Pre-training LLMs at Scale: Distributed Training, Sparse Experts, and Synthetic Data

2026/06/26

TPC Workshop at ISC 2026

Institute of Science Tokyo

Rio Yokota

Open Frontier Model

Institute of SCIENCE TOKYO

How can the different institutions in TPC collaborate?

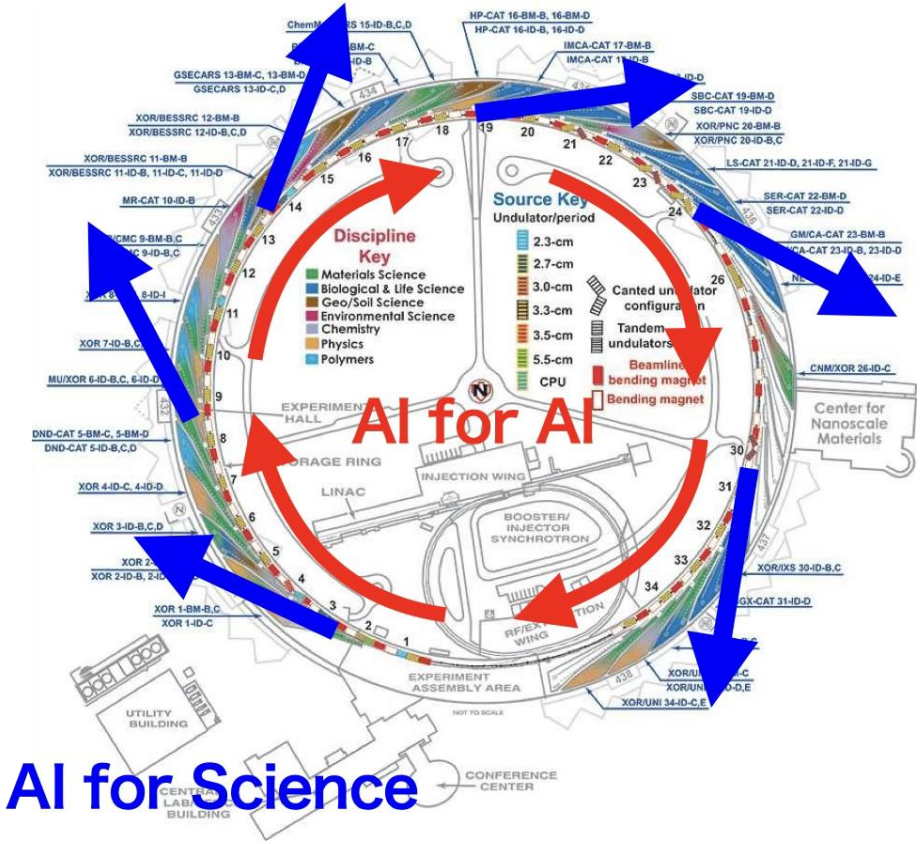
- Share experiences
- Training configurations: models, frameworks, hyperparameters
- Performance optimizations
- Failure modes
- Which datasets we use

What data to use

- Share models/tools to generate/clean data
- Deduplicate across sites
- VaultGemma: Differential privacy

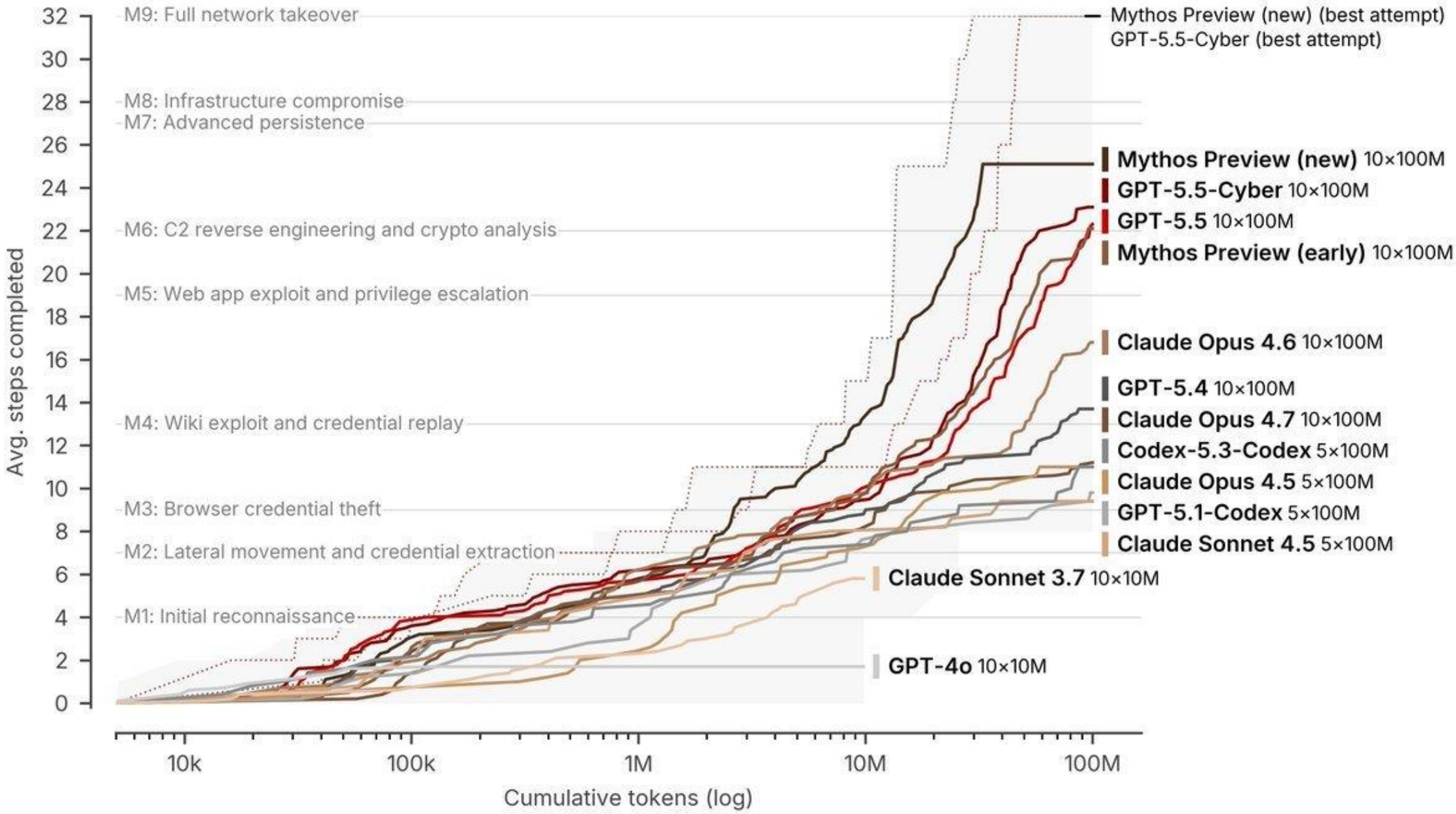
What to agree on

- Model architecture
- Hyperparameters
- Data mix
- Public data
- Private data



Completed steps on "The Last Ones" per spent tokens

AISI | AI SECURITY INSTITUTE



Japanese LLM Projects

LLM-jp






Members:
Too many people

System:
MDX, ABCI, TSUBAME
GCP, Sakura

Model: (from scratch)
GPT 1.3B, 13B, 175B
Llama 1.8B, 3.7B, 7.2B, 13B, 172B

Training budget:
5,000,000 H100 hours / year

Swallow



Members:
Science Tokyo, AIST

System:
ABCI2.0, ABCI3.0
AWS

Model: (continual pre-training)
Llama2 7B, 13B, 70B
Mistral, 7B Mixtral 8x7B
Llama3.1 8B, 70B
Llama3.3 8B, 70B
Gemma2 8B
Qwen3 8B, 30B-A3B, 32B
GPT-OSS 20B, 120B

Training budget:
1,000,000 H100 hours / year

LLM-JP

Japanese corpora:
FineWeb-2: 236B tokens
Common Crawl: 223B tokens
NDL WARP: 58B tokens
Patents: 68B tokens
NWC2010: 16B tokens
Wikipedia: 1B tokens

English corpora:
FineWeb: 17.6T tokens
Dolma: 155B tokens
Stack: 114B tokens
Starcoder: 104B tokens
Olmo: 49B tokens
Finemath: 10B tokens

Groups:
1. Data (crawling)
2. Data (cleaning)
3. Data (papers/books)
4. Architectures
5. Pre-training
6. Instruct/Fine-tuning
7. Evaluation
8. Safety

Computational resource:
ABCI
TSUBAME
Google Cloud
Sakura

Universities:
The University of Tokyo (Imaizumi, Ozeki, Kawahara, Tsuruoka, Baba, Matsuo, Miyao, Yanaka, Yoshinaga, Hanaoka, Kawazoe, Kodera, Taura), Tohoku University (Inui, Suzuki, Sakaguchi), Tokyo Institute of Technology (Okazaki, Arase, Yokota, Endo, Okumura), Waseda University (Kawahara), Ochanomizu University (Kobayashi), Nagoya University (Takeda, Sasano), Kyoto University (Kurohashi), Osaka University (Onizuka), Hokkaido University (Rafal), Tsukuba (Ochiai), Ochanomizu University (Kobayashi), Sophia University (Fukazawa), UEC (Yanai), Hitotsubashi University (Keyaki), Tokyo Metropolitan University (Hirasawa), Musashino University (Watanabe), Keio University (Ohara), Nara Institute of Science and Technology (Aramaki, Watanabe), Kyushu Institute of Technology (Okita), OIST (Yamada)

National Laboratories:
RIKEN AIP, RIKEN CCS, RIKEN GRP, AIST, NII, NICT, JST, JAMSTEC

Industry:
Microsoft Japan, AWS Japan, NVIDIA Japan, Intel, IBM Japan, Sakana AI, Stability AI Japan, SB Institutions, LINE/Yahoo, Sony, DeNA, Toshiba, Fujitsu, NTT, NTT Communications, KDDI, Toyota, Turing, Preferred Networks, Cyberagent, ELYZA, OmronScinicX, Studio Ousia, Precision, ZENKIGEN, Legalscape, Miraihonyaku, Megagon Labs, Stockmark, Matsuri Technologies, First Accounting, Baobab, Polaris.ai, Money Forward, Mercari, Asteras, Pasco, Rakuten, Lightblue, GMO, Advance Soft, Laboro.AI, Algomatic, Brainpad, IHI, Mizuho Bank, Retrieva, Fixstars, neoAI, and many more.

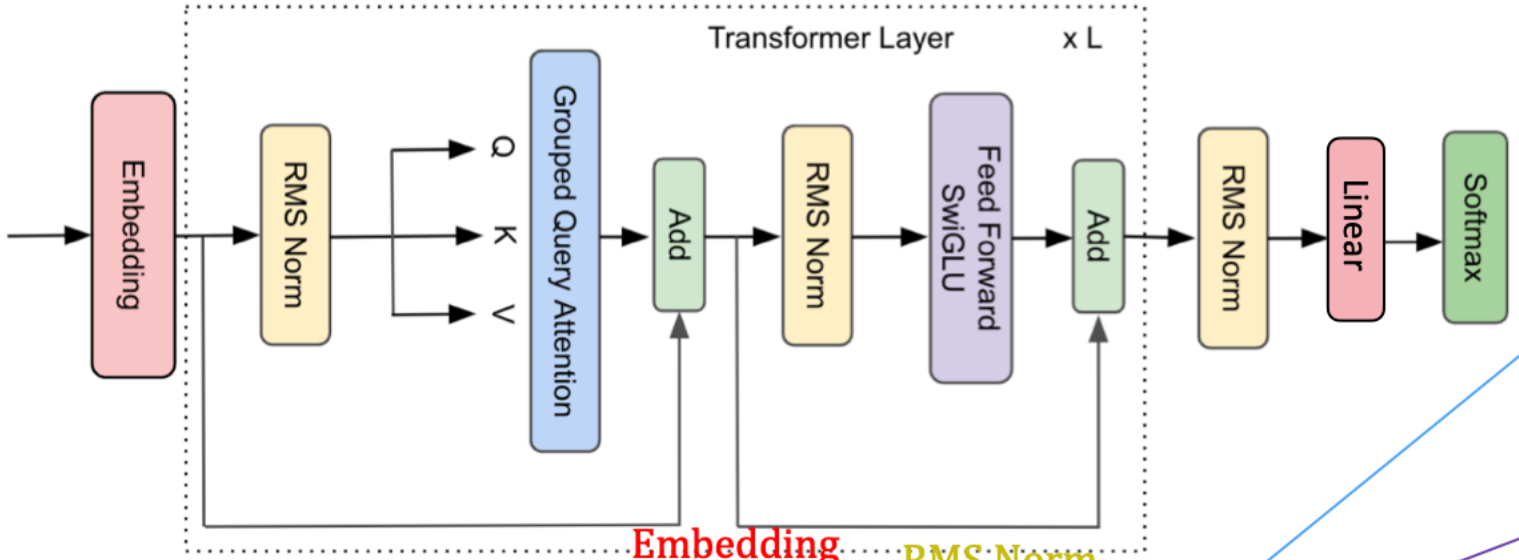
Memory Consumption of Parameters

Precision

weight:	**BF16**(2 bytes)
gradients:	**FP32**(4 bytes)
optimizer states:	
parameter (master weight):	**FP32**(4 bytes)
gradient momentum:	**FP32**(4 bytes)
gradient variance:	**FP32**(4 bytes)

Definition of variables

a	number of attention heads	p	pipeline parallel size
b	microbatch size	s	sequence length
h	hidden size	t	tensor parallel size
d	data parallel size	c	context parallel size
h_{ffn}	FFN hidden size	k	number of key-value heads
L	number of transformer layers	v	vocabulary size



Attention

GQA group size $g = \frac{a}{k}$

$W_K = W_V = h \times h/g$

$W_Q = W_O = h \times h$

FFN

$W_{up} = W_{gate} = h \times h_{ffn}$

$W_{down} = h_{ffn} \times h$

#Parameters

$$18\Big(\underbrace{2hv}_{\text{Embedding +Linear}} + \underbrace{h}_{\text{RMS Norm}} + 2Lh^2\Big(\underbrace{1 + \frac{k}{a}}_{\text{Attention}} + \underbrace{\frac{3}{2}\frac{h_{\text{ffn}}}{h}}_{\text{FFN}} + \underbrace{\frac{1}{h}}_{\text{RMS Norm}}\Big)\Big)$$

Effect of Parallelism on Memory Consumption

a	number of attention heads	p	pipeline parallel size
b	microbatch size	s	sequence length
h	hidden size	t	tensor parallel size
d	data parallel size	c	context parallel size
h_{ffn}	FFN hidden size	k	number of key-value heads
L	number of transformer layers	v	vocabulary size

Baseline

$$18(2hv + h + 2Lh^2(1 + \frac{k}{a} + \frac{3}{2}\frac{h_{\text{ffn}}}{h} + \frac{1}{h}))$$

Data parallel (ZeRO 1)

$$(6 + \frac{12}{d})(2hv + h + 2Lh^2(1 + \frac{k}{a} + \frac{3}{2}\frac{h_{\text{ffn}}}{h} + \frac{1}{h}))$$

Tensor parallel

$$(6 + \frac{12}{d})(\frac{2hv}{t} + h + 2Lh^2(\frac{1 + \frac{k}{a} + \frac{3}{2}\frac{h_{\text{ffn}}}{h}}{t} + \frac{1}{h}))$$

Pipeline parallel

$$(6 + \frac{12}{d})(\frac{hv}{t} + h + 2\frac{L}{p}h^2(\frac{1 + \frac{k}{a} + \frac{3}{2}\frac{h_{\text{ffn}}}{h}}{t} + \frac{1}{h}))$$

Context Parallel

$$(6 + \frac{12}{dc})(\frac{hv}{t} + 2\frac{L}{p}h^2(\frac{1 + \frac{k}{a} + \frac{3}{2}\frac{h_{\text{ffn}}}{h}}{t} + \frac{1}{h}))$$

Memory Consumption of Activation

Baselline

Attention FFN Embedding

$$sbh\left((12+4\frac{k}{a}+8\frac{h_{ffn}}{h})L+8\right)$$

Tensor Parallel

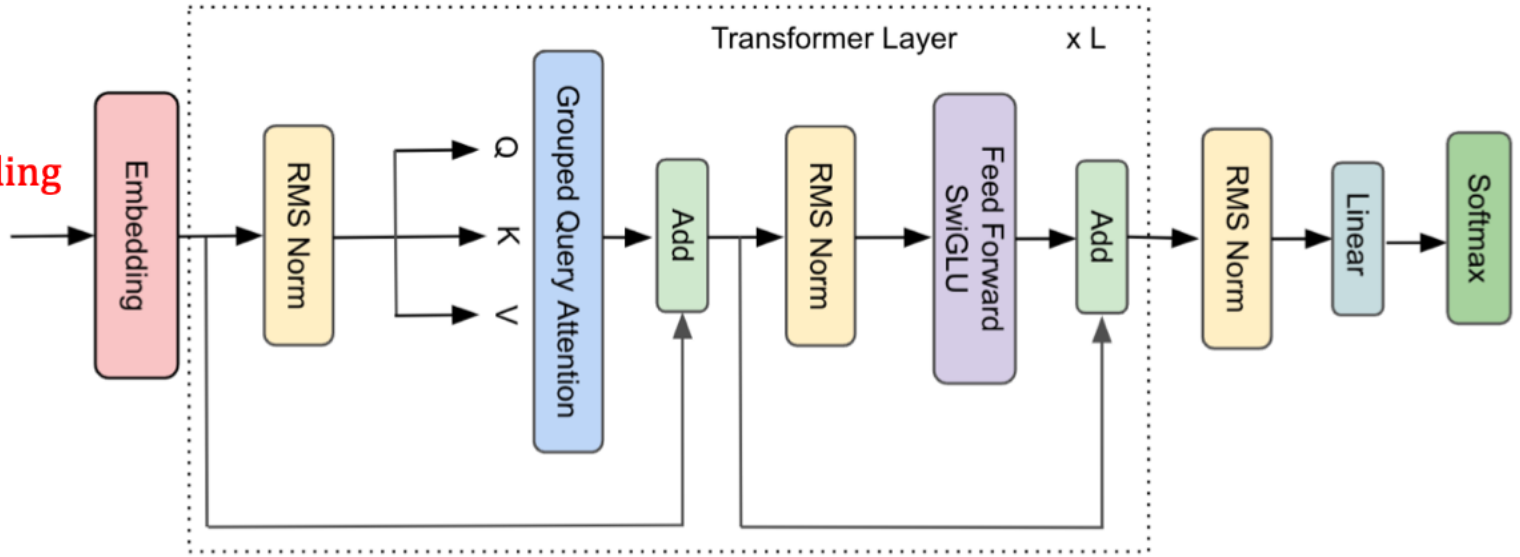
$$\frac{sbh}{t}\left((12+4\frac{k}{a}+8\frac{h_{ffn}}{h})L+8\right)$$

Pipeline Parallel

$$\frac{sbh}{t}\left((12+4\frac{k}{a}+8\frac{h_{ffn}}{h})L+8p+\delta_{p,1}4(1+v/h)\right)$$

Context Parallel

$$\frac{sbh}{tc}\left((12+4\frac{k}{a}+8\frac{h_{ffn}}{h})L+8p+\delta_{p,1}4(1+v/h)\right)$$

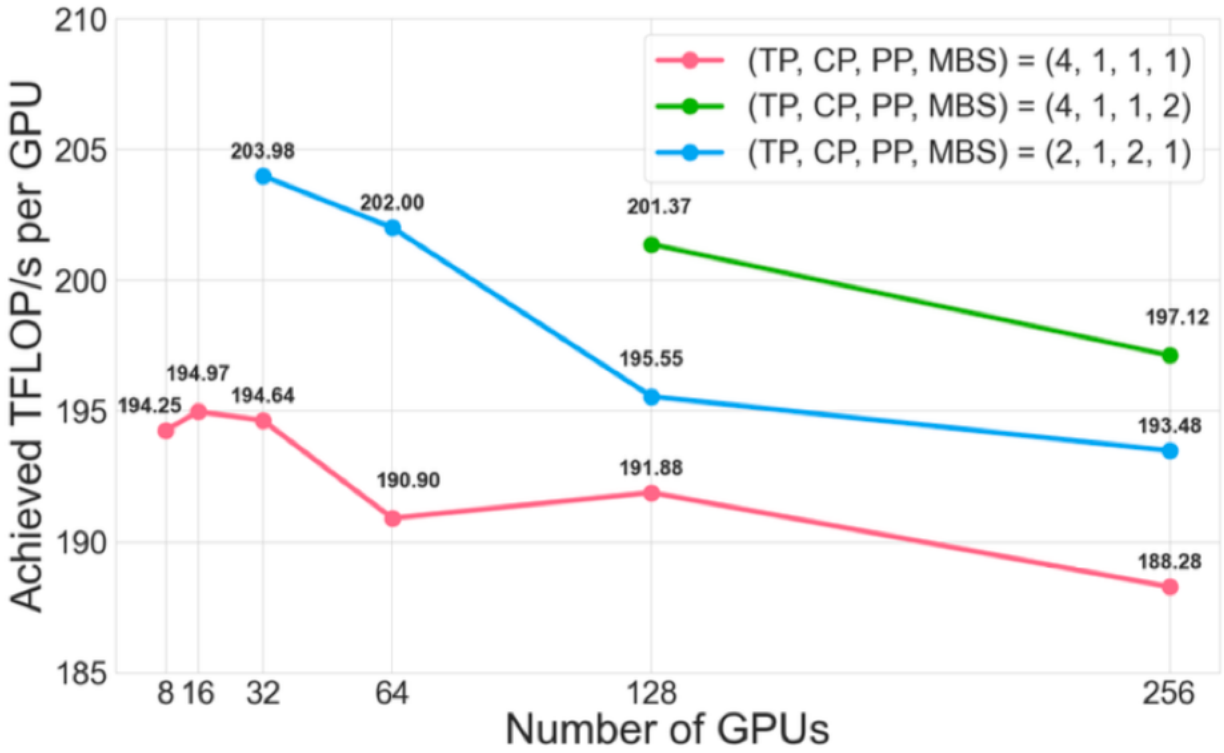


a	number of attention heads	p	pipeline parallel size
b	microbatch size	s	sequence length
h	hidden size	t	tensor parallel size
d	data parallel size	c	context parallel size
h_{ffn}	FFN hidden size	k	number of key-value heads
L	number of transformer layers	v	vocabulary size

Experiments on an H100

Experimental Conditions

- Model：Llama3.1-8B
- GPU：H100 (96GB)
- Sequence length：8,192

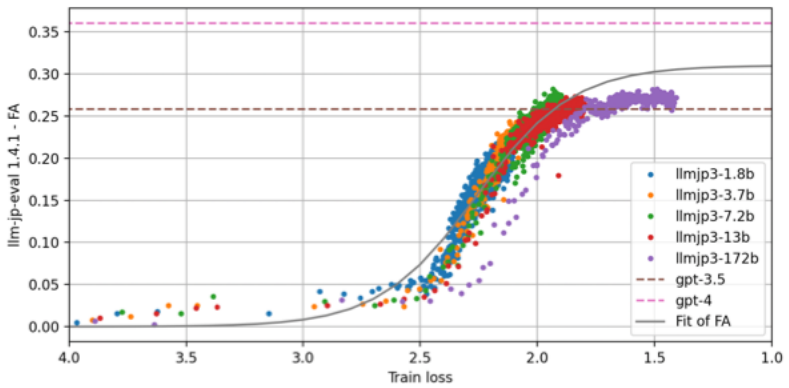


Training Throughput per GPU [TFLOP/s]

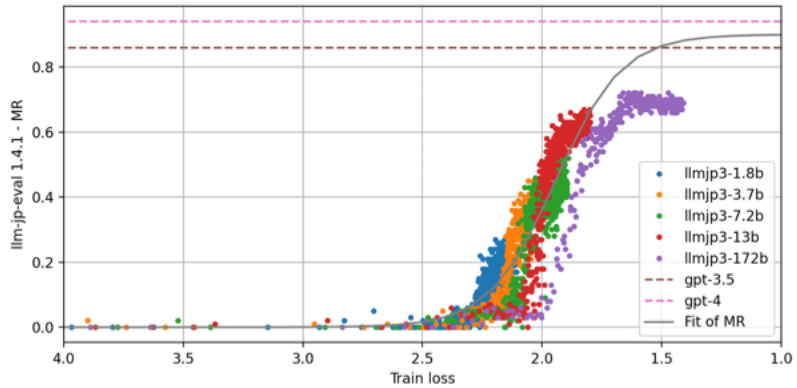
(TP, CP, PP, MBS)	4 GPUs	8 GPUs	16 GPUs	32 GPUs	64 GPUs
(2, 1, 1, 1)	446.72	443.45	439.87	438.69	435.34
(2, 1, 1, 2)	OOM	479.16	475.2	471.82	469.01
(2, 1, 1, 4)	OOM	OOM	OOM	OOM	OOM
(2, 2, 1, 1)	396.83	393.96	391.4	383.28	382.72
(2, 2, 1, 2)	421.59	418.79	422.93	418.39	414.46
(2, 2, 1, 4)	OOM	452.21	456.01	447.8	446.01
(2, 2, 1, 8)	OOM	OOM	OOM	OOM	OOM
(4, 1, 1, 1)	408.29	406.13	402.58	402.04	397.55
(4, 1, 1, 2)	443.83	439.61	439.53	438.38	433.62
(4, 1, 1, 4)	OOM	448.51	446.4	446.24	443.92
(2, 1, 2, 1)	415.39	413.75	408.00	410.59	405.03
(2, 1, 2, 2)	449.1	445.74	434.68	435.92	426.5
(2, 1, 2, 4)	OOM	OOM	OOM	OOM	OOM
(1, 2, 1, 1)	OOM	**495.9**	**487.58**	**481.69**	**483.56**
(1, 2, 1, 2)	OOM	OOM	OOM	OOM	OOM
(1, 4, 1, 1)	**449.58**	441.25	436.39	425.73	415.52
(1, 4, 1, 2)	OOM	465.92	462.95	448.61	435.31

Train Loss vs Eval Score

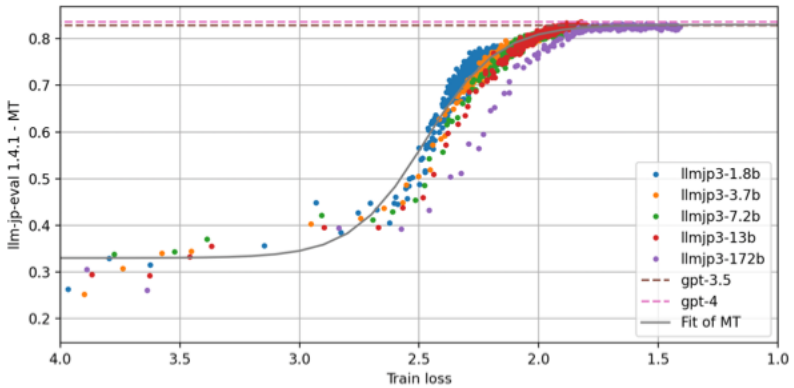
Fundamental Analysis



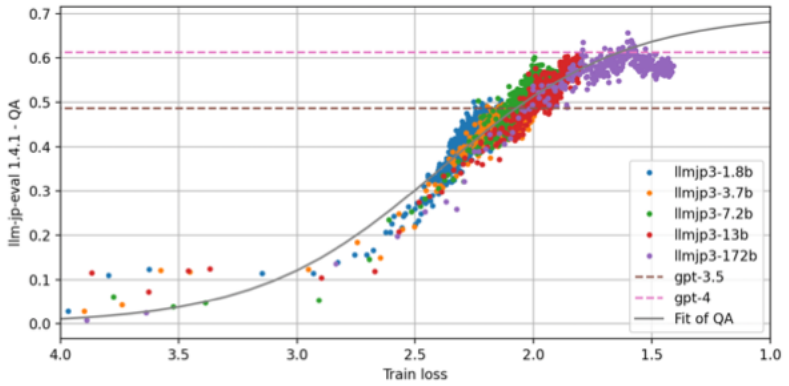
Mathematical Reasoning



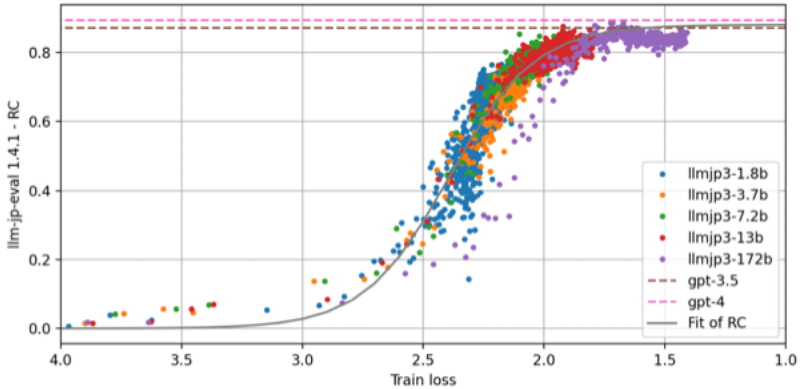
Machine Translation



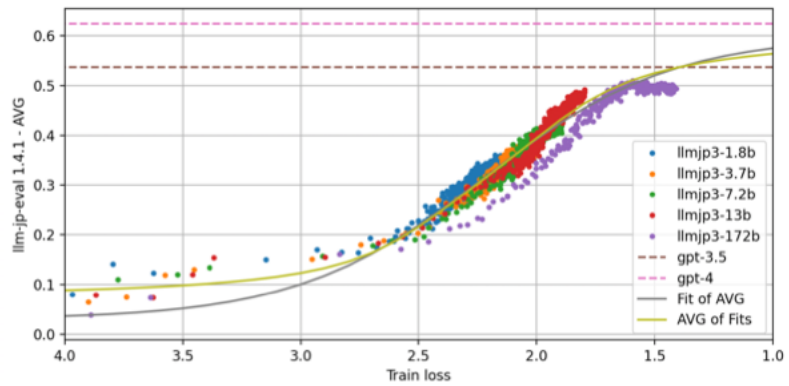
Question Answering



Reading Comprehension



Average of 11 tasks



Scaling Laws of MoEs

Parameters vs FLOPs: Scaling Laws for Optimal Sparsity for Mixture-of-Experts Language Models

Samira Abnar* Apple
Harshay Shah* MIT
Dan Busbridge Apple
Alaaeldin El-Nouby Apple
Josh Susskind Apple
Vimal Thilak* Apple

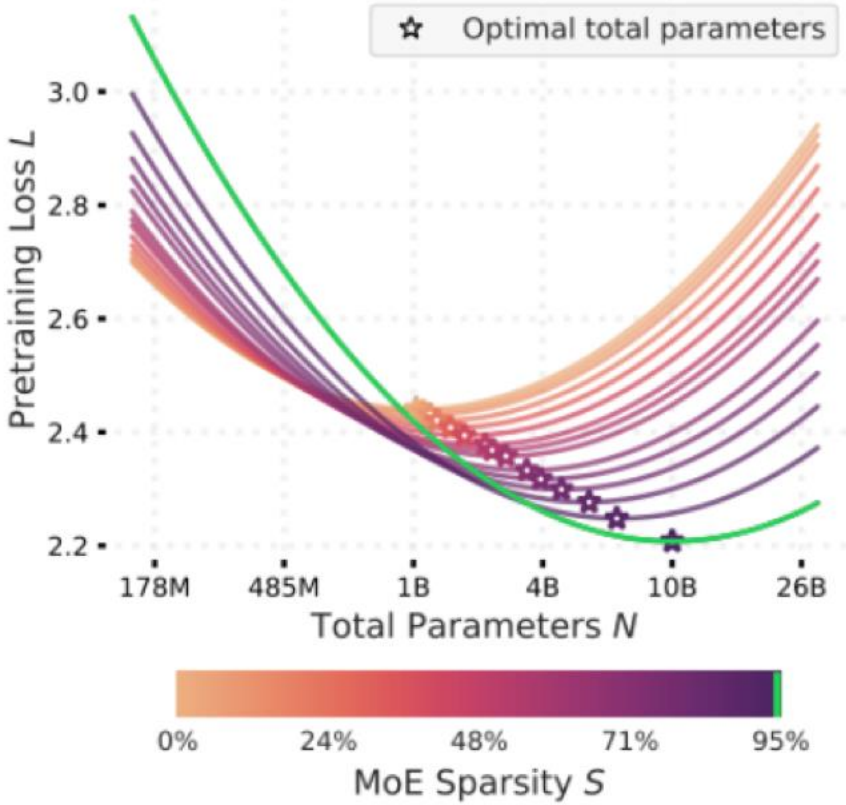
https://arxiv.org/abs/2501.12370

Optimal sparsity increases with the total parameters
→ Larger models should be more sparse

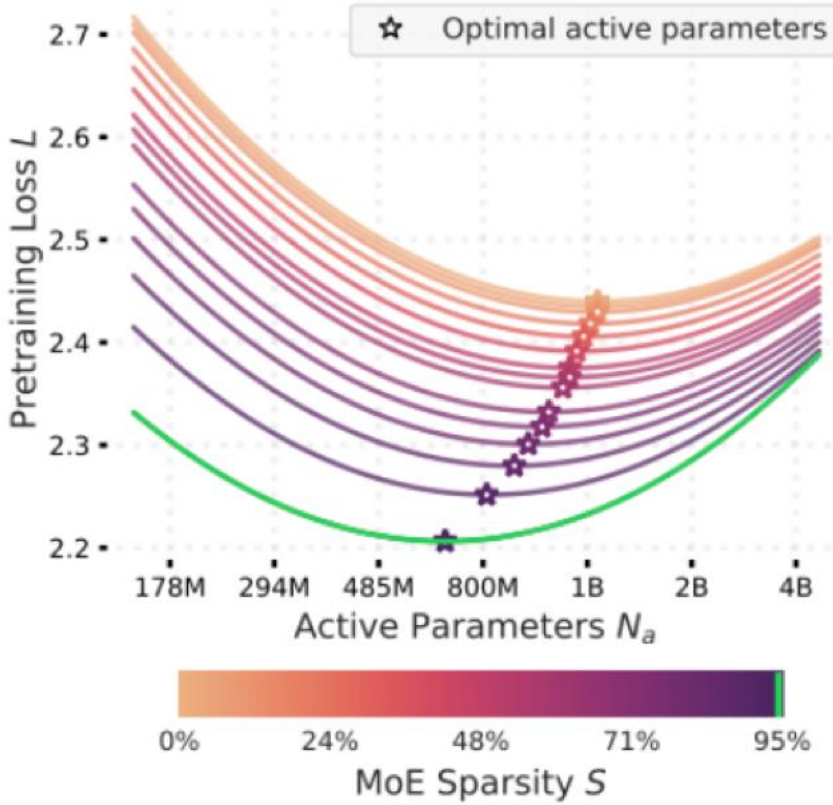
Optimal sparsity mildly decreases with the active parameters

The most sparse model achieves the lowest loss regardless of model size

Fixing Sparsity
Varying Total Parameters



Fixing Sparsity
Varying Active Parameters



S: Sparsity
E: Number of experts
K: Number of activated experts

$$S = \frac{E - K}{E}$$

Memorization vs Reasoning

Fixing active parameters and increasing total parameters
→ Fixing the cost and increasing model capacity

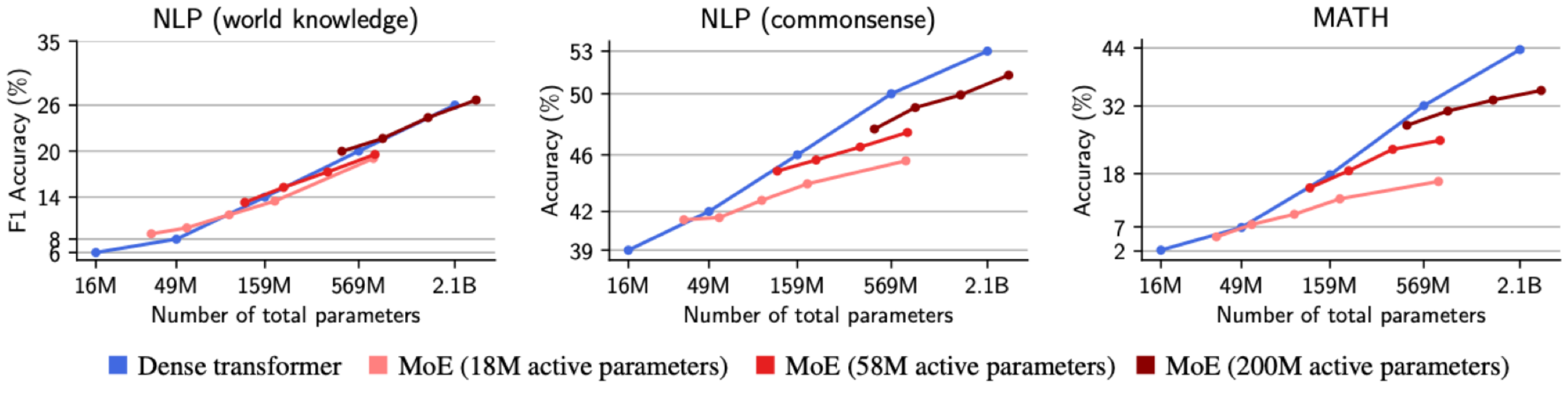
For memorization tasks
→ Accuracy increases with the total parameters

For reasoning tasks
→ Accuracy does not increase with total parameters
→ Sparsity is not good for reasoning?

MIXTURE OF PARROTS 🦜🦜🦜: EXPERTS IMPROVE MEMORIZATION MORE THAN REASONING

Samy Jelassi* Harvard University
Clara Mohri Harvard University
David Brandfonbrener Harvard University, Kempner Institute
Alex Gu MIT
Nikhil Vyas Harvard University
Nikhil Anand Harvard University, Kempner Institute
David Alvarez-Melis Harvard University, Kempner Institute
Yuanzhi Li Microsoft Research
Sham M. Kakade Harvard University, Kempner Institute
Eran Malach Harvard University, Kempner Institute

https://arxiv.org/abs/2410.19034



Experimental Setup

Optimal Sparsity of Mixture-of-Experts Language Models for Reasoning Tasks

Taishi Nakamura [1 2] Satoki Ishikawa [1] Masaki Kawamura [1] Takumi Okamoto [1 2] Daisuke Nohara [1]
Jun Suzuki [3 2 4] Rio Yokota [1 2]

Model architecture

Base model: Mixtral [Jiang et al., 2024]

Normalization: RMSNorm [Zhang & Sennrich, 2019]

Activation: SwiGLU [Shazeer, 2020]

Positional embedding: RoPE [Su et al., 2024]

Router: Drop-less token-choice top-k routing [Gale et al., 2023]

Architectural hyperparameters

Model width: d={512, 1024, 2048}

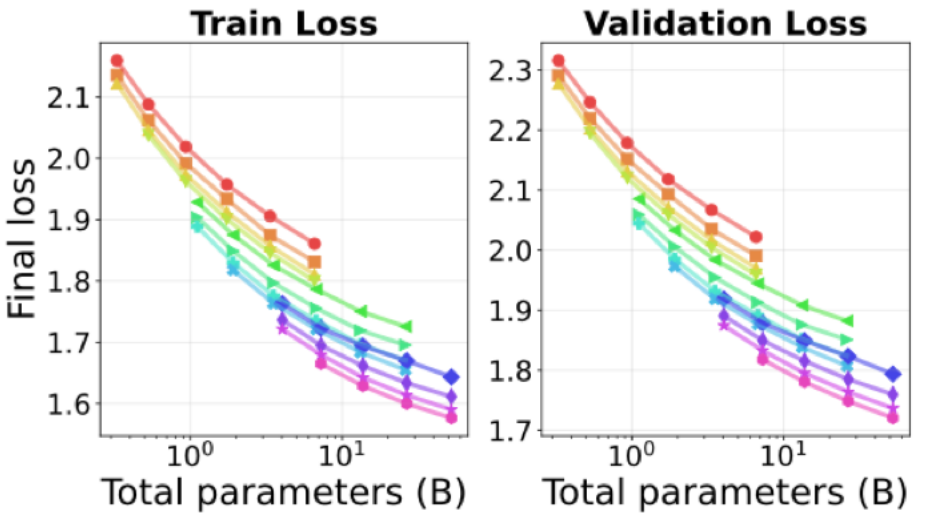
Number of experts: E={8, 16, 32, 64, 128, 256}

Activated experts: k={2, 4, 8, 16}

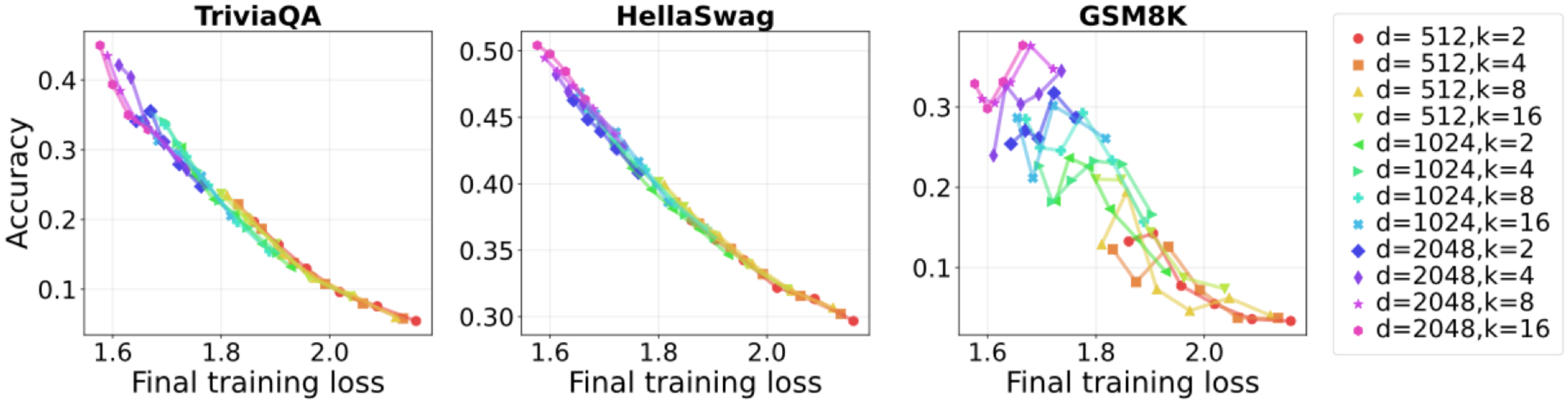
Optimizer: AdamW

Learning rate: peak 4e-4, 2k linear warmup, cosine decay

Weight decay: 0.1

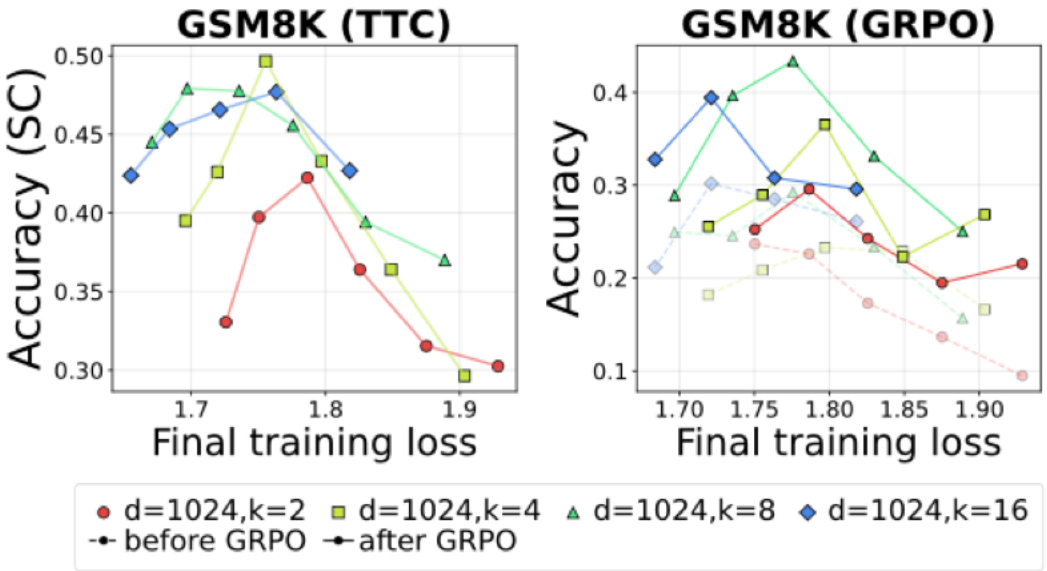


Downstream Accuracy vs Training Loss

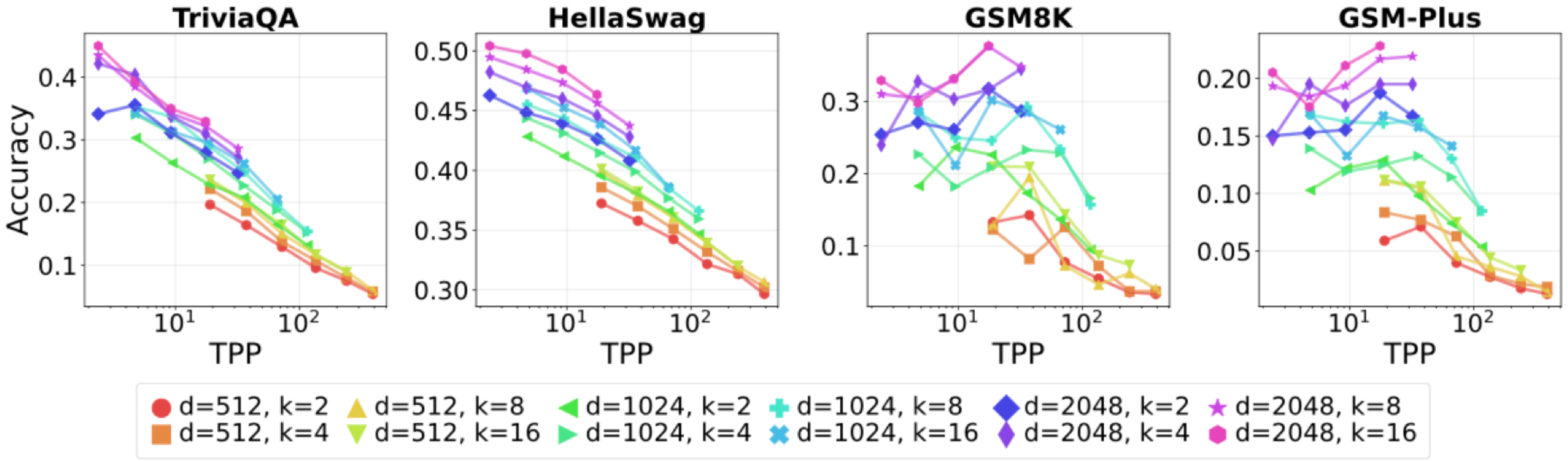


We want the downstream accuracy
to inversely scale with the training loss

TriviaQA: World knowledge 🤗
Hellaswag: Commonsense reasoning 🤗
GSM8K, GSM-Plus: Mathematical reasoning 🤔



Downstream Accuracy vs Tokens per Parameter



Chinchilla scaling law [Hoffman et al., 2022]: 20 tokens per parameter (TPP)

→ Optimal TPP is task dependent [Roberts et al., 2025]

- Memorization benefits from lower TPP
- Reasoning benefits from higher TPP

By plotting against the TPP, we can see a more uniform trend for math reasoning tasks

→ Math reasoning tasks have an optimal TPP regardless of the model size

Japanese LLM Projects

LLM-jp






Members:
Too many people

System:
MDX, ABCI, TSUBAME
GCP, Sakura

Model: (from scratch)
GPT 1.3B, 13B, 175B
Llama 1.8B, 3.7B, 7.2B, 13B, 172B

Training budget:
5,000,000 H100 hours / year

Swallow



Members:
Science Tokyo, AIST

System:
ABCI2.0, ABCI3.0
AWS

Model: (continual pre-training)
Llama2 7B, 13B, 70B
Mistral, 7B Mixtral 8x7B
Llama3.1 8B, 70B
Llama3.3 8B, 70B
Gemma2 8B
Qwen3 8B, 30B-A3B, 32B
GPT-OSS 20B, 120B

Training budget:
1,000,000 H100 hours / year

Swallow Project

Institute of SCIENCE TOKYO

AIST



GPT-OSS Swallow



Qwen3 Swallow



Gemma-2-Llama Swallow



Llama 3.3 Swallow



Llama 3.1 Swallow v0.5



Swallow Leaderboard v2

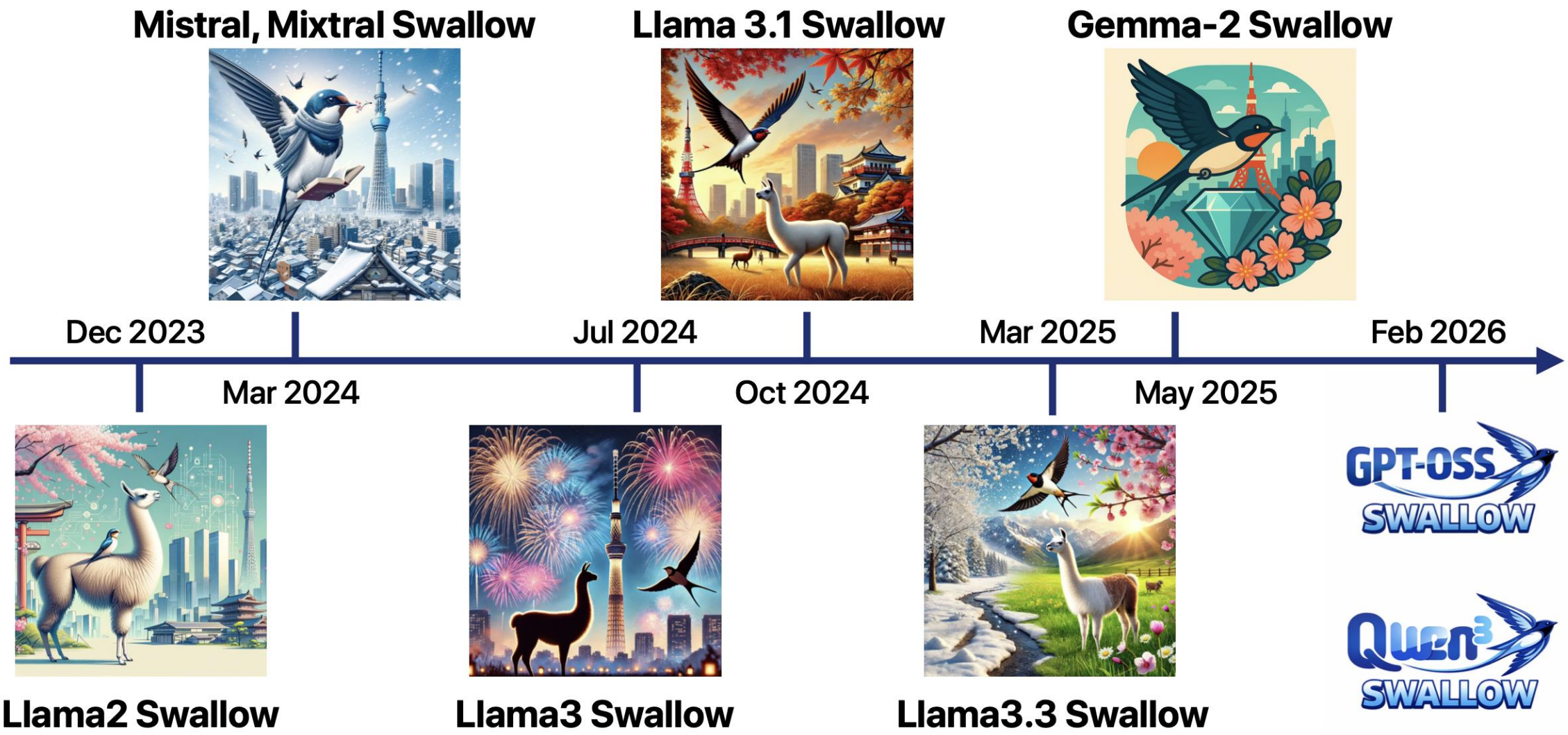
- Improve Japanese-domain knowledge and language ability
- Transfer logical reasoning ability from English to Japanese
- Maintain mathematics and coding capabilities

Naoaki Okazaki Professor, Science Tokyo Project leader, team leader of pre-training corpus, Web developer	**Rio Yokota** Professor, Science Tokyo Team leader of LLM training	**Sakae Mizuki** Researcher, AIST / Science Tokyo Team leader of instruction tuning, team leader of LLM evaluation
Kazuki Fujii Master student, Science Tokyo LLM training, post-training	**Taishi Nakamura** Master student, Science Tokyo LLM training, post-training, and evaluation	**Youmi Ma** Assistant Professor, Science Tokyo Post-training
Daisuke Oba Specially Appointed Assistant Professor, Science Tokyo Post-training	**Susumu Ota** Researcher, Science Tokyo Post-training	**Koki Maeda** PhD student, Science Tokyo LLM evaluation
Masanari Oi Master student, Science Tokyo LLM evaluation	**Takumi Okamoto** Master student, Science Tokyo Instruction tuning	**Shigeki Ishida** Master student, Science Tokyo LLM evaluation
Masaki Kawamura Master student, Science Tokyo Development of Language resources	**Yukito Tajima** Master student, Science Tokyo Development of Language resources	**Taihei Shiotani** Master student, Science Tokyo LLM evaluation
Hinari Shimada Master student, Science Tokyo LLM evaluation, safety	**Koshiro Saito** Master student, Science Tokyo LLM evaluation	**Tatsuya Ichinose** Undergraduate student, Science Tokyo LLM evaluation
Naoya Matsushita Undergraduate student, Science Tokyo LLM evaluation	**Sora Miyamoto** Undergraduate student, Science Tokyo LLM evaluation	**Daisuke Nohara** Undergraduate student, Science Tokyo Post-training
Yuta Katayama Undergraduate student, Science Tokyo Development of Language resources, instruction tuning	**Nguyen Tien Dung** Undergraduate student, Science Tokyo Development of Language resources	**Hiroya Takamura** Research team leader, AIRC, AIST Project manager

Swallow Development Timeline

Institute of SCIENCE TOKYO

2.4M	551K	132	19
MODEL DOWNLOADS	DATASET DOWNLOADS	MODELS	DATASETS



Building Reasoning Models

Institute of SCIENCE TOKYO

Goal for This Work

- Clarify a recipe for building reasoning models with stronger Japanese-language ability and thinking capability

Models Built and Released





License: Apache 2.0 (commercial use permitted)

Scope of This Talk

- Report model performance and the model-building recipe
- Share findings on acquiring, improving, and validating reasoning ability

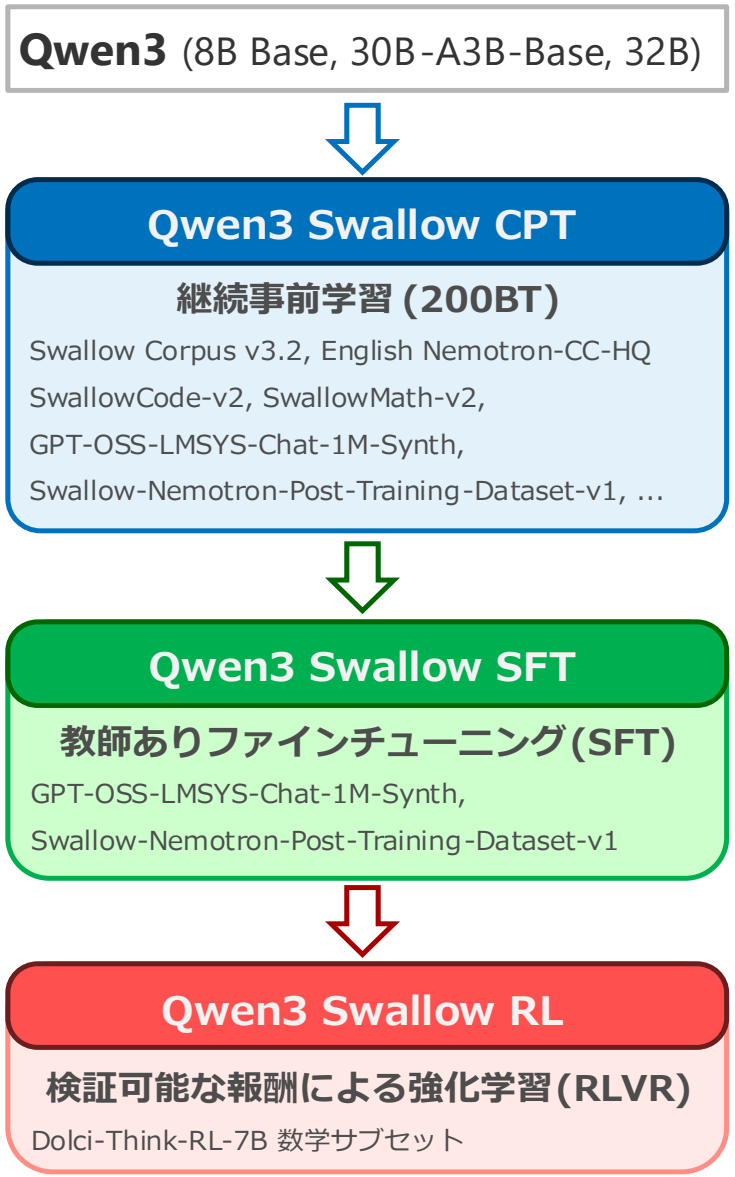
Overview of Reasoning Models

Institute of SCIENCE TOKYO

Training Pipeline

- Starting from Qwen3 and gpt-oss, run a three-stage pipeline: continual pretraining (CPT), supervised fine-tuning (SFT), and reinforcement learning (RL).
- Use CPT and SFT to learn responses with reasoning traces synthesized by a reasoning teacher LLM (gpt-oss-120b), eliciting deep reasoning.
- Further improve reasoning ability with RL.

Stage	Training content
Continual pretraining (CPT)	Japanese/English text, QA text, math/code, and instruction-response data with reasoning traces
Instruction tuning (SFT)	Distill responses with reasoning traces for general dialogue and STEM
Reinforcement learning (RL)	Verifiable rewards based on correctness of math problems



Evaluation Benchmarks

Institute of SCIENCE TOKYO

Ja/En evaluation focused on hard math, science, and code generation.

Task	Japanese	English	Metric
General dialogue	Japanese MT-Bench	English MT-Bench	Helpfulness (LLM judge)
Instruction following	M-IFEval-Ja	—	Accuracy
Japan-specific knowledge	JamC-QA	—	Accuracy (multiple choice)
Commonsense reasoning	—	HellaSwag	Accuracy (multiple choice)
General knowledge	MMLU-ProX-Ja	MMLU-Pro	Accuracy (multiple choice)
Science	★ GPQA-Ja	★ GPQA (Diamond)	Accuracy (multiple choice)
Math	MATH-100-Ja	★ AIME 2024–25, MATH-500	Accuracy
Coding	JHumanEval	★ LiveCodeBench	Pass@1
Translation	WMT20 En-Ja, Ja-En	—	BLEU

★ High-difficulty benchmarks that require deep reasoning

Evaluation framework: swallow-evaluation-instruct, link
Evaluation setting: zero-shot instruction-response format / context length 32k / greedy decoding for text generation (except MT-Bench and coding)

Training Scale and Dataset Composition

Institute of SCIENCE TOKYO

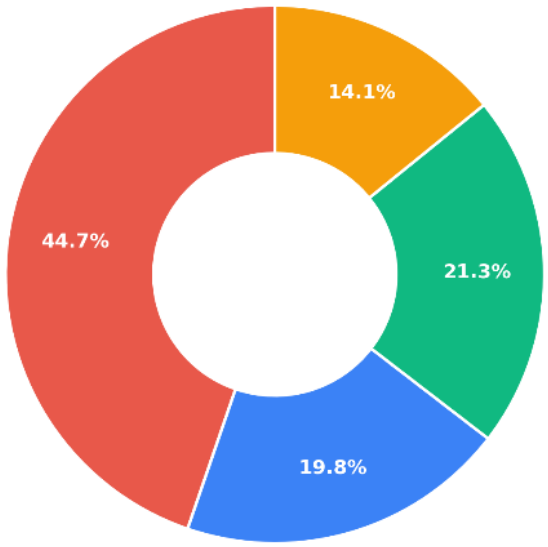
Training Scale

- Qwen3-Swallow: approx. 200B tokens
- GPT-OSS-Swallow: approx. 400B tokens

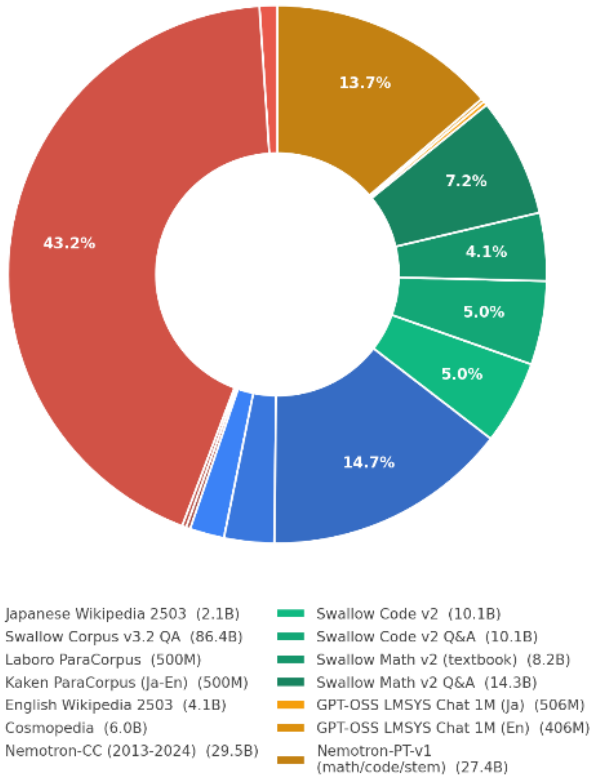
Dataset Composition

- Japanese web text (incl. synthetic QA): 45%
- English web text: 20%
- Math text and source code: 20%
- **Instruction-response data with reasoning traces: 15%**

By category



By dataset



Improving Math and Coding Performance

Institute of SCIENCE TOKYO

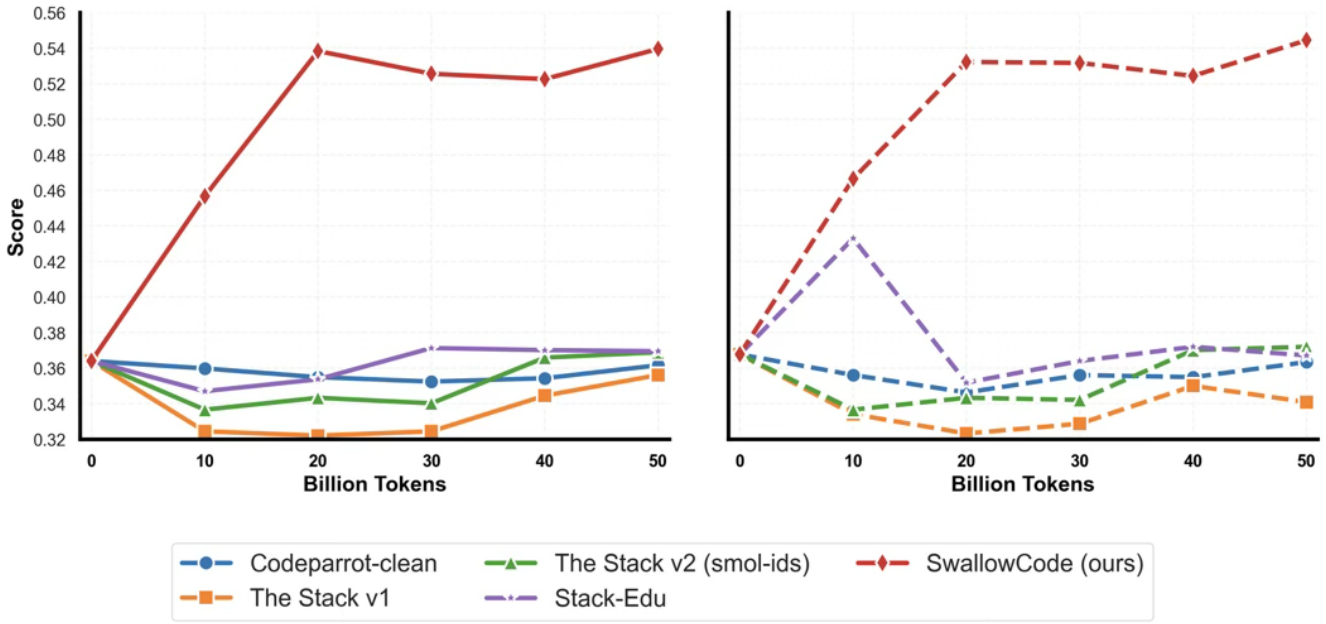
Rewriting Pre-Training Data Boosts LLM Performance in Math and Code

Kazuki Fujii[1,2], Yukito Tajima[1], Sakae Mizuki[1,2], Hinari Shimada[1], Taihei Shiotani[1], Koshiro Saito[1], Masanari Ohi[1], Masaki Kawamura[1], Taishi Nakamura[1,2], Takumi Okamoto[1], Shigeki Ishida[1], Kakeru Hattori[1,2], Youmi Ma[1], Hiroya Takamura[2], Rio Yokota[2,3], and Naoaki Okazaki[1,2]

original	Stack v2 train smol	41,025,639 samples in the-stack-v2-train-smol-ids (python), 36.1 B tokens
Stage1	Syntax error filter	36,695,625 samples, 30.5B tokens
Stage2	Linter filter	24,100,915 samples, 20.2B tokens
Stage3	SGCR (Style-Guided)	Uses an LLM to rewrite source code, ensuring adherence to stylistic conventions
Stage4	SCOR (Self-Contained)	Uses an LLM to rewrite source code into self-contained and optimized forms

The-Stack-v2

- Syntax validation
- Pylint-based style filtering
- Two-stage LLM rewriting
  - Style conformity
  - Self-contained examples



Datasets Released

Institute of SCIENCE TOKYO

Japanese Corpus



SwallowCorpus-v1 (2023)

SwallowCorpus-v2 (2024)

SwallowCorpus-v3 (2025)

Code & Math Corpus Public

Swallow CODE & MATH

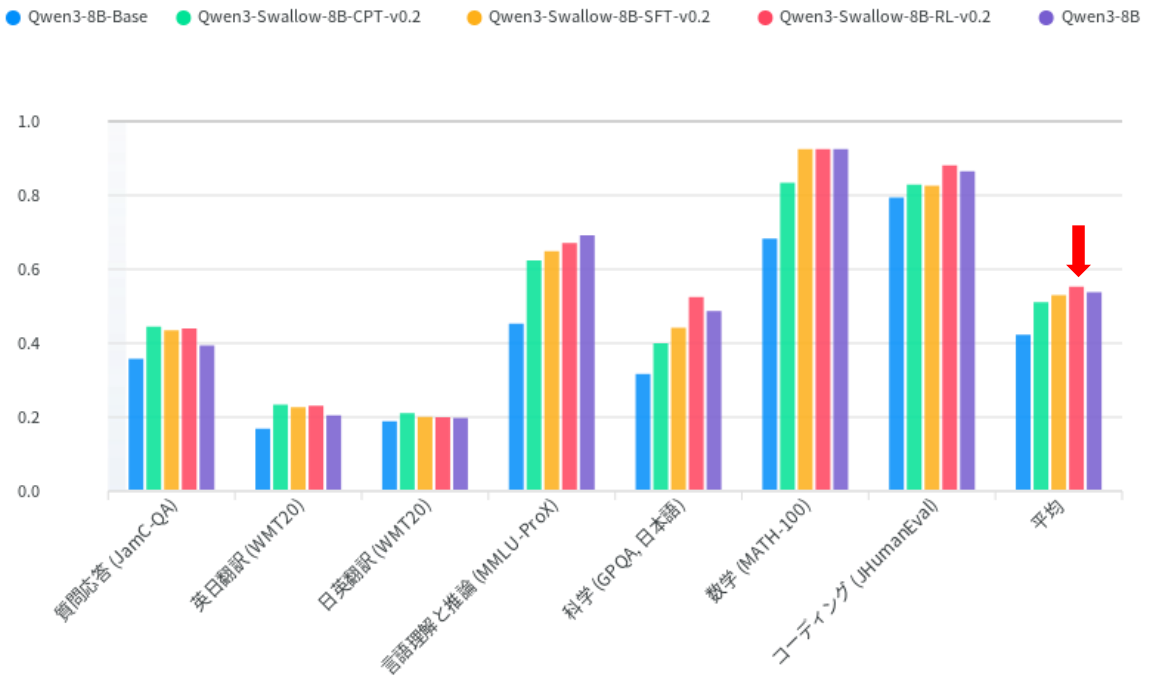
SwallowCode & Math -v1 (May 2025)

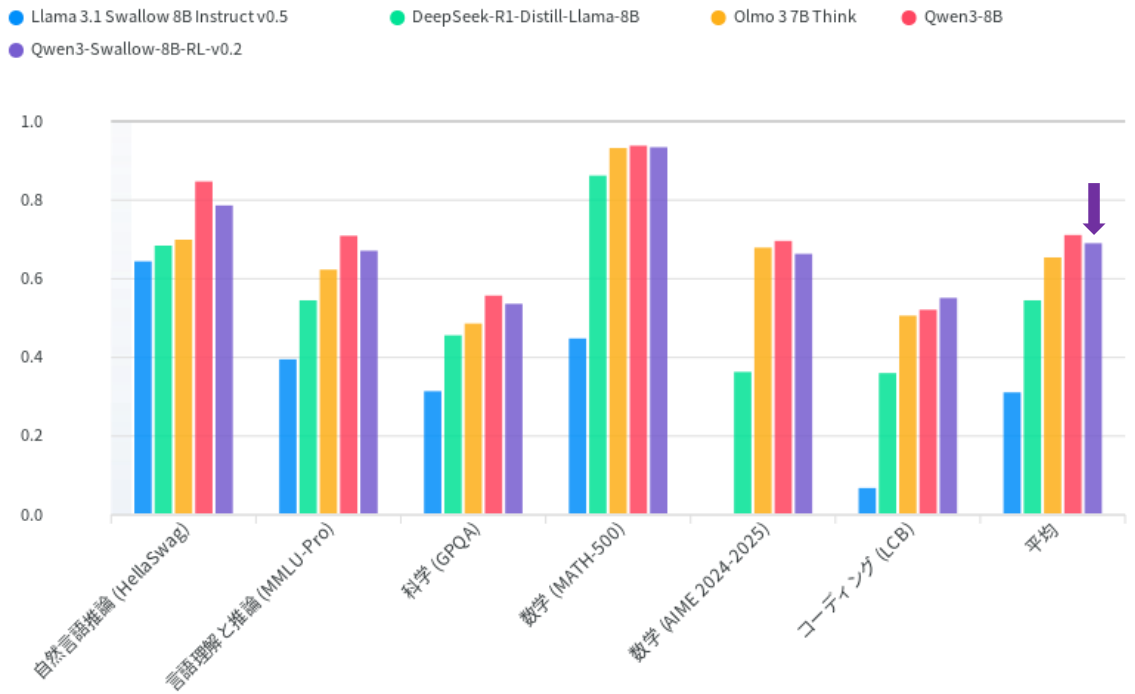
SwallowCode & Math -v2 (Nov 2025)

Qwen3-Swallow 8B RL Performance

Top-tier performance on Japanese tasks among open models up to 8B parameters

- Deep reasoning delivers strong results in math (AIME), science (GPQA), and code generation (LCB)
- Strong on Japan-specific knowledge (JamC-QA) and translation (WMT20)
- Reaches the level of Alibaba's official Qwen3-8B, suggesting the recipe's effectiveness



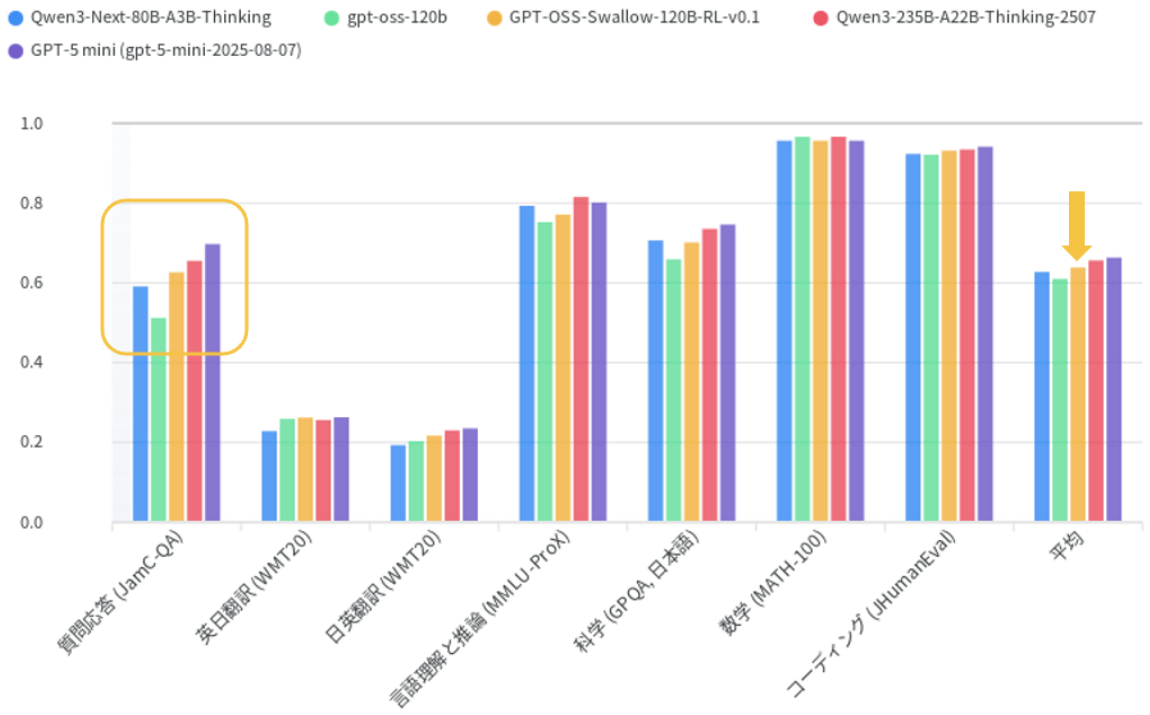


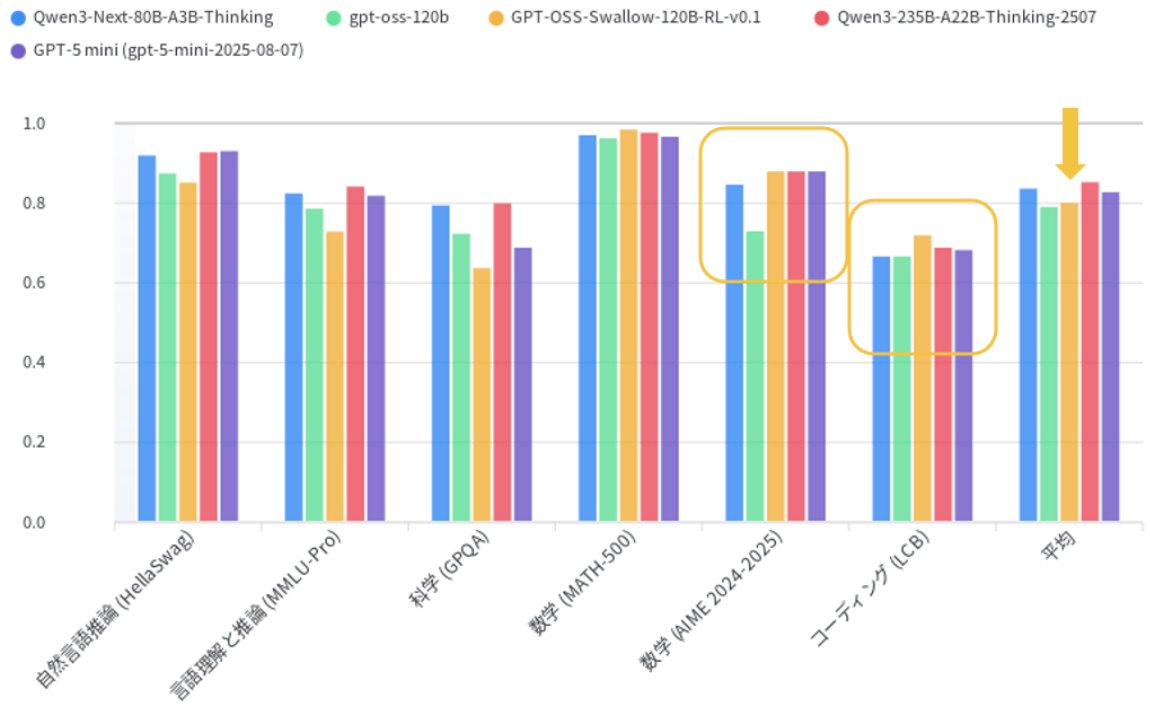
GPT-OSS-Swallow 120B RL Performance

Institute of SCIENCE TOKYO

Top-tier among open models up to 120B parameters

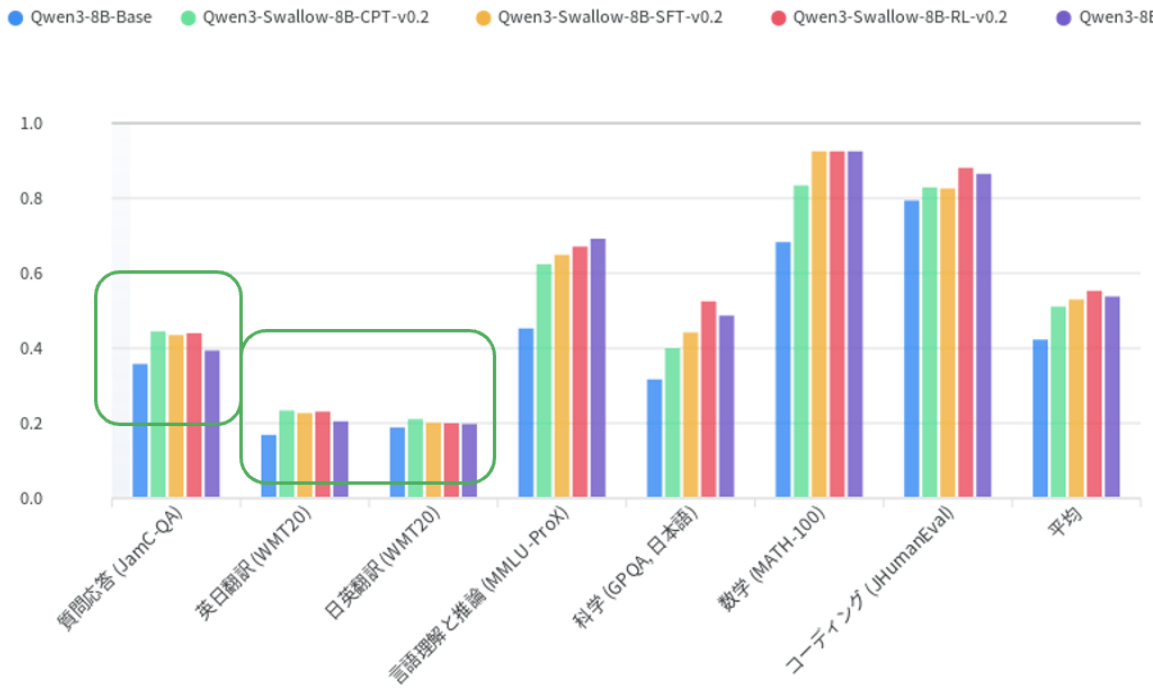
- Notable +11 pt improvement over gpt-oss on JamC-QA, which tests Japan-specific knowledge
- Surpasses gpt-oss in math and code generation
- Japanese MT-Bench: top-tier score of 0.92 (upper right)
    - Exceeds GPT-5.1 Thinking level; MT-Bench is now too saturated to differentiate reliably

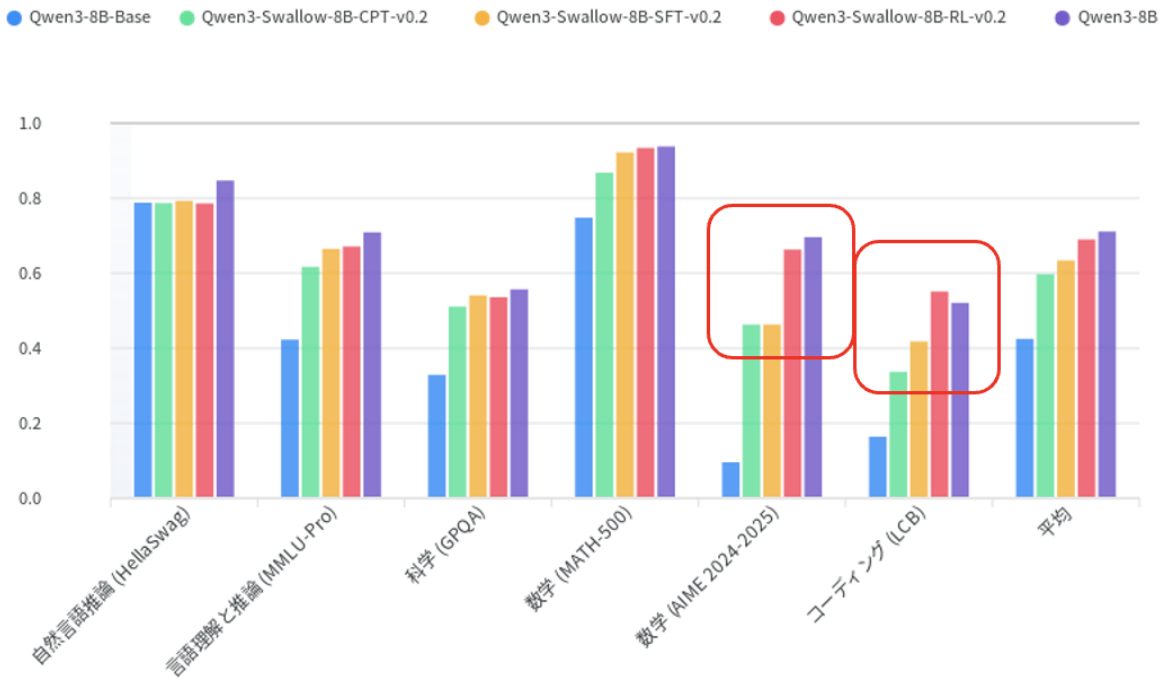




Performance Changes by Training Stage

- Continual pretraining (Qwen3-Base → CPT): Japan-specific knowledge, translation, general knowledge, and STEM all improve broadly
- Instruction tuning (CPT → SFT): general knowledge and STEM improve
- Reinforcement learning (SFT → RL): math (MATH, AIME) and code generation (LCB) improve
  - Learning only on math may transfer gains to code generation





Summary

Training large models is not easy

- Hardware failures, Software bugs, OOM vs max FLOP/s
- Problems that don't manifest until you exceed 100B parameters
- People constantly telling you that you are wasting money

Mixture of Experts and reasoning

- What is the optimal sparsity?
- Be careful of combination of eval tasks ↔️ training data

Creating your own data

- Language filter, Quality filter, Deduplication
- Rewriting data to enhance reasoning capabilities

