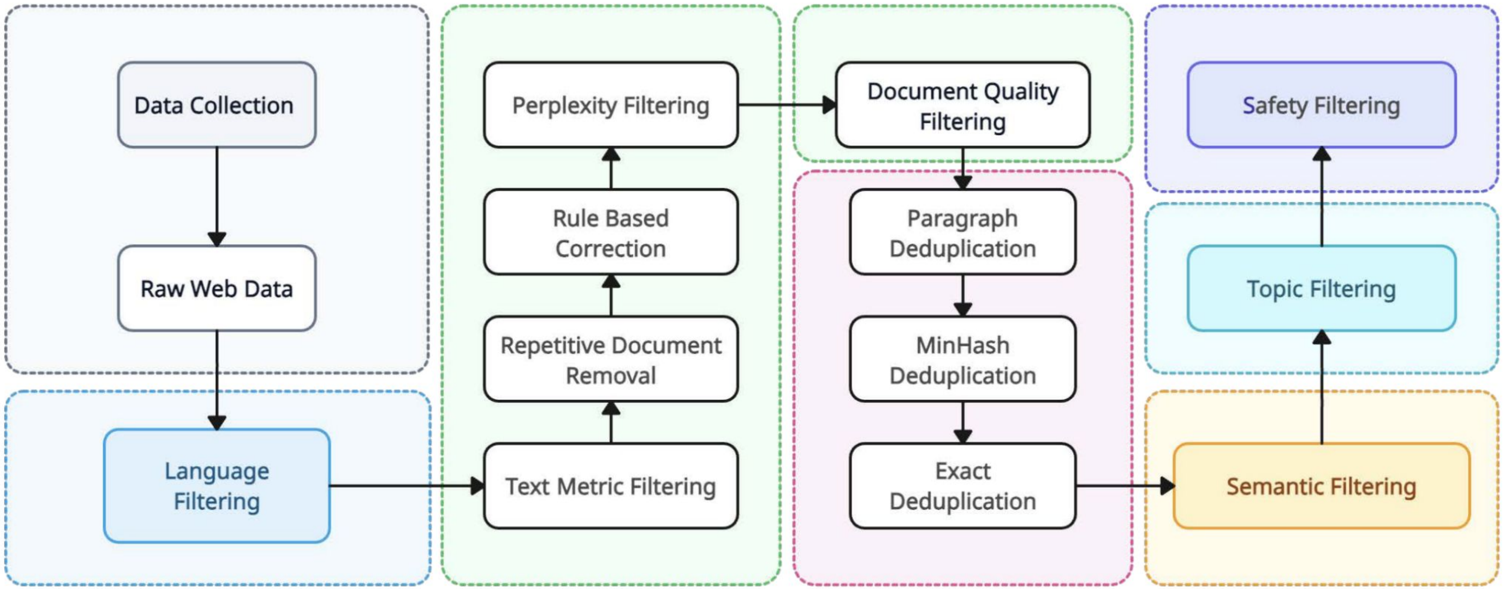


Figure 1: Yi's pretraining data cleaning pipeline.

Data preparation

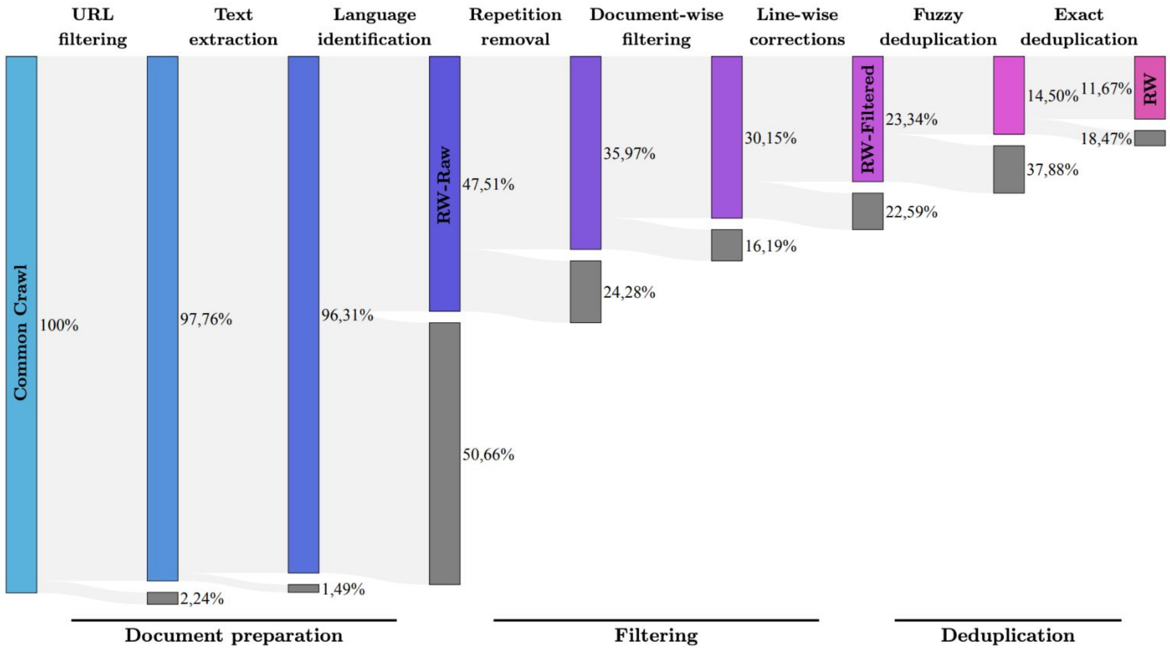


Figure 2. **Subsequent stages of Macrodata Refinement remove nearly 90% of the documents originally in CommonCrawl.** Notably, filtering and deduplication each result in a halving of the data available: around 50% of documents are discarded for not being English, 24% of remaining for being of insufficient quality, and 12% for being duplicates. We report removal rate (grey) with respect to each previous stage, and kept rate (shade) overall. Rates measured in % of documents in the document preparation phase, then in tokens.

Data preparation

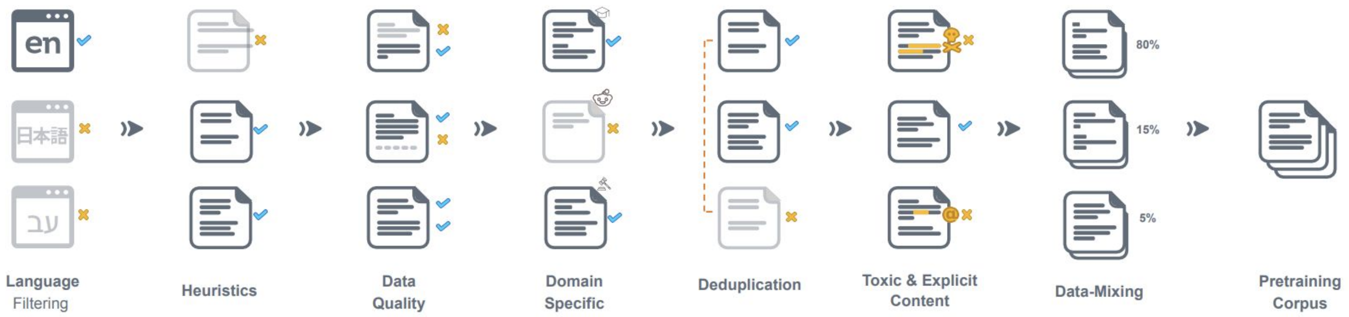


Figure 4: **An overview of the data filtering pipeline for pretraining**. Each filtering component is described in Section 3, and depicts common filters used for preprocessing text data. Note that different works employ different filters, at different stages, and do not necessarily adhere to the order conveyed here.

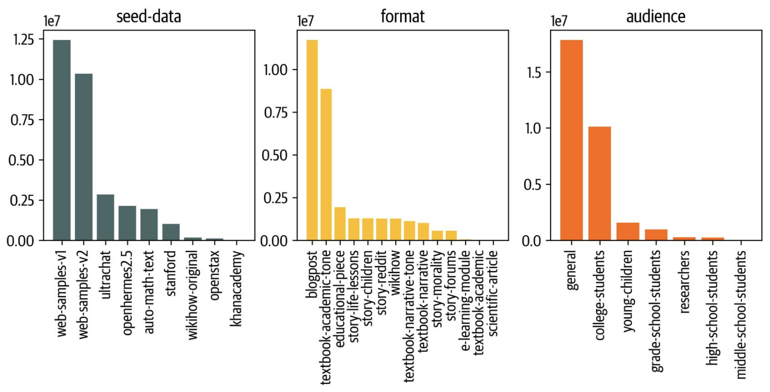
Data sources

Data sources

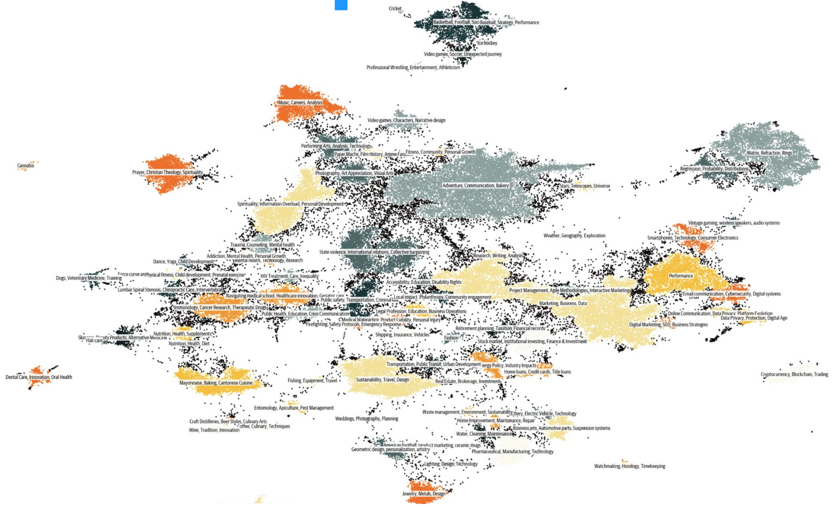
- Very large (> 100B tokens):
  - **Common crawl**: everyone starts from here
  - **Code**: Github and Software Heritage
- Curated:
  - Wikipedia
  - Books: public-domain vs. copyrighted
- More recent trends
  - **Synthetic data**

Synthetic data: Cosmopedia v0.1 🛰️

★ A synthetic dataset of 30M samples
★ generated by Mixtral-8x7B-Instruct-v0.1
★ 8 splits:
  ○ various sources for seed samples: Stanford, OpenStax and KhanAcademy, webdata, instruction-tuning datasets
  ○ model is asked to generate content



Cosmopedia v0.1



Benchmark	WinoGrande	ARC Easy	ARC Challenge
Cosmo-1B	54.2	56.8	33.0
TinyLlama-1.1B-3T*	59.1	55.2	30.1
Qwen1.5-1.8B	59.4	59.0	34.9
phi-1.5	68.7	73.1	48.0

Table 1: Common Sense Reasoning Benchmarks (* from TinyLlama's Github).

Benchmark	PIQA	HellaSwag	MMLU	OpenBookQA
Cosmo-1B	71.3	55.1	32.4	35.4
TinyLlama-1.1B-3T*	73.3	59.2	25.9 (2.5T)	36
Qwen1.5-1.8B	74.1	60.9	45.9	34.2
phi-1.5	75.6	62.6	43.2	48.0

Table 2: Language Understanding and Knowledge benchmarks (* from TinyLlama's Github).

Code data: StarCoder 2 and The Stack v2 💫

- ★ over 3B files in 658 programming and markup languages
- ★ created as part of the BigCode Project pre-training dataset for Code LLMs
- ★ Derived from the Software Heritage archive: largest public archive of software source code

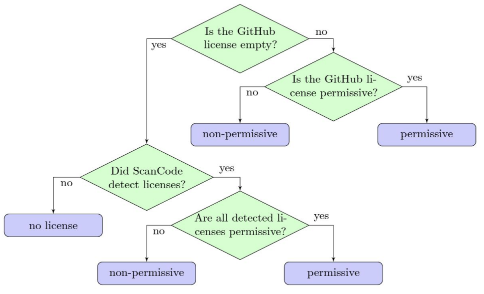


Figure 1: *File-level license assignment logic.*

	The Stack v1	The Stack v2
full	6.4TB	67.5TB
dedup	2.9TB	32.1TB
train (full)	~200B tokens	~900B tokens

StarCoder 2 and The Stack v2: The Next Generation

Published on Feb 29 · ⭐ Featured in Daily Papers on Mar 1

Authors: Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian McAuley, Han Hu, Torsten Scholak, Sebastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, Mostofa Patwary, Nima Tajbakhsh, Yacine Jernite, Carlos Muñoz Ferrandis, Lingming Zhang, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, Harm de Vries

webdata: FineWeb

Coming soon....

Language filtering

fastText

Language Filtering Summary & Ideas

- Language filtering is an important first step of data selection where documents are selected to include only the desired languages.
- Classifier-based methods are the norm when creating both english-only and multilingual datasets.
- For lower-resourced languages, URL-based methods are also useful for classifying the language used in multilingual datasets and can sometimes can be more reliable than classifiers.
- Filtering for code languages is very simple, most works simply using the file extension.

💡 One interesting direction for future research is developing methods that find code data that within documents of natural language. To the best of our knowledge, no such methods currently exist, but a code-language identifier could be used in domains with interleaved code and natural language to find meaningful quantities of new code data.

Quality filtering heuristics

heuristics

Heuristic Category	Common Utility Functions	Example Selection Mechanisms
Item Count	# of characters in a {word/line/paragraph/document} # of {words/lines/sentences} in a document	Remove documents with fewer than 5 words (Raffel et al., 2020)
Repetition Count	# of times a {character/n-gram/word/ sentence/paragraph} is repeated	Remove lines that repeat the same word more than 4 times consecutively (Laurençon et al., 2022)
Existence	Whether a {word/n-gram} is in the document Whether a terminal punctuation is at the end of a line	Remove lines starting with "sign-in" (Penedo et al., 2023)
Ratio	% of alphabetic characters in a document % of numerals/uppercase characters in a {line/document}	Remove documents with a symbol-to-word ratio greater than 0.1 (for "#" and "...") (Rae et al., 2022)
Statistics	The mean length (and standard deviation) of all lines in a document	Remove code files that have mean line length greater than 100 characters (Chen et al., 2021)

Table 2: **Commonly used heuristic utility functions and demonstrative selection mechanisms.**

Quality filtering heuristics

- Advantages
  - controlled
  - robust
  - rather clear priors
- Drawbacks
  - rely entirely on surface level
  - danger of removing too much
  - hyper-parameter tuning

Quality filtering – ML filtering

Given a set of examples of good/bad documents:

- ★ classifier-based quality filtering:
  fastText classification with an n-gram size of 2
- ★ perplexity based filtering:
  5-gram Kneser-Ney model on Wikipedia

Filter based on a threshold

Allow for more “quality/content based filtering” but harder to estimate the impact of the training documents.
May introduce unwanted “bias”

Notes on data filtering

★ Taking care of domains specificities
  - important to inspect the effect on domain specific data
  - extract 10 documents per domains (e.g. top urls)
  - manually inspect the results
  - craft domain specific filters/hyper-parameters
  - same for multiple languages

★ Deterministic vs. stochastic selection
  - hard threshold are strong decision points
  - stochastic smoothing of rules

Data deduplication

Reasoning: a lot of duplicate/near duplicate documents in large scale dumps (internet/github…)

★ increases the distribution density around those areas
★ duplicated data point have more chance of being memorized (Carlini et al. (2023))
★ filtering out duplicates reduce training time
★ reducing duplication improve accuracy on downstream tasks (Lee et al., 2022a; Tirumala et al., 2023)

Data deduplication

- methods:
    - Fuzzy:
        - BLOOM filters (hashing and fixed size vector)
        - MinHash (hashing and sorting)
    - Exact
        - Exact substring with suffix array
        - Sentence dedup
- time/memory consumption
    - MinHash offers a good trade–off of speed/memory with more control than BLOOM filters
- counter intuitive results
    - more deduplication may lead to keeping only bad data

Preparing the data for training

Shuffle:

➜ Important!

Tokenizers – some good practices:

★ Sample quite widely in your dataset (don't overfit to a subset)
★ For math: be careful of numbers – either split digits or add them manually
★ For code: be careful of spaces – handle them well to group them
★ Byte-level BPE is a good standard – Byte fallback is good as well

Scaling tokenization:

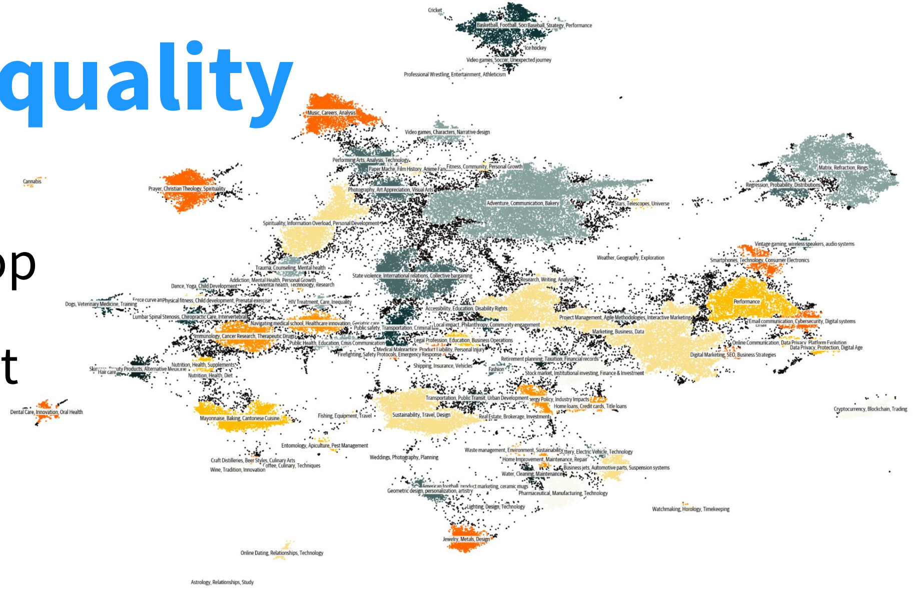
★ Tokenization of trillions token can be non negligible
    ○ tokenize dynamically during training
    ○ parallelize pretokenization in small data sub-sets

How to evaluate data quality

1. Small models trainings: train 1-2B size models on 30GT

- ★ Use a set of “high-signal” benchmarks (in NLP):
  - commonsenseqa
  - hellaswag
  - openbookqa
  - piqa
  - siqa
  - winogrande
  - arc
  - mmlu
- ★ High-signal?
  - **monotonicity**: monotonically increasing during training
  - **low variance**:
    - when comparing two known reference datasets (e.g. The Pile versus C4)
    - when comparing with various sub-parts of data and seeds
    - above random baseline
- ★ Tricky details to maximize signal:
  - Small models like “normalized loglikelihood” better
  - Larger models like “letter answers” better

How to evaluate data quality



2. Manual data inspection
   - ★ Inspect 10 documents for your top domains/URLs
   - ★ Inspect at various stage until post tokenization
   - ★ Inspect kepts and discarded documents
   - ★ Search tools in the dataset

3. Maps and clustering
   - ★ https://github.com/huggingface/text-clustering

4. Uncommon: training a tokenizer and inspect the top/last/longest tokens :)

datatrove

Quick intro

datatrove

- ★ Started as an open reproduction of the RefinedWeb corpus
- ★ Ended up in a fully-fledged lightweight library for processing, filter and deduplicate text data at a very large scale.
- ★ Provides prebuilt commonly used processing blocks with a framework to easily add custom functionality.
- ★ Full python
- ★ Local, remote and other file systems are supported through fsspec.
- ★ Scaled to 20k+ nodes on SLURM

lighteval

Quick intro

lighteval

- ★ Lightweight LLM evaluation suite
- ★ Allow to evaluate models with 3D parallelism
- ★ Custom evals and prompt explorations
- ★ Push to hub/WandB/tensorboard

Training:
2. Modeling

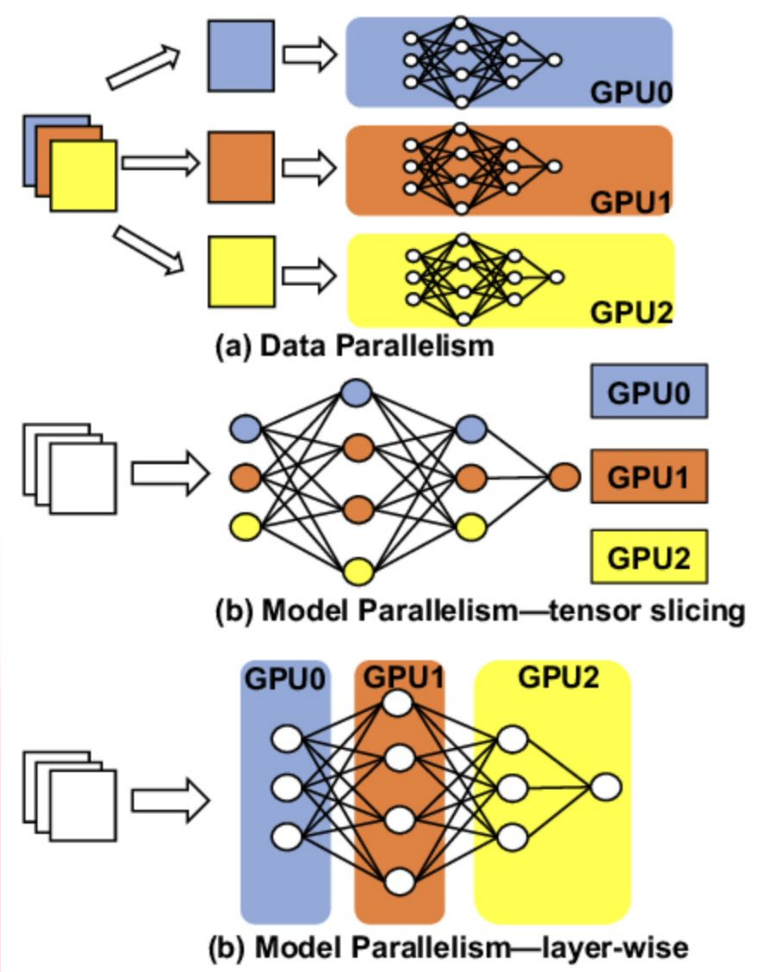
Training the model

Training LLM: the essential elements

- ★ size/efficiency
  - parallelism
  - asynchronicity
  - kernel merging
  - attention
- ★ instabilities
  - stable training recipes
- ★ capacity
  - mixture of experts
  - mamba

When the model is too big: Parallelism

★ 4D Parallelism:
  - Data Parallelism
  - Tensor Parallelism
  - Pipeline Parallelism
  - Sequence Parallelism

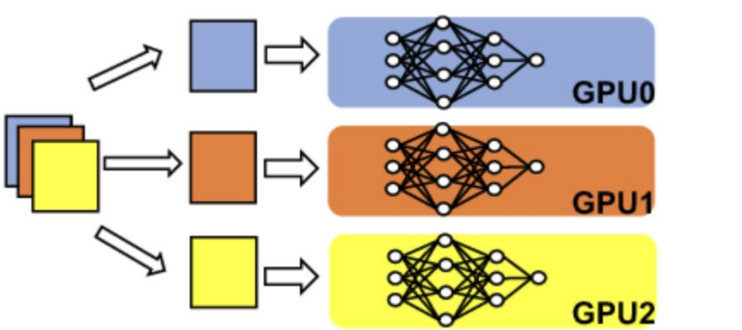


(a) Data Parallelism

(b) Model Parallelism—tensor slicing

(b) Model Parallelism—layer-wise

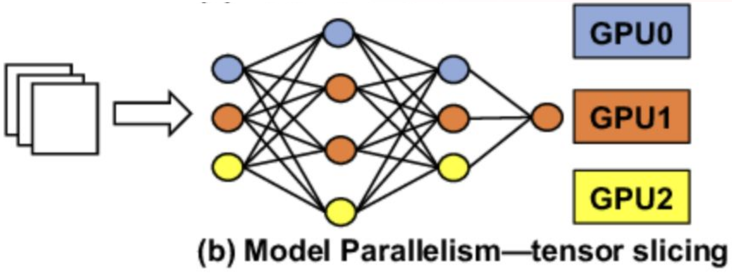
When the model is too big: Parallelism



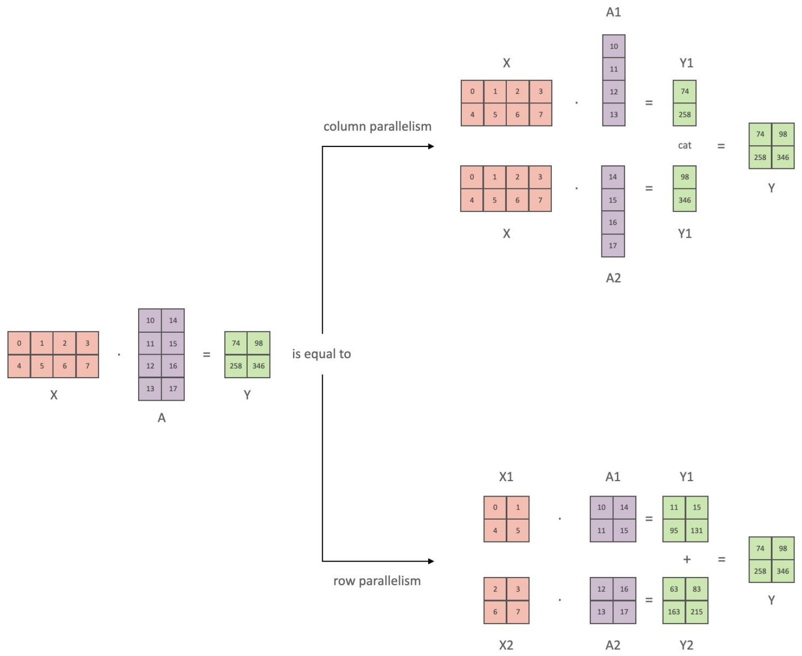
(a) Data Parallelism

- 4D Parallelism:
  - Data Parallelism
    - usually work out-of-the-box
      - just need to be careful with dataloading
    - challenges:
      - compute efficiency for gradient all-reduce
      - training efficiency of batch-size
  - Tensor Parallelism
  - Pipeline Parallelism
  - Sequence Parallelism

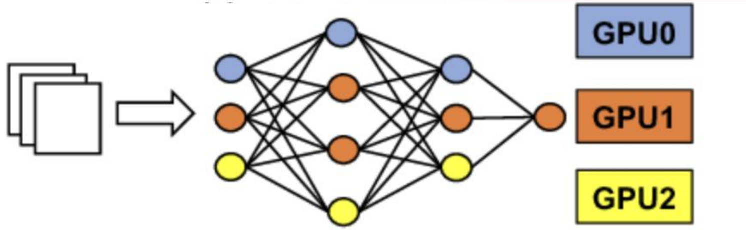
When the model is too big: Parallelism



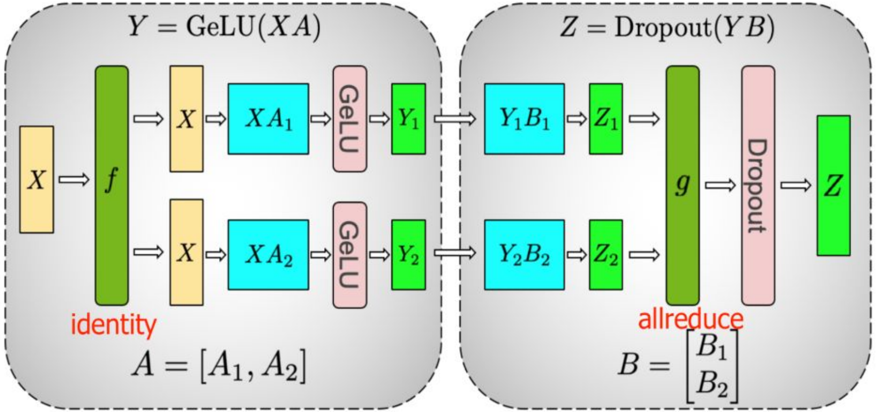
- ★ 4D Parallelism:
  - Data Parallelism
  - Tensor Parallelism
    - Re-write model code
    - Combine collumn/row slicing to reduce sync points
  - Pipeline Parallelism
  - Sequence Parallelism



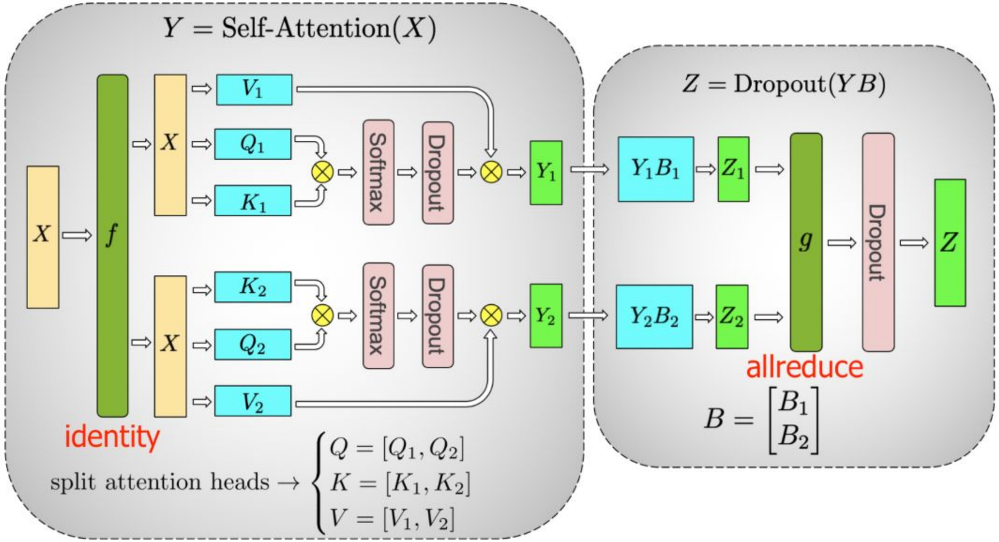
When the model is too big: Parallelism



(b) Model Parallelism—tensor slicing



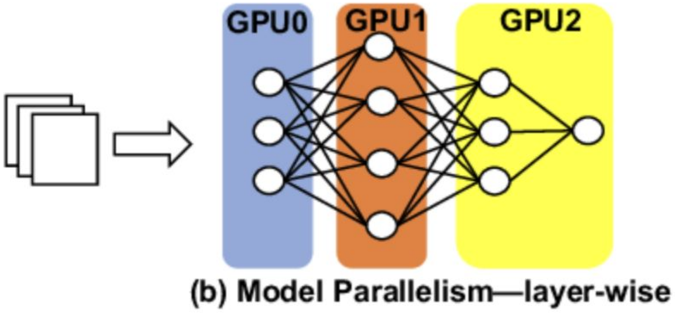
(a) MLP



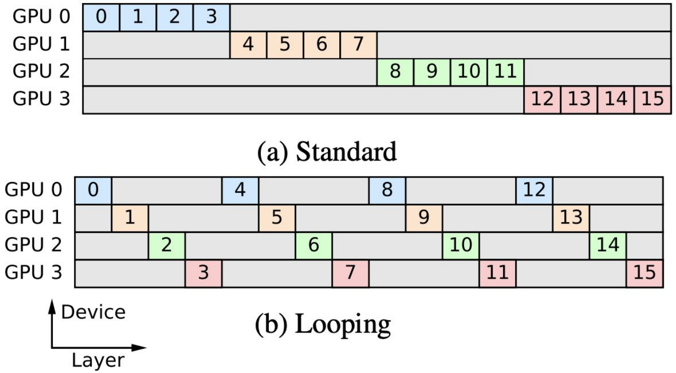
(b) Self-Attention

When the model is too big: Parallelism

★ 4D Parallelism:
- Data Parallelism
- Tensor Parallelism
- Pipeline Parallelism
  - Groupe sub-parts of the network
  - Challenge to keep all GPU busy
- Sequence Parallelism

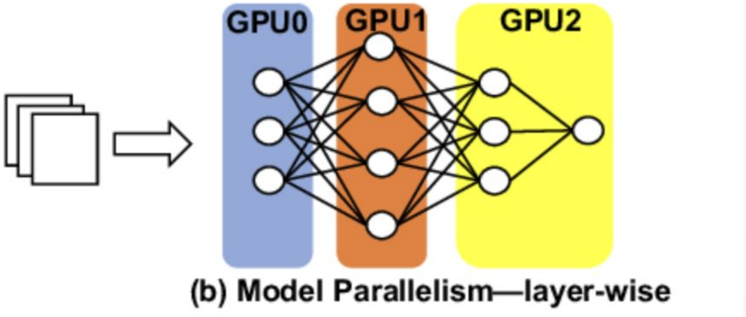


(b) Model Parallelism—layer-wise

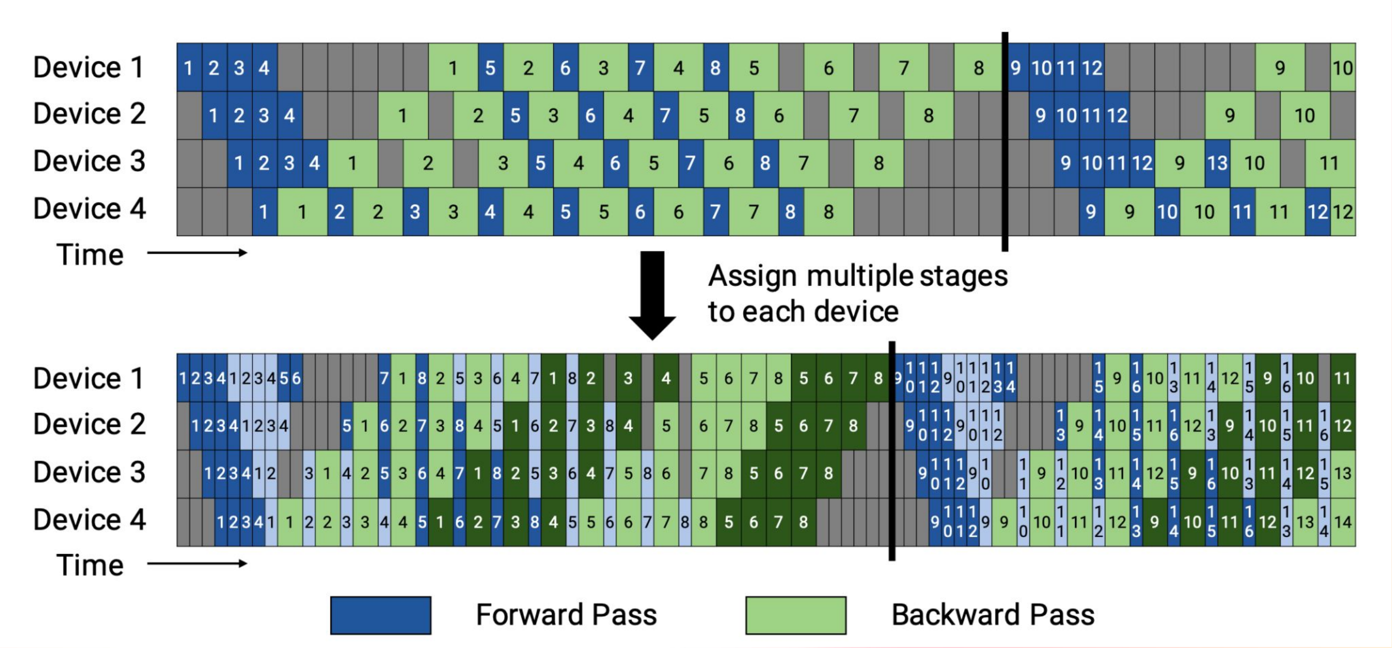


(a) Standard
(b) Looping

When the model is too big: Parallelism

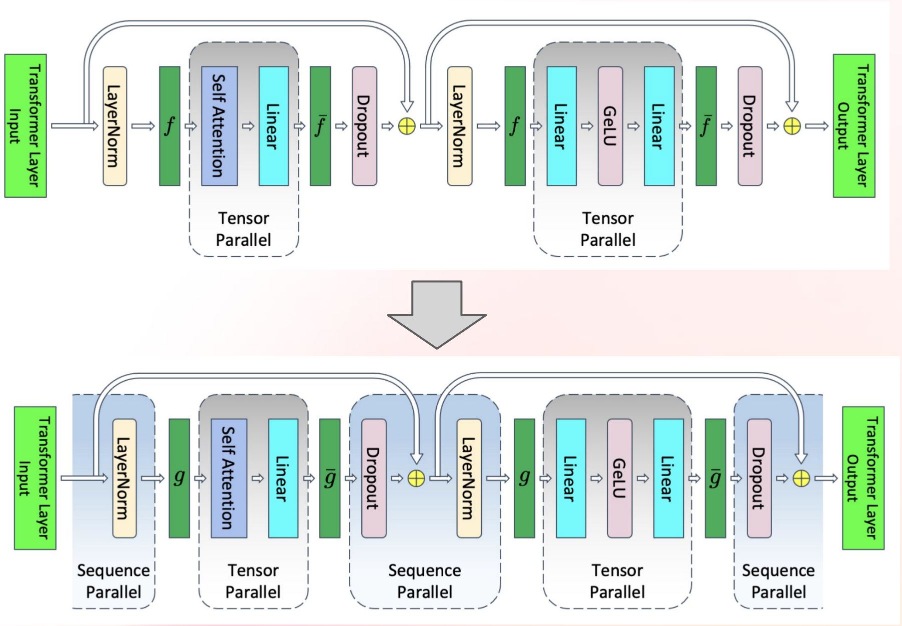


★ 4D Parallelism:



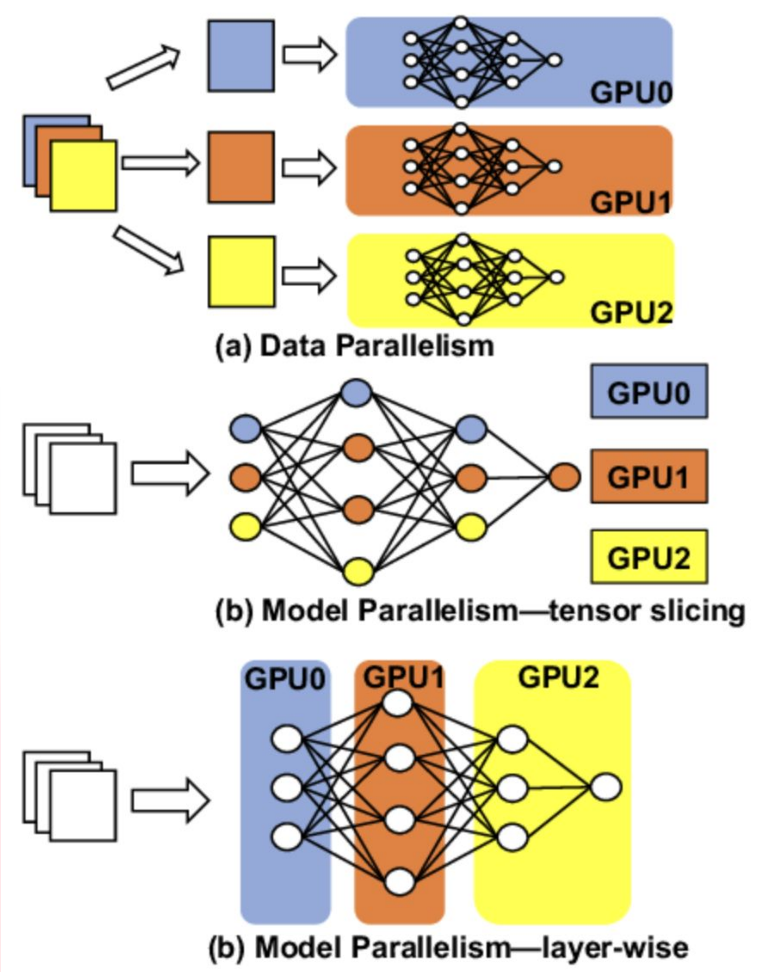
When the model is too big: Parallelism

★ 4D Parallelism:
- Data Parallelism
- Tensor Parallelism
- Pipeline Parallelism
- Sequence Parallelism
  - Add another parallelism
  - Be careful: another "sequence parallelism" exists: "ring attention" (also interesting)
  - Usually only interesting during training



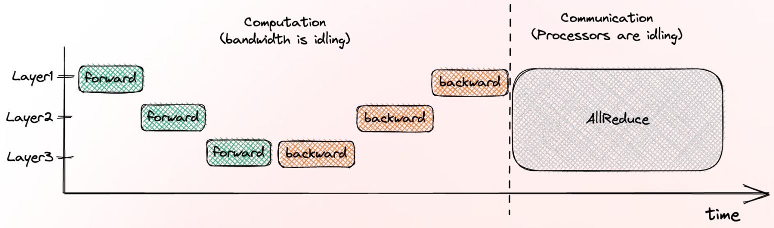
When the model is too big: Parallelism

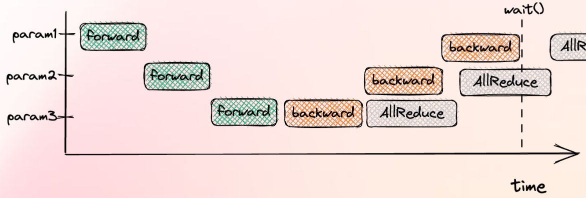
★ References:
- Breadth-First Pipeline Parallelism
  https://arxiv.org/abs/2211.05953
- Reducing Activation Recomputation in Large Transformer Models
  https://arxiv.org/abs/2205.05198
- Sequence Parallelism: Long Sequence Training from System Perspective
  https://arxiv.org/abs/2105.13120

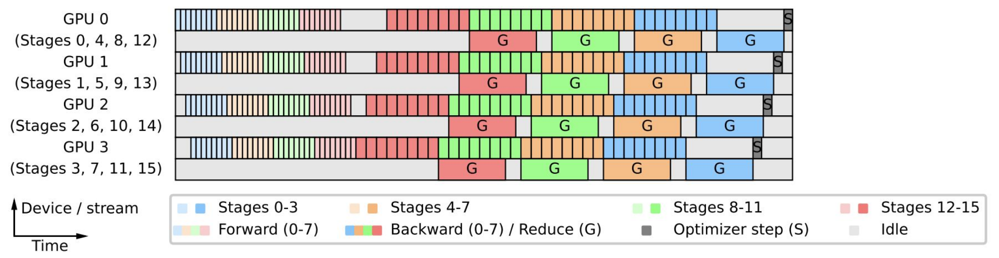


When the model is too big: efficiency

★ Challenges when scaling to multiple GPUs
- ○ Synchronization between:
  - ■ Multiple GPUs
  - ■ CPU and GPUs
- ○ Example:
  - ■ Data parallelism all_reduce overlapping communication and computation







(d) Looped pipeline, breadth-first schedule (PP_{BF}): small bubble, best overlap

Flash Attention

Stop materializing attention matrices!

2.2 Standard Attention Implementation

Given input sequences $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ where N is the sequence length and d is the head dimension, we want to compute the attention output $\mathbf{O} \in \mathbb{R}^{N \times d}$:

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N \times N}, \quad \mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N \times N}, \quad \mathbf{O} = \mathbf{P}\mathbf{V} \in \mathbb{R}^{N \times d},$$

where softmax is applied row-wise.

Standard attention implementations materialize the matrices $\mathbf{S}$ and $\mathbf{P}$ to HBM, which takes $O(N^2)$ memory. Often $N \gg d$ (e.g., for GPT2, $N = 1024$ and $d = 64$). We describe the standard attention implementation in Algorithm 0. As some or most of the operations are memory-bound (e.g., softmax), the large number of memory accesses translates to slow wall-clock time.

Tiling. We compute attention by blocks. Softmax couples columns of $\mathbf{K}$, so we decompose the large softmax with scaling [51, 60, 66]. For numerical stability, the softmax of vector $x \in \mathbb{R}^B$ is computed as:

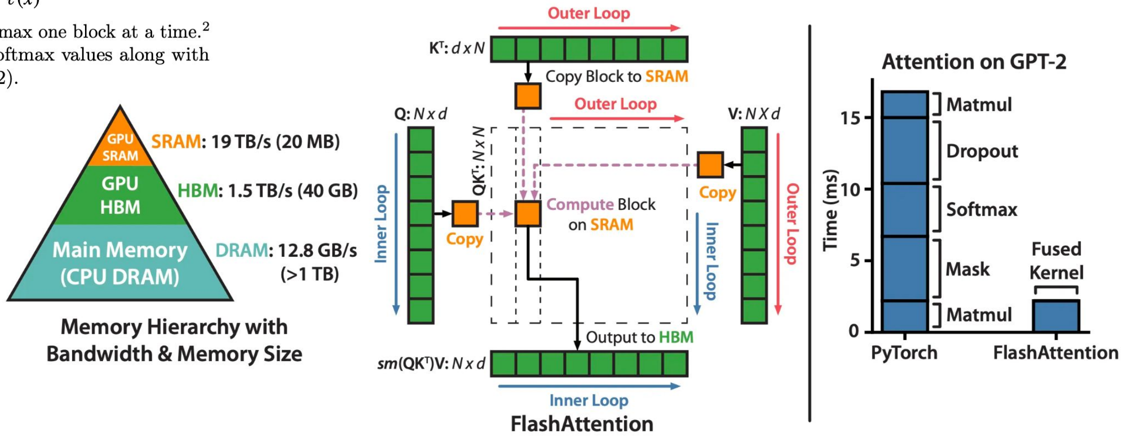
$$m(x) := \max_i \ x_i, \quad f(x) := \begin{bmatrix} e^{x_1 - m(x)} & \ldots & e^{x_B - m(x)} \end{bmatrix}, \quad \ell(x) := \sum_i f(x)_i, \quad \mathrm{softmax}(x) := \frac{f(x)}{\ell(x)}.$$

For vectors $x^{(1)}, x^{(2)} \in \mathbb{R}^B$, we can decompose the softmax of the concatenated $x = \begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix} \in \mathbb{R}^{2B}$ as:

$$m(x) = m(\begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix}) = \max(m(x^{(1)}), m(x^{(2)})), \quad f(x) = \begin{bmatrix} e^{m(x^{(1)}) - m(x)} f(x^{(1)}) & e^{m(x^{(2)}) - m(x)} f(x^{(2)}) \end{bmatrix},$$

$$\ell(x) = \ell(\begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix}) = e^{m(x^{(1)}) - m(x)} \ell(x^{(1)}) + e^{m(x^{(2)}) - m(x)} \ell(x^{(2)}), \quad \mathrm{softmax}(x) = \frac{f(x)}{\ell(x)}.$$

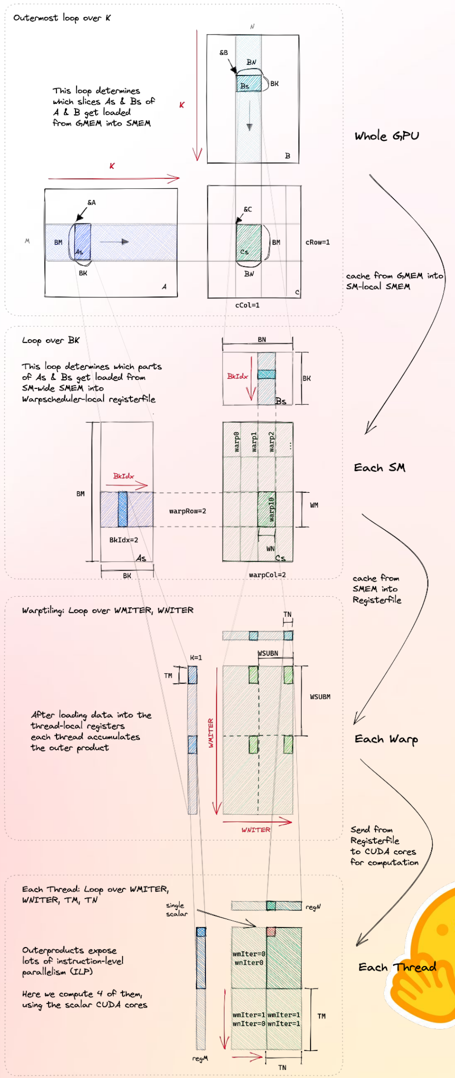
Therefore if we keep track of some extra statistics $(m(x), \ell(x))$, we can compute softmax one block at a time.[2] We thus split the inputs $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ into blocks (Algorithm 1 line 3), compute the softmax values along with extra statistics (Algorithm 1 line 10), and combine the results (Algorithm 1 line 12).



Flash Attention v2

Flash attention V2:

- ★ reduce the number of non-matmul FLOP (division, etc)
  - ○ each non-matmul FLOP is 16× more expensive than a matmul FLOP
- ★ better parallelism (sequences)
- ★ causal masks
- ★ better work partitioning (blocks, wraps)



Stable training recipes

- ★ Initialization
- ★ Stabilisation (see MuTransfer)
- ★ Learning rate: Cosine or not.
- ★ Scaling hyper-parameters results

References:

- ★ MiniCPM blog post
- ★ Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer https://arxiv.org/abs/2203.03466
- ★ Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster https://arxiv.org/abs/2304.03208

Capacity and architectures

Recent developments

- ★ Mixture-of-Experts (MoE)
  - ○ Efficient training on GPUs with Block Sparsity

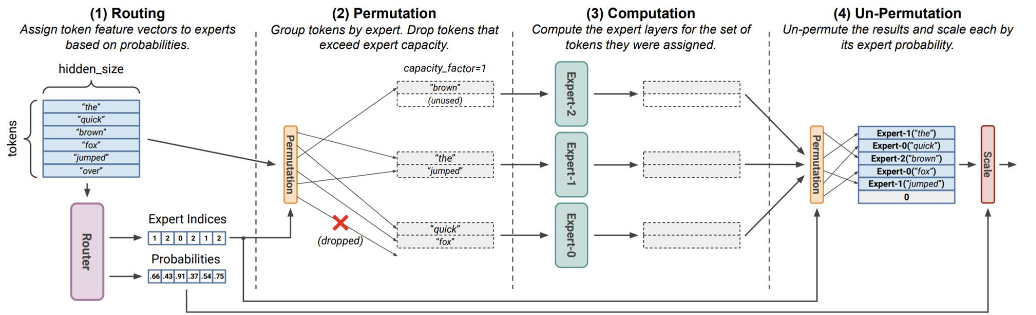


Figure 1. **A Mixture-of-Experts Layer.** Shown for *num_experts=3*, *top_k=1* and *capacity_factor=1* with the prevalent, token dropping formulation. **First (1)**, tokens are mapped to experts by the router. Along with expert assignments, the router produces probabilities that reflect the confidence of the assignments. **Second (2)**, the feature vectors are permuted to group tokens by expert assignment. If the number of tokens assigned to an expert exceeds its capacity, extra tokens are dropped. **Third (3)**, the expert layers are computed for the set of tokens they were assigned as well as any padding needed for unused capacity. **Lastly (4)**, the results of the expert computation are un-permuted and weighted by the router probabilities. The outputs for dropped tokens are shown here set to zero.

MegaBlocks: Efficient Sparse Training with Mixture-of-Experts

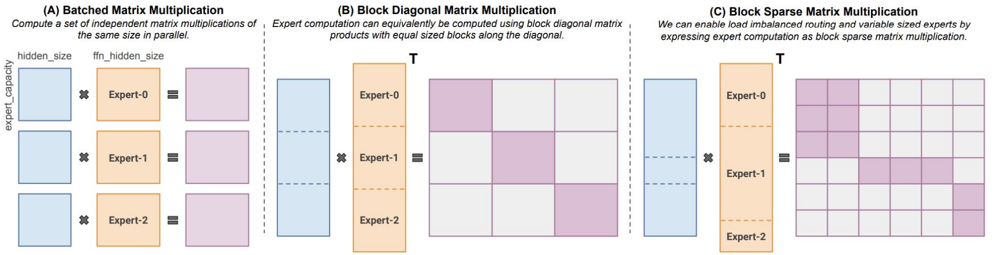


Figure 3. **Expert Computation in an MoE Layer.** Shown with *num_expert=3*. **(A)** State-of-the-art MoE implementations use batched matrix multiplication to compute all experts within a layer in parallel. This introduces the constraints that all experts are assigned the same number of tokens and that all experts have the same shape. **(B)** Expert computation can be analogously posed in terms of block diagonal matrix multiplication with identically sized blocks. **(C)** In order to relax these constraints, we can construct a block diagonal matrix with variable sized blocks made up of many smaller blocks. We can compute this matrix efficiently using block-sparse matrix multiplication.

Capacity and architectures

Recent developments: Mamba

➜ efficient and performant SSM

https://srush.github.io/annotated-mamba/hard.html

2 State Space Models

Structured state space sequence models (S4) are a recent class of sequence models for deep learning that are broadly related to RNNs, and CNNs, and classical state space models. They are inspired by a particular continuous system (1) that maps a 1-dimensional function or sequence $x(t) \in \mathbb{R} \mapsto y(t) \in \mathbb{R}$ through an implicit latent state $h(t) \in \mathbb{R}^N$.

Concretely, S4 models are defined with four parameters $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$, which define a sequence-to-sequence transformation in two stages.

$$h'(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t) \quad (1a)$$
$$y(t) = \boldsymbol{C}h(t) \quad (1b)$$

$$h_t = \overline{\boldsymbol{A}}h_{t-1} + \overline{\boldsymbol{B}}x_t \quad (2a)$$
$$y_t = \boldsymbol{C}h_t \quad (2b)$$

$$\overline{\boldsymbol{K}} = (\boldsymbol{C}\overline{\boldsymbol{B}}, \boldsymbol{C}\overline{\boldsymbol{A}\boldsymbol{B}}, \dots, \boldsymbol{C}\overline{\boldsymbol{A}}^k\overline{\boldsymbol{B}}, \dots) \quad (3a)$$
$$y = x * \overline{\boldsymbol{K}} \quad (3b)$$

Discretization. The first stage transforms the "continuous parameters" $(\Delta, \boldsymbol{A}, \boldsymbol{B})$ to "discrete parameters" $(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}})$ through fixed formulas $\overline{\boldsymbol{A}} = f_A(\Delta, \boldsymbol{A})$ and $\overline{\boldsymbol{B}} = f_B(\Delta, \boldsymbol{A}, \boldsymbol{B})$, where the pair (f_A, f_B) is called a discretization rule. Various rules can be used such as the zero-order hold (ZOH) defined in equation (4).

$$\overline{\boldsymbol{A}} = \exp(\Delta\boldsymbol{A}) \qquad \overline{\boldsymbol{B}} = (\Delta\boldsymbol{A})^{-1}(\exp(\Delta\boldsymbol{A}) - \boldsymbol{I}) \cdot \Delta\boldsymbol{B} \quad (4)$$

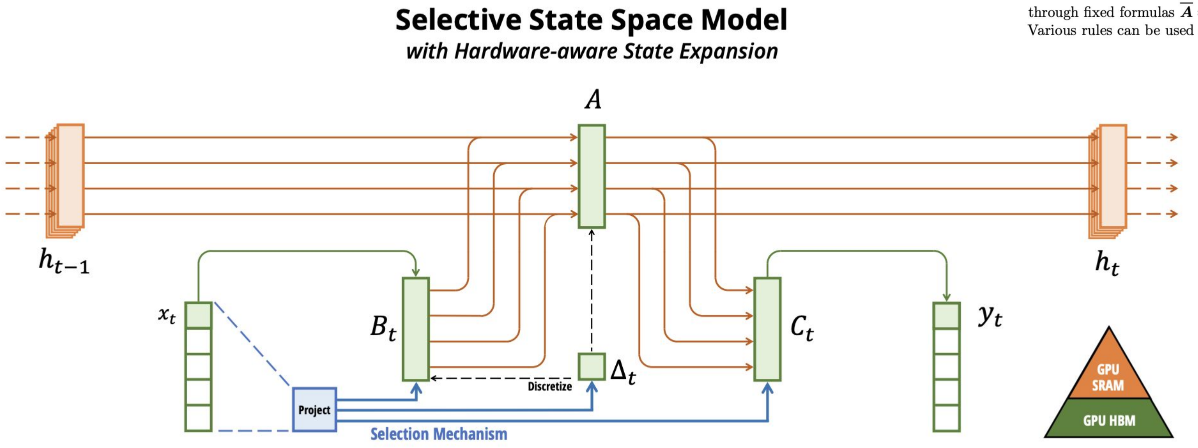
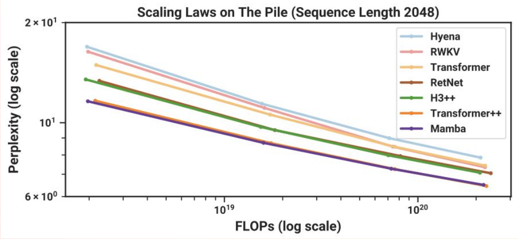


Figure 1: (**Overview**.) Structured SSMs independently map each channel (e.g. $D = 5$) of an input x to output y through a higher dimensional latent state h (e.g. $N = 4$). Prior SSMs avoid materializing this large effective state (DN, times batch size B and sequence length L) through clever alternate computation paths requiring time-invariance: the $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ parameters are constant across time. Our selection mechanism adds back input-dependent dynamics, which also requires a careful hardware-aware algorithm to only materialize the expanded states in more efficient levels of the GPU memory hierarchy.



nanotron

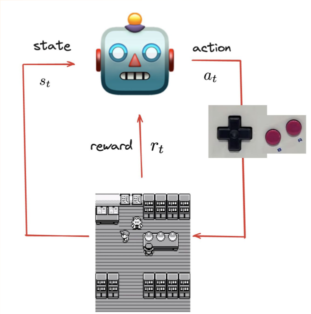
Quick intro

nanotron

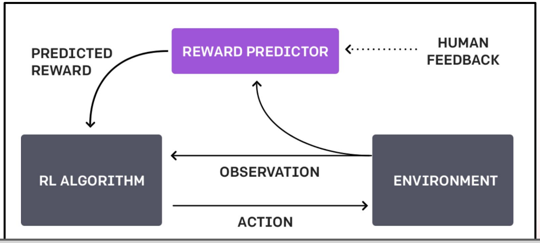
- ★ Minimalistic large language model 3D-parallelism training
- ★ Philosophy
  - ○ Make it fast
  - ○ Make it minimal
  - ○ Make everything explicit instead of transparent
- ★ We support the following
  - ○ 3D parallelism, including 1F1B pipeline engine
  - ○ ZeRO-1 optimizer
  - ○ FP32 gradient accumulation
  - ○ Parameter tying/sharding
- ★ Architectures
  - ○ Llama
  - ○ Mamba
  - ○ MoE

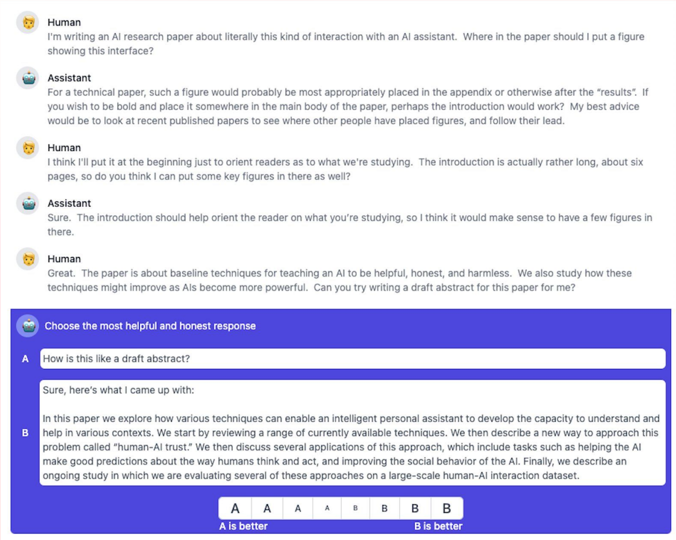
Alignment

★ RLHF in 2024:
  ○ Align the model’s outputs with human’s preferences
  ○ hard to design reward functions for RL




```
def reward_fn(a, ob):
    backroll = -ob[7]
    height = ob[0]
    vel_act = a[0] * ob[8] + a[1] * ob[9] + a[2] * ob[10]
    backslide = -ob[5]
    return backroll * (1.0 + .3 * height + .1 * vel_act + .05 * backslide)
```





Alignment

★ RLFH in 2024: PPO
🤯 results are incredible
❌ implementation is complicated
❌ GPU memory (fit 4 models) 🤯

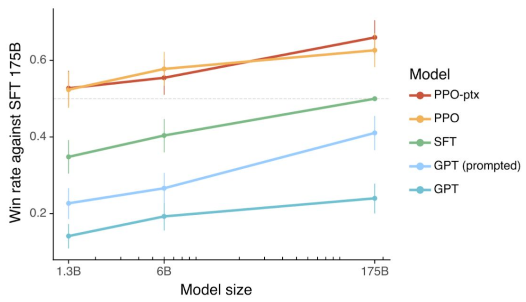
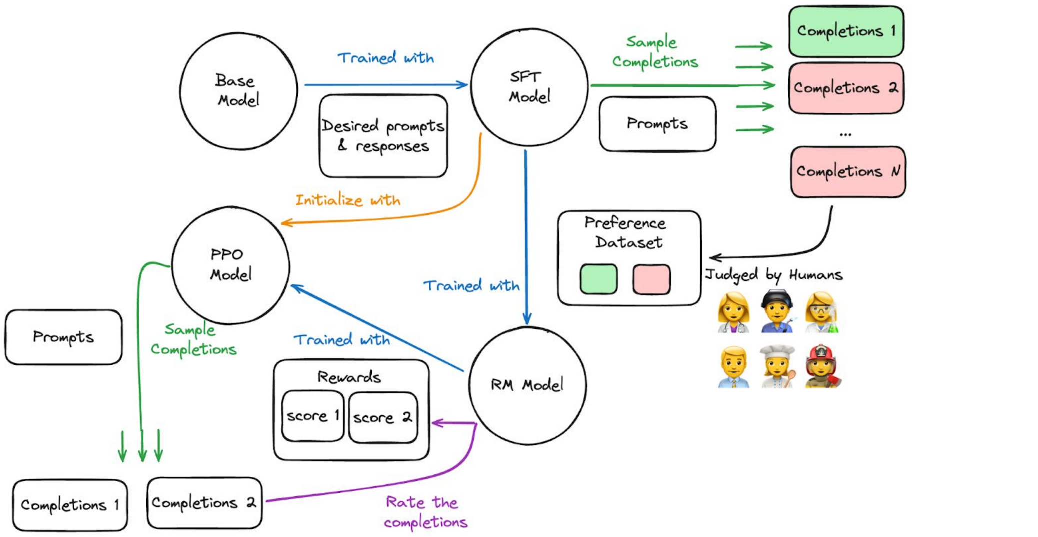


Figure 1: Human evaluations of various models on our API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3. Error bars throughout the paper are 95% confidence intervals.



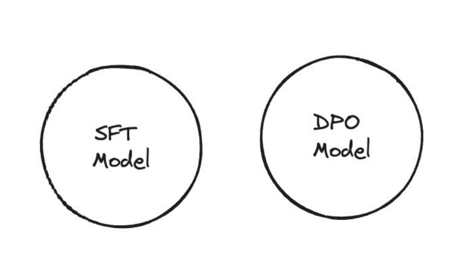
Alignment

★ RLFH in 2024: DPO

✅ results are great
✅ Implementation is easy
✅ Fits in GPU memory well (2 models)

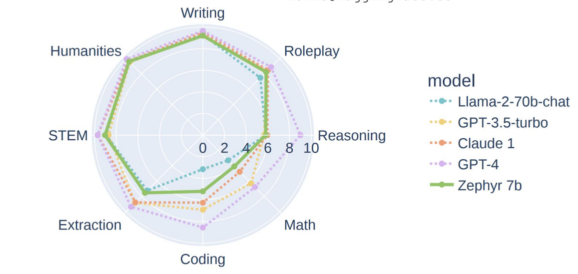
$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta;\pi_{\text{ref}}) = -\beta\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\underbrace{\sigma(\hat{r}_\theta(x,y_l)-\hat{r}_\theta(x,y_w))}_{\text{higher weight when reward estimate is wrong}}\Big[\underbrace{\nabla_\theta\log\pi(y_w\mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log\pi(y_l\mid x)}_{\text{decrease likelihood of } y_l}\Big]\Big],$$

where $\hat{r}_\theta(x,y) = \beta\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ is the reward implicitly defined by the language model π_θ and reference model π_{ref} (more in Section 5). Intuitively, the gradient of the loss function $\mathcal{L}_{\text{DPO}}$ increases the



ZEPHYR: DIRECT DISTILLATION OF LM ALIGNMENT

Lewis Tunstall,* Edward Beeching,* Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf
The H4 (Helpful, Honest, Harmless, Huggy) Team
https://huggingface.co/HuggingFaceH4
lewis@huggingface.co



Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov*† **Archit Sharma***† **Eric Mitchell***†
Stefano Ermon†‡ **Christopher D. Manning**† **Chelsea Finn**†
†Stanford University ‡CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Alignment

★ RLFH in 2024: Is PPO/on-policy RL out yet?

Reka Chat Models

Reka Flash and Reka Edge base models are instruction tuned and then RLHFed with PPO using Reka Flash as the reward model. We conduct a series of human evaluations to evaluate our chat models.

erences from human raters and trained a reward function under the Bradley-Terry model (Bradley and Terry, 1952), similarly to Gemini. The policy was trained to optimize this reward function using a variant of REINFORCE (Williams, 1992) with a Kullback–Leibler regularization term towards the initially tuned model. Similar to the

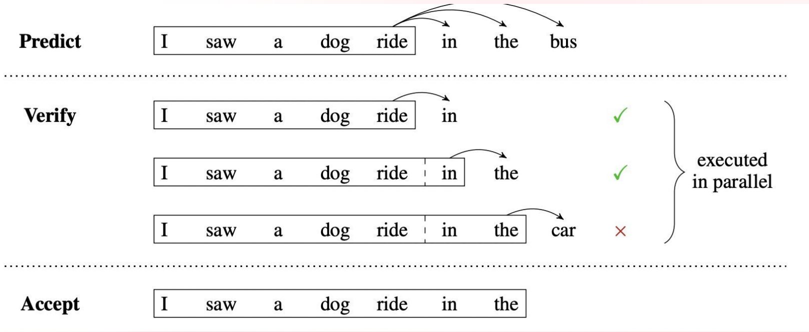
Method	TL;DR	HH (Pythia)	HH (Llama)
RLOO (k=4)	**77.9**	**43.7**	**64.1**
RAFT (k=4)	73.2	42.1	63.3
RLOO (k=2)	**74.2**	**47.6**	**62.2**
RAFT (k=2)	72.1	37.7	58.4
REINFORCE. W/ BASELINE	**70.7**	**37.9**	**55.3**
VANILLA PG	70.4	36.4	52.3
PPO	67.6	29.2	32.0
DPO	66.6	39.0	61.9

Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs

Arash Ahmadian, *Cohere For AI*
Chris Cremer, *Cohere*
Matthias Gallé, *Cohere*
Marzieh Fadaee, *Cohere For AI*
Julia Kreutzer, *Cohere For AI*
Olivier Pietquin, *Cohere*
Ahmet Üstün, *Cohere For AI*
Sara Hooker, *Cohere For AI*

Inference

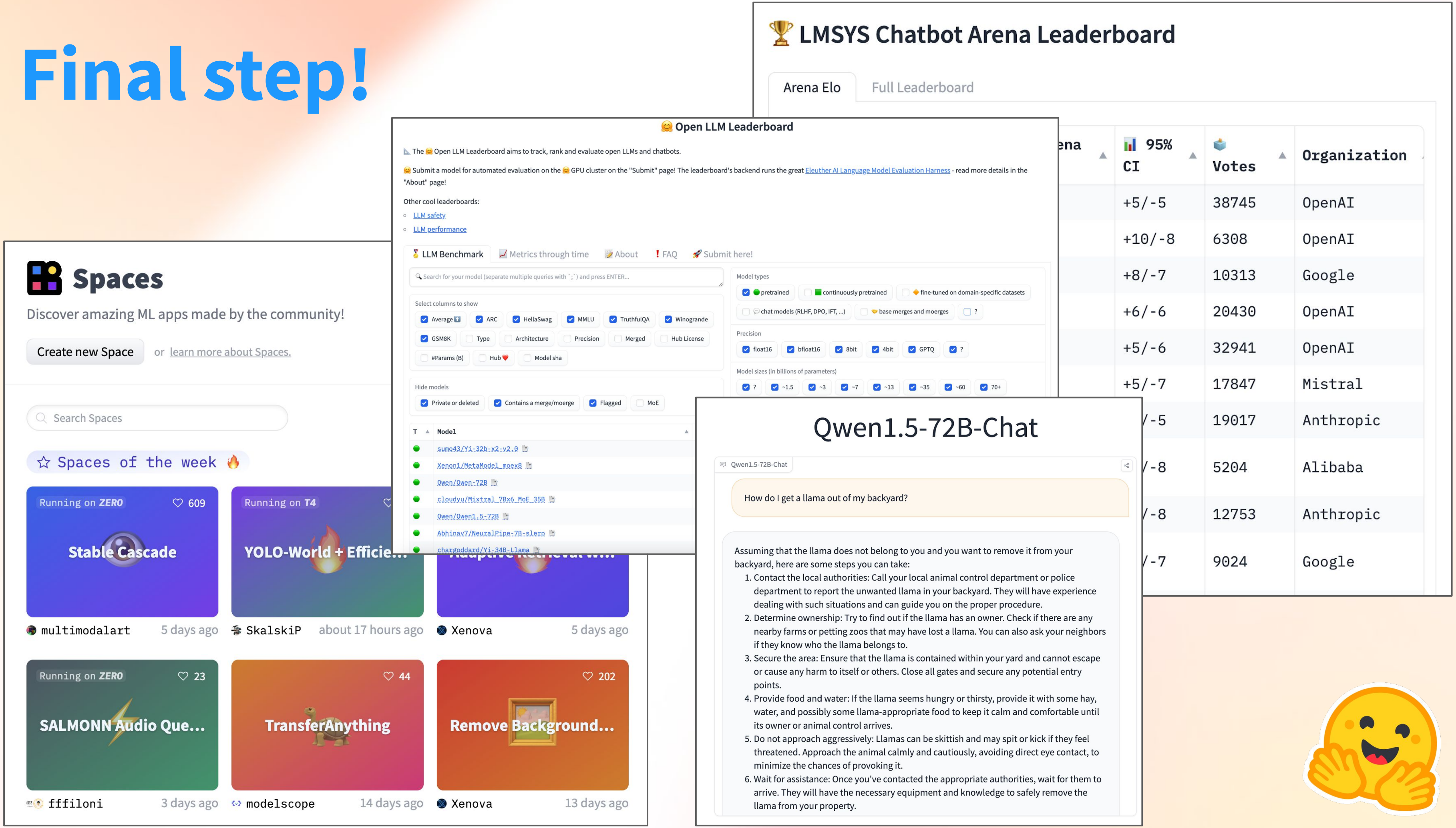
- ★ Quantization:
  - ○ Full quantization for inference: GPTQ/GGML/NF4 https://arxiv.org/abs/2210.17323 Comparison of all three technics
  - ○ AutoGPTQ
  - ○ lama.cpp



- ★ Speculative Decoding
  - ○ Use a small and large model in parallel
  - ○ Medusa: https://arxiv.org/abs/2401.10774

- ★ Compiling and CUDA graphs
  - ○ Accelerating Generative AI with PyTorch II: GPT, Fast

Final step!


LMSYS Chatbot Arena Leaderboard
Arena Elo
Full Leaderboard
95% CI
Votes
Organization
+5/-5 38745 OpenAI
+10/-8 6308 OpenAI
+8/-7 10313 Google
+6/-6 20430 OpenAI
+5/-6 32941 OpenAI
+5/-7 17847 Mistral
19017 Anthropic
5204 Alibaba
12753 Anthropic
9024 Google
Open LLM Leaderboard
The Open LLM Leaderboard aims to track, rank and evaluate open LLMs and chatbots.
Submit a model for automated evaluation on the GPU cluster on the "Submit" page! The leaderboard's backend runs the great Eleuther AI Language Model Evaluation Harness - read more details in the "About" page!
Other cool leaderboards:
LLM safety
LLM performance
LLM Benchmark
Metrics through time
About
FAQ
Submit here!
Spaces
Discover amazing ML apps made by the community!
Create new Space
or learn more about Spaces.
Search Spaces
Spaces of the week
Running on ZERO 609
Stable Cascade
multimodalart 5 days ago
Running on T4
YOLO-World + Efficie...
SkalskiP about 17 hours ago
Xenova 5 days ago
Running on ZERO 23
SALMONN Audio Que...
fffiloni 3 days ago
44
TransferAnything
modelscope 14 days ago
202
Remove Background...
Xenova 13 days ago
Qwen1.5-72B-Chat
How do I get a llama out of my backyard?
Assuming that the llama does not belong to you and you want to remove it from your backyard, here are some steps you can take:
1. Contact the local authorities: Call your local animal control department or police department to report the unwanted llama in your backyard. They will have experience dealing with such situations and can guide you on the proper procedure.
2. Determine ownership: Try to find out if the llama has an owner. Check if there are any nearby farms or petting zoos that may have lost a llama. You can also ask your neighbors if they know who the llama belongs to.
3. Secure the area: Ensure that the llama is contained within your yard and cannot escape or cause any harm to itself or others. Close all gates and secure any potential entry points.
4. Provide food and water: If the llama seems hungry or thirsty, provide it with some hay, water, and possibly some llama-appropriate food to keep it calm and comfortable until its owner or animal control arrives.
5. Do not approach aggressively: Llamas can be skittish and may spit or kick if they feel threatened. Approach the animal calmly and cautiously, avoiding direct eye contact, to minimize the chances of provoking it.
6. Wait for assistance: Once you've contacted the appropriate authorities, wait for them to arrive. They will have the necessary equipment and knowledge to safely remove the llama from your property.

Thank you!

Questions ?

Commercial solutions

1 **Enterprise Hub**
Private and securely hosting and tooling for enterprise collaboration on models, datasets and compute

Spaces
Build, host, and share your ML applications using Spaces in just a few minutes.
Get started with free compute and upgrade easily as needed.

Inference endpoints
Deploy and call your models as APIs in a few clicks

Models
500k+ open models

Meta · MISTRAL AI_ · OpenAI · Microsoft

Datasets
90k+ open datasets

ML Libraries
Transformers, Tokenizers, Autotrain and many more free libraries and tools

Hardware
Specialized hardware optimized for production

habana · intel · AWS Inferentia · GRAPHCORE

2

Expert acceleration program
World-class experts to support your ML team reach impact