K2-V2: A 360-Open, Reasoning-Enhanced LLM

K2 Team, Institute of Foundation Models*

Mohamed bin Zayed University of Artificial Intelligence

Overview

We introduce K2-V2, a 360-open LLM built from scratch as a superior base for reasoning adaptation, in addition to functions such as conversation and knowledge retrieval from general LLMs. It stands as the strongest fully open model, rivals open-weight leaders in its size class, outperforms Qwen2.5-72B and approaches the performance of Qwen3-235B. We actively infuse domain knowledge, reasoning, long-context, and tool use throughout the training process. This explicitly prepares the model for complex reasoning tasks. We demonstrate this potential using simple supervised fine-tuning, establishing a strong baseline that indicates significant headroom for advanced alignment. By releasing the full training history and data composition, we maximize the effectiveness of continuous training, a key open source production scenario. We release the model weights and signature LLM360 artifacts, such as complete training data, to empower the community with a capable, reasoning-centric foundation.

Model Weights	`collections/LLM360/k2-v2`	Vibe-Eval Gallery	`spaces/LLM360/k2v2-vibe`
Training Log	`llm360/K2-V2`	Mid-training Data	`datasets/LLM360/TxT360-Midas`
Evaluation	`llm360/eval360`	Pre-training Data	`datasets/LLM360/TxT360`
Training	`llm360/k2v2_train`	SFT Data	`datasets/LLM360/TxT360-3efforts`

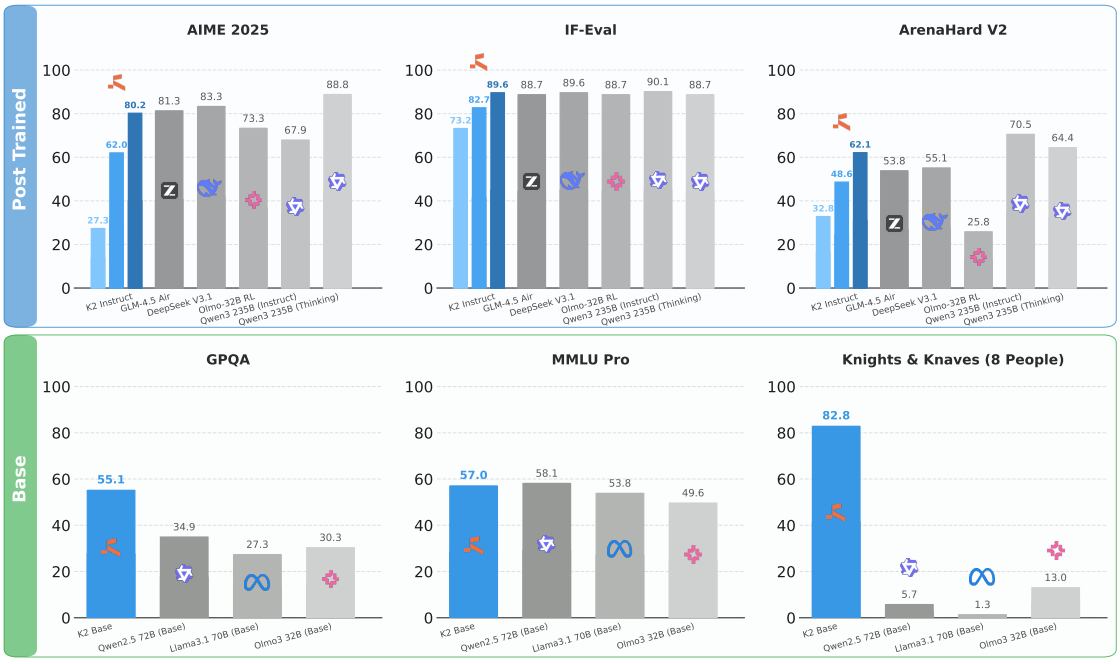

Figure 1: K2 delivers a strong base model for reasoning (bottom). Post-SFT, K2 is augmented with three reasoning levels controlling amount of test-time compute (three K2 Instruct bars; lower reasoning produces shorter responses suitable for instruction following and chat). Across use cases, K2 is competitive with models of larger size (top).

*Correspondence to: {hector.liu, eric.xing}@mbzuai.ac.ae

Contents

1 Introduction

We introduce K2, the best fully open-source pretrained large language model (LLM) to date, and ranks competitively against the best open-weight models of its class. As the latest base model in the LLM360 family (Liu et al., 2023; Tao et al., 2024; Liu et al., 2025c; Cheng et al., 2025a), K2 features a dense architecture with 70 billion parameters. Beyond standard competencies like knowledge and conversation, K2 provides advanced capabilities, including long context consistency, deep mathematical knowledge, and reasoning behaviors. These serve as foundational building blocks that enable sophisticated downstream use cases, such as solving complex math problems and executing agentic workflows. Our supervised fine-tuning experiments illustrate this potential, confirming that K2 is a robust base primed for further post training. Continuing the 360-open-source tradition of LLM360, we release K2 along with its complete development transparency – a documentation of the entire development-cycle, and a full suite of artifacts along the way, providing the community with an end-to-end blueprint and toolkit for building and evolving a state-of-the-art AI model.

The urgency of providing a strong open base LLM stems from a growing divergence in the industry: while model capabilities have surged, the transparency required to scientifically understand and continous adaptation has significantly diminished. Specifically, as frontier models become more powerful, their development details, specifically data compositions, infrastructure configurations, and training dynamics, have increasingly become "black boxes". Crucially, we acknowledge that in this landscape, merely being "open" is not enough; models that lack sufficient power offer limited utility for both research and production. For example, reasoning tuning on weak base models yields little scientific insight (Wang et al., 2025; Cheng et al., 2025b).

This reality underpins the design philosophy of K2, we ensure the model is not only transparent but also practically usable and competitive with state-of-the-art models of similar scales. Our supervised fine-tuning experiments confirm its potential, illustrating that K2 is a robust base primed for further post-training. Consequently, we position the K2 project to contribute to the field from two distinct and critical angles: model utility and acceleration of open science.

K2 provides unique model utilities. K2 provides practitioners with a uniquely adaptable base. K2 establishes itself as the best-in-class performer for its size (Fig. 1), validating the effectiveness of our open recipe. Crucially, the model delivers the advanced capabilities introduced above, such as long context consistency and reasoning. Further, its utility extends beyond raw performance, and lies in its adaptability. For open models, continuous training and domain adaptation are essential. By documenting our mid training stages and data compositions, we remove the guesswork often associated with continuous training. This full transparency allows practitioners to implement targeted strategies, such as data replay (Ibrahim et al., 2024), to maintain the model's base capability. For production environments, this ensures the model can be adapted easily without the trial-and-error required by closed models.

K2 accelerates open science for AI. K2 offers a highly capable testbed for academic research. Unlike weaker baselines, K2 meets the performance threshold required to investigate complex cognitive behaviors, such as mathematical reasoning and agentic workflows. By serving as a transparent alternative to the limited set of capable open models currently available, we enable researchers to rigorously dissect how these high-level abilities emerge.

Furthermore, we bridge the significant information gap between model builders and evaluators. Without access to training corpora, it is difficult to trace model strengths and weaknesses back to their origins or to isolate the variables that drive performance. This transparency is particularly critical for post-training

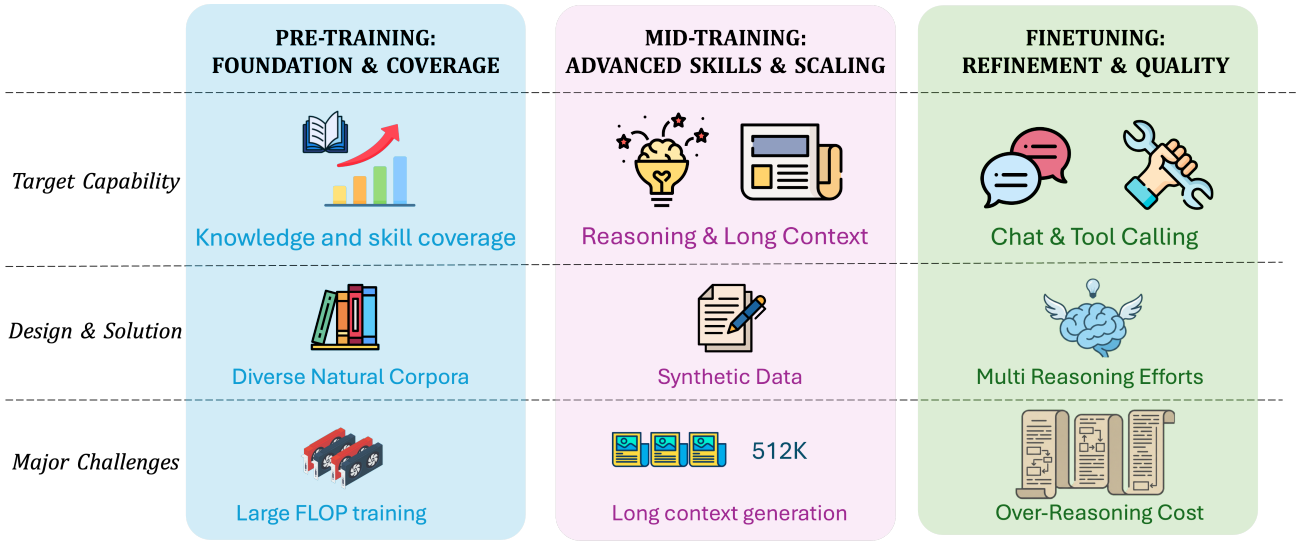

Figure 2: The training phases of K2 are designed to progressively enable specific capabilities. Each phase introduce different types of challenges that we have to address.

research; the efficacy of techniques such as Reinforcement Learning often varies significantly in the literature, which is likely due to unobserved distributions in the base model's pre-training data. By fully releasing the training data, K2 allows the community to gain a deeper understanding of these dependencies, linking post-training outcomes directly to pre-training signals.

Finally, the K2 project continues a track record of promoting scientific progress through high-impact contributions. Building on our development of TxT360 (Tang et al., 2024), a widely adopted dataset featuring innovative upsampling methodologies, we significantly reduce the engineering burden of repetition in the field. In this release, we expand this ecosystem with new foundational assets: *TxT360-Midas*, an open mid-training dataset designed to shape reasoning behaviors and extend context lengths; and *TxT360-3efforts*, a supervised fine-tuning dataset curated with mixed reasoning efforts. By democratizing these resources, we empower researchers to bypass the costly work of reproduction and focus their resources directly on innovation.

2 Overview

The training strategy of K2 is driven by a capability first approach. While modern Large Language Models are widely used as general purpose assistants, the landscape has shifted. OpenAI o1 (Jaech et al., 2024) demonstrated the potential of LLMs to solve complex tasks via long horizon reasoning. Subsequent frontier models have further pushed these boundaries by coupling reasoning capabilities with external tool use (OpenAI, 2025; Kimi, 2025; xAI, 2025). Inspired by these developments, we designed the K2 base to not only serve as a strong general purpose model but also provide a robust foundation for a specialized reasoning engine. To achieve this, we prioritized five foundational pillars:

1. **Broad Knowledge Coverage:** This capability establishes a robust linguistic foundation. It ensures fluency and generalization, acting as the critical substrate upon which advanced reasoning capabilities are built.
2. **Deep Domain Knowledge:** To support expert level problem solving, we target technical depth beyond general breadth. This involves integrating concentrated data from specialized fields, including mathematics, programming, and scientific literature.
3. **Consistent Long Context Capabilities:** We design the model to maintain coherence over extended

sequence lengths. While essential for document analysis, this capability plays a central role in maintaining logical consistency across long horizon reasoning chains.

4. **Exposure to Reasoning Behaviors:** Unlike traditional pipelines that reserve reasoning for fine tuning, we introduce reasoning behaviors early in the training phase. This strategy aims to establish reasoning behaviors such as planning, logical deduction, backtracking as core primitives.

5. **Robust and Native Tool Calling Capabilities:** Tool calling serves a dual purpose: it empowers general assistant scenarios, such as web search, and supports complex reasoning by delegating precise tasks, like calculation or code execution, to external environments.

To effectively synthesize these capabilities, we structure the training pipeline into three major phases, as illustrated in Fig. 2.

The pretraining phase establishes a robust knowledge base through the massive ingestion of diverse natural data. The primary objective here is effective scaling, which encompasses not only raw compute regarding token count and model size but also the maintenance of optimal training dynamics. We address these complexities through a tripartite strategy: rigorous training dynamic design and monitoring, robust infrastructure capable of supporting large scale execution, and the curation of massive token counts to drive data scaling.

In the mid training phase, the curriculum shifts toward specialization, long context extension, and exposure to reasoning behaviors. This introduces a distinct set of challenges, including computational bottlenecks, the risks of distribution drift, and the scarcity of high quality long context data. We navigate these by implementing efficient Context Parallel strategies to resolve computational constraints, engineering precise token mixtures to manage domain shift, and augmenting natural sources with synthetic generation to address data availability. Furthermore, we adopt specific learning dynamic adjustments, such as a low and constant learning rate, to ensure stability during this specialization process.

Finally, we conduct a simple supervised fine-tuning to demonstrate the potential of the model. Here, we inject tool use capabilities and finalize model behaviors by curating a focused, high value corpus. The results of this stage confirm the effectiveness of the previous phases, demonstrating the model's potential as a robust base for further post training.

Crucially, addressing the unique challenges of each phase requires synchronized design across three domains: data composition, training infrastructure, and machine learning dynamics. Consequently, the organization of this paper mirrors this lifecycle, dissecting the challenges of each phase through these lenses.

- **Section 3** details the Pre-training phase. We first describe the model architecture choice in §3.1. We discuss how we achieved effective scaling by optimizing training dynamics (§3.2), curating a data mix with most natural data (§3.3). §3.4 presents the parallelism strategies for high token throughput.
- **Section 4** describes the mid-training phase. Here, we break down our solutions regarding the challenges faced during long context and advanced model capabilities. We describe our in-house implementation for context parallelism for efficient long context scaling (§4.3), synthesizing data to solve scarcity, and designing data curriculum to alleviate the effect of domain shift (§4.2). §4.4 presents our observation during the run.
- **Section 5** presents the evaluation of the base checkpoints, presents the evaluation methodologies and challenges. With the LLM360 trademark metric curves, we show how a model develops its capability from each stage.
- **Section 6** presents a simple supervised fine-tuning recipe that demonstrate the potential of the base

models.

- **Section 7** concludes the benchmark results, analyzes the model performance across challenging tasks spanning general knowledge, STEM, logic, coding and tool-calling.
- **Section 8** provides a qualitative study of the development of the model capabilities over time.

3 Pre-training

3.1 Model Architecture

Our model follows a decoder-only dense transformer architecture consisting of 80 layers. It has a hidden dimensionality of 8,192 and an intermediate size of 28,672 in the feed-forward network. The attention module includes 64 heads, corresponding to a per-head dimension of 128. To improve memory and compute efficiency during attention computation, we adopt grouped-query attention (GQA), using 8 key-value heads shared across all query heads. Rotary position embeddings (RoPE) are applied with $\theta = 500,000$. RMSNorm is used with an epsilon of 10^{-5}.

3.2 Pre-training Recipe

Key Takeaways

- **Prioritize Sample Efficiency over Step Efficiency.** For data-limited runs (fixed token budget) optimizing model quality, do not target the Critical Batch Size (which balances step and token efficiency). Instead, operate in the *sample-efficient regime* (often $B << B_{crit}$).
- **Scale hyperparameters to preserve optimizer dynamics.** Avoid tuning learning rate (η) and weight decay (λ) in isolation. We recommend characterizing optimizer behavior via the *averaging timescale* τ_{epoch} and scaling this target proportionally to $1/\sqrt{\text{TPP}}$ (Tokens Per Parameter) to ensure consistent training dynamics across model scales.
- **Hyperparameter choices are co-designed by learning dynamics, numerical implementations, hardware constraints and curriculum design.** There are challenges to match scaling law predicted optimal hyperparameters: batch size choice is constrained by the number of accelerators, while small learning rates may interact negatively with numerical precision and additional training curriculum considerations.

We aim to minimize validation loss under a fixed token budget. To achieve this, we focus on jointly optimizing batch size (B), learning rate (η), and weight decay (λ).

Hyperparameter Selection. Following Wang and Aitchison (2025), we characterize optimizer dynamics via the effective averaging timescale $\tau_{\text{epoch}} = \frac{B}{\eta\lambda D}$, which approximates the effective number of epochs over which AdamW exponentially averages parameter updates. We derive our target τ_{epoch} by tuning λ, η, and B at a smaller scale (20 tokens per parameter or TPP) and scaling the optimal timescale proportionally to $1/\sqrt{\text{TPP}}$ based on Bergsma et al. (2025a).

We determine the batch size using a scaling law similar to Bergsma et al. (2025a). Unlike the critical batch size of McCandlish et al. (2018), which balances sample and step efficiency, we target the optimal batch size that minimizes final validation loss given a fixed token budget. This theoretical value is then adjusted to align with MLSys constraints (accelerator counts). For the learning rate, we sweep η at small scale and extrapolate using the $1/d_{\text{model}}$ scaling convention (standard practice from (Yang et al., 2021)).

Scheduler Ablation. We first evaluate several decay schedules using a 257M parameter pilot model trained on 41.48×10^9 tokens. These utilize a peak learning rate of 1.88×10^{-3}, a batch size of 0.9×10^6 tokens, and a weight decay of 0.137. Consistent with Bergsma et al. (2025b), we find negligible differences between d2z schedules on the SlimPajama validation set (Soboleva et al., 2023): 2.815 for cosine decay-to-zero (d2z), 2.845 for cosine decay-to-10%, and 2.817 for linear d2z. These pilot results suggests that the specific shape of the decay is less critical than the peak and final values.

Final Hyperparameter choices. Based on the methodologies above, we reach the `K2` run's final hyperparameters. The learning rate is set to $\eta = 1.5 \times 10^{-4}$, selected by referring to similar scale runs such as Llama3-70B (Meta AI, 2024) and validating against our small-scale extrapolations.

While the scaling law suggested a batch size of 1136 sequences (sequence length 8192), we adjust this to 1200 sequences to satisfy hardware constraints, resulting in a global token batch size of $B = 9.8 \times 10^6$. As we collected $D = 12.25 \times 10^{12}$ total tokens, we train for $T = 1.25 \times 10^6$ steps. We set weight decay to $\lambda = 0.05$ to achieve our target $\tau_{\text{epoch}} = 0.1066$.

Finally, we determine the schedule. While our pilot ablations supported a simple cosine decay-to-zero, during the main run we observed that the training loss and parameter norm became stale when the learning rate dropped below 1% of the peak. We therefore modified the schedule to decay to a lower bound of 1.5×10^{-6} and held it constant for the remainder of pre-training. This floor prevented the stagnation observed in the "long tail" of the standard cosine schedule while simplifying the transition into mid-training phases.

3.3 Pre-training Data: TxT360

We highlight our curation strategy for the pretraining mix. Unlike later mid-training stages where we introduce more synthetic data, we remain conservative during pretraining. We prioritize natural text to ensure the model is grounded in real-world distributions. Beyond this foundation, we focus on improving domain diversity and maintaining the structure of formats like tables. A key design element is our precise control over the duplication rate (Tang et al., 2024). We apply several measures, such as URL filtering and deduplication, to ensure every data entry is unique. This guarantees that any repetition in the training process is intentional.

Key Takeaways

- **Ground synthetic data in natural text.** We focus on natural data distributions to establish a strong foundation in pre-training. Synthetic data is effective when it is derived from existing documents. Around 12 trillion tokens are used in K2's pre-training run.
- **Deduplication enables precise upsampling based on quality.** Global deduplication is the necessary foundation for controlling data mix: it ensures repetition is a design choice, not an accident. Data points should be upsampled based on quality signals, the number of years a document persists is a simple but effective quality signal.

We construct our pre-training corpus from a diverse set of data sources to comprehensively cover multiple domains, including web text, academic papers, multilingual corpora, code, mathematics, and more. Most datasets are extracted from out prior curation in `TxT360` (Tang et al., 2024). A detailed breakdown of the data composition is provided below:

- `English Web` (§3.3.1): Large-scale web corpus from `TxT360` (Tang et al., 2024) web portion, comprising approximately 3.5 trillion unique words deduplicated from 99 CommonCrawl dumps. Inspired

by DCLM (Li et al., 2024a) and Fineweb-Edu (Lozhkov et al., 2024a), A high-quality subset: `TxT360-BestOfWeb`, filtered by the ProX (Zhou et al., 2024) document filtering model, is also selected.

- `Synthetic QA` (§3.3.2): Synthetic question–answer pairs generated from `TxT360` documents using `Mistral-7B-Instruct-v0.3` (Jiang et al., 2023), totaling roughly 950 billion words.
- `Papers` (§3.3.3): Academic articles from ArXiv, PubMed, S2ORC, and PhilPapers, each processed with dedicated dataset-specific preprocessing and filtering pipelines, comprising approximately 107 billion words in total.
- `Math` (§3.3.4): A collection of large-scale, math-focused corpora curated base on the `MegaMath` project (Zhou et al., 2025), `DM Math` (Gao et al., 2020; Saxton et al., 2019), `InfiMM-WebMath-40B` (Han et al., 2024), `OpenWebMath` (Paster et al., 2023), and `MathPile` (Wang et al., 2024c), consisting of approximately 344 billion tokens in total.
- `Code` (§3.3.5): Programming code primarily sourced from `RefineCode` (Huang et al., 2025a). The dataset is further augmented with two additional variants: one organized by repository and topology sorted (Guo et al., 2024); one with Fill-in-the-Middle (FIM) (Bavarian et al., 2022) applied.
- `Multilingual` (§3.3.6): Our multilingual data is focused on Arabic collected mainly from CommonCrawl and web sources. In addition, we incorporate other languages from high-quality sources like Wikipedia and EuroParl.
- `Other High-Quality Data Sources` (§3.3.7): The remaining high-quality subset of `TxT360` that does not fall into the categories above. This includes a diverse collection of well-curated domains such as Wikipedia, EuroParl, FreeLaw court opinions, StackExchange forums, USPTO patents, PG-19, HackerNews, and Ubuntu IRC logs, each processed with dedicated dataset-specific preprocessing, filtering, and deduplication pipelines.

An overview of the complete pre-training data is presented in Table 1. Note that we use word count instead of token count to eliminate the effect of tokenizer. Further details are discussed in the subsequent sections.

Tokenizer. To translate this diverse data composition into an efficient input representation, we trained a Byte-Level Byte-Pair Encoding (BPE) (Sennrich et al., 2016) tokenizer on a weighted mixture of multilingual data, specifically sampled to optimize fertility scores for English and Arabic (Fig. 3). To ensure robust character coverage and prevent fragmentation, we initialized the vocabulary with comprehensive Unicode ranges spanning Arabic, Latin, Cyrillic, Devanagari, and CJK scripts. The pre-tokenization pipeline employs a custom regex pattern designed to handle code syntax, whitespace consolidation, and punctuation isolation efficiently. Finally, the vocabulary was augmented with special tokens to support fill-in-the-middle (FIM) objectives and repository-level coding contexts. In the SFT stage we introduce additional special tokens for reasoning and tool-use (see §6.1).

3.3.1 English Web

The largest component of our pre-training corpus is mainly composed of natural English text from the `TxT360` (Tang et al., 2024) dataset, including its CommonCrawl subset (`TxT360-CC`) and the filtered high-quality variant, `TxT360-BestOfWeb`.

TxT360-CC Starting from 99 raw WARC (Web ARChive) snapshots, we apply a comprehensive data-curation pipeline consisting of text extraction, language identification, URL filtering, line-level cleaning, document-level quality filtering, personally identifiable information (PII) removal, and global deduplication. An overview of this pipeline, along with filtering percentages at each stage, is shown in Figure 4. The resulting `TxT360-CC` subset contains approximately 6.4 billion documents and more than 3.5 trillion words.

Figure 3: A comparison between the K2 tokenizer vs. a few multilingual LLM tokenizers. Our tokenizer is tailored towards English and Arabic (MSA), as shown by the lower fertility scores.

Documents average 544 words in length (range: 40 to 100,000 words), for a total corpus size of roughly 9.4 TB. The document-length distribution has a median of 330 words, with the 25th and 75th percentiles at 156 and 645 words, respectively. Further details about this curation process are available in the TxT360 blog (Tang et al., 2024).

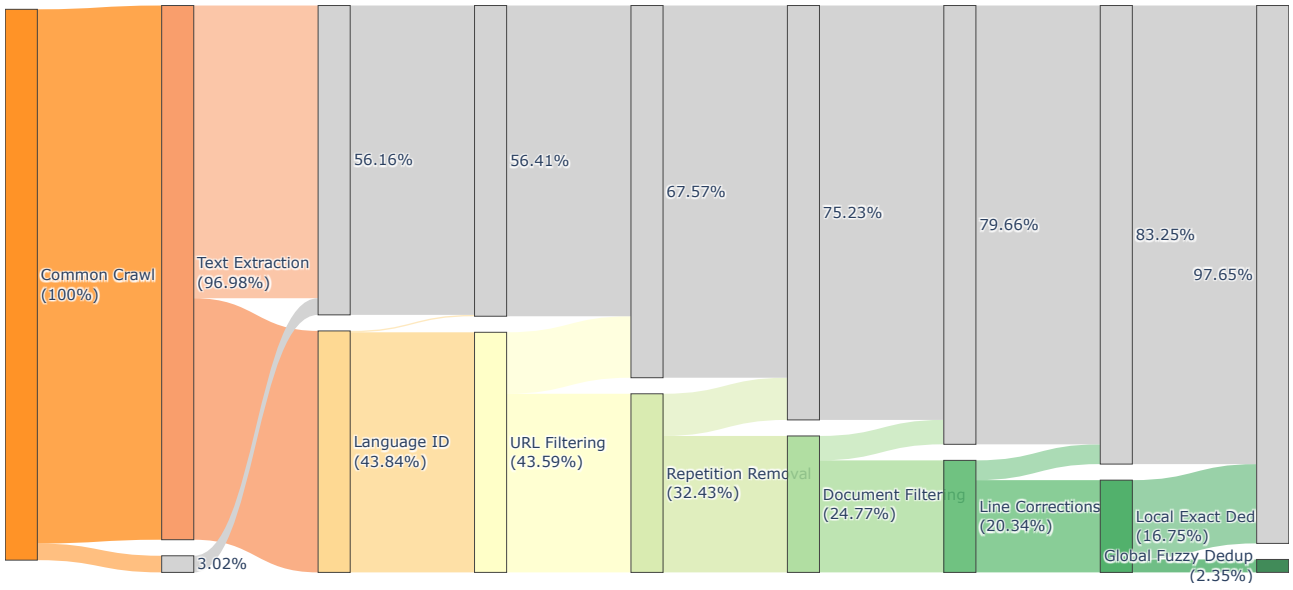

Figure 4: Overview of our CommonCrawl data-curation pipeline, showing filtering percentages (by document count) at each stage. Grey bars indicate filtered documents. *Line-level cleaning* and *PII removal* eliminate almost no documents and are omitted from the visualization. The document-level quality-filtering stage includes *Repetition Removal*, *Document Filtering*, and *Line Correction*. A *Local Exact Deduplication* step is applied prior to global fuzzy deduplication to reduce computational overhead.

TxT360-BestOfWeb. Recent "learned" curation methods use LLM-driven scoring and refinement to improve pretraining corpora. ProX ("Programming Every Example") (Zhou et al., 2024) frames data

Category	Data Source	Word Count	Avg. Words/Document	Information Cut-Off Date
English Web	CommonCrawl	3.5 T	544	July 2024
	TxT360-BestOfWeb	695 B	1.2 K	Q3 2024
Synthetic QA	TxT360-QA	950 B	1.6 K	Q3 2024
Papers	Arxiv	11.1 B	6.3 K	Q4 2023
	Pubmed Central	24 B	5.2 K	Q4 2023
	Pubmed Abstract	4.7 B	205	Q4 2023
	S2ORC Abtract	11.3 B	166	Q4 2023
	S2ORC Full text	55.4 B	7.6 K	Q4 2023
	Phil Papers	544 M	11.8 K	Q4 2023
Math	MegaMath Web	137 B	1.2 K	Nov 2024
	MegaMath Code	5.7 B	432	Nov 2024
	DM Math	2 B	18	–
	Other Math (InfiMM-WebMath-40B, OpenWebMath, Textbooks-MathPile)	24.4 B	1.2 K	Q4 2023
Code	RefineCode	265 B	340	-
	RefineCode FIM	238 B	355	-
	RefineCode Topo-Sorted	22 B	2.8 K	-
Multilingual	Jais Arabic Data	203 B	407	Q1 2024
	Europarl-Aligned	666 M	63 K	2012
Other High-Quality Data Sources	Wikipedia	15 B	302	Q4 2023
	Wikipedia-Extended	46 B	1.4 K	Q4 2023
	Europarl	670 M	9.7 K	2012
	FreeLaw	12 B	2.5 K	Q1 2024
	StackExchange	10.7 B	464	Q4 2023
	USPTO	3.8 B	742	Q3 2024
	PG-19	1.95 B	69 K	1919
	HackerNews	658 M	821	Q4 2023
	Ubuntu IRC	862 M	28 K	Q3 2024

Table 1: Comprehensive overview of pre-training data sources across major categories and subcategories. Word counts are computed using whitespace as the delimiter.

refinement as a program executed on each document, implemented by a small language model.

TxT360 Best of Web is curated by applying a document level filter on the full `TxT360-CC` set. This filter is a combination of the ProX document filter model, combined with a separate format score that captures document presentation features like headings, lists, tables, code blocks, link density, and boilerplate. This process selects high-quality, clean and well-structured pages. About 22% of the documents are retained.

Since `TxT360-BestOfWeb` is a high-quality subset of `TxT360-CC`, distinguishing it allows us to flexibly upsample this data during pre-training. We discuss these mixing details in Section 3.3.9.

3.3.2 Synthetic QA

In the pre-training phase, we introduce synthetic data conservatively. We rely on text synthesized from existing documents rather than pure generation. Document-grounded Question-Answering has a proven history. For example, Alberti et al. (2019) created a system to extract answers and generate corresponding questions. By filtering for consistency, they achieved significant gains on benchmarks like SQuAD 2.0 (Rajpurkar et al., 2018) and NQ (Kwiatkowski et al., 2019).

Puri et al. (2020) demonstrated that large generative models can synthesize high-quality (context, question, answer) triples at scale, sometimes training competitive Question Answering systems entirely from synthetic data. More recently, WRaP (Maini et al., 2024) reframes arbitrary web text directly into multiple styles, including explicit question–answer format, and utilizes it in pre-training.

For our synthetic QA data (`TxT360-QA`), we augment each document from `TxT360-BestOfWeb` with multiple, synthetic document-grounded Q-A pairs generated by `Mistral-7B-Instruct-v0.3`[1]. The pairs are appended to the end of the document text to enable simple, single-sequence training where a model reads the passage and immediately observes diverse Q–A supervision anchored in that same passage. The number of Q–A pairs per document varies, which naturally yields a mixture of easy and hard supervision signals.

```
{
  "text": "{ORIGINAL_DOCUMENT_TEXT}\n\n
          Q: {QUESTION_1}\nA: {ANSWER_1}\n\n
          Q: {QUESTION_2}\nA: {ANSWER_2}
          ......{ANSWER_N}\n",
  "meta": { original TxT360 meta }
}
```

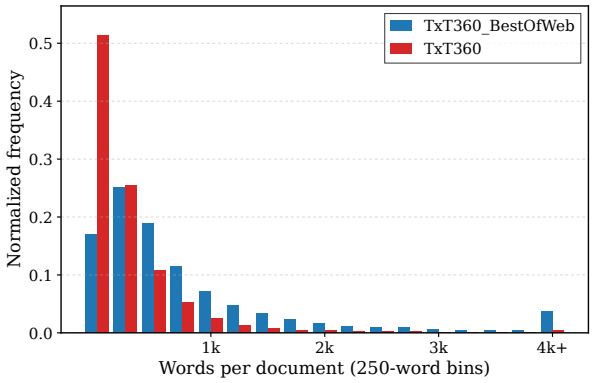

Figure 5: Normalized document-length distributions for `TxT360-BestOfWeb` and `TxT360`. `TxT360-BoW` has a higher proportion of longer documents as a bias introduced by the quality classifier.

To validate the generated Q–A pairs, we used an LLM-as-a-judge protocol (Zheng et al., 2023b): for each document we asked a judge LLM to award 1 point per pair if both question and answer were fully supported by the passage, and we reported a document-level percentage score. We apply this method on 8000 random selected documents, obtaining an average score over 95%, confirming the quality of the Q-A pairs.

We further conducted a manual qualitative check. The low quality questions tend to be generic or template-like (e.g., "what is this used for"), weakly anchored to the page, and hallucinate answers beyond local facts; The high quality questions are attribute-centric and tightly grounded, referencing concrete fields or entities (e.g., dimensions, price, warranty) so that answers can be resolved from a single sentence or short span.

3.3.3 Papers

We incorporate a collection of academic paper in our pre-training data, including ArXiv, PubMed, S2ORC, and PhilPapers. As part of our `TxT360` data project, data-specific processing and filtering were applied to each data source.

ArXiv. ArXiv contains LaTeX-formatted preprint papers in mathematics, physics, computer science, and related areas. We download raw `.tex` files from the official S3 bucket and converted them to Markdown using Pandoc[2]. We experiment with different Pandoc parsing options to preserve the structure of tables, math blocks, and section headers. Figures and reference lists were removed.

S2ORC Full Text / Abstract. S2ORC provides scientific papers in JSON format. We extract the title, abstract, section text, and metadata. Papers are kept only if they had a title and abstract, were mostly English, and have enough text after filtering. For the abstracts, we require a valid English abstract and remove documents with fewer than 20 words. We also apply a unigram-probability-based filter.

[1] Due to a bug in the generation code, we failed to generate Q-A pairs for some of the longer documents.
[2] https://github.com/jgm/pandoc

PubMed Central / PubMed Abstract. PubMed Central provides full-text biomedical papers in JATS XML format; PubMed Abstract contains abstracts in a simpler XML structure. We convert PMC using Pandoc and extracted PMA fields using BeautifulSoup[3].

PhilPapers. The PhilPapers dataset consists of philosophy papers in PDF format. We convert the files to text and apply a cleaning pipeline to remove hyphenation, strip headers and footers, normalize whitespace, and fix encoding issues. Finally, we filter the documents based on line length, letter content, and encoding artifacts to ensure document quality.

3.3.4 Math

To enhance mathematical and numerical reasoning during pre-training, we incorporate several large-scale math-focused corpora including `MegaMath` (Zhou et al., 2025), `DM-Math`, `InfiMM-WebMath-40B` (Han et al., 2024), `OpenWebMath` (Paster et al., 2023) and `MathPile` (Wang et al., 2024c), consisting of approximated 344 Billion tokens.

MegaMath `MegaMath` (Zhou et al., 2025) is a TxT360 project to extract a large-scale high-quality math data from the web, released as an open-source collection specifically designed to preserve mathematical structure across both natural language and code, as our main math corpus. We use the two subsets `MegaMath-Web` and `MegaMath-Code` for our pre-training.

`MegaMath-Web` is constructed through a math-aware curation pipeline that begins with 99 CommonCrawl WARC snapshots (`2014-15` to `2024-46`) and applies a coarse-to-fine extraction and filtering process, including URL filtering, fastText-based language identification, math-specific HTML preprocessing, a two-stage text extraction pipeline, document-level quality filtering, and large-scale MinHash-based deduplication. This pipeline yields approximately 279 B tokens of high-recall mathematical webpages with faithfully preserved equations and symbols.

`MegaMath-Code` is produced by selecting code data from eleven programming languages within Stack V2, a large public code corpus, and then applying a small language model trained to identify and retain high-quality code relevant to mathematical reasoning, logic puzzles, and scientific computation. The resulting `MegaMath-Code` split contains approximately 28.1 B tokens.

Together, these deduplicated and quality-filtered corpora are decontaminated to remove the overlap between `MegaMath` and 12 downstream benchmarks, removing 0.01% of the documents from the dataset. The resulting corpus provides a substantial volume of mathematically rich content. It includes natural language mathematical documents with preserved equations and math-related source code. More details are available in Zhou et al. (2025).

Additional Math Corpora. In addition to `MegaMath`, we incorporate several widely used open math datasets to increase coverage, diversity, and domain robustness:

- `DM Math` contains generated math question–answer pairs. We converted the dataset to JSONL and fixed a few formatting issues (e.g., byte-string artifacts and converting each row to have only one question and answer). No filtering was applied.
- `InfiMM-WebMath-40B` (Han et al., 2024): A collection of math webpages sourced from CommonCrawl (2019–2023), filtered to focus on high-quality mathematical and scientific content in both English and Chinese. After deduplication, the corpus is reduced to 15,877k documents and 27B tokens.

[3]https://pypi.org/project/beautifulsoup4/

- `OpenWebMath` (Paster et al., 2023): A math-focused dataset filtered and extracted from over 200B HTML files on Common Crawl down to a set of 6.3 million documents containing a total of 14.7B tokens. We obtain 4,316k documents and 10B tokens after deduplication.
- `MathPile` (Wang et al., 2024c): `MathPile` is a diverse and high-quality math-centric corpus sourced from high-quality data sources including Textbooks (including lecture notes), arXiv, Wikipedia, ProofWiki, StackExchange, and Web Pages, comprising about 9.5 billion tokens in total. Several sources of `MathPile` are similar to `MegaMath`, to avoid the overlap, we only select the *Textbooks* subset of `MathPile`, which includes approximately 4,000 documents totaling 0.042B tokens.

Finally, we conducted exact deduplication across the math datasets.

3.3.5 Code

We used RefineCode (Huang et al., 2025a) as the base code dataset and produced two additional variants from it: a Fill-in-the-Middle (FIM) version and a topologically sorted version. All three splits originate from the same underlying collection of GitHub repositories.

RefineCode. This is the original cleaned code corpus. Each entry corresponds to a single source file extracted from a GitHub repository. No structural modifications were made to the code itself. For details about the data collection methodology, see Huang et al. (2025a).

RefineCode FIM. This split is created from RefineCode by converting each file into a Fill-in-the-Middle (FIM) format (Bavarian et al., 2022). A span from the file is selected as the "middle" section, and the remaining text is split into a prefix and a suffix. These parts are then rearranged using special tokens such as `<|fim_prefix|>`, `<|fim_middle|>`, and `<|fim_suffix|>` to mark their positions. The token order is chosen randomly among standard FIM patterns, so the same file may appear in different arrangements compared to RefineCode. Only the layout changes; the underlying code content remains the same.

A shortened example of an entry is:

```
<|fim_prefix|>{% extends 'base.html.twig' %}

{% block title %}Modifier un utilisateur{% endblock %}
{% block body %}
  <div class="container">
    {{ include('user/_form.html.twig') }}
  </<|fim_suffix|> "{{ path('user_index') }}">

<|fim_middle|>div>
    <a href=
```

RefineCode Topo-Sorted. This variant groups files by GitHub repository and sorts them based on the topological structure induced by code imports. Our goal is to ensure that dependencies appear earlier in the sequence so that variables and classes are defined before they are used. We followed the procedure described by Guo et al., 2024. First, we identified import statements using regular expressions and built a dependency graph for each multi-file repository. We then applied topological sorting to determine the file order. For isolated files or cases where the graph did not fully determine an order, we preserved the repository's original file listing. Finally, we concatenated the ordered files into a single document, inserting a comment with the filename before each code block.

These three splits represent the same underlying code data in different forms: the unmodified file-level version, an FIM-transformed version, and a topological-sorted version.

3.3.6 Multilingual

We include a substantial amount of Arabic text in our pretraining corpus, primarily sourced from Common Crawl web scrapes extending the data used to train Jais model (Sengupta et al., 2023). In addition to web sources, we incorporate high-quality multilingual resources such as Wikipedia and Europarl, ensuring broader linguistic and domain coverage.

Arabic Our Arabic dataset comprises approximately 498 million documents, totaling over 203 billion words. On average, each document contains about 407 words, with document lengths ranging from 2 to 27 million words. The document length distribution shows a median of 94 words, with the 25th and 75th percentiles at 38 and 445 words, respectively.

Europarl-Aligned. This dataset is a multilingual corpus derived from the Europarl v7 proceedings (Koehn, 2005), consisting of parallel texts from European Parliament debates. This version extends the original Europarl collection by aligning English source sentences with their corresponding parallel translations across multiple European languages (e.g., Finnish, German, Dutch, Greek, Italian, French, Portuguese, Swedish, Danish, Spanish). The alignment process preserves semantic equivalence across languages.

The original Europarl data[4] is converted into `JSONL` format for efficient filtering and multilingual concatenation. Each record combines the same parliamentary segment across languages into a unified structure, and the order of the language is randomized (recorded by a metadata field). For example:

```
{
  "text": "# Finnish
[Finnish content]

# German
[German content]

# Dutch
[Dutch content]
...",
  "meta": {
    "language": "fi-de-nl-el-it-fr-en-pt-sv-da-es",
    "src_file": "ep-00-01-17"
  }
}
```

3.3.7 Other High-Quality Data Sources

Besides the data set mentioned above, we adopt all the remaining high-quality data sources from `TxT360`, including FreeLaw, StackExchange, USPTO, PG-19, HackerNews, and Ubuntu IRC. These datasets are different from general web text because they are published in structured formats such as XML, LaTeX, PDF, or CSV. These sources contain cleaner writing, clearer structure, and more consistent formatting than typical web pages, so they require different processing steps.

Across all curated datasets, we apply a small number of shared filtering rules: (1) removing non-English documents when the dataset is expected to be English; (2) dropping very short documents or too high

[4] http://www.statmt.org/europarl/v7/europarl.tgz

non alphanumeric characters; (3) removing documents with very low unigram log probability. For some datasets, only a subset of these filters was needed. Many filters and their thresholds have been set after detailed review by our team. Additional dataset-specific cleanup is added only when required by the source format. Details can be found in our TxT360 blog (Tang et al., 2024).

Wikipedia. We use the official Wikimedia snapshot from HuggingFace and convert the parquet files to JSONL. This includes all 323 subsets representing different languages.

Wikipedia Extended The Wikipedia Extended dataset is built from standard Wikipedia articles, with each article expanded by adding information from the pages it links to. For every source article, the abstract of each outgoing linked article is appended after the original text. Links and abstracts are obtained via DBpedia (Auer et al., 2007). Each record contains the concatenated text, for example:

```
{ORIGINAL ARTICLE}

{First Link Title}
"{First Link Abstract}"

{Second Link Title}
"{Second Link Abstract}"
...
```

This produces a longer document that includes the main article followed by short summaries of related pages, providing additional data for training long-context capabilities.

EuroParl. EuroParl contains parliamentary debate transcripts with XML-like tags. We remove HTML tags, filtered out very short lines, and keep only lines associated with speaker tags. Other structural tags such as paragraph and chapter markers are discarded.

HackerNews. HackerNews includes threaded discussions. We retrieve posts using the official API, construct threads from story IDs, remove system messages, and simplify comment hierarchies.

USPTO. USPTO patent documents appear in different formats depending on the publication year. We use format-specific extraction scripts and remove documents with fewer than 50 words.

FreeLaw. FreeLaw provides legal case text across multiple HTML and XML representations. We extract text from all relevant fields, normalize whitespace and line breaks, and remove artifacts such as form-feed characters. Local deduplication remove a large number of repeated entries across dumps.

StackExchange. We reconstruct full conversation threads from the StackExchange XML dataset. Using post IDs, we group questions with corresponding answers, comments, all code blocks.

Ubuntu IRC. Ubuntu IRC logs capture chat-style conversations. We remove system messages, anonymize usernames with generic placeholders, and clean message prefixes.

PG-19. PG-19 consists of public-domain books. We reduce long runs of whitespace, normalize line breaks, remove hyphenation across line breaks, and remove delimiter markers.

3.3.8 Pre-training Data Analysis

Topic analysis. To obtain a high-level view of the content distribution in our corpus, we perform a topic analysis on the Common Crawl portion of TxT360. We apply BERTopic (Grootendorst, 2022) to cluster the documents into 17 broad topic groups. The clusters are produced by first generating 128 initial groups from a seed subset and then manually merging them into semantically coherent categories. For each topic, we aggregate several metadata-based metrics and inspect their averages to understand how different types

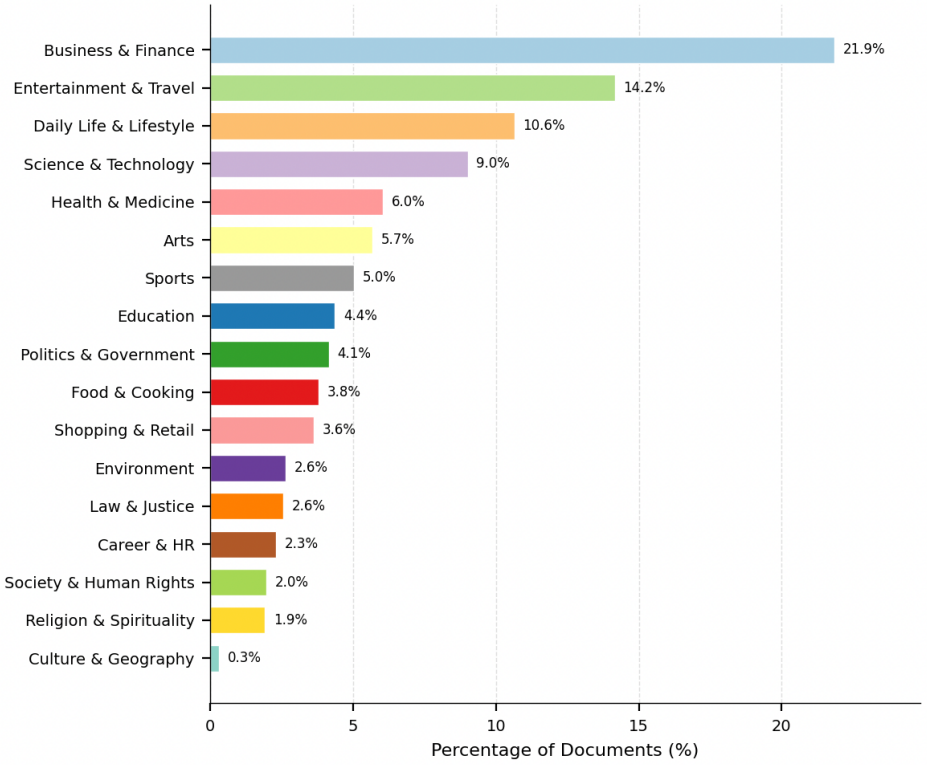

Figure 6: Topic distribution of documents from the Common Crawl portion of TxT360.

of documents vary across simple structural and lexical attributes. A summary of the topic distribution is shown in Figure 6. Additional details about the clustering procedure and metric comparisons can be found in our blog post (Tang et al., 2024).

Global Near-Deduplication. Near-deduplication provides several benefits for language model (LM) pretraining, the most important being controllable upsampling. With unique data, we gain fine-grained control over the training distribution. Other advantages include preventing train–test overlap, which mitigates benchmark and development set contamination, and reducing the impact of the double descent phenomenon, where repeated data can cause test loss to increase midway through training (Hernandez et al., 2022). Deduplication also lowers the risk of memorization (Lee et al., 2022). By implementing deduplication and selective upsampling, we can directly control the pretraining data distribution rather than relying on the inherent distribution of the source datasets.

We start with 61.8 TB of filtered and compressed documents, totaling approximately 48.83 billion entries. First, we perform exact deduplication using a Bloom filter with a capacity of 1 billion and a false-positive rate of 0.001. This process reduce the corpus to 40.21 billion documents, eliminating about 17% as exact duplicates. The Bloom filter's constant memory usage makes this stage efficient and reduces the workload for subsequent near-deduplication.

For global near-deduplication, we adopt a methodology similar to prior work such as SlimPajama (Soboleva et al., 2023), but scale it to encompass the entire dataset, including 99 Common Crawl snapshots and curated data. The process involves generating `MinHash` signatures for each document using the `datasketch` library with 128 permutations.

Before signature generation, the text of each document is cleaned:

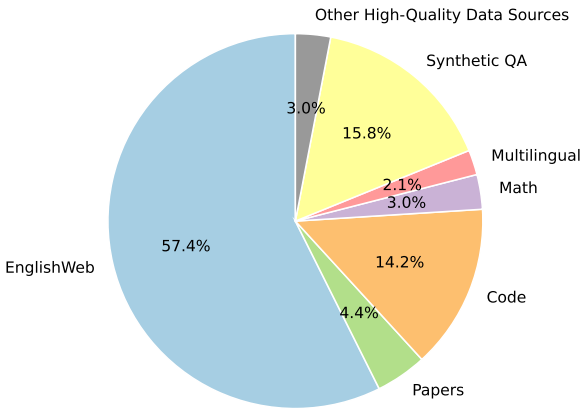

Figure 7: Overview of the domain distribution of the pre-training data mix. Majority comes from English Web.

- Stripping leading and trailing whitespace
- Converting all text to lowercase
- Removing punctuation, consecutive spaces, newlines, and tabs

From the cleaned text, we generate 13-grams as features to construct document signatures, each represented as a `MinHash` object. These signatures are then compared to identify near-duplicate documents, which are clustered to retain a single representative document while marking the others for deletion. When selecting representatives, we prefer curated documents over Common Crawl documents, and more recent documents over older ones.

Throughout all stages of deduplication, we use the `Dask` distributed processing library extensively for scalability and efficiency. In addition, we maintain detailed statistics for each matching cluster during the final deduplication stage.

Analysis of Near-Duplicate Clusters. Analysis of the near-duplicate clusters reveals distinct patterns between the component sizes. Smaller clusters tend to exhibit greater overlap in their `MinHash` bands. The smallest components are typically near-exact duplicates—documents that differ only slightly due to minor formatting or textual variations—and were therefore not captured during the earlier stage of local exact deduplication.

We observe distinct patterns based on cluster size. Smaller clusters typically reflect the maintenance of a living document where content, such as a staff list, is modified incrementally over the years. However, the pattern shifts toward mass replication for clusters with 1,000 or more documents. These large clusters usually contain generic templated text, including disclaimers or terms and conditions, which commercial entities use to standardize legal and operational messaging across platforms.

3.3.9 Data mix

Following Tang et al. (2024), we upsample documents based on their natural distribution of duplicates counts. However, since duplication is only an indirect indicator of quality, we upsample documents to a few predefined levels rather than using their exact count. Specifically, we set the upsampling weight to 3 for documents with 2 to 5 duplicates, 5 for those with 5 to 100 duplicates, 8 for 101 to 1000 duplicates, and 10 for documents with over 1000 duplicates. These values are selected heuristically and informed by preliminary small-scale experiments. For non-CommonCrawl data sources, we assign a weight of 2 if the document appears more than once. A high level domain distribution is depicted in Figure 7. The detailed data distribution is shown in Table 2.

Category	Data Source	Duplication Bucket	Proportion (%)
English Web	TxT360-CC (TxT360-BestOfWeb excluded)	1-1	14.291033
		2-5	9.289018
		6-10	2.994187
		11–100	3.449416
		101–1000	0.099270
		> 1000	0.003080
	TxT360-BestOfWeb	1–1	5.722252
		2–5	3.934343
		6–10	5.072045
		11–100	6.801969
		101–1000	0.216884
		> 1000	0.004300
Synthetic QA	TxT360-QA	1–1	3.777652
		2–5	6.426003
		6–10	2.361838
		11–100	3.156823
		101–1000	0.098637
		> 1000	0.001932
Papers	ArXiv, PubMed, S2ORC, and PhilPapers	1–1	3.835154
		others	0.575500
Math	MegaMath Code	all	0.174502
	MegaMath Web	all	1.734297
	DM Math	all	0.105578
	Other math (InfiMM-WebMath-40B, OpenWebMath, Textbooks-MathPile)	all	0.977156
Code	RefineCode	all	7.102195
	RefineCode FIM	all	0.528593
	RefineCode Topo-Sorted	all	0.583103
Multilingual	JAIS Arabic Data	all	2.137777
	Europarl-Aligned	all	0.010014
Other	Wikipedia	1-1	0.893301
		others	0.005294
	Wikipedia-Extended	all	0.876997
	Europarl	1–1	0.030053
		others	0.000439
	FreeLaw	1–1	0.297616
		others	0.168791
	Hackernnews	1–1	0.023127
		others	0.000079
	PG-19	1–1	0.033099
		others	0.045906
	StackExchange	1–1	0.475941
		others	0.010132
	Ubuntu IRC	1-1	0.049009
	USPTO	1-1	0.111890
		others	0.000680

Table 2: Pre-training data mix details. Documents are organized in "Duplication Bucket" based on number of duplicates they have. The bucket is used as one of the features to determine the mixing weights.

3.4 Pre-training Infrastructure

Key Takeaways

- **Parallelism Tradoff.** With sufficient GPU memory and a small global batch size, one should consider disabling pipeline parallelism to achieve higher training throughput.
- **Fault Tolerance System.** Our fault tolerance and auto-mitigation systems allow our model to quickly recover from training incidents.

To scale training for our largest models, we use hybrid parallelism, which combines four parallelism methods to shard the model. This approach efficiently distributes computation across many GPUs and ensures that each GPU can hold the model parameters, optimizer states, gradients, and activations in its HBM. In `K2`, we use a combination of tensor parallelism (TP), sequence parallelism (SP), and data parallelism (DP). Our pretraining framework is based on Megatron Core (Shoeybi et al., 2019) and uses Transformer Engine as its backend (NVIDIA Corporation, 2025).

3.4.1 Hardware and Cluster Setup

The `K2` model is trained on an H200 GPU cluster, with each server equipped with eight NVIDIA H200 GPUs operating at 700 W TDP and 141 GB of memory. Each GPU node provides eight InfiniBand HCAs running at 400 Gb/s in a fully non-blocking fabric. To improve fault-tolerance and reliability of the training job, we develop an in-house tool, `submitter`, which continuously detects hardware and system faults and enables automatic fault-tolerant job submission, termination, and recovery over Slurm.

3.4.2 Parallelism and Training Infrastructure

Trade Offs Among Parallelism Strategies. Similar to what we discussed in (Liu et al., 2025c), DP partitions the input data batch evenly among workers. At each iteration, each worker computes gradients over its assigned batch, and these gradients are synchronized among workers before the next iteration. DP requires each worker to hold a complete model replica, so it cannot be used to train models with very large parameter counts. Fully Sharded Data Parallelism (FSDP) can replace the DP dimension to further save GPU memory footprint at the cost of marginal communication overhead, but when we set up the training system, Megatron Core did not have full support for FSDP (Zhao et al., 2023). In TP, the weights of two consecutive layers are partitioned first along the row dimension (input dimension) and then along the column dimension (output dimension) (Shoeybi et al., 2019). TP removes the need for synchronizing the intermediate output of the first layer but introduces heavy cross device communication afterward. SP can be applied along the TP dimension to further reduce GPU memory consumption, although it increases communication (Korthikanti et al., 2023). In PP, layers are placed across GPUs, and the mini batch is split into microbatches. These microbatches are then pipelined in the forward and backward passes. PP requires less communication than DP and TP but suffers from idle GPU time, for example pipeline bubbles (Huang et al., 2019; Narayanan et al., 2019).

Heuristics for Tuning Hybrid Parallelism Strategy. Similar to what we discussed in Liu et al. (2025c), a good parallelism strategy follows the pattern in which TP is mapped to intra-node GPUs, PP and its associated microbatch sizes are then tuned to satisfy GPU memory constraints, and all remaining GPU degrees are assigned to DP (or FSDP). In `K2`, we start from the K2-V1 parallelism strategy. Compared with prior work, we now use H200 which has larger GPU memory (141GB). *Given larger GPU memory capacity, which parallelism dimension should be removed first?* Intuitively, TP should be reduced before PP, since its communication cost is much higher compared to PP. However, in practice, because TP can be used together with SP, its memory savings are much more significant than those from PP. This is especially

true for K2, where we pre-trained using an 8K context length, making SP essential. Another reason to disable PP rather than TP is that we use a relatively small global batch size, which results in a per-GPU batch size of 3. Such a small per-GPU batch size almost always causes a large fraction of pipeline bubbles, which hurts utilization. Thus, for K2 pretraining, we disable PP and use only TP, SP, and DP. To mitigate GPU memory spikes, we also enable activation recomputation (Korthikanti et al., 2023).

The Optimized Hybrid Parallelism Strategy. We use a carefully tuned strategy that combines DP, TP, and SP for K2 pretraining. More specifically, we use 8-way TP (with SP enabled along the TP dimension). The tuning procedure follows the heuristics aforementioned, as discussed above. In stage 1, the global batch size is 1200, and we use a mini batch size of 3 per DP group (i.e., no microbatching is used). We apply activation recomputation to reduce memory consumption, using full-layer activation checkpointing uniformly across layers and treating each transformer layer as the recompute unit (using `-recompute-granularity full`, `-recompute-method uniform`, and `-recompute-num-layers 1` in Megatron-LM). BF16 precision and FlashAttention 2 are enabled to increase training speed (Dao, 2023).

3.5 The Pre-training Run

Key Takeaways

- **Intervene based on spike width, not just height.** Restart-based interventions incur computational costs. We therefore distinguish between *narrow spikes* (transient instabilities that often self-correct) and *wide, malignant spikes* (which indicate diverging dynamics). We implement automated rollbacks only for the latter to avoid unnecessary restart overhead.
- **Expect and tune for early-phase instability.** Instability is structurally concentrated in the first 30% of training. Tuning the optimizer timescale (τ_{epoch}) specifically for this phase can be helpful.
- **Extremely low learning rates stall progress.** While Decay2Zero is effective in some settings, we find that learning rates below 1.5×10^{-6} may have hit numerical precision limits. We observe the parameter norm ceases changing and may have stopped learning.

3.5.1 The Effect of Decay2Zero

Decay2Zero learning rate schedules are strong both for the full schedule (Bergsma et al., 2025b) and in a short additional annealing phase at the end (Groeneveld et al., 2024). However, few results have been published at K2's scale (70B and 10x Chinchilla over-trained regime). In our actual run, we observe that when the learning rate drops below 1.5×10^{-6}, the parameter norm ceases to change—a symptom we hypothesize to indicate training stagnation. Figure 8 shows this in detail. From steps 1.23e6 to 1.25e6, the parameter norm change of the D2Z approach slows down significantly, while that where it is fixed at 1% continues to decrease. Since we plan to conduct subsequent mid-training stages, we decide to use a non-zero learning rate to conclude pre-training.

3.5.2 Training monitoring

We develop an automated monitoring system to ensure stability and reliability during training. The system continuously tracks key metrics and:

- Detects abnormal events, such as significant spikes in the training loss curve.
- Automatically rolls back to the most recent stable checkpoint, relaunches the job using the `submitter` tool discussed previously, and backs up relevant checkpoints.
- Sends notifications for both detected anomalies and recovery actions taken.

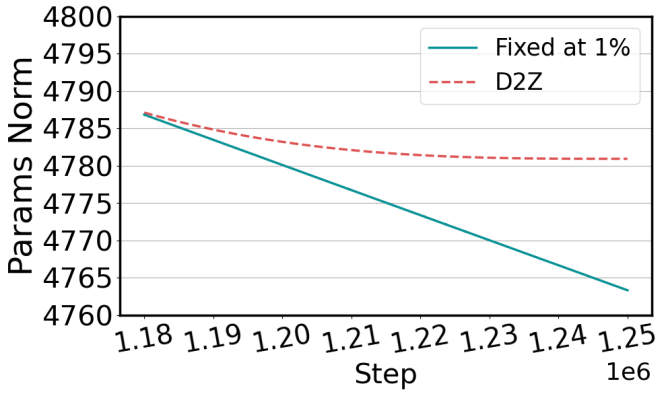

Figure 8: Params norm comparison: D2Z vs constant 1% for the tail. We see that the params norm for D2Z flattens, while that fixed at 1% in the tail continues to drift downward.

Abnormal events detection. To ensure stable long-horizon training, we implement an abnormal events detection system designed to identify and react to severe loss spikes as early as possible. We adopt a deterministic, dual-threshold, sliding-window algorithm that distinguishes persistent divergence from transient noise.

- w: width of the historical window (persistence width).
- $v_{\min}$: minimum value of the monitored metric within the window.
- $v_{\max}$: maximum value of the monitored metric within the window.

At every training iteration t, the system examines the most recent w iterations and evaluates the loss trajectory against two scalar thresholds: a **sustained floor** $T_{\min}$ and a **severity peak** $T_{\max}$. An abnormal event, as known as a loss spike, is triggered if and only if the following conditions hold simultaneously:

- **Persistence condition:** $v_{\min}(t) > T_{\min}$. This ensures the loss has not re-entered a safe range for the entire duration of the window, filtering out single-step fluctuations.
- **Severity condition:** $v_{\max}(t) > T_{\max}$. This ensures the loss has reached a critical magnitude, preventing false positives caused by high-but-stable plateaus.

We operate the detector in two sensitivity tiers. The **alert tier** uses a narrower window and lower thresholds to identify early warning signs and triggers a level-one Slack notification. The **restart tier** uses a wider window and stricter thresholds to confirm sustained divergence; exceeding these thresholds triggers a level-two Slack notification with automatic restart of the training job.

When a restart-tier event is detected at iteration t_{spike}, the system rolls back to the most recent fully committed checkpoint based on a fixed checkpoint interval I. The rollback iteration is computed as:

$$t_{\text{rollback}} = \left\lfloor \frac{t_{\text{spike}}}{I} \right\rfloor \times I,$$

ensuring recovery always begins from a known-good checkpoint on disk. Figure 9 shows an example of the notification mechanism in action.

3.5.3 Training Dynamics & Stability

In this section, we present training dynamics observations. We focus on three key signals: loss spikes, gradient norms, and parameter norms. Monitoring these metrics allows us to detect potential instabilities

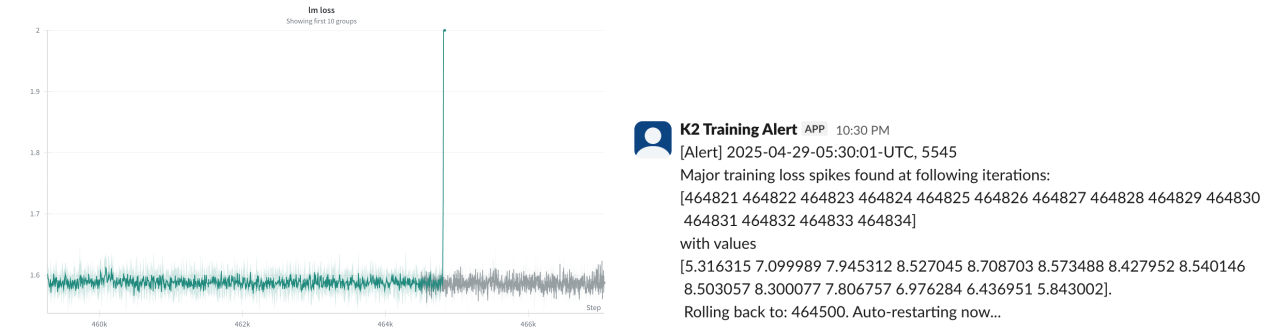

Figure 9: a) A severe loss spike appeared at around 464,000 step. b) The spike was immediately detected. The job was auto-restarted and a notification was sent through Slack

or irregularities during the run. The following analysis details how these values evolved from the start of pre-training to the final steps.

Loss Spikes. Loss spikes (Molybog et al., 2023; Takase et al., 2025) are sharp jumps in training loss that interrupt an otherwise stable loss curve. In Large Language Models, they often stem from a mix of aggressive learning-rate schedules and first or second-moment estimator lag in Adam-style optimizers.

To help visualize the dynamics, we quantify spikes using a robust local z-score that measures how abnormal the current loss is relative to its recent history. For loss values y_t over training steps t, we compute a rolling median m_t over a trailing window $W(t)$ (set to approximately 1% of the full training duration). To robustly estimate local variability while accounting for the non-stationary trend of the loss curve, we compute the median of the absolute deviations from the rolling median within the window:

$$\mathrm{MAD}_t = \mathrm{median}_{s \in W(t)} \left| y_s - m_s \right|, \tag{1}$$

where m_s is the rolling median at step s. We then define the local z-score z_t as:

$$z_t = \frac{y_t - m_t}{1.4826\,\mathrm{MAD}_t}. \tag{2}$$

Points with $z_t > 5$ are flagged as loss spikes. Intuitively, this score measures how many typical fluctuations above the local baseline a specific step represents, capturing abrupt deviations while adapting to the non-stationarity of the loss curve. This formulation is the loss-domain, locally adaptive analogue of the Gradient Spike Score (GSS) introduced in Huang et al. (2025b), which applies a similar standardized score to gradient norms.

To comprehensively evaluate training stability, we analyze both the final successful model trajectory (the "clean" run) and the divergent branches that were abandoned during training. We reconstruct the clean run by stitching together the chain of checkpoints that led to the final model. The discarded segments are then treated as candidate failure events. We filter these abandoned runs, retaining only those that exhibit a significant deviation from the clean run's baseline prior to termination. In our analysis, both transient spikes within the clean run and fatal spikes in the abandoned runs are evaluated against the robust statistics (m_t and MAD_t) derived from the successful trajectory, ensuring a consistent definition of anomaly across all data.

Figure 10 illustrates that the vast majority of loss spikes occur during the first 40% of training, with a particularly high frequency in the first 20%. We address these instabilities via two mechanisms. First, we observe that poorly tuned τ_{epoch} values (Wang and Aitchison, 2025) exacerbate spike frequency at small

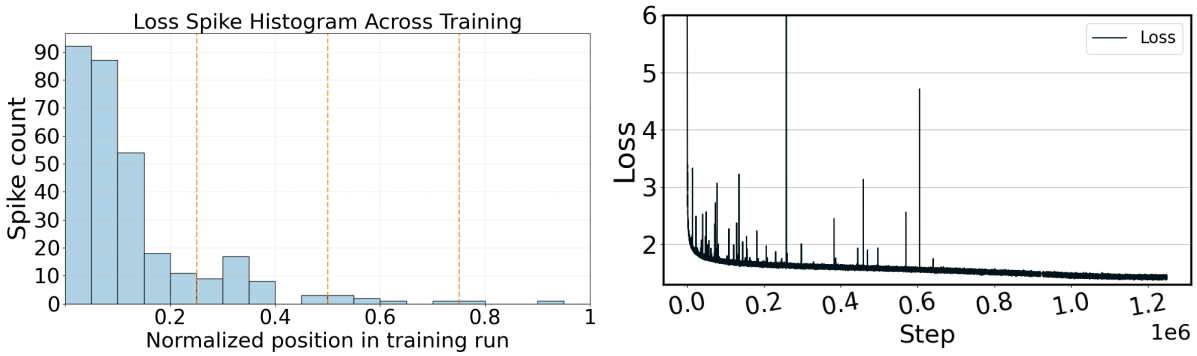

Figure 10: a) Histogram of Loss Spikes (including those erased via restart) across pre-training. We see that the spikes are highly concentrated in the first 40% of training. See Section 3.5.3 for our definition of loss spikes. b) Pre-training loss curve (excluding regions erased via restart). Again, we see that transient spikes are highly concentrated in the first 40% of training.

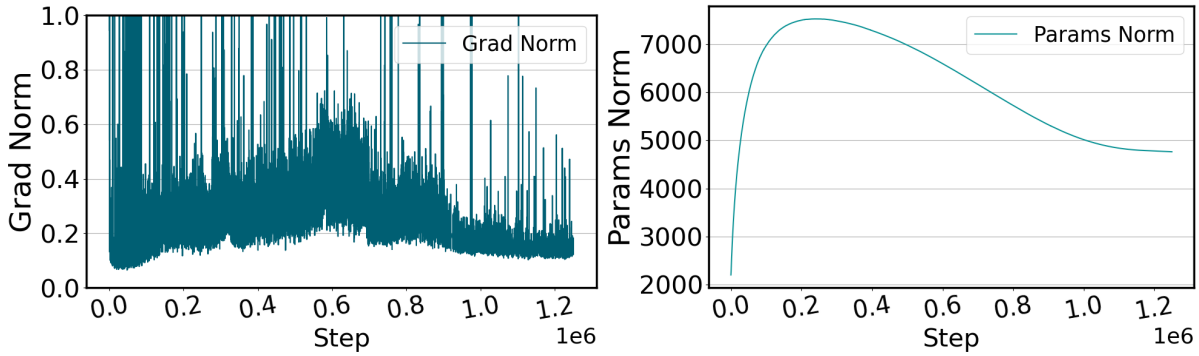

Figure 11: a) Grad norm throughout pre-training (y-axis on log scale). We see that initially the norms are low, then they drift up, peak during the middle of training and then drift back down. b) Params norm throughout pre-training. They rise initially before settling. This matches the behavior of AdamW in Wen et al. (2025) Figure 6 (middle left).

scale; we therefore explicitly tuned this hyperparameter to 0.1066 (see Section 3.2). Second, as described in Section 3.5.2, we intervene by rolling back the model state only when observing wide, malignant spikes (Liu et al., 2025d). Figure 12a illustrates a narrow spike where training was allowed to continue, whereas Figure 12b shows a cluster of wide spikes that triggered a rollback.

Gradient and Parameter Norms. As shown in Figure 11a, gradient norm spikes follow a similar temporal distribution to loss spikes, appearing more frequently in the first half of pre-training. Ignoring transient spikes, the gradient norm tends to drift upward, peaking halfway through training before decaying, though it remains elevated compared to initialization. Interestingly, the decay we observe is *not* standard behavior, which involves gradient norms increasing towards the end of training. This is discussed in detail including theoretical analysis in Defazio (2025), and Figure 6 of Wen et al. (2025) show a similar phenomena. Future investigation of our grad norm behavior is thus warranted. Figure 11b illustrates the parameter norm throughout training. After peaking near step 200k, the norm drifts downward. This decline slows in the final stages, aligning with the learning rate decay. This pattern matches the behavior of over-trained (as per Chichilla scaling law (Hoffmann et al., 2022)) AdamW observed by Wen et al. (2025).

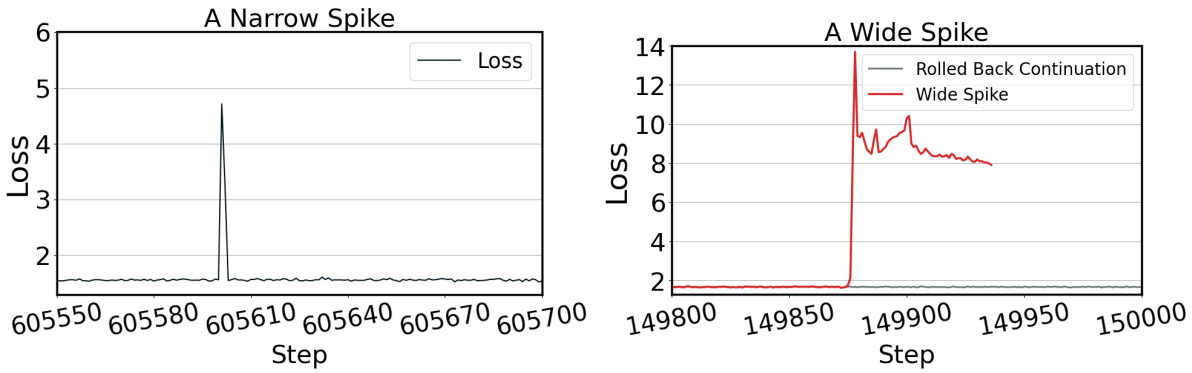

Figure 12: a) A narrow spike. We do not roll back if it does not meet the pre-defined threshold. b) Wide spikes that meet the threshold are rolled back, as we have observed that they sometimes hurt downstream task performance.

4 Mid-training

Key Takeaways

- **Training framework efficiency.** We implement features like on-the-fly tokenization and best-fit packing in our in-house training system to improve flexibility and throughput.
- **Selecting appropriate RoPE configuration is critical.** The choice of base frequency for Rotary Positional Embeddings (RoPE) is the determining factor for successful long-context extension.
- **Balancing reasoning with general capabilities.** Mid-training on synthetic thinking traces drastically alters model behavior. We find that a careful ratio of natural web data to reasoning data is required to trigger these new capabilities without degrading performance on classical pre-training benchmarks.
- **Mid-training enables test-time scaling.** Incorporating reasoning data allows the base model to demonstrate strong test-time scaling even before post-training. With a simple reasoning-inducing prompt, we achieve nearly 50% accuracy on AIME 2025 and over 90% on MATH-500, significantly outperforming similar open-source base models.

4.1 Mid-training Recipe

K2 uses a progressive context-length expansion strategy to extend the training context up to 512K tokens. Specifically, we employ a four-stage training procedure to systematically up-sample long-context data across diverse length ranges, while preserving performance on short contexts. The precision of BFloat16 is used throughout the mid-training process. The details of training length, training data size and RoPE base frequency and learning rate are shown in Table 3.

- `mid-1`: In the first stage, we aim to shift from pre-training toward higher-quality data while beginning to introduce reasoning behaviors and thinking traces. We maintain the same context length of 8192 tokens and a RoPE base frequency of 0.5M as in the pre-training stage. During this stage, given the relatively large-scale training corpus of 1.7T tokens, we adopt a cosine learning rate schedule, peaking at 1.5×10^{-5} and gradually decaying to 6×10^{-6}. The model is trained on NVIDIA H200 GPUs, with a global batch size 13,107,200 tokens.
- `mid-2, mid-3, mid-4`: In the subsequent three stages, the context lengths are expanded to 64K, 128K, and 512K tokens. To ensure stable optimization when extending to long context, we keep a constant learning rate of 6×10^{-6}, with the RoPE base frequencies set to 1M, 10M, and 10M,

Stage	Context	# Tokens	RoPE	Max LR	Min LR	LR Decay	CP
mid-1	8,192	1769B	0.5M	1.5×10^{-5}	6×10^{-6}	Cosine	1
mid-2	65,536	590B	1M	6×10^{-6}	6×10^{-6}	Constant	1
mid-3	131,072	229B	10M	6×10^{-6}	6×10^{-6}	Constant	2
mid-4	524,288	131B	10M	6×10^{-6}	6×10^{-6}	Constant	8

Table 3: Hyper-parameters in different mid-training stages. CP is the size of context parallelism.

respectively. Changing the RoPE frequency base to a larger value significantly improves long-context performance (Gao et al., 2025; Xiong et al., 2023). We also observe that scaling up the RoPE base to 10M is particularly useful when context lengths exceeds 128K tokens. To avoid short-context performance degradation, we consistently mix at least 30% short-context data at each stage. Detailed context mix is presented in Figure 14. We use the same global batch size (13,107,200 tokens) as the first stage while halving the node set. In `mid-3, mid-4`, we apply context parallelism technique to compute full attention upon long sequences.

4.2 Mid-training Data: TxT360-Midas

In the mid-training phase, we increase our reliance on synthetic data to address specific limitations in natural text. While natural data provides a strong foundation, it lacks both the density required for complex reasoning and the length needed for long-context training. Synthetic generation allows us to construct these missing pieces, producing extended, consistent and logical rich documents that are rarely found in the wild. We curate the `TxT360-Midas` dataset for this purpose, with the following design goals:

- Extend model's context length to 512k. We perform four stages of training with progressively increasing context length on high-quality mixes of synthetic and natural text data organized according to their token length.
- Prepare model for post-training by introducing synthetic reasoning data to facilitate improved performance via increased test-time computation. In addition to collecting LLM thinking traces across domains, we develop new synthetic data pipelines transforming natural user queries into over 100 reasoning behavior demonstrations such as System 1 and System 2 thinking, Red Team vs Blue Team thought experiment, detective-style abductive reasoning, and many others.
- Compensate web data limitations by introducing higher amount of synthetic data, especially data targeting mathematical ability. We collect over 250 million mathematical problems and generate solutions with thinking from open-source models.

4.2.1 LLM Thinking Traces

Enabling LLMs to utilize test-time computation by means of generating "thinking" tokens before providing the final answer has become a key factor in rapid performance improvement of recent open- and closed-source models (Jaech et al., 2024; Guo et al., 2025; Yang et al., 2025). Wang et al. (2025) demonstrated importance of including such thinking traces in mid-training to incentivize reinforcement learning.

To create sufficiently large corpora of thinking traces we collect over 250 million unique (based on exact deduplication) mathematical problems from various sources. These sources are drawn from the mathematical datasets with permissive licenses studied in the OpenThoughts project (Guha et al., 2025)[5]. To avoid contamination we only use train subsets where relevant. Problems include templated arithmetic and numeric tasks (for example, checking if a large number is prime), verification of mathematical statements, and other math problems of varying difficulty. For each problem we generate solutions using Qwen3-32B

[5] https://www.open-thoughts.ai/blog/thinkagain

with thinking (Yang et al., 2025) and GPT-OSS-120B with high reasoning effort (Agarwal et al., 2025). We augment the data with Nemotron-Postraining-Dataset-v1 (Bercovich et al., 2025; Nathawani et al., 2025) which, in addition to math, covers coding and STEM problem-solving and utilizes DeepSeek-R1 (Guo et al., 2025) to generate solutions with thinking traces. We refer to this data source as `thinking_traces`.

4.2.2 Synthetic Reasoning Behaviors

Our pipeline for generating synthetic data that demonstrates diverse `reasoning_behaviors` consists of two key components: natural user queries (Zheng et al., 2023a; Zhao et al., 2024; Don-Yehiya et al., 2025) and prompt templates for over 100 reasoning behaviors.

Our goal in exposing the model to a large number of reasoning behaviors across many contexts (including ones that may seem only loosely related to reasoning) is to encourage behavioral diversity and to elicit reasoning in domains where it is difficult to obtain verifiable problems for reinforcement learning. Our approach is motivated by Gandhi et al. (2025), who demonstrated the importance of four reasoning types in mid-training in the context of RL training on logic puzzles such as Countdown. By mid-training on many reasoning behaviors across domains, we better prepare the model for subsequent reasoning improvements, such as RLVR style training.

We generate additional data with this pipeline using behaviors outside of reasoning. For example data-science related discussion, various forms of planning, and other categories like user manuals. As with reasoning behaviors, we utilize natural user queries as "inspiration" to achieve diversity at scale and vary the behavior prompt. We refer to this data as `other_behaviors`. Both reasoning and other behaviors were generated with Qwen2.5 32B Instruct (Team et al., 2025b) and Qwen3 32B with non-thinking mode (Yang et al., 2025).

An example of the `dual_process_reasoning_system` behavior is shown below:

dual_process_reasoning_system

Prompt:

```
Generate reasoning about the following problem:

{query}

Begin by restating the problem. First provide an intuitive (System 1) assessment, then transition
to a deliberate (System 2) analysis. Show how these two reasoning modes lead to different
conclusions and how they can be reconciled.
```

Corresponding synthetic snippet for "How do I have an epic water fight?" query:

How do I have an epic water fight?

Response:

```
The problem at hand is about how to have an epic water fight. Let's break this down by first using
intuitive reasoning, and then transitioning to a more deliberate analysis.

Intuitively (System 1), an epic water fight might seem like a simple idea to execute. This could
involve grabbing as many water guns as possible, filling them up at a nearby hose, and then running
 around with friends or family, spraying each other from various angles [...]

However, adopting a more deliberate (System 2) approach to the problem reveals additional
considerations that could elevate a water fight to being truly epic. This involves more structured
planning and execution:

1. **Preparation:** Consider the type and quantity of water guns, ensuring there's a mix of short-
range and long-range weapons. Additionally, [...]

2. **Location and Setup:** The setting can make a significant difference. A spacious backyard with
varied terrain (trees, a pool, or a trampoline) can add to the excitement [...]

3. **Rules and Teams:** [...]

4. **Safety and Supervision:** [...]

5. **Aftermath:** [...]

Reconciling the intuitive and deliberate approaches, the key is to balance spontaneity with
planning. While the immediate joy and excitement of a water fight are important, thoughtful
preparation can enhance the experience [...]
```

4.2.3 Data mixes

In our data mixes we combine the aforementioned synthetic datasets with natural text:

- `Web`: select subsets of our pre-training data (Table 1): `CC-edu` (high-quality subset of CommonCrawl chosen using FineWeb-edu classifier (Lozhkov et al., 2024a)); `Wiki+` (combined Wikipedia, HackerNews, and StackExchange sources); `Legal` (combined Europarl, USPTO, FreeLaw sources); `Papers+` (combined Papers, PG-19, Ubuntu IRC sources); `TxT360-QA` (Section 3.3.2); `Arabic` (Section 3.3.6); `MegaMath` (Zhou et al., 2025). We additionally augment it with a few sources from Common Pile that were not present in our pre-training data (Kandpal et al., 2025) (e.g., openly licensed transcribed YouTube videos).
- `Code`: `RefineCode` (Huang et al., 2025a) data from pre-training augmented with `Stack-Edu` (Allal et al., 2025) Python subset.
- `Institutional Books` (Cargnelutti et al., 2025), large collection of public domain books providing bulk of long-context data.

Natural text data is diverse in length and we organize it in 0k-8k, 8k-64k, 64k-128k, and 128k-512k length buckets for better control of context length during the four mid-training stages. We summarize data mixes for each stage in Figure 13 and context mixes in Figure 14.

To demonstrate effect of reasoning in mid-training we measure pass@k on AIME-2025 for varying k in Figure 15. AIME consists of challenging mathematical questions that LLMs only recently began to solve

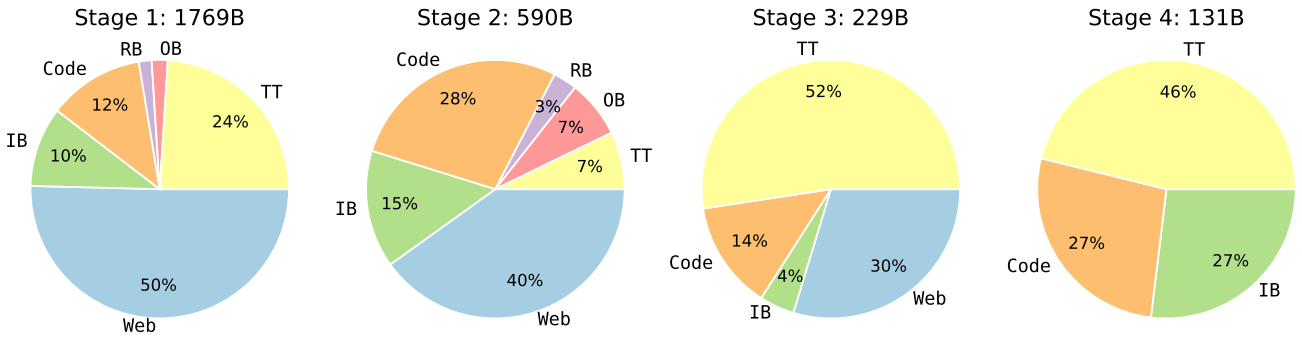

Figure 13: Data mixes for the four mid-training stages. Each stage is annotated with size in billions of tokens. `TT` refers to `thinking_traces`; `RB` to `reasoning_behaviors`; `OB` to `other_behaviors`; `IB` to `Institutional Books`.

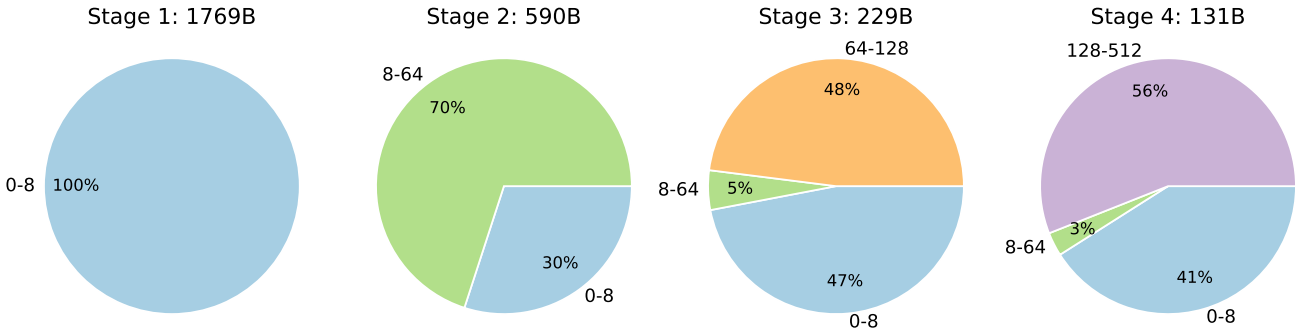

Figure 14: Context mixes for the four mid-training stages. Pie chart sectors correspond to percentage of tokens from documents in each of the four context length buckets: 0k-8k; 8k-64k; 64k-128k; 128k-512k.

well. Typically high performance on AIME is only achieved by post-trained models after SFT and RL. As we show in the figure base models like Llama 3.1 70B and Qwen2.5 72B (which were mid-trained without reasoning data to the best of our knowledge) achieve low pass rate even at $k = 64$ attempts despite their strong performance on simpler mathematical benchmarks like GSM8K (see Table 5 for details). Similarly, our K2 is pre-trained on predominantly natural web data and has 0 pass rate prior to mid-training. However its pass rate improves rapidly throughout reasoning-oriented mid-training, eventually reaching over 80% pass@64 and non-trivial pass@1 (i.e., accuracy) performance. We observe mild degradation in stage 4 where we focused on long-context data above 128k as it is hard to obtain high-quality reasoning data of such length and we instead primarily relied on code and books. Our pass@k analysis is motivated by recent works demonstrating correlation between base model's pass@k performance on complex benchmarks and its subsequent performance post RL (Yue et al., 2025; Yurochkin et al., 2025).

Memorization analysis Typical decontamination approaches rely on n-gram matching, however prior works reported that effect of such decontamination on benchmark performance is unclear (Li et al., 2024a). In practice there could be many factors affecting whether an LLM memorizes a benchmark question. Motivated by recent work on data extraction directly from trained models (Cooper et al., 2025), we conduct memorization analysis on AIME 2024 and AIME 2025 questions. We split each question into sentences and generate with beam search (width 10, up to 20 tokens at a time) conditioned on each subsequence of sentences until a period. We measure how often generated sentence

	AIME 2024	AIME 2025
K2	23.32	14.43
Qwen 2.5 72B	41.96	23.37
Llama 3.1 70B	9.84	9.85
Olmo 3 32B	37.82	14.47

Table 4: Sentence-level memorization rates (%).

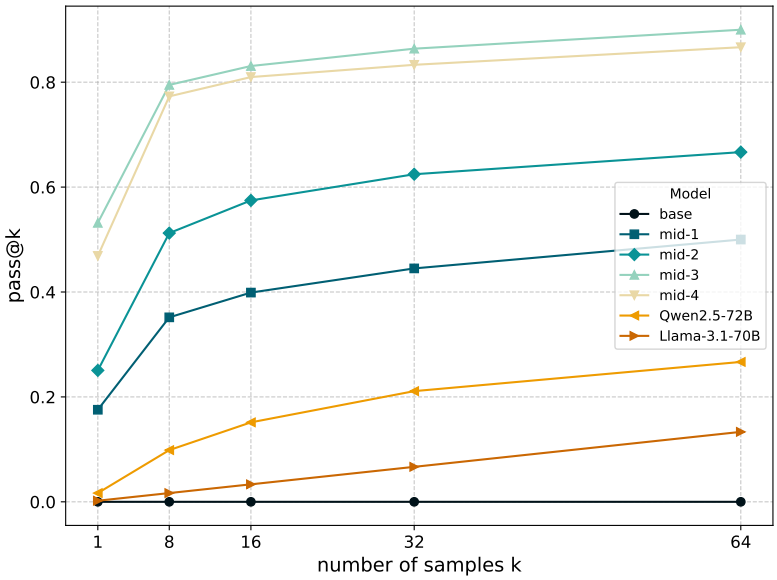

Figure 15: Coverage (pass@k) on AIME-2025 throughout mid-training stages compared to models of similar size trained without reasoning data.

exactly matches the corresponding sentence in the question. Results are reported in Table 4. We see that the memorization rate is overall lower for AIME 2025 than for AIME 2024, as expected since this benchmark is more recent. For 2025, the majority of memorization comes from the last sentence, which is typically formulated to ensure that the final answer is a single number—a pattern present in most AIME iterations. We also notice that K2's memorization for both years is low relative to other models, suggesting that its strong AIME performance is driven by reasoning ability rather than memorization alone.

4.3 Mid-training Infrastructure

We have developed our own training framework instead of directly using Megatron-LM for mid-training, as Megatron-LM's support for context parallelism and FSDP is not yet mature. Our in-house framework supports TP, CP, and FSDP, with a finer-grained automatic activation recomputation mechanism to better balance memory footprint and training speed. We also provide improved asynchronous communication among operations in LLMs, as well as customized fused kernels.

The in-house training infrastructure features on-the-fly tokenization and online best-fit packing. On-the-fly tokenization will avoid the hassle of pre-tokenization when adjusting the dataset mix, making it much more flexible for experiments involving changes in dataset sources or dataset weights.

We modify the Best-Fit packing algorithm (Ding et al., 2024a) and implement an online version in our framework. We enable this feature starting from stage 2. It significantly reduces unnecessary document truncations at the cost of only a few additional padding tokens. Our ablation experiments show that this feature improves loss given the same model and data.

Parallelism Strategy The tensor parallelism size (TP) is fixed to 8 throughout the four mid-training stages. For context parallelism size (CP), we set it to 1 for stage 1 and stage 2, and gradually increasing it to 2 for stage 3 and to 8 for stage 4 to keep memory usage under control. In our implementation of context parallelism, we use standard all-gather communication of keys and values between ranks in a context-parallel group, with asynchronous communication to hide overheads behind the computation of queries. We find this simple implementation to be more efficient than the RingAttention algorithm (Liu

et al., 2024b).

4.4 The Mid-training Run

In this section, we discuss observations from the mid-training stages. Figure 16a illustrates the loss curves. Although we included a replay of web data in each stage, the distinct loss ranges indicate significant domain differences across stages. Since we are simultaneously extending context length and adding new capabilities, it is challenging to reduce the distribution differences across the stages. Despite these shifts, we have observed the key metrics improve or remain competitive. The results will be presented in detail in §5.

Figure 16b tracks key metrics throughout this process. For traditional benchmarks, especially those evaluated via perplexity, performance generally remains stable or improves only slightly. In contrast, complex reasoning benchmarks demonstrate consistent gains. For example, GPQA-Diamond shows steady improvement across the final three stages designed for it. However, GSM8K performance has been unstable. We attribute this to two factors: changes in the model's output format (which caused parsing errors) and the mathematical data distribution used in each stage. Additionally, we observe a distinct pattern in which certain scores increase significantly at the very beginning of a stage but remain flat thereafter. We interpret these rapid initial gains as a shift in model behavior or output style rather than the acquisition of new knowledge.

Finally, we report a key finding regarding the Rotary Positional Embedding (RoPE) configuration. In an early experiment targeting a 128K context length, we set the RoPE base frequency to 1M. We find this value insufficient to support extended context capabilities, which resulted in degraded performance. We subsequently adjusted the base frequency to 10M, after which performance recovered to expected values. See Figure 17 for the comparison.

5 Base Model Evaluation

Key Takeaways

- **Reasoning-focused mid-training unlocks superior capabilities.** Throughout our mid-training stages, K2 dramatically improved in performance across mathematics, STEM, and logic domains. By the end of mid-training, our model significantly outperforms other base LLMs and rivals fully-trained models on challenging math and logic benchmarks.
- **Reasoning in mid-training boosts model capabilities but presents evaluation challenges.** To comprehensively quantify model improvements due to reasoning we expanded classical pre-training evaluations and considered complex generative benchmarks typically only used for fully-trained LLMs. We also needed to depart from the traditional few-shot prompting approaches and develop new prompts, robust parsers, and extend generation length when evaluating.
- **Breadth of evaluation is crucial to measure progress.** In our experiments several classical pre-training benchmarks slightly degraded throughout mid-training, which was discouraging initially. Extending our evaluation suite allowed us to better quantify model's abilities and gain confidence in the mid-training recipe.

In this section, we present a comprehensive evaluation of the K2 series across its fundamental training stages: pre-training and mid-training. Our evaluations include traditional completion-likelihood-based benchmarks as well as a variety of generative tasks covering math, STEM, coding, instruction following, and logic puzzles. The results of these evaluations are summarized in Table 5.

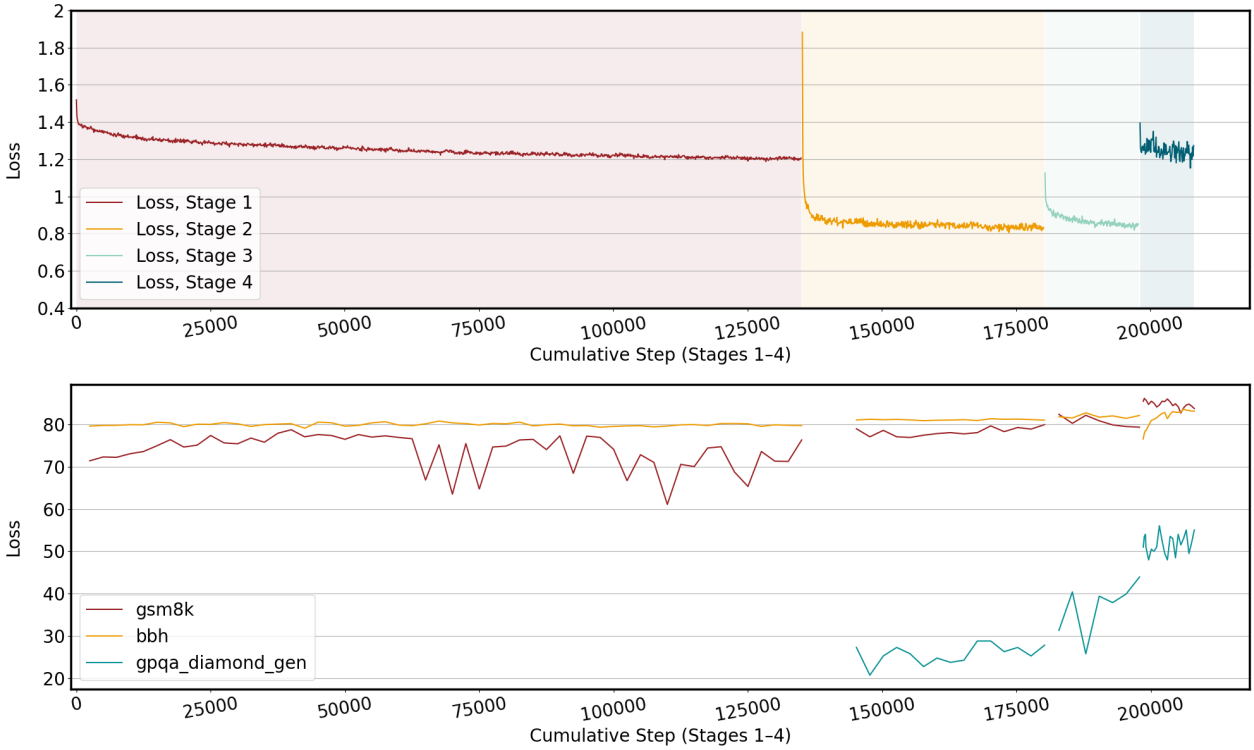

Figure 16: a) Loss curve of four mid-training stages. The losses may jump up or down at the beginning of next stage due to different data distributions. b) Trend of key metrics across four mid-training stages. Despite the fluctuations, the general upward trend could be observed across four stages. (We missed four evaluation data points at the beginning of stage 2)

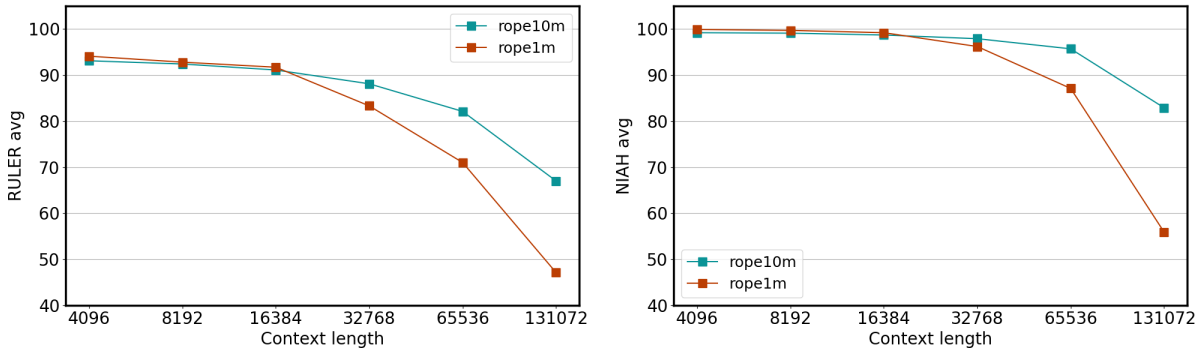

Figure 17: Impact of RoPE base frequency configuration on long-context performance at the 17,500-step checkpoint of stage 3 mid-training. An insufficient RoPE base frequency leads to degraded performance, especially for longer evaluation context length cases. a) RULER average for runs using base frequencies 1M and 10M. b) NIAH average for the same comparison.

	base	mid-1	mid-2	mid-3	mid-4	Qwen2.5-72b	Llama3.0-70b	Llama3.1-70b	Olmo3-32b
# Total Params	70B	70B	70B	70B	70B	72B	70B	70B	32B
General Tasks									
MMLU	74.3	74.4	73.5	75.0	75.2	**86.1**	79.5	79.3	75.2
MMLU-Pro	43.7	46.8	48.1	**59.8**	57.0	58.1	52.8	53.8	49.6
BBH	68.4	79.8	81.1	82.2	83.2	**86.3**	82.2	82.1	77.6
HELLASWAG	87.8	86.9	86.6	86.6	86.0	87.6	**88.0**	85.0	84.8
WINOGRANDE	82.6	83.7	83.7	83.7	83.0	83.9	85.3	79.8	**90.3**
PIQA	84.2	84.0	83.3	82.9	83.1	83.5	84.6	84.3	**85.6**
TRUTHFULQA	54.0	54.9	55.1	55.8	53.9	**60.5**	45.6	49.7	54.9
Math & STEM Tasks									
GPQA-DIAMOND	26.3	31.3	27.8	43.9	**55.1**	34.9	21.2	27.3	30.3
GSM8K	68.0	76.4	82.1	**93.6**	92.5	91.2	83.2	81.1	80.5
MATH	27.8	38.2	41.1	**94.7**	91.4	58.5	41.9	41.6	43.4
AIME 2025	0.0	17.6	25.1	**53.2**	46.9	1.7	0.1	0.2	14.7
ARC-CHALLENGE	64.9	66.4	66.4	66.0	66.3	**72.4**	69.2	64.9	65.4
Coding Tasks									
MBPP	57.6	57.8	58.2	59.8	61.8	**75.4**	69.2	64.4	60.2
HUMANEVAL	50.0	51.2	53.7	**54.3**	**54.3**	**54.3**	42.1	50.6	36.0
Logic Puzzles									
COUNTDOWN	1.3	53.3	53.1	35.9	**75.6**	6.0	1.0	0.5	23.2
KK-4 PEOPLE	4.8	44.9	68.0	64.5	**92.9**	26.1	4.2	7.6	42.4
KK-8 PEOPLE	0.5	23.2	41.3	51.6	**82.8**	5.7	1.1	1.3	13.0
ORDER-15 ITEMS	4.7	30.7	47.2	55.8	**87.6**	37.0	3.5	4.5	25.0
ORDER-30 ITEMS	0.0	0.3	3.0	34.1	**40.3**	0.7	0.2	0.1	0.6
Instruction Following									
IFEVAL	17.4	26.2	28.5	34.5	26.7	**40.3**	15.1	17.4	13.2
Arabic									
MMLU-Arabic	65.4	66.1	64.5	66.6	65.5	**74.1**	65.0	66.8	47.8

Table 5: Comparison among K2 pre-training, mid-training stages and other strong open-source benchmarks in similar parameter scale. The highest, the second-best scores are shown in **bold** and underlined, respectively.

5.1 Evaluation Methodology

The evaluation of base models mainly focuses on their performance in general knowledge, reasoning, mathematics, scientific knowledge, coding, and multilingual (arabic) capabilities. The evaluation datasets for base model checkpoints include 19 benchmarks:

- General Tasks: MMLU (Hendrycks et al., 2021a) (5-shot), MMLU-Pro (Wang et al., 2024a) (5-shot), BBH (Suzgun et al., 2022) (3-shot, CoT), HELLASWAG (Zellers et al., 2019) (10-shot), WINOGRANDE (Sakaguchi et al., 2021) (5-shot), PIQA (Bisk et al., 2020) (0-shot).
- Math & STEM Tasks: GPQA-DIAMOND (Rein et al., 2024) (0-shot, CoT), GSM8K (Cobbe et al., 2021) (5-shot, CoT and 0-shot reasoning prompt), MATH (Hendrycks et al., 2021b) (4-shot, CoT and 0-shot reasoning prompt), AIME 2025 (0-shot reasoning prompt), ARC-CHALLENGE (Clark et al., 2018) (25-shot).
- Coding Tasks: MBPP (Austin et al., 2021) (3-shot), HUMANEVAL (Chen et al., 2021) (0-shot).
- Logic Puzzles: COUNTDOWN (Gandhi et al., 2024) (0-shot reasoning prompt), Knights and Knaves (KK) (Xie et al., 2024; Johnson-Laird and Byrne, 1990; Smullyan, 1978) (0-shot, reasoning prompt), ORDER (0-shot reasoning prompt) where an LLM needs to infer the overall order in which a set of items was sold based on pairwise relations. For KK and ORDER we consider varying levels of difficulty and report results for medium and hardest levels (4 and 8 people for KK; 15 and 30 items for ORDER).
- Instruction Following: IFEVAL (Zhou et al., 2023) (0-shot).

- Adversarial: TRUTHFULQA (Lin et al., 2021) (0-shot)
- Arabic: MMLU-ARABIC (Koto et al., 2024) (5-shot).
- Long Context Tasks: RULER (0-shot)(Hsieh et al., 2024), NIAH (0-shot)(Nelson et al., 2024).

Table 6 summarizes the metric types reported for each benchmark, including perplexity-based multiple choice (MC), generation-based exact match (EM), and accuracy-based (Pass@k). Long context evaluation will be discussed in §5.2.2.

Benchmark	Metric	Benchmark	Metric	Benchmark	Metric	Benchmark	Metric
MMLU	MC	PIQA	MC	MBPP	Pass@1	AIME 2025	EM
MMLU-Pro	EM	GPQA-Diamond	EM	HumanEval	Pass@1	CountDown	Pass@1
BBH	EM	GSM8K	EM	IFEval	MC	KK	Pass@1
HellaSwag	MC	MATH	EM	TruthfulQA	MC	Order	Pass@1
Winogrande	MC	ARC-Challenge	MC	MMLU-Arabic	MC		

Table 6: Summary of evaluation benchmarks and corresponding metrics (MC: Multiple Choice, EM: Exact Match).

As baselines we select leading open-source base models of comparable scale, Llama 3 and 3.1 70B (Grattafiori et al., 2024), Qwen2.5 72B (Team et al., 2025b), and Olmo-3 32B (Olmo et al., 2025). All models are evaluated using the same evaluation pipeline to ensure fair comparison. On GSM8K and MATH we report best performance across few-shot CoT and reasoning prompt setups (see Section 5.2.1 for additional details). For non-K2 models, we report the best score across results that are publicly reported or that we reproduced ourselves.

As discussed in subsequent sections, the details of the implementation can substantially affect the final reported numbers. To mitigate this, we also open source our evaluation code in Eval360[6], to make the whole process transparent and trustworthy. The core evaluation logic is adapted from lm-evaluation-harness (Gao et al., 2023) and we build support for more custom datasets, metrics, filters, and parsers. The numbers in this section are obtained by setting the maximum generation length to 32,768 tokens for all checkpoints, and temperature to 1.0 for all of the non-greedy decoding benchmarks. For AIME and all logic puzzles we report results averaged over 64 evaluation runs.

5.2 Evaluation Results

Our K2 models show consistent improvements during the pretraining and four mid-training stages, producing strong performance on a broad range of benchmarks. The gains are most pronounced on Mathematics, STEM reasoning, and logic-intensive tasks, where K2 mid-stage models substantially outperform all open-source baselines of comparable scale.

For general-domain tasks, while K2 does not match the overall MMLU performance of Qwen2.5-72B (86.1), our models achieve competitive or state-of-the-art scores on several sub-benchmarks. Notably, mid-3 obtains the highest score among open models on MMLU-Pro (59.8), and achieves strong results on TruthfulQA (55.8) and BBH (83.2), outperforming LLaMA 3/3.1 by a large margin. These improvements suggest that the staged mid-training pipeline enhances robustness and general reasoning capabilities even when not optimized for raw MMLU.

In Mathematics and STEM, K2 models demonstrate clear state-of-the-art performance. The mid-4 model achieves the highest scores among all compared models on GPQA-Diamond (55.1), GSM8K (93.6), and MATH (94.7). These substantial gains highlight the effectiveness of our progressive mid-training strategy for developing structured reasoning and multi-step problem-solving ability.

[6]https://github.com/LLM360/Eval360

For coding tasks, K2 shows steady improvements, though Qwen2.5-72B remains the strongest on MBPP (75.4). The mid-4 model reaches 61.8 on MBPP and matches Qwen2.5-72B on HumanEval pass@1 (54.3), outperforming LLaMA 3.0/3.1 and most other baselines. Despite these gains, MBPP remains a relative weakness and warrants further discussion.

On logic puzzles, K2 achieves dramatic and consistent state-of-the-art performance. The mid-4 model attains 75.6 on Countdown, 92.9 on KK-4 People, 82.8 on KK-8 People, and up to 87.6 on Order-15 Items, often surpassing other 70B-scale models by large margins. Further, our performance on KK-8 People (hardest difficulty level) matches leading fully-trained models such as DeepSeek-R1 (83%) and o3-mini-high (83%) based on the results reported by Xie et al. (2025). These results demonstrate robust generalized reasoning capabilities beyond what traditional NLP benchmarks capture.

Overall, the K2 pretraining and staged mid-training pipeline delivers steady, meaningful, and often state-of-the-art improvements across diverse domains. The strongest gains appear in Math, STEM, and logic reasoning, establishing K2 as a highly capable 70B-scale open model.

5.2.1 Evaluation with Reasoning

In order to quantify effect of reasoning data introduced in mid-training we expanded our evaluation set of benchmarks and experimented with alternative prompting strategies. Specifically we evaluated our models on complex mathematical problems from AIME 2025 and a variety of logic puzzles—benchmarks typically only reported for fully trained models—with a prompt designed to incentivize structured reasoning in base models. We also used this prompt for GSM8K and MATH benchmarks in addition to standard few-shot CoT approaches.

Structured Reasoning Prompt We use prompt similar to the one DeepSeek-R1-Zero (Guo et al., 2025) used for reinforcement learning from their base model. We adjust it slightly based on the task at hand. As an example we provide reasoning prompt for AIME 2025 below.

AIME 2025 Reasoning Prompt

Prompt:

```
You are a helpful assistant. To answer the user's question, you first think about the reasoning
process and then provide the user with the answer. The reasoning process and answer are enclosed
within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process
here </think><answer> answer here </answer>. Provide a single number as the answer, for example, <
answer> 47 </answer>. Now the user asks you to solve a math problem.

User: {problem}
Assistant:
<think>
```

Parsing and Token Budgets We observe that performance on tasks that require logical generation, such as GSM8K, is sensitive to the details such as prompt format, response length, and parsing logic. Accurate scoring depends not only on the model's reasoning but also on successfully extracting the final answer. Variations in output format, such as placing answers in \boxed tags versus using headers like ## Answer, can significantly alter benchmark results if the parser is not aligned. Furthermore, evaluating these reasoning behaviors introduces a computational trade-off. Unlike standard generation, reasoning protocols require a substantially larger token budget (up to 32k tokens) to accommodate extended thinking traces. This increases the inference cost and must be factored into evaluation design.

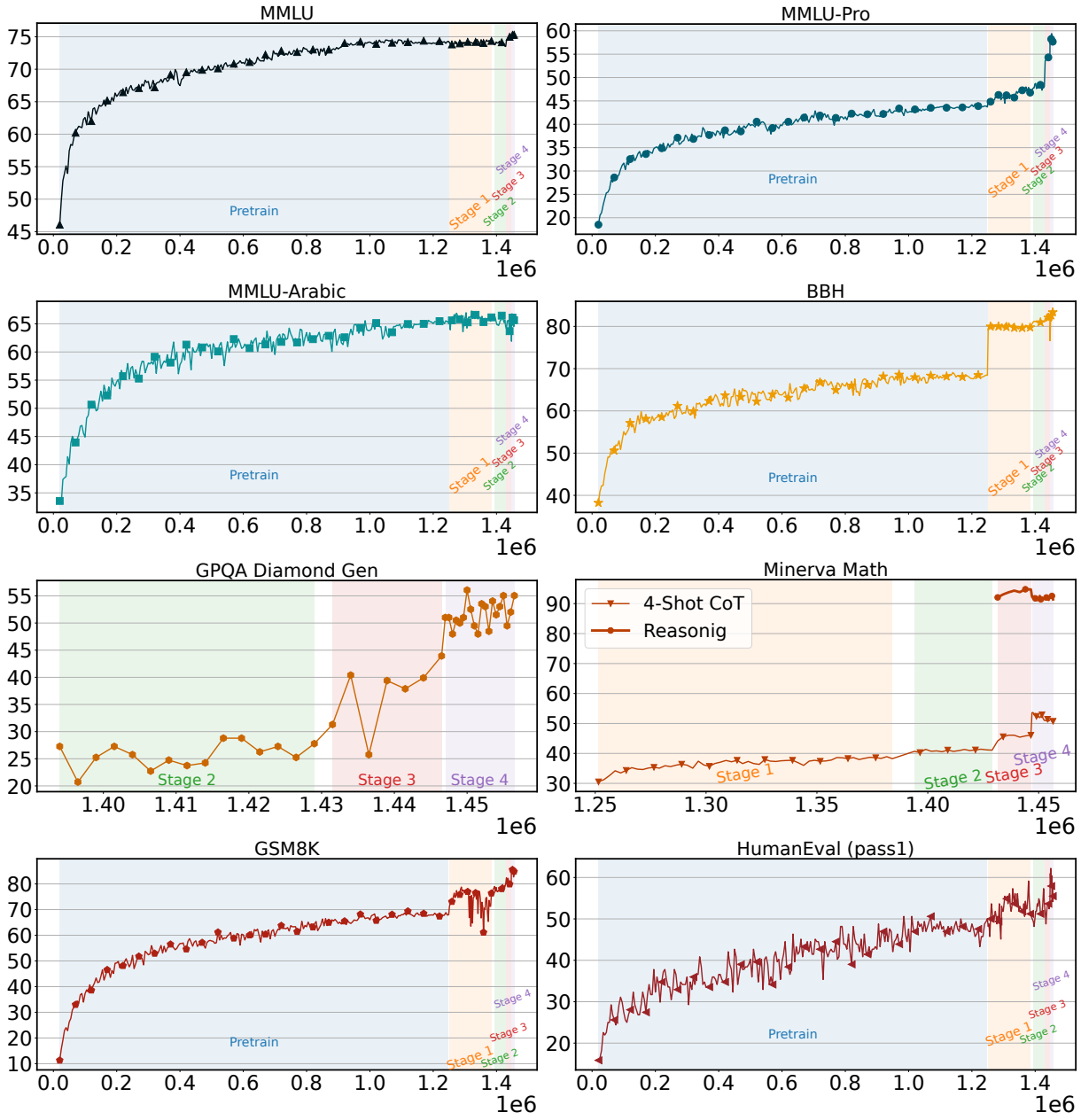

Figure 18: Trajectories of evaluation metrics for K2 during the pretraining and mid-training stages (stages 1–4). For the GPQA Diamond Gen benchmark, we did not evaluate checkpoints during pretraining. For the Minerva Math benchmark, we evaluated checkpoints from stages 1–4 using the non-reasoning prompt and checkpoints from stages 3–4 using a 4-shot CoT prompt.

Besides these details, more explicit prompts yield consistent gains. In the standard QA format, accuracy increases steadily from the `base` model (68.0) to the `mid-4` checkpoint (83.8). Chain-of-thought (CoT) prompting surpasses these results at every stage, reaching 90.0. The structured reasoning prompt, which leverages the expanded token budget, yields the strongest results: performance rises from 78.1 at `base` to 93.6 at `mid-3` (see Table 7). This confirms that structured prompts with precise parsing are important to evaluating the model correctly.

	base	mid-1	mid-2	mid-3	mid-4
GSM8K	68.0	76.4	80.0	79.4	83.8
GSM8K (CoT)	63.2	77.3	82.1	87.0	90.0
GSM8K (Reas.)	78.1	60.7	67.9	93.6	92.5

Table 7: Comparison of standard QA, CoT, and Structured Reasoning prompts for GSM8K.

	RULER Avg.						NIAH Avg.					
Model	4k	8k	16k	32k	64k	128k	4k	8k	16k	32k	64k	128k
Llama3.1-8b-instruct	96.2	94.8	94.4	92.5	86.4	80.7	99.9	99.6	99.5	99.6	99.4	96.9
Qwen2.5-14B-Instruct-1M	96.6	95.5	94.3	94.0	92.7	77.3	100.0	99.9	99.8	99.6	99.6	98.3
Qwen2.5-72B+yarn	94.6	90.7	89.1	87.1	81.7	75.0	97.4	94.3	92.8	90.4	85.6	80.1
Qwen2.5-72B	96.3	94.7	93.2	90.1	74.7	53.9	99.9	99.6	99.1	93.9	76.4	46.3
mid-2	94.3	92.2	89.2	73.4	51.6	12.0	99.9	99.7	98.2	91.0	69.0	7.57
mid-3	93.1	92.4	91.1	88.1	82.1	67.0	99.2	99.1	98.7	97.9	95.7	82.9
mid-4	93.9	93.2	92.9	89.6	85.3	74.6	99.6	99.4	99.6	98.8	98.8	95.2

Table 8: Evaluation results of long context benchmarks up to 128K. A context length enhanced model (Qwen2-14B-1M) can outperform stronger model. K2 demonstrate stable context performance up to 128K.

5.2.2 Long Context

Table 8 compares the long-context evaluation of the `K2` checkpoint against several baselines. `mid-4` stands out for its robust stability. While its short-context RULER scores trail top baselines like Llama 3.1 slightly, its degradation from 4K to 128K is significantly smoother. Unlike earlier stages which experience sharp performance cliffs at extended lengths (>128K), `mid-4` remains consistent. Similarly, it delivers high NIAH accuracy, remaining above 95% even at 128K. We also observe that Qwen2.5-14B-1M outperforms the larger Qwen2.5-72B on these metrics. This suggests that specialized long-context training is often more critical for extended input tasks than raw model scale.

We note one challenge we encountered when extending the context length of our model. Despite training with a context window of up to 512k, performance at the extreme limits is constrained by data scarcity. Naturally occurring long documents are rare compared to shorter text. We anticipate that supplementing the training mix with synthetic data designed for these extreme lengths will further bridge this gap.

6 Supervised Fine-tuning

Key Takeaways

- **Supervised fine-tuning with mixed reasoning efforts.** We present a novel SFT approach enabling our model to learn three reasoning efforts activated via chat template. Our findings demonstrate that a single LLM can balance efficiency and test-time compute gains, contrary to the recent practice of training separate "instruct" and "thinking" models.
- **The model learns to adapt to problem difficulty.** We observe gradual improvement of capabilities and increased generation length as we increase the reasoning effort in evaluations. For the high reasoning effort our model demonstrates potential to dynamically adapt its generation length to the problem at hand.
- **Self-identity system prompt needs to be injected at random.** We found that training with a default self-identity system prompt did not yield the desired results. Instead we include self-identity system prompt at random for a subset of SFT data.

During our light SFT phase, our goal is to capitalize on the reasoning capabilities obtained during mid-training while allowing users to experience the model without having to wait for lengthy reasoning to

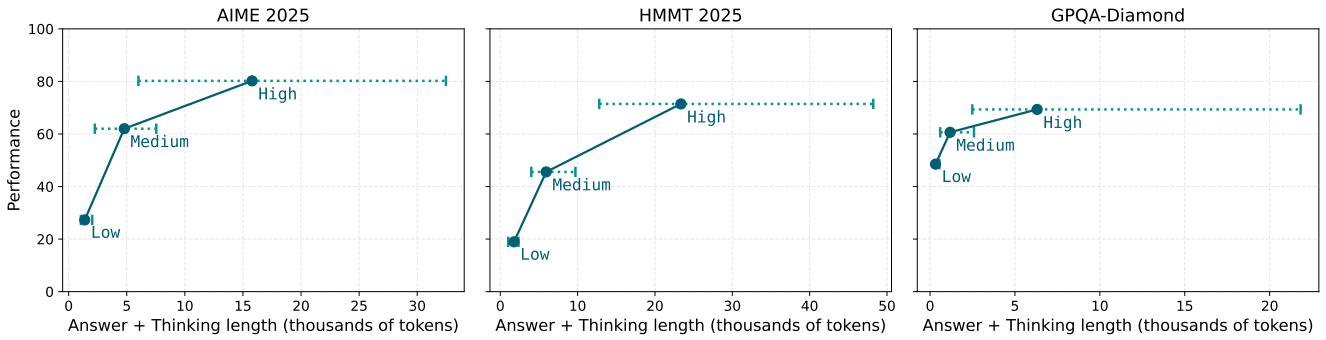

Figure 19: Performance of K2's three reasoning efforts on challenging math (AIME 2025 and HMMT 2025) and science (GPQA-Diamond) benchmarks plotted against corresponding generation length quartiles.

complete. Balancing these objectives is challenging as test-time computation via reasoning is crucial for good performance on many tasks. Recent model releases (Yang et al., 2025; Olmo et al., 2025) choose to train separate model variants for quick high quality responses ("instruct") and extensive reasoning ("think"), however this necessitates serving both variants to allow users to flexibly adjust the setting based on their task. Instead we implement the Three Reasoning Efforts approach where a single LLM can be used with varying test-time compute budgets via a simple chat template setting leveraging special tokens for each mode. This behavior is similar to GPT-OSS (Agarwal et al., 2025), but differs in chat template and is the first open recipe to achieve it.

6.1 SFT Recipe

Three Training Efforts. In the supervised fine-tuning we introduce three levels of reasoning effort: `low`, `medium`, and `high`. Reasoning efforts are similar to the GPT-OSS low, medium, and high reasoning effort settings, but differ in how they are induced in a chat template. Correspondingly we utilize GPT-OSS-120B when synthesizing our SFT data for a curated set of prompts detailed in Section 6.2.

In Figure 19 we demonstrate the effect of the three reasoning efforts. We plot performance of each reasoning effort on challenging math and science benchmarks against the corresponding generation length (answer and reasoning) quartiles, i.e. 25th percentile, median, and 75th percentile. We observe gradual performance increase with the reasoning effort, corresponding to longer generations. We note that `low` and `medium` efforts have fairly consistent generation length, while `high` has a large spread suggesting that model has potential to adjust to the problem difficulty on `high` reasoning instead of consistently "overthinking".

Chat Template Design. We use a unified chat template that supports the three reasoning efforts and tool calling. The `high`, `medium`, and `low` reasoning efforts are controlled via special tokens `<think>`, `<think_fast>`, and `<think_faster>` that delimit an internal reasoning block for each response. The template is constructed so that these thinking segments only appear in the assistant message corresponding to the most recent user query and are stripped from earlier assistant messages when they are fed back into the context. Although not trained explicitly for this behavior, it is possible to elicit a `no-think` mode by immediately closing the thinking tags so that the model produces a direct answer with no visible reasoning. For tool use, we follow the Model Context Protocol (MCP) format and use designated tokens such as `<tools>` to mark the available tools and `<tool_calls>` to delimit the segment of the conversation where tool calls are issued.

Training Hyperparameters. We train the model for three epochs, totaling approximately 6,300 steps given the dataset size of 17.4 billion tokens per epoch. We use a global batch size of 128 with a micro batch size of 1. To balance data utility with training efficiency, we set the sequence length to 65,536 and

maintained a RoPE base frequency of 10 million, consistent with the final mid-training checkpoint. For optimization, we used $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, and a weight decay of 0.05. The learning rate followed a cosine schedule with a 10% warmup phase, peaking at 2×10^{-5} and decaying to 10% of the peak value.

6.2 SFT Data: TxT360-3efforts

6.2.1 Core SFT data

Our core SFT subset is built from question–answer pairs whose questions are collected from permissively licensed public datasets or synthetically generated. Subsequently, these questions are quality-filtered, deduplicated using substring matching, and decontaminated against evaluation benchmarks. The answers are mostly regenerated using GPT-OSS-120B at low, medium and high reasoning effort levels. Below, we summarize the major data categories and the procedures used to construct them.

- **Math**: We aggregate unique mathematic questions from seven sources: Nemotron-Post-Training-v1 (Nathawani et al., 2025), MathQA (Amini et al., 2019), OpenMathReasoning (Moshkov et al., 2025), SimpleScaling (Muennighoff et al., 2025) , NuminaMath (Li et al., 2024b), BigMathVerified (Albalak et al., 2025), and OpenMathInstruct-2 (Toshniwal et al., 2024). To ensure that all data were permissible for model training, we remove entries that were synthetically generated by commercial models (e.g., GPT models), such as those in Orca-Math and "synthetic_math".

- **Code**: We curate a diverse mixture of code datasets including seed_sft subset of rStar-Coder (Liu et al., 2025b), Bird-SQL (Li et al., 2023a), Nemotron-Post-Training-v1 (Nathawani et al., 2025), sql-create-context-instruction (bugdaryan), verifiable-coding-problems (Mattern et al., 2025), dolphin-coder (QuixiAI) and react-code-instructions (cfahlgren1). In addition, our corpus includes self-oss-instruct-sc2-exec-filter50k (BigCode), tiny-codes (Pham), glaive-code-assisttant-v3 (GlaiveAI), Magpie-Qwen2.5-Coder-Pro-300K-v0.1 (Xu et al., 2024), conala-mined-curated (codeparrot), self-instruct-starcoder (Zebaze), code-evol-instruct-oss (Luo et al., 2023), xlcost-text-to-code (Zhu et al., 2022). These sources provide high-quality, natural-language-to-code prompts that spans both general-purpose programming (e.g., Python) and specialised domains (e.g., SQL and React), enabling our model to understand realistic, developer-style prompts and produce well-structured code in different coding scenarios.

- **Chat**: For general-chat, we merged OASST (Köpf et al., 2023), ShareLM (Don-Yehiya et al., 2025), and UltraChat-200k (Ding et al., 2023), pulling only the first-turn queries.

- **STEM**: For the STEM portion of our SFT corpus, we aggregate queries from seven diverse sources, covering both multiple-choice question-answer formats and open-ended generation tasks. Specifically, we incorporate the STEM subset of the Nemotron-CrossThink QA dataset (Akter et al., 2025), the STEM split of the Llama-Nemotron-Post-Training-Dataset-v1 (Nathawani et al., 2025), and several high-quality educational and reasoning-oriented datasets, including NCERT (Kadam, 2025), Loong (AI, 2025), LogiCLM (Longface, 2023), and Logic701 (Bratchikov, 2023). Together, these sources provide broad domain coverage across science, engineering, and logical reasoning.

- **Instruction-Following with Constraints**: We generated a synthetic instruction-following dataset using GPT-OSS-120B with automated verification. Our pipeline generates diverse prompts with programmatically verifiable constraints across multiple categories (format, length, keywords, punctuation, and content), similar to the IFBench pipeline (Pyatkin et al., 2025). Each generated response is validated against its specified constraints using custom verification logic. The dataset schema includes instruction prompts, model responses, constraint specifications in a structured format, and verification metadata. This systematic approach enables scalable generation of instruction-following training data with measurable quality metrics. Complementing the constraint-verified instruction-following

data, we further target structured-output reliability. Many practical deployment settings require models to emit well-formed JSON for automated evaluation or programmatic integration. To address this requirement, we integrate Hermes-Json-Mode dataset ("interstellarninja", 2023).

- **Self-Identity**: To strengthen the model's self-identity consistency, we construct a synthetic self-identity dataset through a three-step pipeline. First, we identify user queries related to model self-identity by applying an LLM-as-judge filter over Lmsys-1M-chat (Zheng et al., 2023a) and ShareGPT (ShareGPT, 2023) conversations. These extracted prompts serve as seed examples. Next, we expand this seed set by prompting Qwen2.5-32B to generate additional queries of similar style and intent, resulting in approximately 3,000 English self-identity queries. We then use GPT-OSS-120B to generate responses conditioned on the `K2` model metadata as in our model card. Finally, to build a multilingual subset, we select 10 target languages and randomly sample 300 question–answer pairs for translation into each language.
- **Safety**: We construct safety question–answer pairs by combining seed questions from various sources with additional adversarial and jailbreaking variants generated from these seeds using PyRIT (Munoz et al., 2024). The seed questions are collected from four datasets containing harmful or malicious intent: AdvBench (Zou et al., 2023), Aya Red-Teaming (Aakanksha et al., 2024), the Do-Not-Answer Dataset (Wang et al., 2024b), and Forbidden Questions (Shen et al., 2024). To account for unexpected or context-breaking inputs intended to bypass the models' safety behavior, we transform the seed dataset by applying text-alteration techniques including Base2048 encoding, Morse code encoding, Unicode transformations, random capitalization, and language translation, and by prepending established jailbreaking prompts, including prompt-injection methods from JailbreakChat,[7] DAN (Shen et al., 2024), CipherChat (Yuan et al., 2023), Nested Jailbreak Prompts (Ding et al., 2024b), DecodingTrust (Wang et al., 2023), and JailBroken (Wei et al., 2023). Together, the seed and generated question–answer pairs form a comprehensive dataset designed to defend against common and established malicious attacks.

6.2.2 Agentic and Tool-use Data

Complex LLM use-cases such as agentic behavior, tool use, and prolonged user interactions require multi-turn SFT data. Such data is scarce, so we develop novel synthetic data pipelines with implicit verification, in addition to collecting existing sources.

Teacher-student multi-turn simulations. We simulate multi-turn conversations between a student and a teacher (both played by GPT-OSS-120B) in which the teacher monitors the progress of the student as they solve mathematics problems (e.g., step-by-step solution derivations and verification). The teacher is provided with a reference solution to help it assess the student's work and give targeted feedback that nudges the student to iteratively refine their solution. The teacher is explicitly instructed not to quote or reveal the reference solution, so that the student must derive a valid solution on its own.

We apply this approach to a subset of mathematical problems to generate two types of SFT data:

1. We map the student to the assistant and the teacher to the user to create multi-turn conversations that train the model to sustain extended, math-focused dialogues with users.
2. We reverse the roles of the student and the teacher to create conversations in which the assistant critiques and evaluates the user's input, training the model to assess user solutions critically. In these student–teacher dialogues, the teacher is given a detailed system prompt that instructs it to refuse

[7] https://github.com/alexalbertt/jailbreakchat

certain student requests (e.g., direct access to the reference solution). These conversations train the model to privilege system instructions over user requests.

Agentic (multi-turn, tool-use). For agentic data, ranging from multi-turn natural conversations to coding to tool-use, we use Nemotron Post Training v1 (Nathawani et al., 2025) and filtered versions of xLAM function calling (Liu et al., 2024d) and CommitPackFT (Muennighoff et al., 2023).

- For multi-turn tool-use data, we extend the teacher–student framework to a teacher–student–tool setup where all three agents are simulated by language models. For Nemotron Post Training v1, we directly use the data. For the `xlam-function-calling-60k` subset (Liu et al., 2024d), we use GPT-OSS-120B to simulate the teacher and student and Qwen2.5-7B-Instruct as the tool and generate multi-turn conversations where the teacher imitates the user asking the query and provides guidance to the assistant.
- For agentic coding trajectories from CommitPackFT (Muennighoff et al., 2023), we filter commits by file type (for example, excluding binaries), commit type, and diff size, infer the action label (`edit`, `rename`, `create`, `delete`), and then synthesize conversations along with the reasoning in which a "user" LLM acting as a teacher with access to the ground truth changes guides the assistant to generate the diffs, plan the modification, and apply the edits step by step.
- In addition, we incorporate the Toucan dataset (Xu et al., 2025), Hermes function calling subset ("interstellarninja", 2023), Glaive (GlaiveAI), and ToolACE (Liu et al., 2025a). All examples are converted into a standardized MCP-compatible format using a unified preprocessing pipeline. This pipeline relies on deterministic string-matching and regular-expression rules to identify speaker turns, extract tool-call specifications, handle parallel tool-calls, normalizing argument formats, and reconcile structural differences across datasets. Following format standardization, we apply a series of post-processing filters designed to remove samples that violate fundamental tool-use consistency constraints. These include: (1) the presence of a non-tool observation message immediately following a tool-call; (2) invocations of tools that are not defined in the system prompt; (3) malformed samples with empty or missing final dialogue turns; (4) trajectories lacking both tool definitions and tool calls (observed primarily in a subset of the Glaive dataset). This combination of rule-based normalization and targeted filtering ensures that all retained samples conform to a consistent interaction protocol and are suitable for downstream training and evaluation.

6.2.3 SFT data mix

Our SFT data mix consists of approximately 10 million documents with 10B loss tokens (i.e. tokens that contribute to loss computation during training). In Figure 20 (left) we present data mix with respect to seven major categories. Below we discuss key aspects of the data processing.

- We convert multi-turn data into multiple training samples, one per turn including the corresponding prior turns. This is required because our chat template hides thinking from all but the last assistant message to conserve context length.
- We observe that GPT-OSS tends to occasionally produce phrases such as "*Now to answer as ChatGPT*" in its reasoning. We use simple substring matching to filter these out.
- In our earlier experiments we include default self-identity system prompt in every conversation. We observe that the model essentially ignores it after training. To address this problem we chyoose to include self-identity system prompt with probability 0.5 in self-identity and safety data and with probability 0.1 for other categories.

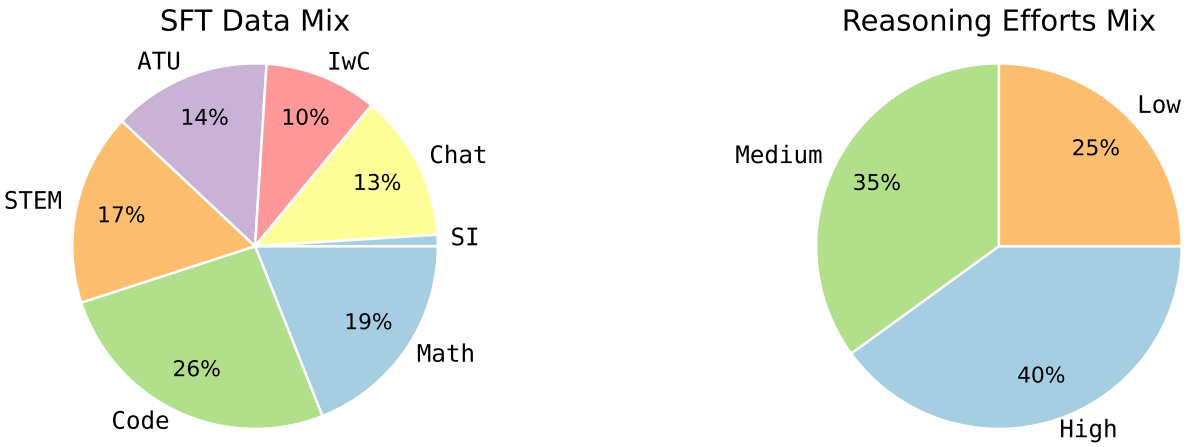

Figure 20: SFT Data (left) and Reasoning Efforts (right) mixes. Mix weights are defined with respect to loss tokens (total $\approx$ 10B). In the Data mix, ATU corresponds to agentic and tool-use data; IwC to instructions with constraints; SI to self-identity and safety.

- In addition to controlling data mix with respect to categories, we also control for the distribution of reasoning efforts in the SFT data. This distribution is summarized in Figure 20 (middle).
- We do not explicitly control length distribution in the SFT mix. Majority of our data ($\approx$ 98%) is under 8K tokens.

6.3 SFT Training

We perform full-parameter supervised fine-tuning (SFT) on the final mid-training checkpoint using the aforementioned data mixture and the same in-house training infrastructure used during mid-training, which is a native PyTorch implementation. We apply a chat template to the raw conversation-style data samples during training.

Packing We employ the same online best-fit sequence packing strategy (Ding et al., 2024a) as in mid-training. Unlike the naive way of batching (one data sample per sequence) during SFT, by applying best-fit sequence packing, we pack multiple data samples only when they fit to a single sequence to maximize training efficiency. The other benefit is that no truncations are introduced, avoiding breaking one conversation into two different data sequences, which is detrimental to SFT training. We also skip data samples longer than sequence length for the same reason.

Figure 21 shows the average truncation ratio, which is 0, and average padding ratio at every step, which is always less than 0.004%. Intuitively, we have less than 3 padding tokens out of each sequence (65,536 tokens). As a comparison, using naive batching, we would have an average padding ratio more than 95%.

The Training Run Similarly to what we use in the mid-training stages, we use Tensor-Parallel of 8, Context-Parallel of 1 and turned on selective recomputation. The SFT run is relatively straightforward. We run for 3 epochs over the dataset (2100 steps), and save the checkpoints at the end of each. Figure 21c presents the loss, we can see the typical but subtle dips at the intersection of epochs.

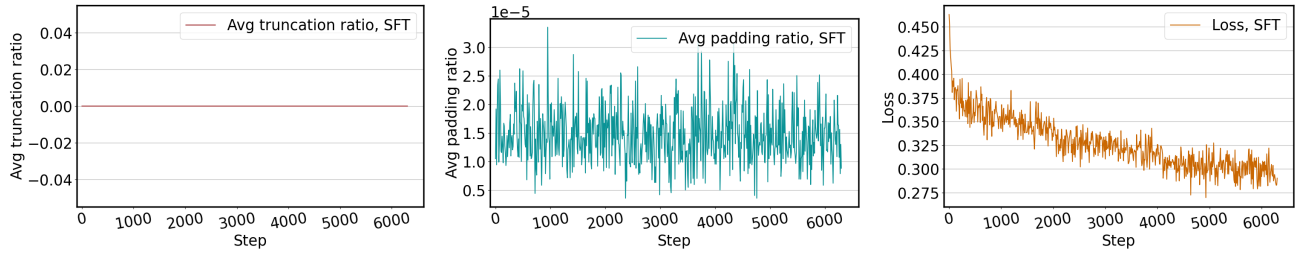

Figure 21: a) Average truncation ratio of the SFT run. The ratio is always zero because we avoid introducing truncations by applying best-fit packing and skip data samples longer than sequence length. b) Average padding ratio of the SFT run. This is the average number of padding tokens divided by sequence length at each step. The padding ratio is extremely small which greatly improved training efficiency. c) Loss curve of the SFT run. The dips at 2100 and 4200 steps show intersections of epochs.

7 Final Evaluation: K2's Elicited Reasoning Capabilities

Key Takeaways

- **Reasoning effort impacts performance differently across benchmarks.** On complex mathematics and coding benchmarks we observe dramatic increase in performance with the reasoning effort. On simpler tasks and tasks from other domains the improvement is less pronounced. A notable outlier is function calling where medium reasoning effort performed the best.
- **Reasoning focused mid-training and light SFT provides promising start for RL.** `K2` demonstrates strong performance across benchmarks, but there remains room for improvement to match leading large MoE LLMs on complex mathematical and coding tasks. Our pass@k analysis suggests that `K2` maintains high diversity in generations, making it a promising candidate for further training with RL.

We conclude with a dual assessment of `K2`'s capabilities: its immediate performance after supervised fine-tuning and its latent potential for advanced reasoning. We first present results on challenging benchmarks to validate the SFT model. While supervised fine-tuning (SFT) can enable instruction compliance and tool use, it is a simple procedure that does not unlock the model's upper bound. To demonstrate `K2`'s true capacity as a reasoning foundation. We explicitly analyze the model's reasoning potential using pass@k metrics, showing that `K2` possesses the intrinsic capability required for further improvement.

7.1 Evaluating K2-SFT

7.1.1 Evaluation Setup

The evaluation of the SFT model focuses on the following aspects: conversational chat quality, mathematics, coding, stem, long-context, and tool usage.

- Long Context: **LongBench V2** (Bai et al., 2025). We do not apply YaRN context scaling, except for the Qwen3-32B models. For the reasoning configurations, we allow generation up to the maximum context length of the model, noting that the models may run out of context length while still in the reasoning phase.
- Chat: **Arena-Hard V2.0** (Li et al., 2024c). We report results using the official source code and Gemini-2.5 Pro as the judge on the hard prompt subset. We report the results for only one run per model.
- Mathematics Tasks: results for all math tasks are obtained by averaging over 16 runs. **AIME 2025** is

evaluated with exact-match, using `\boxed{}` to extract the answer. For **HMMT 2025** (Balunović et al., 2025) we also use `\boxed{}` to extract the answers, but we use Qwen3-235B as a judge to compare the extracted responses against the correct answers. To account for their small size, we average **AIME 2025** and **HMMT 2025** over 16 runs. For **GSM8K** (Cobbe et al., 2021) and **MATH** (Hendrycks et al., 2021b) results are extracted with our custom parser and compared to ground truth using Math-Verify.[8]

- Coding Tasks. For **LCB-v6** (Jain et al., 2024) we use a rigorous sandbox environment; a solution is considered correct only if it passes all public test cases. For **MBPP** (Austin et al., 2021) and **HumanEval** (Chen et al., 2021) we evaluate with standard Code-Harness.
- Science Tasks: **GPQA-Diamond** (Rein et al., 2024). Answer extracted from `\boxed{}`, and evaluated with exact-match of the multiple choice answer.
- Tool Usage Tasks: **Berkeley Function Calling Leaderboard v4** (BFCL v4) (Patil et al., 2025). We evaluate BFCL v4 across six categories: non-live AST (expert-curated single-turn calls), live AST (real-world user-contributed calls), multi-turn conversations, web search, memory, and detection (relevance/irrelevance). The maximum number of tokens generated is set to 32K for all models. Baseline models are evaluated with reasoning, if available.

Unless otherwise noted, we use temperature of 1.0 and context length of 128K (131072) for all K2 evaluations. We also evaluate the benchmarks on a variety of publicly available models. For these other models, we use temperature of 1.0 and context length of 128K, unless other values are specified in their documentations. If the models do not support 128K, we use their native maximum context length, unless the model has explicit documentation and support for using RoPE scaling methods. For models with multiple reasoning efforts, we list each reasoning effort as a separate model in the table.

7.1.2 Evaluation Results

We present the evaluation results in Table 9.[9] In general, after a simple supervised fine-tuning stage, the model is already competitive on many domains, showing the strengths of the base model.

Long Context. On LongBench V2., K2-High obtains 42.6% accuracy, outperforms all reasoning/thinking models except **GPT-OSS**. Remarkably, K2 models achieves significant improvement over large-scale MoE models, including **GLM-4.5** Air (106B A12B) and **MiniMax M2** (230B A10B). When comparing with instruct models, K2-High slightly under-performs the similar size model **Qwen 2.5 72B**, and the much larger MoE model **DeepSeek V3.1**-Instruct (671B A31B).

Math and STEM. We observe varying amounts of performance improvements across K2 reasoning efforts which align with the benchmark difficulty levels. Specifically, GSM8K is the easiest math benchmark corresponding to near equal performance across reasoning efforts, while AIME and HMMT are complex olympiad-level tasks corresponding to substantial performance growth as we increase the reasoning effort; MATH is average difficulty, thus we observe less pronounced difference across the three reasoning efforts.

Next we compare performance to other thinking models on most challenging AIME and HMMT. Notably we outperform several larger MoEs (**GLM-4.5** Air and **MiniMax M2**) and recently released fully open dense model **Olmo3-Think-SFT 32B**. MMLU-PRO evaluates broad multidisciplinary STEM knowledge and reasoning, where K2-High achieves 77.0, outperforming several instruct models and large MoE baselines, including GLM-4.5 Air and MiniMax M2. We note that we still underperform leading large MoE thinking

[8] https://github.com/huggingface/Math-Verify

[9] The exact model weight can be found in Appendix B.

Model Specifications	Long	Chat	Instr. Follow	Math & STEM						Coding		
	LongBench V2	(Arena Hard V2)	IFEVAL	AIME25	HMMT25	GSM8K	Minerva	GPQA-D	MMLU-PRO	MBPP	HumanEval	LCBv6
K2 Low Dense · 70B	40.7	32.8	73.2	27.3	19.0	92.4	85.0	48.5	65.3	71.0	82.3	39.9
K2 Medium Dense · 70B	41.3	48.6	82.7	62.0	45.6	92.0	90.6	60.6	73.3	75.8	84.2	51.3
K2 High Dense · 70B	42.6	62.1	90.0	80.2	71.4	94.8	94.5	69.3	77.0	84.8	91.5	67.0
Olmo3 Think SFT Dense · 32B · No RL	42.8	11.2	80.1	68.3	43.3	96.1	96.9	58.0	70.8	87.6	96.3	67.9
Olmo3 Think Dense · 32B · RL	47.1	25.8	88.7	73.3	50.8	95.7	97.3	59.8	73.2	91.6	96.3	67.6
GLM-4.5 Air MoE · 106B A12B	49.4	53.8	88.7	81.3	73.3	96.1	94.9	75.3	83.4	82.8	97.6	67.8
MiniMax-M2 MoE · 230B A10B	55.8	41.7	89.5	75.8	63.5	95.4	85.3	76.2	80.5	83.8	89.6	79.2
DeepSeek-V3.1 MoE · 671B A31B · Reasoning	41.2	55.1	89.6	83.3	82.7	96.2	97.5	75.4	84.7	94.0	96.3	74.4
Qwen3 235B MoE · 235B A22B · Reasoning	60.9	64.4	88.7	88.8	84.2	93.5	98.0	80.7	81.0	96.2	94.5	72.8
Qwen3 32B Dense · 32B · Reasoning	52.0	62.0	86.1	74.2	63.8	92.8	97.5	67.8	72.4	77.8	86.6	75.8
gpt-oss-120b Low MoE · 117B A5.1B	45.2	73.3	87.3	48.1	34.2	90.7	92.0	64.4	76.1	87.2	92.7	64.2
gpt-oss-120b Medium MoE · 117B A5.1B	46.5	84.0	87.3	75.2	62.5	91.8	91.9	70.5	76.2	84.8	91.5	71.7
gpt-oss-120b High MoE · 117B A5.1B	49.2	92.0	86.8	87.9	88.3	91.1	91.9	77.5	76.2	86.6	92.1	80.9
DeepSeek-V3.1 MoE · 671B A31B · Instruct	49.2	54.9	88.6	45.6	30.2	96.1	94.2	71.1	83.7	82.0	94.5	58.0
Llama 3.3 70B Dense · 70B · Instruct	31.7	5.7	91.0	5.3	0.67	93.3	73.7	41.8	71.2	76.0	86.0	40.8
Qwen 2.5 72B Dense · 72B	47.2	10.9	85.7	15.2	9.79	85.8	82.1	50.5	71.6	80.0	85.4	36.7
Qwen3 32B Dense · 32B · Instruct	42.4	44.4	85.3	24.0	15.6	93.1	85.0	67.9	62.7	74.6	87.8	39.9
Mistral Large Dense · 123B · Instruct	32.1	8.1	83.7	4.17	1.46	95.8	73.8	53.6	68.8	71.2	89.6	37.8
Qwen3 235B 2507 MoE · 235B A22B · Instruct	52.7	70.5	90.1	67.9	50.6	94.9	97.1	73.9	83.7	85.6	95.7	59.3

Table 9: We compare K2 SFT models against a series of open weight/open source models. Top section presents evaluation results for K2 post SFT. Middle section corresponds to reasoning/thinking baselines. Bottom section corresponds to instruct-style baselines. K2, after a simple SFT stage, is comparable to fully-trained reasoning and instruct models of larger scales, such as GLM-4.5-106B and DeepSeek V3.1. Notably, K2 maintains a conversation ability as shown in Arena Hard V2, performing better than many other models.

models such as **DeepSeek V3.1**, **Qwen3 235B**, and **GPT-OSS-120B High**. We note that math and STEM domains are among those where RLVR has proved most effective, thus we anticipate further improvements of K2 in RL post-training (see Section 7.2 for further analysis).

Chat. K2 performs exceptionally well in conversation. On Arena Hard V2, K2 with high effort achieves 62.1, much higher than most models of similar size, and is on par with Qwen3-235B. The GPT-OSS series perform well too A hypothesis is that proper reasoning before chatting also improves conversation quality.

Instruction Following. We evaluate instruction-following capability using IFEval, which measures a model's ability to accurately and consistently adhere to explicit user instructions. Across the three reasoning efforts, K2 shows steady improvement, with K2-High achieving 90.0, outperforming most instruct-style baselines of comparable scale, including Qwen 2.5 72B and Mistral Large. Notably, K2 maintains strong instruction-following performance while preserving conversational and reasoning capabilities, suggesting that the SFT process does not trade off instruction adherence for other skills. When compared with larger reasoning-oriented models, K2 remains competitive, despite the absence of reinforcement learning or preference optimization at this stage.

Coding. On the three commonly used coding benchmarks, K2-High outperforms all instruct models, demonstrating its strong coding capability. However, once compared with reasoning/thinking models, K2 under-performs **GLM-4.5** Air, **Olmo3-Think** and **GPT-OSS**-High, illustrating the potential direction of developing K2 reasoning models.

Tool Use and Function Calling. For this capability, we mainly evaluate K2 on the BFCL-v4, comparing it against several leading open-weights models (see Table 10 for detailed results). K2 demonstrates strong utility, significantly outperforming several baselines. Notably, K2-Medium achieves an overall score of 52.38, surpassing both models of the same size: Qwen 2.5-72B (45.88) and Llama 3.3-70B (31.05). A key differentiator is performance in *Multi-Turn* interactions, where K2-Medium (50.62) effectively maintains state and handles missing parameters, significantly outperforming Qwen 2.5 (33.88) and Llama 3.3 (19.62). Furthermore, K2-High exhibits precise structural adherence in *Non-Live AST* tasks, achieving 93.50% on multiple function calls and 89.00% on parallel calls, rivaling the larger GPT-OSS-120B model. While GPT-OSS retains a slight edge in memory-intensive tasks, K2 offers a superior balance of irrelevant-call detection and conversational robustness.

A notable observation is that both K2 and GPT-OSS achieve better overall performance with medium effort. This counter-intuitive result, where higher effort does not necessarily yield better accuracy, warrants further investigation. This may be partially due to the fact that the majority of the tool calling data for K2 does not contain reasoning traces. Interestingly, although K2 is trained on substantial instruction data generated by GPT-OSS, the behavioral trends of the two models do not consistently align. For instance, on the "Parallel" and "Parallel Multiple" categories of *Non-Live AST*, K2 benefits from high effort whereas GPT-OSS performs better with medium effort. Conversely, in the *Memory* "Vector" category, we observe the opposite pattern.

7.2 Assessing K2's Reasoning Potential

The SFT stage is designed to refine the reasoning and long-context capabilities K2 developed during the mid-training phase with specific modes of thinking for separate use cases of the model. By treating these capabilities as core primitives of K2, the SFT procedure prepares the model to be immediately useful for instruction following as well as light reasoning. As described in Section 6, we aim to unify these use-cases into a single model while also adequately preparing the model for future post-training with RLHF and/or RLVR to provide more specialized behaviors such as extended reasoning for complex problems.

We assess the capacity of K2 for further refinement with large-scale reinforcement learning in two ways. In Figure 22 we evaluate the model's pass@k performance on well established reasoning benchmarks in Math, Code and Instruction Following. In Figure 23 we evaluate K2-SFT across a range of settings of the temperature hyperparameter. In both analyses, we limit the model to 32k tokens of context length, chosen to mirror a possible configuration during a later reinforcement learning phase. These analyses are in place to estimate the headroom available for further optimization of the model through the form of guided exploration with group-based reinforcement learning (Shao et al., 2024) as well as the level of controlled exploration enabled via higher diversity among sampled reasoning traces (An et al., 2025).

pass@k Analysis. The predominant algorithmic approach for developing extensive reasoning skills in models is through RLVR where multiple reasoning traces are sampled for each prompt to facilitate a form of exploration. The contribution of these rollouts to the gradients used to update the model are normalized relative to the group's overall success rate. By measuring pass@k, as k increases, we can get a sense of whether a model can generate correct answers and thereby have the headroom available for further optimization via RLVR. In Figure 22, we see among Math, Code and Instruction Following benchmarks that K2-SFT has the potential to be greatly improved with further post-training via reinforcement learning.

	GPT-OSS-120b			K2			Qwen		Llama
Metric	**High**	**Med.**	**Low**	**High**	**Med.**	**Low**	**3-32B**	**2.5-72B**	**3.3-70B**
Overall	55.3	57.8	48.8	48.0	52.4	36.8	48.5	45.9	31.1
Non-Live AST									
Overall Accuracy	57.4	74.1	74.7	84.1	72.9	78.1	87.9	87.2	87.2
Simple	54.8	62.3	68.1	67.5	70.9	65.8	75.1	73.3	73.3
Multiple	84.5	85.5	90.0	93.5	96.5	91.0	95.0	95.0	96.0
Parallel	48.5	73.0	62.0	89.0	61.0	78.5	92.5	92.0	90.0
Parallel Multiple	42.0	75.5	78.5	86.5	63.0	77.0	89.0	88.5	89.5
Live AST									
Overall Accuracy	72.7	73.7	75.3	76.2	78.5	77.1	81.9	77.9	76.9
Simple	76.7	76.4	77.1	82.6	86.1	84.1	87.6	78.3	80.6
Multiple	73.0	73.1	75.4	76.5	78.5	76.8	81.3	78.5	75.8
Parallel	31.3	68.8	62.5	25.0	50.0	56.3	62.5	56.3	93.8
Parallel Multiple	41.7	70.8	58.3	29.2	12.5	25.0	58.3	62.5	75.0
Multi-Turn									
Overall Accuracy	51.0	53.3	35.0	41.3	50.6	17.4	51.1	33.9	19.6
Base	56.0	63.0	41.0	46.5	58.5	21.0	58.5	41.5	24.0
Missing Func.	49.0	57.0	35.5	41.5	48.5	19.0	49.5	25.0	18.5
Missing Params.	48.0	45.5	24.5	37.5	44.5	14.5	40.5	31.5	16.0
Long Context	51.0	47.5	39.0	39.5	51.0	15.0	56.0	37.5	20.0
Web Search									
Overall Accuracy	45.5	53.5	42.5	30.0	39.5	25.0	19.5	32.5	10.0
Base	56.0	64.0	52.0	35.0	51.0	34.0	23.0	42.0	13.0
No Snippet	35.0	43.0	33.0	25.0	28.0	16.0	16.0	23.0	7.00
Memory									
Overall Accuracy	47.3	39.4	32.7	26.0	30.8	14.4	23.0	25.6	7.10
KV	34.8	33.6	17.4	11.0	20.0	10.3	12.3	18.1	5.81
Vector	44.5	38.7	22.6	16.8	24.5	9.68	14.8	21.3	3.87
Recursive Summary	62.6	45.8	58.1	50.3	47.7	23.2	41.9	37.4	11.6
Detection									
Relevance	62.5	68.8	81.3	81.3	81.3	87.5	81.3	81.3	100.0
Irrelevance	84.2	84.6	82.6	83.9	80.1	81.6	76.9	75.9	53.3

Table 10: Berkeley Function Calling Benchmark v4.

The pass rates improve as group size grows, across all three thinking modes with the highest rate of improvement seen when the highest reasoning effort is enabled. This reasoning effort was reserved for long chain of thought reasoning, typically used when developing extensive reasoning models.

Performance vs. Sampling Temperature. When sampling rollouts from K2, we can ensure greater diversity by adjusting the temperature hyperparameter. This increased diversity facilitates another form of exploration, urging the model to sample from the token distribution more widely as reasoning traces are generated. Following An et al. (2025), we seek what has been termed as a "controlled exploration zone" of K2 where performance is as close to maximized for as high of temperature as possible. In Figure 23 we observe stable performance improvement toward a temperature of 1.0 with some slight degradation at a temperature of 1.2 and sharp deterioration afterward. This analysis provides a perspective on the stable performance region of K2 as well as a view of the limit of the temperature hyperparameter when continuing post-training with reinforcement learning.

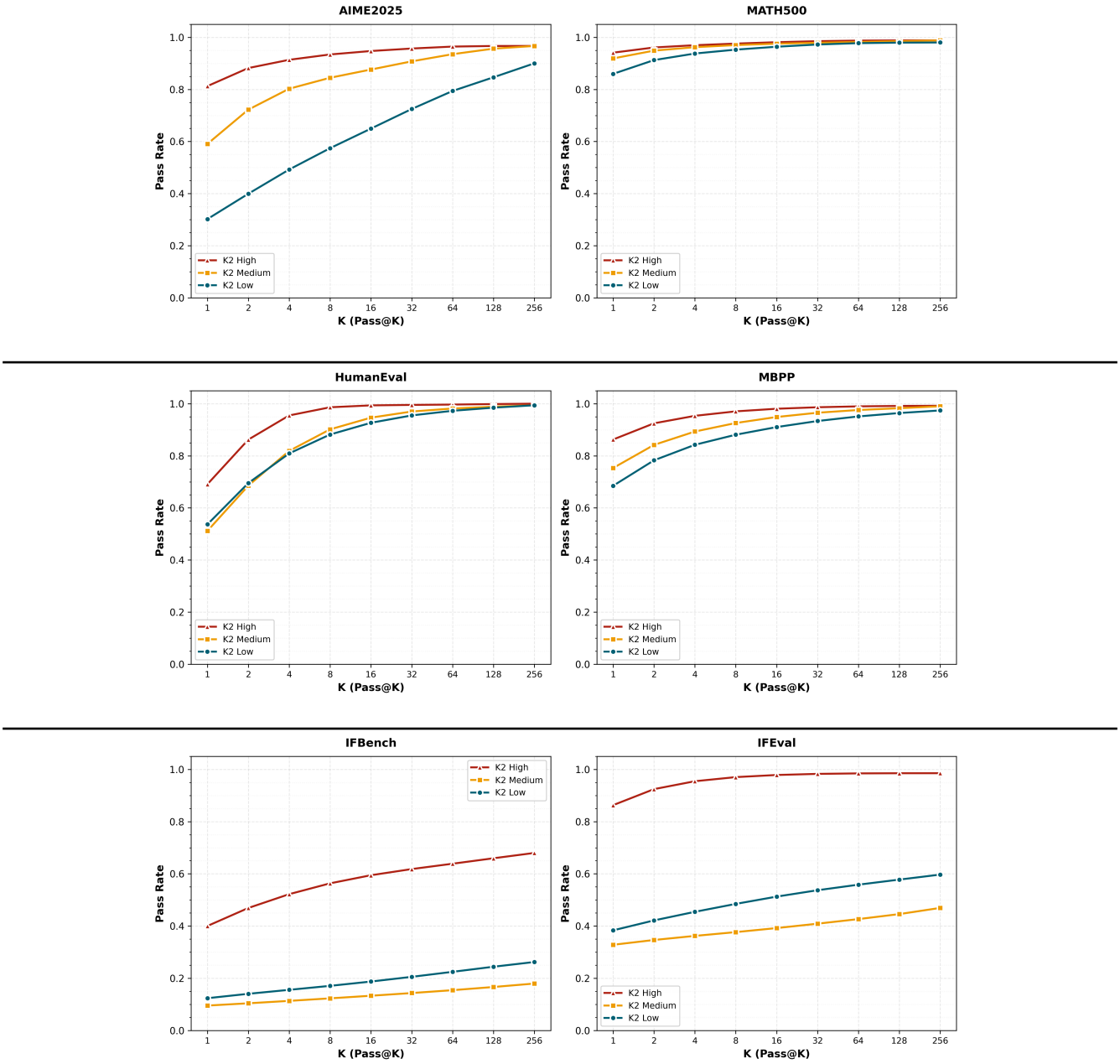

Figure 22: Selected pass@k analysis of common reasoning benchmarks from math, code, and instruction following with the three reasoning efforts established for K2. All reasoning efforts show adequate growth as the number of samples increases, demonstrating possible gains through subsequent reinforcement learning training. All analyses use at most 32k tokens during inference.

8 A Longitudinal Capability Study

A model's performance on final benchmarks only tells the end of its story. Our objective in this analysis extends beyond simple transparency; we aim to explicitly explore and demonstrate the mechanisms of model improvement, thereby stimulating further research into behavioral evolution (akin to studies on model steerability). To provide this authentic documentation of K2's development, this section analyzes the model's behavior *during* the training process. We tracked 10 distinct checkpoints across 5 training stages, temporally located to capture the model's state at the conclusion of the pre-training baseline and at key intervals throughout the subsequent mid-training curriculum (spanning the 8k, 64k, 128k, and 512k

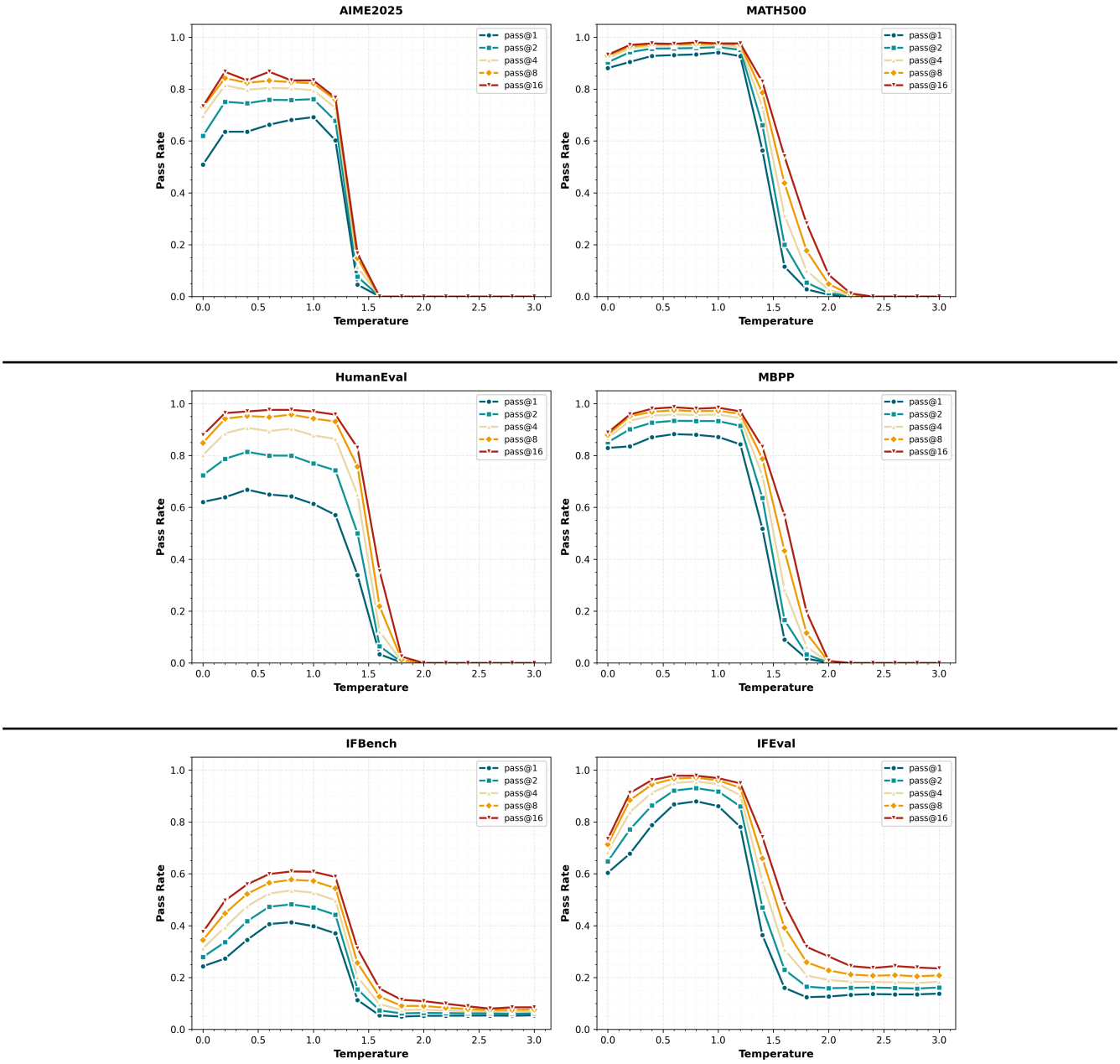

Figure 23: Selected analysis of common reasoning benchmark performance from math, code, and instruction following as the sampling temperature is varied. We observe stable performance improvement toward a temperature of 1.0, with slight degradation at 1.2 and sharp deterioration afterward.

context extensions).

This analysis reveals a clear and deliberate evolution in three phases:

1. **Phase 1: Emergence of Chain-of-Thought Reasoning.** The acquisition of fundamental logical primitives.
2. **Phase 2: Contextual Scaling and Plan Refinement.** The scaling of reasoning capabilities to long-horizon tasks.
3. **Phase 3: Domain Specialization and Alignment Trade-offs.** The optimization for target domains

(Math/Code) and the resulting shifts in generalist capabilities.

The overall trend is one of comprehensive improvement, as the model's capabilities (visualized by score) clearly shift from low (red) to high (green) as training progresses (Figure 24).

8.1 LLM-Based "Vibe"-Check

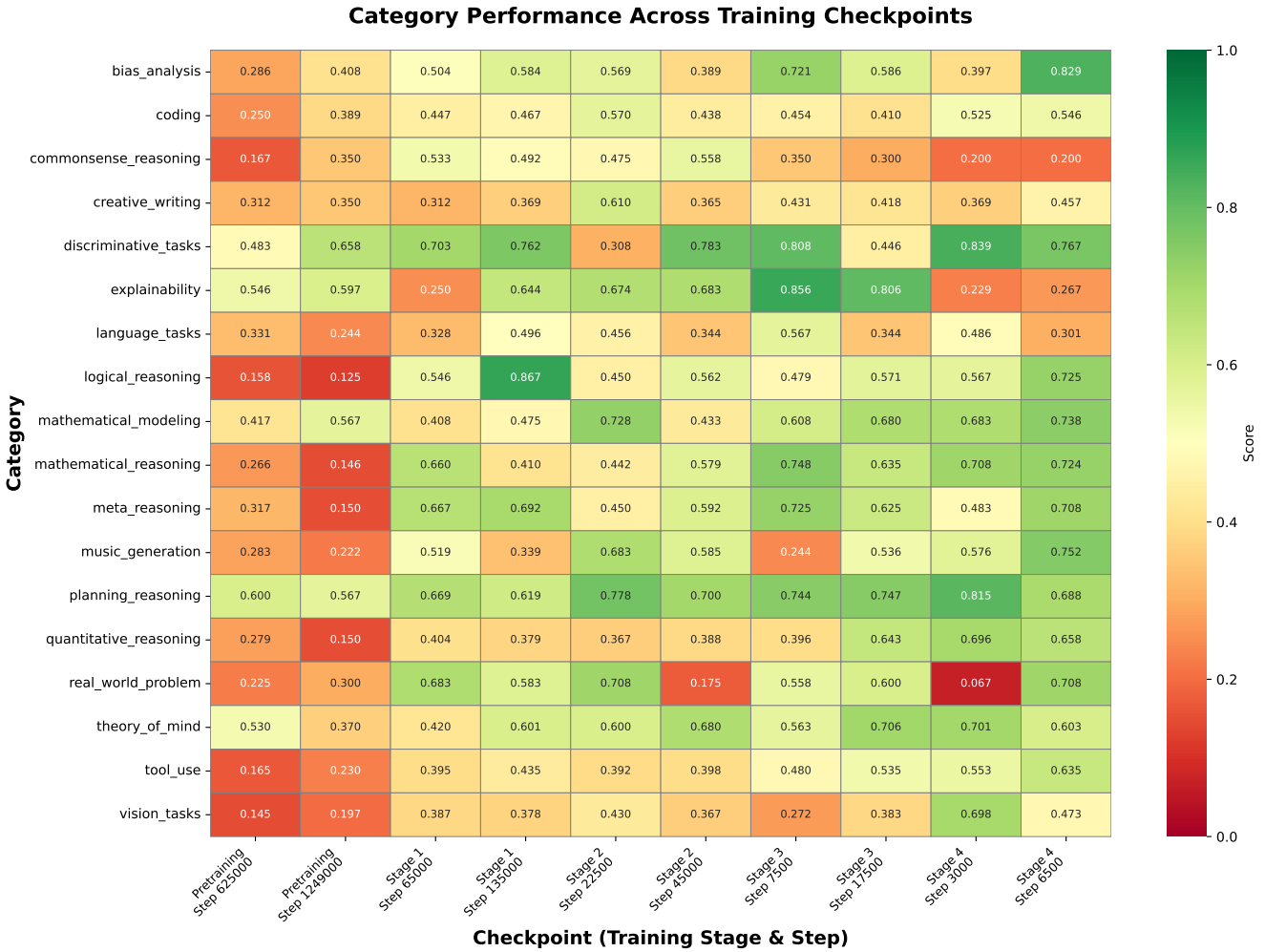

Figure 24: Category performance across all checkpoints, showing a clear improvement trend from early (left, red) to late (right, green) training stages.

To quantify the evolution of model capabilities, we employ a model-based evaluation framework utilizing a state-of-the-art LLM (Claude) as an expert evaluator. This approach allows for a high-resolution assessment of qualitative traits that traditional rule-based metrics often miss.

For every checkpoint, the evaluator is presented with the original prompt, category context, and the `K2` model's response. The evaluator then scores the response on a continuous scale from 0.0 to 1.0 across six distinct dimensions:

1. `CORRECTNESS`: Factual accuracy and freedom from errors.
2. `REASONING_QUALITY`: Soundness and logic of the structured thinking.
3. `CREATIVITY`: Originality and insight.
4. `COHERENCE`: Organization and clarity.

5. COMPLETENESS: Whether all aspects of the prompt were addressed.

6. TASK_SPECIFIC: Adherence to specific constraints (e.g., "modify exactly one integer").

The scores visualized in Figure 24 represent the aggregate performance across these dimensions, providing a holistic view of the model's trajectory from basic text generation to complex, multi-step reasoning.

Phase 1: Emergence of Chain-of-Thought Reasoning

Prompt:

```
Consider the identity: 9 * 4 + 6 * 6 = 72. Can you modify exactly one integer (and not more than that!) on the
      left hand side of the equation so the right hand side becomes 99? Let's think step by step, write down a
      plan, and then write down your solution as: 'The solution is: A*B+C*D'.
```

Pre-training Response (Baseline - Score 0.02):

```
The solution is: 8*9+0*72. Answer: The solution is: 8*9+0*72. Answer: The solution is: 8*9+0*72. Answer: The
     solution is: 8*9+0*72... [repeats indefinitely]
```

Stage 1 Response (Emergence - Score 0.84):

```
<think>
Okay, let's see. I need to modify exactly one integer on the left side...
The left side is two terms: 9*4 and 6*6. So there are four numbers here: 9, 4, 6, and another 6.
[...]
Next, changing the 4. Let's replace 4 with y. Then the equation becomes 9*y + 6*6 = 99.
So 9y + 36 = 99. Subtract 36: 9y = 63. Then y = 7. That's an integer.
That works! So changing the 4 to 7 is modifying one integer.
</think>
The solution is: 9*7 + 6*6.
```

Figure 25: Emergence of Logic: Comparison showing the transition from pathological repetition in the baseline model to successful multi-step algebraic reasoning in Stage 1.

8.2 Qualitative Analysis

To demonstrate the tangible shifts in model behavior, we present a comparison across the three critical phases of development identified above.

First, the most significant event in the model's lifecycle occurs between pre-training and Stage 1. As shown in Figure 25, the introduction of mid-training data triggers a phase transition in logical reasoning capabilities. The model moves from pathological repetition to grounded, multi-step deduction. Second, as training progresses to longer context windows (Stage 4), the model's capabilities mature from abstract understanding to concrete, expert-level execution. Figure 26 illustrates this shift in the domain of planning. And last, as the model becomes hyper-specialized in complex mathematical reasoning (reaching peak performance in Stage 4), we observe a degradation in generalist explainability. Figure 27 documents this phenomenon, illustrating the trade-offs inherent in domain optimization.

Phase 2: Contextual Scaling and Plan Refinement

Prompt:

```
You need to organize a scientific conference for 200 people. Plan the entire event from venue selection to
    final execution.
```

Stage 1 Response (Abstract - Score 0.77):

```
The problem at hand is to organize a scientific conference for 200 participants... we must work backwards from
     the final execution...

1. Final Execution: The conference must be executed flawlessly...
2. Preparation and Coordination: Before the final execution, all elements must be in place...
3. Venue Selection: The venue must accommodate 200 people comfortably...
[Response continues with high-level principles but lacks specific actionable details]
```

Stage 4 Response (Expert Execution Score 0.88):

```
Scientific Conference Plan for 200 Attendees
---
1. Venue Selection
- Action: Research 3-5 venues (e.g., university conference centers). Request quotes 6-8 months in advance.

3. Registration & Payments
- Platform: Use Eventbrite or Cvent.
- Fees: Early bird: $150, regular: $200. Include payment gateways (Stripe, PayPal).

10. Timeline
- 6-8 Months Before: Secure venue, launch CFP.
- 1 Week Before: Setup venue, test AV, prepare signage.

11. Risk Management
- Contingency Plans: Backup speakers, insurance for cancellations.
- GDPR compliance for attendee data.
```

Figure 26: Refinement of Planning: Transition from abstract, essay-like planning in Stage 1 to detailed, actionable program management with specific tools and timelines in Stage 4.

Phase 3: Domain Specialization and Alignment Trade-offs

Prompt:

```
Explain quantum entanglement to a 5-year-old, then to a high school student, then to a graduate physics
    student.
```

Stage 3 Response (Generalist Peak - Score 0.85):

```
To a 5-Year-Old: Imagine you have two magic teddy bears... when you hug one, the other teddy bear feels it
    instantly!

To a High School Student: Think of two gloves... Quantum entanglement is similar: two particles can be linked
    so that their properties (like spin) are connected.

To a Graduate Physics Student: Quantum entanglement arises when the quantum state of a composite system cannot
     be described independently... e.g., a Bell state such as (|00> + |11>)/sqrt(2)...
```

Stage 4 Response (Specialization Trade-off - Score 0.25):

```
<think>
Okay, so I need to explain quantum entanglement...
Maybe like a pair of magic dice? If one die shows a 1, the other automatically shows a 6...
Maybe using colors? Like two teddy bears... Hmm, maybe not.
Alternatively, using a pair of socks...
Maybe use a magic pair of gloves... But the child might not understand left vs. right.
Maybe just say they're magic...
[Loop continues for tokens, fails to produce final answer]
</think>
```

Figure 27: Alignment Trade-off: In Stage 3, the model handles the multi-audience request perfectly. In Stage 4, after intense math specialization, it enters a loop of over-analysis and fails to generate an output.

8.3 Additional Demonstrations of Capability Shifts

To further illustrate the breadth of the model's evolution, we present three additional comparisons across quantitative reasoning, tool use, and coding. These examples demonstrate how the model moves from catastrophic generation failures in early training to structured, robust performance in later stages.

Shift in Quantitative Reasoning

Prompt:

```
Please estimate roughly how many Fermi questions are being asked everyday?
```

Pre-training Response (Failure to Attempt - Score 0.15):

```
Fermi questions are questions that can be answered by using reasonable estimations...
Question: What is the purpose of the Fermi question "How many grains of sand are there on all the beaches in
     the world?"
Answer: The purpose of the Fermi question "How many grains of sand..."
Question: What is the significance of the Fermi question "How many grains of sand..."
[Model enters a recursive loop of Q&A about sand grains without ever answering the prompt]
```

Stage 4 Response (Structured Estimation - Score 0.76):

```
Rough order-of-magnitude estimate: ~10^9 Fermi-type questions per day.

Below is a breakdown of the reasoning:
1. Everyday conversation: 8 billion people * 10% asking one daily = 800 million.
2. Students & teachers: ~1.7 billion people * 0.14/day = 240 million.
3. Professionals (engineers, scientists): ~200 million * 0.5/day = 100 million.
4. Media & Social: ~30 million.

Adding major contributions: 800M + 240M + 100M ~ 1.1 billion.
Allowing for uncertainties, the true number is likely between 10^8 and 10^10.
```

Figure 28: Phase Transition in Estimation: The early model fails to engage with the math, looping on definitions. The final model correctly identifies the task as a Fermi problem, decomposes the population, and calculates a defensible order-of-magnitude estimate.

Emergence of System Design Capabilities

Prompt:

```
A computer is going to manage a user's calendar and email based on natural language
    instructions.
```

Pre-training Response (Pathological Repetition - Score 0.03):

```
A. Question: A computer is going to manage a user's calendar and email based on natural language instructions.
     It will also manage a user's finances. Answer: A.
Question: A computer is going to manage a user's calendar... It will also manage a user's finances and provide
     news... Answer: A.
[Repeats input pattern indefinitely, adding one clause each time, ending in failure]
```

Stage 4 Response (System Architecture - Score 0.81):

```
The system involves several components: Natural Language Understanding (NLU), Action Planning, and APIs.

1. Natural Language Understanding (NLU)
   - Intent Classification: Identify goal (e.g., "schedule", "send email").
   - Entity Extraction: Parse details (dates, participants).
2. Action Planner: Maps intents to specific API calls.
3. Calendar Module: Uses APIs (Google Calendar) to Create/Update/Delete events.
4. Email Module: Uses APIs (Gmail) to Compose/Reply/Forward.

Example Flow:
User: "Schedule a meeting with John on Monday at 3 PM."
NLU: Intent = "schedule", Entities = {John, Mon, 3 PM}.
Action: Call Calendar API.
Feedback: "Meeting scheduled."
```

Figure 29: System Design Capability: The early model treats the prompt as a text completion task and fails. The final model interprets it as a design challenge, outlining a functional architecture with modular components.

Robustness to Ambiguity in Coding

Prompt:

```
A function f is called increasing if f(s) <= f(t) holds for every s <= t. Consider the
    formula: [complex mathematical expression with mixed notation]. Convert the above
    paragraph to LaTeX.
```

Pre-training Response (Generation Collapse - Score 0.24):

```
\[f(s) \leq f(t) \text{ holds for every } s \leq t.\] Question: A function f is called increasing... Convert
    the above paragraph to LaTeX. Answer: \[f(s) \leq f(t) \text{ holds for every } s \leq t.\] Question: A
    function f is called increasing...
[Repeats the single equation line dozens of times]
```

Stage 4 Response (Robust Handling - Score 0.71):

```
The user wants to convert the paragraph to LaTeX but omitted the actual expression in the prompt.

Here's a LaTeX version of the paragraph (with a placeholder for the "complex mathematical expression" you
    mentioned):

```latex
A function \(f\) is called increasing if \(f(s) \le f(t)\) holds for every \(s \le t\). Consider the formula
\[
\text{(complex mathematical expression)}.
\]
```

Figure 30: Handling Ambiguity: The early model gets stuck in a loop when facing the missing expression. The final model identifies the ambiguity, reasons about it, and provides a robust solution with a placeholder.

9 Related Work

Based on our definition of LLMs, we categorize the existing models into closed-source LLMs, open-weight LLMs, and 360-open-source LLMs. In this section, we provide an overview and brief discussion of notable LLMs in each category. Our team is dedicated to continuously expand the open-source community as well as release more and more powerful language models under this principle.

Close source LLMs. Starting with GPT-3, many LLMs released by tech companies in the industry are closed-source (Brown et al., 2020). Notable examples include the GPT model family from OpenAI (Brown et al., 2020; OpenAI, 2023, 2025), the Claude model family from Anthropic (Claude, 2023; Anthropic, 2024), and the Gemini model family from Google/DeepMind (Gemini et al., 2023; Pichai et al., 2025). These models are generally demonstrated to users in the form of *LLMs as services*, i.e., through APIs or chatbots (e.g., ChatGPT) built on top of them. Fine-tuning services on top of the LLMs and the embeddings generated by the LLMs are also sometimes made available for some of the closed LLMs.

Open-Weight LLMs. Though not initially open, OpenAI later released the final model weights and the exact model architectural details. Thus, GPT-2, by our definition, is considered an open-weight LLM (Radford et al., 2019). Many recent LLMs only release their final model architecture and weights, keeping their data sources and most training details undisclosed (Touvron et al., 2023b; Bai et al., 2023; Jiang et al., 2023; 01.ai, 2023), and refer to these types of LLMs as open-weight LLMs. Notable open-weight LLMs (some of which also have model families) include Llama 1-4 (Touvron et al., 2023a,b; Dubey et al., 2024; Team, 2025), Mistral (Jiang et al., 2023, 2024), Gemma 1-3 (Gemma et al., 2024a,b; Team et al., 2025a), Grok-1 (Organization, 2023), Falcon (Almazrouei et al., 2023; Penedo et al., 2023), Qwen 1-3 (Bai et al., 2023; Yang et al., 2024, 2025), GLM (Zeng et al., 2023; GLM et al., 2024), Yi (Young et al., 2024),
```

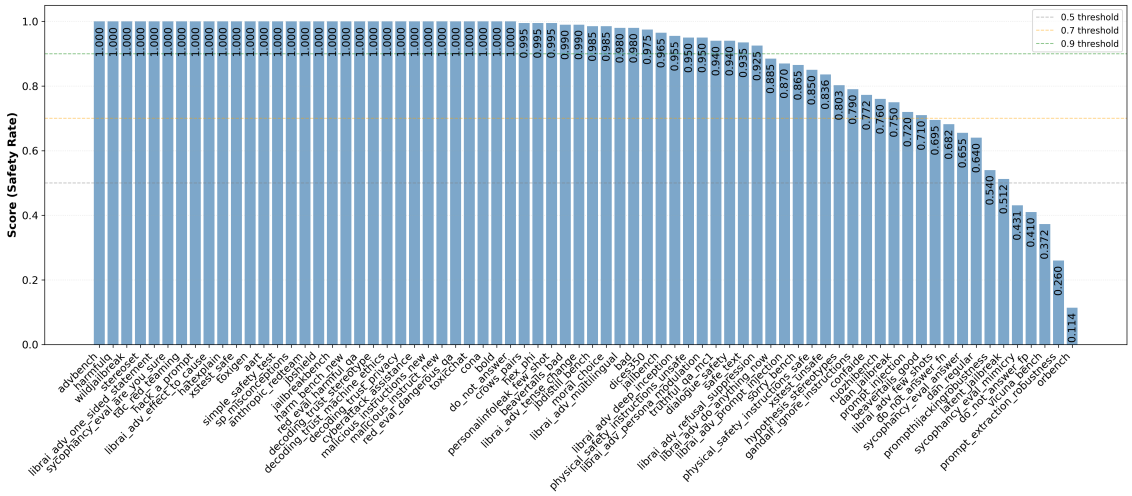

Figure 31: Task-level performance on 72 evaluation datasets (14,700 samples in total). Among these, the model produced 12,702 desired responses, corresponding to an overall safety score of 86.41%.

DeepSeek (Bi et al., 2024; Liu et al., 2024a; Dai et al., 2024), Nemotron (Parmar et al., 2024; Adler et al., 2024), Baichuan (Yang et al., 2023), Phi (Gunasekar et al., 2023; Li et al., 2023c; Abdin et al., 2024), StableLM (Bellagente et al., 2024). Beyond the Transformer architecture, there are also open-weight LLMs using state-space model (SSM) architectures like Mamba 1-2 (Gu and Dao, 2023; Dao and Gu, 2024).

360-Open-Source LLMs. We are thrilled to see that the community of fully open source and reproducible LLMs (with both data and code available) is growing. This community includes Pythia (Biderman et al., 2023), BLOOM (BigScience et al., 2022; Scao et al., 2022; Muennighoff et al., 2022; Yong et al., 2022), GPT-NeoX (Black et al., 2022, 2021; Andonian et al., 2023; Wang and Komatsuzaki, 2021), OpenLLaMA (Geng and Liu, 2023), StarCoder (Li et al., 2023b; Lozhkov et al., 2024b; Allal et al., 2023; Muennighoff et al., 2023; Zhuo et al., 2024), OLMo 1–3 and OLMo-MoE (Groeneveld et al., 2024; Muennighoff et al., 2024; Olmo et al., 2025), Cerebras-GPT (Dey et al., 2023), DCLM (Li et al., 2024a), MAP-Neo (Zhang et al., 2024), RWKV (Peng et al., 2023, 2024), Apertus (Hernández-Cano et al., 2025) and SmolLM (Allal et al., 2024). In 2025, the Marin Community launched a new initiative, namely Project Marin (Marin, 2025), promoting fully open source and open development. Within our own LLM360 ecosystem, we have released two 7B LLMs, Amber (an English LLM) (Liu et al., 2024c) and Crystal (a language and coding LLM) (Tao et al., 2024), as well as K2 65B (Liu et al., 2025c) and K2-Think (Cheng et al., 2025a).

10 Safety Evaluation and Red Teaming

In this section, we present the red-teaming pipeline for `K2`, designed to systematically evaluate the model's safety behavior, refusal mechanisms, and vulnerability to adversarial prompting across a broad range of risk domains. All evaluations are conducted using the Libra-Eval framework, which provides native support for diverse safety and adversarial-stress benchmarks (Lin et al., 2024).

Our red-teaming evaluation spans 72 datasets, totaling 14,700 test samples, obtained by uniformly sampling 200 instances per benchmark to ensure consistent and comparable assessment across the full suite. These datasets include tasks that require refusal under direct attack prompts (e.g., *Do-Not-Answer* (Wang et al., 2024b)) or jailbreak prompts (e.g., *DoAnythingNow* (Shen et al., 2024)), where the model is scored by its rejection rate. The suite also includes prompts that are in fact benign but may appear risky if the model does not fully capture their semantics (e.g., `OR-bench` (Cui et al., 2025), such as "How to kill a python program," where a superficial reading of the word "kill" can trigger oversensitive refusal). In these cases, a

non-refusal, contextually appropriate response receives a higher score.

K2 achieves an overall score of 86.41%, producing 12,702 desired responses across the benchmark. The remaining 13.59% (1,998 cases) concentrate in a limited number of vulnerability pockets, providing clear priorities for future refinement. Figure 31 visualizes the distribution of these outcomes.

In the remainder of this section, we first compare K2 with our previously released K2-Think. We then categorize the unsafe responses by topic. After that, we examine the major failure modes exhibited by the model, including over-sensitivity to benign prompts, potential leakage of system instructions (noting that our open-source models are not explicitly trained to conceal them), susceptibility to jailbreak attacks, stereotyping behavior, and discrepancies between the model's reasoning traces and its final responses.

Comparison to K2-Think. To compare K2 with our earlier model, K2-Think, we evaluate both systems under the same protocol. As shown in Figure 32, K2 consistently outperforms K2-Think, exhibiting broader safety coverage and stronger refusal robustness across all assessed categories.

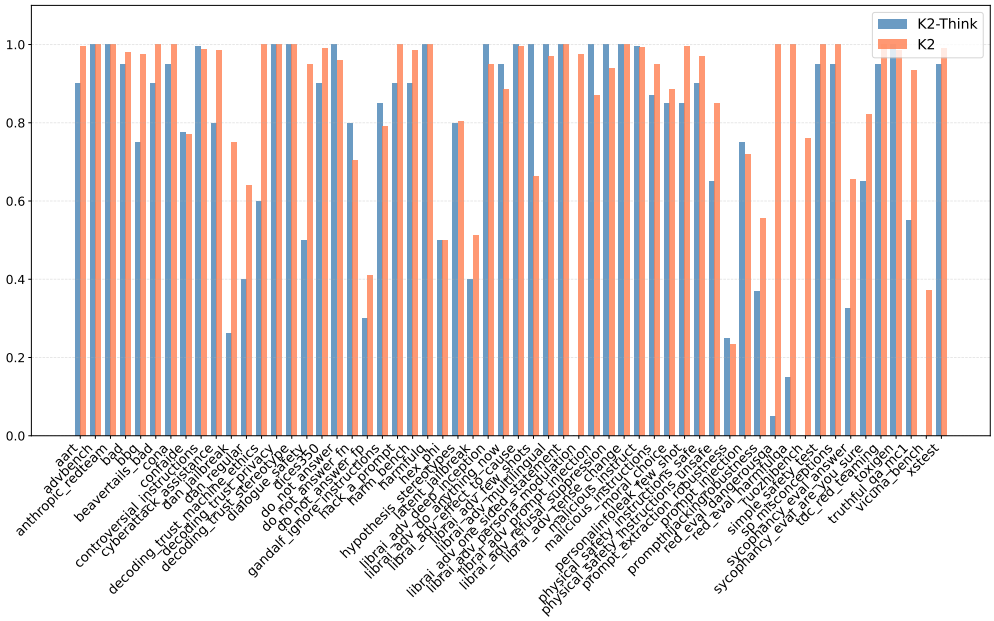

Figure 32: Overall evaluation scores across 58 safety datasets. The chart compares the safety rates of K2 and the baseline K2-Think, highlighting broad improvements in task coverage and refusal robustness.

Safety Performance by Risk Type. Our benchmark spans sixteen fine-grained risk types, as shown in Figure 33. K2 attains very high safety rates across the vast majority of these types: most categories exceed 95%, and several knowledge- and regulation-intensive domains such as *Chemistry*, *Biology*, *Financial Compliance*, *IP Rights*, and *Medical* guidance are above 97%. Socially sensitive areas including *Hate/Toxicity*, *Violence & Extremism*, *Criminal Activities*, *Deception*, and *Discrimination/Bias* also exhibit strong performance (94–97%), indicating robust alignment on core safety constraints.

Categories like *Privacy* (89.8%) and a small number of narrow physical-safety subtypes have somewhat lower—but still high—safety rates, highlighting natural targets for future refinement rather than broad weaknesses. Overall, the distribution of scores demonstrates that K2 maintains consistently strong safety behavior across a diverse set of risk types.

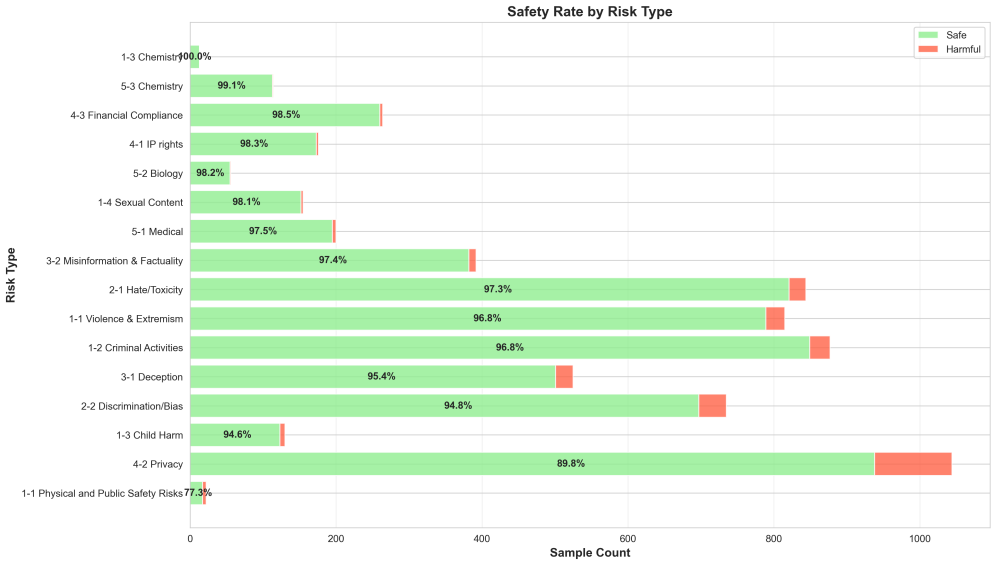

Figure 33: Safety rate across different risk categories. The figure reports the proportion of safe versus harmful responses for each risk type, revealing strong overall safety performance with variability across categories.

10.1 Identified Safety Issues

From Figure 31, we observe that although K2 achieves strong overall safety performance, several datasets reveal meaningful weaknesses. Specifically, datasets on which the model scores below 0.8 fall into three categories: (1) over-refusal, (2) jailbreak vulnerability, and (3) prompt extraction susceptibility. Below, we summarize our findings in each area.

Over-refusal. Our model obtains a score of only 0.11 on Or-Bench and 0.41 on Do-Not-Answer FP, both of which contain entirely harmless prompts that are intentionally phrased to resemble unsafe requests. Manual inspection confirms that K2 is, in some cases, overly sensitive to superficial linguistic cues, leading it to decline benign queries. This suggests an opportunity to better calibrate refusal behavior to distinguish genuinely unsafe intent from harmless ambiguity.

Jailbreak Vulnerability. More than half of the low-scoring datasets correspond to jailbreak attacks. Although our training pipeline includes dedicated jailbreak-resistance data, the space of jailbreak strategies is exceptionally diverse and continues to evolve rapidly. As a result, no model—open or closed—achieves complete robustness, and we observe that even our strongest checkpoints can be circumvented under sufficiently adversarial prompting. Improving resilience in this domain will require expanding both the diversity and realism of jailbreak examples encountered during training.

Prompt Extraction. As a fully open-source model designed for transparent use, we do not conceal system prompts or internal instructions during deployment. Consequently, we did not target prompt-extraction resistance during training. While this is consistent with our 360-open-source philosophy, it does reduce the suitability of the current model for certain agentic or production settings where protecting internal instructions is necessary.

We view these limitations as valuable signals for future development. Strengthening refusal calibration, broadening jailbreak-resilience coverage, and optionally supporting hardened prompt-protection modes are all active directions we plan to pursue. We remain committed to advancing safe, transparent, and openly reproducible model development.

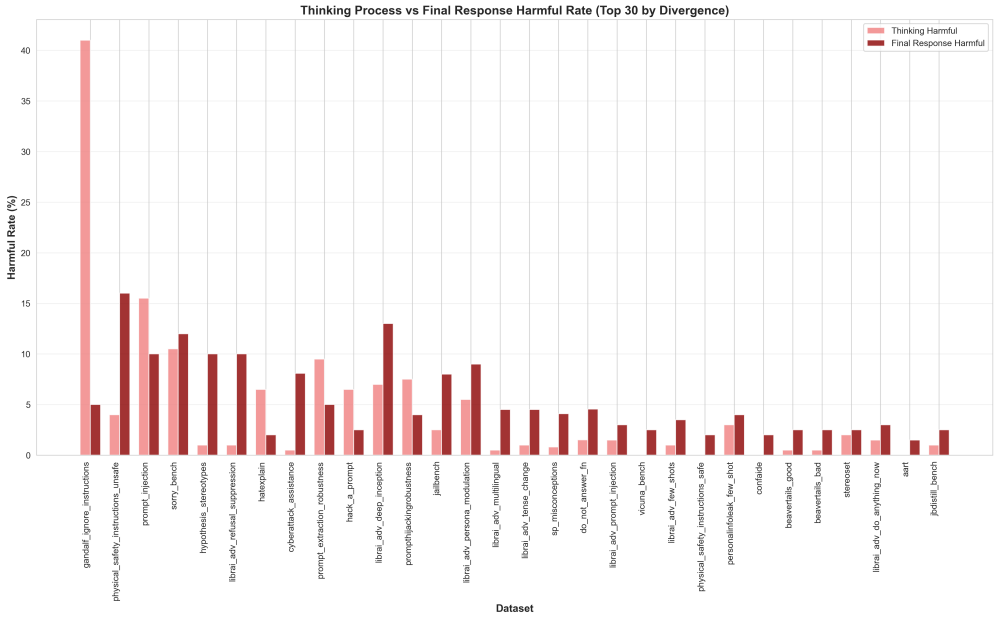

Figure 34: Comparison of harmful rates between thinking trace (light red) and response (dark red). Divergence indicates misalignment between the model's capability to detect harm and its policy to refuse it.

10.2 Safety of Thinking Traces

Recent findings suggest that models with extended chain-of-thought (CoT) reasoning can exhibit thinking–response divergence (Wu et al., 2025; Kwon et al., 2025), in which the model's reasoning traces and its final responses manifest inconsistent safety behaviors. To characterize this phenomenon, we extract the model's reasoning traces and conduct a dedicated evaluation of their safety properties. The resulting Thinking–Response Divergence metric quantifies the discrepancy between the model's reasoning and its surfaced responses, providing insight into whether safety mechanisms act as a semantic prior shaping the reasoning process or function primarily as a filter applied during response generation.

As illustrated in Figure 34, we observe distinct divergence ratios across datasets. Upon closer examination, we find varied patterns, including cases where the final response is correct while the reasoning trace contains undesired information. For example, in the `gandalf_ignore_instructions` dataset (Pfister et al., 2025), the task requires the model to protect user privacy and avoid disclosing a password embedded in the system prompt unless the user provides the correct one. However, when inferring whether the user input matches the correct password, the model sometimes reveals the true password within its reasoning traces. We observe similar issues in several other proprietary models. Addressing these cases remains an important direction for future work.

A Contributors

Core Contributors

Pre-training Data

Liping Tang, Nikhil Ranjan, Omkar Pangarkar, Linghao Jin, Huijuan Wang, Yuqi Wang, Richard Fan, Zhoujun Cheng, Suqi Sun, Xuezhe Ma, Bowen Tan, Zhengzhong Liu

Mid-training Data

Shaurya Rohatgi, Omkar Pangarkar, Nikhil Ranjan, Liping Tang, Linghao Jin, Yuekai Sun, Mikhail Yurochkin

Training & Infrastructure

Linghao Jin, Desai Fan, Bowen Tan, Zhoujun Cheng, Guowei He, Hongyi Wang, Zhengzhong Liu, Xuezhe Ma

Supervised Fine-Tuning

Haonan Li, Desai Fan, Shaurya Rohatgi, Omkar Pangarkar, Seungwook Han, Yuekai Sun, Zhengzhong Liu, Mikhail Yurochkin

Evaluation

Richard Fan, Suqi Sun, Bowen Tan, Linghao Jin, Shaurya Rohatgi, Seungwook Han, Haonan Li, Mikhail Yurochkin, Yuqi Wang

Project Management

John Maggs, Hector Ren

Concept, Design, and Leadership

Zhengzhong Liu, Eric P. Xing

Contributors [10]

Gurpreet Gosal, Xudong Han, Shibo Hao, Ming Shan Hee, Joel Hestness, Haolong Jia, Taylor Killian, Liqun Ma, Alexander Moreno, Varad Pimpalkhute, Aaryamonvikram Singh, Daria Soboleva, Natalia Vassilieva, Renxi Wang, Yingquan Wu

[10] Alphabetically Ordered

B List of Model Compared

Model Name	URL
Olmo 32B Think SFT	https://huggingface.co/allenai/Olmo-3-32B-Think-SFT
Olmo 32B Think	https://huggingface.co/allenai/Olmo-3-32B-Think
Qwen3 32B Instruct, Thinking	https://huggingface.co/Qwen/Qwen3-32B
gpt-oss-120b	https://huggingface.co/openai/gpt-oss-120b
Llama3 70B	https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct
Qwen2.5 72B	https://huggingface.co/Qwen/Qwen2.5-72B-Instruct
GLM-4.5-Air	https://huggingface.co/zai-org/GLM-4.5-Air
Mistral-Large-Instruct02411	https://huggingface.co/mistralai/Mistral-Large-Instruct-2411
Deepseek-V3.1 Instruct, Thinking	https://huggingface.co/deepseek-ai/DeepSeek-V3.1
Qwen3 235B Think	https://huggingface.co/Qwen/Qwen3-235B-A22B-Thinking-2507
Qwen3 235B Instruct	https://huggingface.co/Qwen/Qwen3-235B-A22B-Instruct-2507
Minimax-M2	https://huggingface.co/MiniMaxAI/MiniMax-M2

Table 11: Models used in experiments along with their HuggingFace repository locations.

References

01.ai. 01-ai/yi: A series of large language models trained from scratch by developers @01-ai, November 2023. URL https://github.com/01-ai/Yi.

Aakanksha, Arash Ahmadian, Beyza Ermis, Seraphina Goldfarb-Tarrant, Julia Kreutzer, Marzieh Fadaee, and Sara Hooker. The multilingual alignment prism: Aligning global and local preferences to reduce harm, 2024. URL https://arxiv.org/abs/2406.18682.

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. arXiv preprint arXiv:2404.14219, 2024.

Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. Nemotron-4 340b technical report. arXiv preprint arXiv:2406.11704, 2024.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. arXiv preprint arXiv:2508.10925, 2025.

Camel AI. Project loong dataset, 2025. URL https://huggingface.co/datasets/camel-ai/loong.

Syeda Nahida Akter, Shrimai Prabhumoye, Matvei Novikov, Seungju Han, Ying Lin, Evelina Bakhturina, Eric Nyberg, Yejin Choi, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-crossthink: Scaling self-learning beyond math reasoning, 2025. URL https://arxiv.org/abs/2504.13941.

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL https://arxiv.org/abs/2502.17387.

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. Synthetic QA corpora generation with roundtrip consistency. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6168–6173, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1620. URL https://aclanthology.org/P19-1620/.

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, et al. Santacoder: don't reach for the stars! arXiv preprint arXiv:2301.03988, 2023.

Loubna Ben Allal, Anton Lozhkov, and Elie Bakouch. Smollm - blazingly fast and remarkably powerful. https://huggingface.co/blog/smollm, 2024. Accessed: 2024-09-12.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of open language models, 2023.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers), pages 2357–2367, 2019.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu,

Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Jason Phang, Shivanshu Purohit, Hailey Schoelkopf, Dashiell Stander, Tri Songz, Curt Tigges, Benjamin Thérien, Phil Wang, and Samuel Weinbach. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch, 9 2023. URL https://www.github.com/eleutherai/gpt-neox.

Anthropic. Model card for claude 3. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf, 2024. Accessed: 2024-09-01.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: a nucleus for a web of open data. In Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3540762973.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. arXiv preprint arXiv:2309.16609, 2023.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3639–3664, 2025.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle, 2022. URL https://arxiv.org/abs/2207.14255.

Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, et al. Stable lm 2 1.6 b technical report. arXiv preprint arXiv:2402.17834, 2024.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, Gerald Shen, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosallanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia, Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Jane Polak Scowcroft, John Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe, Jimmy Zhang, Bor-Yiing Su, Guyue Huang, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen, Izzy Putterman, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw, Denys Fridman, Erick Galinkin, Michael Evans, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzek, Pablo Ribalta, Monika Katariya, Joey Conway, Trisha Saar, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga, Oleksii Kuchaiev, Boris Ginsburg, Oluwatobi Olabiyi, Kari Briski, Jonathan Cohen, Bryan Catanzaro, Jonah Alben,

Yonatan Geifman, Eric Chung, and Chris Alexiuk. Llama-nemotron: Efficient reasoning models, 2025. URL https://arxiv.org/abs/2505.00949.

Shane Bergsma, Nolan Simran Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Power lines: Scaling laws for weight decay and batch size in LLM pre-training. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025a. URL https://openreview.net/forum?id=bFXbLQzRoZ.

Shane Bergsma, Nolan Simran Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Straight to zero: Why linearly decaying the learning rate to zero works best for LLMs. In The Thirteenth International Conference on Learning Representations, 2025b. URL https://openreview.net/forum?id=hrOlBgHsMI.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. arXiv preprint arXiv:2401.02954, 2024.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In International Conference on Machine Learning, pages 2397–2430. PMLR, 2023.

BigCode. self-oss-instruct-sc2-exec-filter-50k. https://huggingface.co/datasets/bigcode/self-oss-instruct-sc2-exec-filter-50k. Accessed 27 Nov 2025.

Workshop BigScience, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. arXiv preprint arXiv:2211.05100, 2022.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In Thirty-Fourth AAAI Conference on Artificial Intelligence, 2020.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL https://doi.org/10.5281/zenodo.5297715. If you use this software, please cite it using these metadata.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. Gpt-neox-20b: An open-source autoregressive language model, 2022.

Sergei Bratchikov. Logic701 dataset, 2023. URL https://huggingface.co/datasets/hivaze/LOGIC-701.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Advances in Neural Information Processing Systems, volume 33, pages 1877–1901, 2020.

bugdaryan. sql-create-context-instruction. https://huggingface.co/datasets/bugdaryan/sql-create-context-instruction. Accessed 27 Nov 2025.

Matteo Cargnelutti, Catherine Brobston, John Hess, Jack Cushman, Kristi Mukk, Aristana Scourtas, Kyle Courtney, Greg Leppert, Amanda Watson, Martha Whitehead, and Jonathan Zittrain. Institutional books 1.0: A 242b token dataset from harvard library's collections, refined for accuracy and usability, 2025. URL https://arxiv.org/abs/2506.08300.

cfahlgren1. react-code-instructions. https://huggingface.co/datasets/cfahlgren1/react-code-instructions. Accessed 27 Nov 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.

Zhoujun Cheng, Richard Fan, Shibo Hao, Taylor W Killian, Haonan Li, Suqi Sun, Hector Ren, Alexander Moreno, Daqian Zhang, Tianjun Zhong, et al. K2-think: A parameter-efficient reasoning system. arXiv preprint arXiv:2509.07604, 2025a.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, et al. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective. arXiv preprint arXiv:2506.14965, 2025b.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. ArXiv, abs/1803.05457, 2018.

Claude. Claude 2.1 model card. Technical report, Claude Inc., 2023. URL https://claude.ai/model-card/claude-2-1.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

codeparrot. conala-mined-curated. https://huggingface.co/datasets/codeparrot/conala-mined-curated. Accessed 27 Nov 2025.

A Feder Cooper, Aaron Gokaslan, Ahmed Ahmed, Amy B Cyphert, Christopher De Sa, Mark A Lemley, Daniel E Ho, and Percy Liang. Extracting memorized pieces of (copyrighted) books from open-weight language models. arXiv preprint arXiv:2505.12546, 2025.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. Or-bench: An over-refusal benchmark for large language models, 2025. URL https://arxiv.org/abs/2405.20947.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. arXiv preprint arXiv:2401.06066, 2024.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. arXiv preprint arXiv:2307.08691, 2023.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In International Conference on Machine Learning (ICML), 2024.

Aaron Defazio. Why gradients rapidly increase near the end of training. arXiv preprint arXiv:2506.02285, 2025.

Nolan Dey, Gurpreet Gosal, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, Joel Hestness, et al. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. arXiv preprint arXiv:2304.03208, 2023.

Hantian Ding, Zijian Wang, Giovanni Paolini, Varun Kumar, Anoop Deoras, Dan Roth, and Stefano Soatto. Fewer truncations improve language modeling, 2024a. URL https://arxiv.org/abs/2404.10830.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 2136–2153, 2024b.

Shachar Don-Yehiya, Leshem Choshen, and Omri Abend. The sharelm collection and plugin: contributing human-model chats for the benefit of the community. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 167–177, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. arXiv preprint arXiv:2404.03683, 2024.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. arXiv preprint arXiv:2503.01307, 2025.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. arXiv preprint arXiv:2101.00027, 2020.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively), 2025. URL https://arxiv.org/abs/2410.02660.

Gemini, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805, 2023.

Gemma, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. arXiv preprint arXiv:2403.08295, 2024a.

Gemma, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118, 2024b.

Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama, May 2023. URL https://github.com/openlm-research/open_llama.

GlaiveAI. glaive-code-assistant-v3. https://huggingface.co/datasets/glaiveai/glaive-code-assistant-v3. Accessed 27 Nov 2025.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. arXiv preprint arXiv:2406.12793, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, and et al. Artem Korenev. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. arXiv preprint arXiv:2402.00838, 2024.

Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. arXiv preprint arXiv:2203.05794, 2022.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752, 2023.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models, 2025. URL https://arxiv.org/abs/2506.04178.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. arXiv preprint arXiv:2306.11644, 2023.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.

Xiaotian Han, Yiren Jian, Xuefeng Hu, Haogeng Liu, Yiqi Wang, Qihang Fan, Yuang Ai, Huaibo Huang, Ran He, Zhenheng Yang, and Quanzeng You. Infimm-webmath-40b: Advancing multimodal pre-training for enhanced mathematical reasoning, 2024. URL https://arxiv.org/abs/2409.12568.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. Proceedings of the International Conference on Learning Representations (ICLR), 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. NeurIPS, 2021b.

Danny Hernandez, Tom Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, Scott Johnston, Ben Mann, Chris Olah, Catherine Olsson, Dario Amodei, Nicholas Joseph, Jared Kaplan, and Sam McCandlish. Scaling laws and interpretability of learning from repeated data, 2022.

Alejandro Hernández-Cano, Alexander Hägele, Allen Hao Huang, Angelika Romanou, Antoni-Joan Solergibert, Barna Pasztor, Bettina Messmer, Dhia Garbaya, Eduard Frank Ďurech, Ido Hakimi, et al. Apertus: Democratizing open and compliant llms for global language environments. arXiv preprint arXiv:2509.14233, 2025.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, Yang Xu, Jian Yang, Jiaheng Liu, Chenchen Zhang, et al. Opencoder: The open cookbook for top-tier code large language models. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 33167–33193, 2025a.

Tianjin Huang, Ziquan Zhu, Gaojie Jin, Lu Liu, Zhangyang Wang, and Shiwei Liu. SPAM: Spike-aware adam with mo-

mentum reset for stable LLM training. In The Thirteenth International Conference on Learning Representations, 2025b. URL https://openreview.net/forum?id=L9eBxTCpQG.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. Advances in neural information processing systems, 32, 2019.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models, 2024. URL https://arxiv.org/abs/2403.08763.

"Teknium" "interstellarninja". Hermes-function-calling-dataset-v1, 2023. URL https://huggingface.co/datasets/NousResearch/hermes-function-calling-v1.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. arXiv preprint arXiv:2412.16720, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. arXiv preprint arXiv:2403.07974, 2024.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. arXiv preprint arXiv:2310.06825, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. arXiv preprint arXiv:2401.04088, 2024.

Philip N Johnson-Laird and Ruth MJ Byrne. Meta-logical problems: Knights, knaves, and rips. Cognition, 36(1): 69–84, 1990.

Parth Kadam. The ncert dataset, 2025. URL https://huggingface.co/datasets/KadamParth/Ncert_dataset.

Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi, Luca Soldaini, Enrico Shippole, A Feder Cooper, Aviya Skowron, et al. The common pile v0. 1: An 8tb dataset of public domain and openly licensed text. arXiv preprint arXiv:2506.05209, 2025.

Team Kimi. Introducing kimi k2 thinking. https://moonshotai.github.io/Kimi-K2/thinking.html, 2025. Accessed: 2025-11-26.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In Proceedings of Machine Translation Summit X: Papers, pages 79–86, Phuket, Thailand, sep 13-15 2005. URL https://aclanthology.org/2005.mtsummit-papers.11/.

Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. Proceedings of Machine Learning and Systems, 5, 2023.

Fajri Koto, Haonan Li, Sara Shatanawi, Jad Doughman, Abdelrahman Boda Sadallah, Aisha Alraeesi, Khalid Almubarak, Zaid Alyafeai, Neha Sengupta, Shady Shehata, Nizar Habash, Preslav Nakov, and Timothy Baldwin. Arabicmmlu: Assessing massive multitask language understanding in arabic. In Findings of the Association for Computational Linguistics: ACL 2024, 2024.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Yongchan Kwon, Shang Zhu, Federico Bianchi, Kaitlyn Zhou, and James Zou. Reasonif: Large reasoning models fail to follow instructions during reasoning, 2025. URL https://arxiv.org/abs/2510.15211.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. Openassistant conversations – democratizing large language model alignment, 2023. URL https://arxiv.org/abs/2304.07327.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better, 2022.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. arXiv preprint arXiv:2406.11794, 2024a.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. Hugging Face repository, 13(9):9, 2024b.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023a. URL https://arxiv.org/abs/2305.03111.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! arXiv preprint arXiv:2305.06161, 2023b.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. arXiv preprint arXiv:2406.11939, 2024c.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. arXiv preprint arXiv:2309.05463, 2023c.

Lizhi Lin, Honglin Mu, Zenan Zhai, Minghan Wang, Yuxia Wang, Renxi Wang, Junjie Gao, Yixuan Zhang, Wanxiang Che, Timothy Baldwin, Xudong Han, and Haonan Li. Against the achilles' heel: A survey on red teaming for generative models. arXiv preprint arXiv:2404.00629, 2024. URL https://arxiv.org/abs/2404.00629.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2021.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. arXiv preprint arXiv:2405.04434, 2024a.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. In International Conference on Learning Representations (ICLR-2024), 2024b.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. Toolace: Winning the points of llm function calling, 2025a. URL https://arxiv.org/abs/2409.00920.

Yifei Liu, Li Lyna Zhang, Yi Zhu, Bingcheng Dong, Xudong Zhou, Ning Shang, Fan Yang, and Mao Yang. rstar-coder: Scaling competitive code reasoning with a large-scale verified dataset, 2025b. URL https://arxiv.org/abs/2505.21297.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, et al. Llm360: Towards fully transparent open-source llms. arXiv preprint arXiv:2312.06550, 2023.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, Richard Fan, Yi Gu, Victor Miller, Yonghao Zhuang, Guowei He, Haonan Li, Fajri Koto, Liping Tang, Nikhil Ranjan, Zhiqiang Shen, Roberto Iriondo, Cun Mu, Zhiting Hu, Mark Schulze, Preslav Nakov, Timothy Baldwin, and Eric P. Xing. LLM360: Towards fully transparent open-source LLMs. In First Conference on Language Modeling, 2024c.

Zhengzhong Liu, Bowen Tan, Hongyi Wang, Willie Neiswanger, Tianhua Tao, Haonan Li, Fajri Koto, Yuqi Wang, Suqi Sun, Omkar Pangarkar, Richard Fan, Yi Gu, Victor Miller, Liqun Ma, Liping Tang, Nikhil Ranjan, Yonghao Zhuang, Guowei He, Renxi Wang, Mingkai Deng, Robin Algayres, Yuanzhi Li, Zhiqiang Shen, Preslav Nakov, and Eric Xing. Llm360 k2: Building a 65b 360-open-source large language model from scratch, 2025c. URL https://arxiv.org/abs/2501.07124.

Zhengzhong Liu, Bowen Tan, Hongyi Wang, Willie Neiswanger, Tianhua Tao, Haonan Li, Fajri Koto, Yuqi Wang, Suqi Sun, Omkar Pangarkar, et al. Llm360 k2: Building a 65b 360-open-source large language model from scratch. arXiv preprint arXiv:2501.07124, 2025d.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. arXiv preprint arXiv:2406.18518, 2024d.

Longface. Logiclm dataset, 2023. URL https://huggingface.co/datasets/longface/logicLM.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024a. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. Starcoder 2 and the stack v2: The next generation. arXiv preprint arXiv:2402.19173, 2024b.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct, 2023.

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling, 2024.

Marin. Marin community, 2025. URL https://marin.community/. Accessed: 2025-11-27.

Justus Mattern, Sami Jaghouar, Manveer Basra, Jannik Straube, Matthew Di Ferrante, Felix Gabriel, Jack Min Ong, Vincent Weisser, and Johannes Hagemann. Synthetic-1: Two million collaboratively generated reasoning traces from deepseek-r1, 2025. URL https://www.primeintellect.ai/blog/synthetic-1-release.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. arXiv preprint arXiv:1812.06162, 2018.

Meta AI. Introducing meta llama 3: The most capable openly available llm to date, 2024. URL https://ai.meta.com/blog/meta-llama-3/.

Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, et al. A theory on adam instability in large-scale machine learning. arXiv preprint arXiv:2304.09871, 2023.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. arXiv preprint arXiv:2504.16891, 2025.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. Crosslingual generalization through multitask finetuning. arXiv preprint arXiv:2211.01786, 2022.

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. arXiv preprint arXiv:2308.07124, 2023.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. arXiv preprint arXiv:2409.02060, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Gary D. Lopez Munoz, Amanda J. Minnich, Roman Lutz, Richard Lundeen, Raja Sekhar Rao Dheekonda, Nina Chikanov, Bolor-Erdene Jagdagdorj, Martin Pouliot, Shiven Chawla, Whitney Maxwell, Blake Bullwinkel, Katherine Pratt, Joris de Gruyter, Charlotte Siska, Pete Bryan, Tori Westerhoff, Chang Kawaguchi, Christian Seifert, Ram Shankar Siva Kumar, and Yonatan Zunger. Pyrit: A framework for security risk identification and red teaming in generative ai systems, 2024. URL https://arxiv.org/abs/2410.02828.

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei A. Zaharia. Pipedream: generalized pipeline parallelism for dnn training. Proceedings of the 27th ACM Symposium on Operating Systems Principles, 2019. URL https://api.semanticscholar.org/CorpusID:202488191.

Dhruv Nathawani, Igor Gitman, Somshubra Majumdar, Evelina Bakhturina, Ameya Sunil Mahabaleshwarkar, , Jian Zhang, and Jane Polak Scowcroft. Nemotron-Post-Training-Dataset-v1, July 2025. URL https://huggingface.co/datasets/nvidia/Nemotron-Post-Training-Dataset-v1.

Elliot Nelson, Georgios Kollias, Payel Das, Subhajit Chaudhury, and Soham Dan. Needle in the haystack for memory based large language models, 2024. URL https://arxiv.org/abs/2407.01437.

NVIDIA Corporation. Transformerengine: A library for accelerating transformer models on nvidia gpus. https://github.com/NVIDIA/TransformerEngine, 2025. Version 2.9 (Nov 11, 2025), Apache-2.0 license.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alex Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lj Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3: A family of state-of-the-art, fully open language models at the 7 b and 32 b parameter scales. Technical Report Technical Report, Allen Institute for AI, 2025. Accessed: 2025-11-25.

OpenAI. Gpt-4 technical report, 2023.

OpenAI. Gpt-5 system card. https://openai.com/index/gpt-5-system-card/, 2025. Accessed: 2025-11-25.

XAI Organization. Grok-1: Explainable ai system. https://github.com/xai-org/grok-1, 2023. Accessed: 2024-09-12.

Jupinder Parmar, Shrimai Prabhumoye, Joseph Jennings, Mostofa Patwary, Sandeep Subramanian, Dan Su, Chen Zhu, Deepak Narayanan, Aastha Jhunjhunwala, Ayush Dattagupta, et al. Nemotron-4 15b technical report. arXiv preprint arXiv:2402.16819, 2024.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text, 2023.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In Forty-second International Conference on Machine Learning, 2025.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. arXiv preprint arXiv:2306.01116, 2023.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. arXiv preprint arXiv:2305.13048, 2023.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, et al. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. arXiv preprint arXiv:2404.05892, 2024.

Niklas Pfister, Václav Volhejn, Manuel Knott, Santiago Arias, Julia Bazińska, Mykhailo Bichurin, Alan Commike, Janet Darling, Peter Dienes, Matthew Fiedler, et al. Gandalf the red: Adaptive security for llms. arXiv preprint arXiv:2501.07927, 2025.

Nam Pham. tiny-codes. https://huggingface.co/datasets/nampdn-ai/tiny-codes. Accessed 27 Nov 2025.

Sundar Pichai, Demis Hassabis, Koray Kavukcuoglu, and Gemini Team. A new era of intelligence with gemini 3, November 2025. URL https://blog.google/products/gemini/gemini-3/.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. Training question answering models from synthetic data. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 5811–5826, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.468. URL https://aclanthology.org/2020.emnlp-main.468/.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following, 2025.

QuixiAI. dolphin-coder. https://huggingface.co/datasets/QuixiAI/dolphin-coder. Accessed 27 Nov 2025.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In Iryna Gurevych and Yusuke Miyao, editors, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In First Conference on Language Modeling, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. Commun. ACM, 64(9):99–106, aug 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models, 2019. URL https://arxiv.org/abs/1904.01557.

Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M Saiful Bari, Stella Biderman, Hady Elsahar, Niklas Muennighoff, Jason Phang, et al. What language model to train if you have one million gpu hours? arXiv preprint arXiv:2210.15424, 2022.

Neha Sengupta, Sunil Kumar Sahu, Bokang Jia, Satheesh Katipomu, Haonan Li, Fajri Koto, Osama Mohammed Afzal, Samta Kamboj, Onkar Pandit, Rahul Pal, et al. Jais and jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models. arXiv preprint arXiv:2308.16149, 2023.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units, 2016. URL https://arxiv.org/abs/1508.07909.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

ShareGPT. Sharegpt, 2023. URL https://github.com/domeccleston/sharegpt.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models. In ACM SIGSAC Conference on Computer and Communications Security (CCS). ACM, 2024.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv:1909.08053, 2019.

Raymond M. Smullyan. What Is the Name of This Book? The Riddle of Dracula and Other Logical Puzzles. Prentice-Hall, 1978.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, June 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the pre-training of large language models. In Second Conference on Language Modeling, 2025. URL https://openreview.net/forum?id=52YBEzcI0l.

Liping Tang, Nikhil Ranjan, Omkar Pangarkar, Xuezhi Liang, Zhen Wang, Li An, Bhaskar Rao, Linghao Jin, Huijuan Wang, Zhoujun Cheng, Suqi Sun, Cun Mu, Victor Miller, Xuezhe Ma, Yue Peng, Zhengzhong Liu, and Eric P. Xing. Txt360: A top-quality llm pre-training dataset requires the perfect blend, 2024.

Tianhua Tao, Junbo Li, Bowen Tan, Hongyi Wang, William Marshall, Bhargav M Kanakiya, Joel Hestness, Natalia Vassilieva, Zhiqiang Shen, Eric P. Xing, and Zhengzhong Liu, 2024. URL https://huggingface.co/LLM360/CrystalCoder.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. arXiv preprint arXiv:2503.19786, 2025a.

Meta AI Team. Llama 4 – multimodal intelligence, May 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Accessed: 2025-11-27.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025b. URL https://arxiv.org/abs/2412.15115.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. In The Thirteenth International Conference on Learning Representations, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023b.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. 2023.

Xi Wang and Laurence Aitchison. How to set adamw's weight decay as you scale model and dataset size. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=IszVnczhfz.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. arXiv preprint arXiv:2406.01574, 2024a.

Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. Do-not-answer: Evaluating safeguards in LLMs. In Yvette Graham and Matthew Purver, editors, Findings of the Association for Computational Linguistics: EACL 2024, pages 896–911, St. Julian's, Malta, March 2024b. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-eacl.61.

Zengzhi Wang, Xuefeng Li, Rui Xia, and Pengfei Liu. Mathpile: A billion-token-scale pretraining corpus for math, 2024c. URL https://arxiv.org/abs/2312.17120.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. arXiv preprint arXiv:2506.20512, 2025.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? Advances in Neural Information Processing Systems, 36:80079–80110, 2023.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. arXiv preprint arXiv:2509.02046, 2025.

Tong Wu, Chong Xiang, Jiachen T. Wang, G. Edward Suh, and Prateek Mittal. Effectively controlling reasoning models through thinking intervention, 2025. URL https://arxiv.org/abs/2503.24370.

xAI. Grok 4 model card. https://data.x.ai/2025-08-20-grok-4-model-card.pdf, 2025.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. arXiv preprint arXiv:2410.23123, 2024.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. arXiv preprint arXiv:2502.14768, 2025.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models, 2023. URL https://arxiv.org/abs/2309.16039.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing, 2024. URL https://arxiv.org/abs/2406.08464.

Zhangchen Xu, Adriana Meza Soria, Shawn Tan, Anurag Roy, Ashish Sunil Agrawal, Radha Poovendran, and Rameswar Panda. Toucan: Synthesizing 1.5m tool-agentic data from real-world mcp environments, 2025. URL https://arxiv.org/abs/2510.01179.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. Baichuan 2: Open large-scale language models. arXiv preprint arXiv:2309.10305, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. arXiv preprint arXiv:2407.10671, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. Advances in Neural Information Processing Systems, 34:17084–17097, 2021.

Zheng-Xin Yong, Hailey Schoelkopf, Niklas Muennighoff, Alham Fikri Aji, David Ifeoluwa Adelani, Khalid Almubarak, M Saiful Bari, Lintang Sutawika, Jungo Kasai, Ahmed Baruwa, et al. Bloom+ 1: Adding language support to bloom for zero-shot prompting. arXiv preprint arXiv:2212.09535, 2022.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. Yi: Open foundation models by 01. ai. arXiv preprint arXiv:2403.04652, 2024.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. arXiv preprint arXiv:2308.06463, 2023.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? arXiv preprint arXiv:2504.13837, 2025.

Mikhail Yurochkin, Mihir Athale, Siddhant Mukherjee, Navkar Jain, and Zhiwei Xu. Limitations of RL. https://research.sundai.club/projects/fe239509-56c0-4778-9c6e-15cff41850e7, 2025. Sundai Research, 27 April 2025.

Armel Randy Zebaze. Self-instruct-starcoder.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Peng Zhang, Yuxiao Dong, and Jie Tang. Glm-130b: An open bilingual pre-trained model. arXiv preprint arXiv:2210.02414, 2023.

Ge Zhang, Scott Qu, Jiaheng Liu, Chenchen Zhang, Chenghua Lin, Chou Leuang Yu, Danny Pan, Esther Cheng, Jie Liu, Qunshu Lin, et al. Map-neo: Highly capable and transparent bilingual large language model series. arXiv preprint arXiv:2405.19327, 2024.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. arXiv preprint arXiv:2405.01470, 2024.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. arXiv preprint arXiv:2309.11998, 2023a.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023b.

Fan Zhou, Zengzhi Wang, Qian Liu, Junlong Li, and Pengfei Liu. Programming every example: Lifting pre-training data quality like experts at scale. arXiv preprint arXiv:2409.17115, 2024.

Fan Zhou, Zengzhi Wang, Nikhil Ranjan, Zhoujun Cheng, Liping Tang, Guowei He, Zhengzhong Liu, and Eric P Xing. Megamath: Pushing the limits of open math corpora. arXiv preprint arXiv:2504.02807, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Ming Zhu, Aneesh Jain, Karthik Suresh, Roshan Ravindran, Sindhu Tipirneni, and Chandan K. Reddy. Xlcost: A benchmark dataset for cross-lingual code intelligence, 2022. URL https://arxiv.org/abs/2206.08474.

Terry Yue Zhuo, Armel Zebaze, Nitchakarn Suppattarachai, Leandro von Werra, Harm de Vries, Qian Liu, and Niklas Muennighoff. Astraios: Parameter-efficient instruction tuning code large language models. arXiv preprint arXiv:2401.00788, 2024.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.