LLM-jp FT-LLM コンペにおける 数学推論能力向上の取り組み

尾崎 大晟 [1]　力岡 友和 [1]　渡部 泰樹 [1]　Jeong Seong Cheol[2]
[1] 株式会社松尾研究所 [2] 東京大学 松尾・岩澤研究室

概要

本稿では，LLM-jp FT-LLM コンペティションにおける我々のチームの取り組みを報告する．本コンペティションでは，llm-jp-4-8b をベースモデルとし，中学校・高等学校レベルの数学問題に対する正答率を競う．我々は，(1) YaRN によるコンテキスト長拡張，(2) gpt-oss-120b を活用した合成推論データの構築，(3) SFT および GRPO による学習，の 3 つを組み合わせた．特にデータ合成では，GSM8K および StackMathQA を基に Chain-of-Thought（CoT）と Tool-Integrated Reasoning（TIR）の 2 種類の推論データを生成し，LLM-as-a-Judge による多段階品質管理を行った．

1　はじめに

LLM-jp FT-LLM コンペティション [1] は，llm-jp が主催する LLM ファインチューニングの公開コンペティションである．参加チームは llm-jp-4-8b をベースモデルとして，日本の中学校・高等学校レベルの数学問題 500 問に対する正答率を競う．推論時には llm-jp-4-8b をベースとしたモデルのみ使用可能であり，外部 LLM の利用やインターネットアクセスは禁止されている．

我々のチームでは，以下の 3 つの戦略を組み合わせた．

1. **コンテキスト長の拡張:** YaRN[2] により，長い推論過程を扱えるようコンテキスト長を拡張する
2. **合成推論データの構築:** gpt-oss-120b を用いて CoT および TIR の推論データを合成し，多段階の品質管理を行う
3. **SFT および GRPO:** 合成データを用いた教師ありファインチューニングと GRPO による強化学習を実施する

以下，第 2 節で予備調査とコンテキスト長拡張について，第 3 節で合成推論データの構築について，第 4 節で SFT および GRPO について，第 5 節で推論パイプラインについて述べる．

2　予備調査とコンテキスト長拡張

本節では，ベースモデル llm-jp-4-8b の推論能力を把握するための予備実験と，コンテキスト長拡張について述べる．

2.1　ベースライン実験

コンペティションで配布されたテストデータに対して評価を行ったところ，単純な指示文のみのプロンプトでは正答率は 0.08 に留まった．一方，SFT で使用するプロンプト形式（指示・入力・応答）に揃えることで正答率は 0.32 まで改善し，さらに英語プロンプトを用いることで 0.40 まで向上した．この結果から，推論性能はプロンプト設計および使用言語に大きく依存することが確認された．比較のため大規模モデルも評価し，`gpt-oss-120b` では 0.81，`Qwen3-235B-A22B-Thinking` では 0.74 の正答率を得た．

2.2　コンテキスト長の拡張

数学問題の推論では長い Chain-of-Thought 推論過程を生成する必要があるため，YaRN (Yet another RoPE extensioN)[2] によりベースモデルのコンテキスト長を拡張した．

3　合成推論データの構築

本節では，SFT に用いる合成推論データの生成パイプラインについて述べる．パイプラインのソースコードは公開されている[1)]．

1) https://github.com/ft-llm-team-mkj/synthetic-data-pipeline

3.1 パイプライン概要

本パイプラインは，既存の数学 QA データセットを入力とし，高品質な推論過程を付与した SFT 用データセットを出力する．処理は大きく 3 段階からなる: (1) データソースの前処理，(2) CoT/TIR 推論の生成，(3) 多段階品質検証．

3.2 データソースと前処理

データソースとして GSM8K[3] および StackMathQA[4] を用いた．

GSM8K GSM8K は小学校レベルの算数文章題約 8,500 問からなるデータセットである．各問題の回答に含まれる「####」マーカー以降を最終回答として抽出した．

StackMathQA StackMathQA は Stack Exchange の数学関連 Q&A を収集したデータセットである．前処理として，(1) 1 問あたりの回答数が 5 以下，(2) 回答テキストの長さが 3,000 文字以下のフィルタリングを適用し，30,000 件をサンプリングした．複数回答がある場合は，gpt-oss-120b により最適な回答を選定し，質問文の簡潔化も行った．

3.3 CoT 拡張

各 QA ペアに対し，問題から回答に至る Chain-of-Thought 推論過程を生成した．多様性を確保するため各問題につき 3 候補を生成した．

生成された候補は LLM-as-a-Judge 方式で評価した．評価基準は直接性（30%），明瞭性（30%），完全性（20%），効率性（20%）の 4 項目（各 1〜10 点）であり，重み付き総合スコアを算出する．3 候補中の最高スコア候補を選択し，後段の品質検証（第 3.5 節）で総合スコア ≥7 のフィルタリングを適用した．

3.4 TIR 拡張

CoT 拡張に加え，Tool-Integrated Reasoning（TIR）データも生成した．TIR では，推論過程に加えて実行可能な Python コードを<tool_call>タグで付与する．CoT と同様に 3 候補を生成し，LLM-as-a-Judge によりコード正確性，推論明瞭性，完全性の 3 項目（各 1〜10 点）で評価し，最高スコア候補を選択した．

TIR データに対しては後処理として以下の 2 段階の検証を実施した:

1. **実行検証**: Python サンドボックス経由でコードを実行し，エラーなく動作するかを確認
2. **意味的一致検証**: 実行結果と期待回答の意味的一致を LLM で評価（数値フォーマットや単位の差異を許容）

3.5 品質管理とマージ

生成段階で最良候補を選択した後，後処理として品質の再評価を行った．CoT データに対しては，生成時と同じ 4 基準（直接性・明瞭性・完全性・効率性）で LLM による品質再評価を実施し，総合スコア ≥7 のデータのみを有効とした．TIR データに対しては，前述の 2 段階検証に加え，コード正確性と推論明瞭性の平均スコア ≥7 を有効条件とした．

生成されたデータセットは HuggingFace Hub 上に公開している[2)]．

4 SFT および GRPO

第 3 節で構築した合成データを用いて，教師ありファインチューニング（SFT）および GRPO（Group Relative Policy Optimization）[5] による強化学習を実施した．学習基盤には NVIDIA の NeMo-RL[6] を採用した．NeMo-RL は Ray ベースの分散学習フレームワークであり，DTensor によるモデル並列化と vLLM[7] による高速な推論生成を，同一 GPU 上で時分割（colocated mode）により実現する．

4.1 SFT

第 3 節のパイプラインで生成した CoT データ（訓練 33,905 件，検証 1,784 件）を用いて SFT を実施した．データは<think>タグ付きの CoT 推論を含む OpenAI チャット形式である．YaRN により 8k/16k/32k の 3 種類のコンテキスト長拡張を行ったが，32k でも計算リソースに十分な余裕があったため，最終的に 32k バリアントを採用した．SFT データの系列長が最大でも 8,192 トークンに収まったため，SFT では最大系列長を 8,192 トークンに設定して学習を行った．

主要なハイパーパラメータを表 1 に示す．DTensor によるテンソル並列（TP=2）と activation checkpointing を有効化し，8 基の H100 GPU を用いた．Sequence packing により異なる長さの系列を効率的にバッチ化し，GPU 利用率を向上させた．チェックポイントは検証損失に基づき上位 3 つを保

2) https://huggingface.co/datasets/ft-llm-team-mkj/cot-merged

図 1　SFT における訓練損失の推移

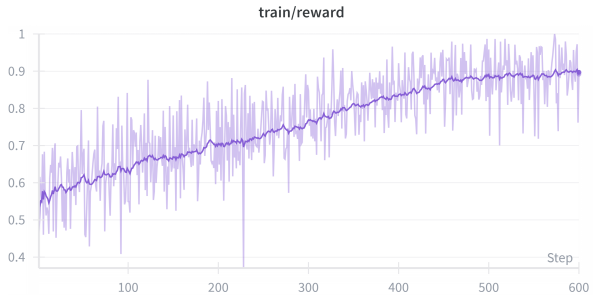

図 2　GRPO における訓練報酬の推移

表 1　SFT の主要ハイパーパラメータ

項目	値
GPU	H100 × 8（1 ノード）
精度	bfloat16
テンソル並列（TP）	2
エポック数	3
グローバルバッチサイズ	32
最適化手法	AdamW
学習率	2×10^{-5}
(β_1, β_2)	$(0.9, 0.98)$
重み減衰	0.1
勾配クリッピング	1.0
Sequence packing	有効（MFFD）

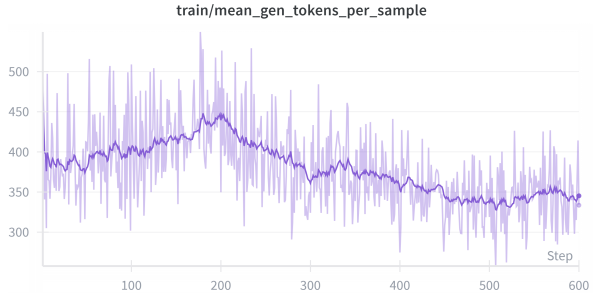

図 3　GRPO における平均生成トークン数の推移

存した．

図 1 に SFT の訓練損失の推移を示す．損失は学習初期に急速に低下し約 0.55 に収束した後，3 エポック目の終盤でさらに 0.4 付近まで低下した．

4.2　GRPO

SFT で得られたチェックポイントを初期方策とし，GRPO による強化学習を実施した．GRPO はプロンプトごとに複数の応答を生成し，グループ内の相対的な報酬に基づいて方策を更新するアルゴリズムである [5]．学習データとして数学問題 180 件（検証 20 件）を用い，SFT よりも長い推論生成を許容するため最大系列長を 16,384 トークンに拡張して最大 600 ステップの学習を行った．

報酬設計　報酬関数は，バイナリ正誤判定と LLM による部分点の 2 段構成とした．まず hf_math_verify によりモデル出力の最終回答を正誤判定する．正解の場合は報酬 1.0 を付与し，不正解の場合は gpt-oss-120b による推論過程の評価に基づき部分点を付与する．部分点は LLM スコア（0〜1）に重み $\alpha = 0.2$ を乗じた値とした:

$$r = \begin{cases} 1.0 & \text{(正解)} \\ \alpha \cdot r_{\text{LLM}} & \text{(不正解)} \end{cases} \tag{1}$$

この設計により，最終回答が誤りであっても推論過程が適切であれば勾配情報が得られ，学習の安定化に寄与する．

学習設定　NeMo-RL の colocated mode を用い，DTensor による学習（TP=2）と vLLM による生成（TP=1，各 GPU 独立）を同一の 8 基の H100 上で時分割実行した．主要なハイパーパラメータを表 2 に示す．Leave-one-out baseline による分散低減と，応答が最大長を超過した場合のペナルティ（reward shaping）を適用した．参照方策との KL ダイバージェンスには k3 推定量 [8] を用い，係数 $\beta = 0.01$ で正則化した．

図 2 に GRPO の訓練報酬の推移を示す．報酬は学習開始時の約 0.55 から単調に増加し，600 ステップで約 0.9 に達した．また，図 3 に生成トークン数の推移を示す．学習の進行に伴い平均生成トークン数は約 390 から約 340 へ減少しており，モデルがより簡潔な推論で正解に到達できるようになったことが示唆される．なお，生成トークン数は終始 500 トークン以下で推移しており，16,384 トークンの最大系列長は過剰であった可能性がある．

5　推論パイプライン

推論パイプラインは，(1) 問題の日英翻訳，(2) マルチエージェント並列推論，(3) 回答抽出と多数決

表 2　GRPO の主要ハイパーパラメータ

項目	値
GPU	H100 × 8（1 ノード）
精度	bfloat16
プロンプト数/ステップ	16
生成数/プロンプト	8
最大ステップ数	600
生成温度	0.7
学習率	1×10^{-6}
(β_1, β_2)	$(0.9, 0.999)$
重み減衰	0.01
PPO クリップ ϵ	0.2
KL 係数 β	0.01
報酬正規化	有効
LLM 部分点重み α	0.2

投票の 3 段階からなる．推論エンジンには vLLM を使用し，8 基の GPU 上で並列に実行する．なお，第 3 節で述べた TIR データは学習してみたが，学習後のモデルにおいて Python コードの生成が安定しなかったため，最終的な推論パイプラインでは CoT ベースの推論のみを採用した．

5.1　問題の日英翻訳

入力された日本語の数学問題を，まず SFT で構築した翻訳モデル（math-translator_v2[3]）により英語に変換する．各問題に対して 4 つの翻訳候補を生成する．一部誤訳が含まれる可能性があるため，複数の翻訳候補を生成することで，翻訳の多様性を確保し，後段の推論における回答の多様性を高める．英語への変換を導入した動機は，ベースモデル llm-jp-4-8b の事前学習データにおいて英語の比率が高く，英語の数学問題に対してより高い推論性能を発揮すると期待されるためである．

5.2　マルチエージェント並列推論

翻訳された問題に対し，8 つの独立した推論エージェントをマルチプロセスにより並列に起動する．モデルの割り当てはエージェントごとに異なり，エージェント 0–5 には math-reasoner-v2[4] を，エージェント 6–7 には math-reasoner-v5[5] を使用する．math-reasoner-v2 は幅広い難易度の問題で GRPO を適用したモデルであり，math-reasoner-v5 は高難易度問題に特化して GRPO を適用したモデルである．2 種類のモデルを混在させることで，アンサンブル効果による回答精度の向上を狙った．

各エージェントは，全翻訳候補（1 問あたり 4 候補）に対して 5 サンプルずつ推論を実行する．したがって，1 問あたり 4（翻訳候補）×8（エージェント）× 5（サンプル）= 160 個の回答候補が得られる．推論時の最大コンテキスト長は 16,384 トークンとし，第 2 節で述べた YaRN による拡張を適用している．これにより，長い Chain-of-Thought 推論過程を途中で打ち切ることなく生成できる．

5.3　回答抽出と多数決投票

各エージェントの出力から最終回答を抽出する．モデルは<think>タグ内に Chain-of-Thought 推論過程を生成するため，</think>タグ以降のテキストを最終回答として取り出す．

抽出された 160 個の回答候補に対して多数決投票を行い，最終回答を決定する．単純な文字列一致ではなく，math-verify ライブラリを用いた数学的等価性判定に基づくグルーピングを行う．具体的には，新たな回答候補を既存の全グループの代表回答と比較し，数学的に等価なグループが存在すればそこに分類する．最終的に，最大グループに属する回答の代表値を最終回答として出力する．

以上の推論パイプラインにより，テストデータにおける最終的な正答率 0.70 を達成した．

6　おわりに

本稿では，LLM-jp FT-LLM コンペティションにおける我々のチームの取り組みを報告した．YaRN によるコンテキスト長拡張，合成データによる SFT，GRPO による強化学習を組み合わせ，8B パラメータモデルでの数学推論能力の向上を目指した．データ合成パイプラインでは，CoT および TIR の 2 種類の推論データを生成し，LLM-as-a-Judge とコード実行検証を組み合わせた多段階品質管理により，高品質な SFT データを構築した．一方で，TIR データは学習に用いたものの，推論時のコード生成が安定しなかったため最終的な推論パイプラインには採用しなかった．

今後の課題として，GRPO の学習安定化，より多様なデータソースの活用，および推論時における Tool Use 統合の実現が挙げられる．

3）https://huggingface.co/ft-llm-team-mkj/math-translator_v2
4）https://huggingface.co/ft-llm-team-mkj/math-reasoner-v2
5）https://huggingface.co/ft-llm-team-mkj/math-reasoner-v5

謝辞

本取り組みは LLM-jp コミュニティおよび ABCI 3.0 の計算資源の提供を受けて実施された．関係者の皆様に感謝申し上げる．

参考文献

[1] LLM-jp ファインチューニングコンペティション 2026. https://llm-jp.github.io/tuning-competition/2026/feature.html, 2026. Accessed: 2026-02-24.

[2] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. **arXiv preprint arXiv:2309.00071**, 2023.

[3] 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.

[4] math-ai. StackMathQA: A curated collection of mathematical q&a from Stack Exchange. https://huggingface.co/datasets/math-ai/StackMathQA, 2024.

[5] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. **arXiv preprint arXiv:2402.03300**, 2024.

[6] NVIDIA. NeMo-RL: A scalable and efficient library for reinforcement learning of LLMs. https://github.com/NVIDIA/NeMo-RL, 2025.

[7] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In **Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)**, 2023.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. **arXiv preprint arXiv:1707.06347**, 2017.

[9] Angeliki Giannou, Shashank Rajput, Jy yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In **Proceedings of the 40th International Conference on Machine Learning (ICML)**, 2023.

[10] Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In **Proceedings of ICLR**, 2024.

[11] Yongyi Fan, et al. What makes looped transformers perform better than non-recursive ones (provably). **arXiv preprint arXiv:2510.10089**, 2025.

[12] Sean McLeish, Ang Li, John Kirchenbauer, et al. Teaching pretrained language models to think deeper with retrofitted recurrence. **arXiv preprint arXiv:2511.07384**, 2025.

[13] Qinyuan Ye, Xiaoyang Wang, Hongming Zhang, Xiang Yue, Xin Liu, Haitao Mi, and Dong Yu. Polaris: Open-ended long-context reasoning through co-evolution of tasks and solutions. https://hkunlp.github.io/blog/2025/Polaris/, 2025.

A　付録

本コンペティションにおいて検討したアプローチのうち不採用とした Looped Transformer と，追加実験について報告する．

A.1　Looped Transformer

Looped Transformer は，固定の Transformer ブロック（重み共有）を N 回ループさせることで任意の「深さ」を実現するアーキテクチャである．Giannou ら [9] はチューリング完全性を証明し，Yang ら [10] は通常の 10%以下のパラメータで同等性能を示した．Fan ら [11] はループ構造が River-V-Valley と呼ばれる損失景観の帰納バイアスを誘導することを理論的に示した．

McLeish ら [12] は，事前学習済み LLM にループ構造を後付けしカリキュラム学習でループ回数を段階的に増加させる手法を提案し，1B クラスのモデルで数学推論性能の向上を報告している（コード[6)]）．

本コンペへの適用は，8B モデルへの適用可能性が不明確であること，MATH レベルでの改善が限定的であること，学習コストが高い（32 GPU・50B トークン規模）ことから見送り，CoT SFT→GRPO パイプライン（第 4 節）を優先した．

A.2　YaRN の拡張度合いの影響

YaRN のスケーリング係数が GRPO 学習に与える影響を調べるため，コンテキスト長拡張を 8k，16k，32k に設定した 3 条件で GRPO を実施した．図 4 に各条件の訓練報酬の推移を示す．3 条件の報酬曲線はほぼ一致しており，拡張度合いによる大きな差は見られなかった．ただし，100 ステップ以降では拡張度合いの大きい 32k バリアントがやや高い報酬を示す傾向が見られた．Ye ら [13] は YaRN の推論時適用により長文脈での精度が大幅に向上することを報告しており，YaRN の主たる効果は学習時よりも推論時のコンテキスト長外挿にあると考えられる．

A.3　コンテキストウインドウの影響

GRPO 学習時のコンテキストウインドウサイズの影響を調べるため，YaRN 16k モデルに対し，最大系列長を 1k，2k，4k，8k，16k に変えた 5 条件で GRPO を実施した．図 5 に訓練報酬の推移を，図 6 に平均生成トークン数の推移を示す．報酬の最終到達値はいずれの条件でもおおむね同程度であったが，1k の条件では学習初期（約 100 ステップまで）の報酬立ち上がりが他条件と比べて遅れる傾向が見られた．一方，平均生成トークン数は全条件で 500 トークン以下で推移しており，本タスクでは 2k 以上のコンテキストウインドウがあれば十分であることが示唆される．Ye ら [13] も，事前学習時のコンテキスト長を超える領域での RL 学習はサンプルの大部分が長文脈を活用せず効率が低いことを報告しており，本実験の結果もこの知見と整合的である．

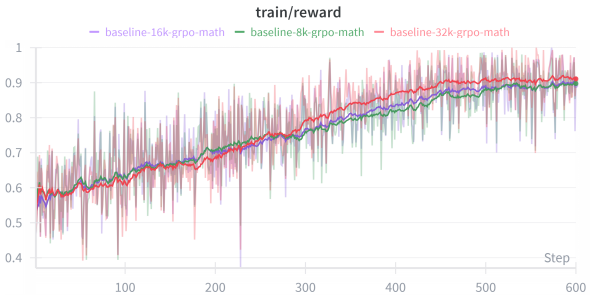

図 4　YaRN の拡張度合い別の GRPO 訓練報酬

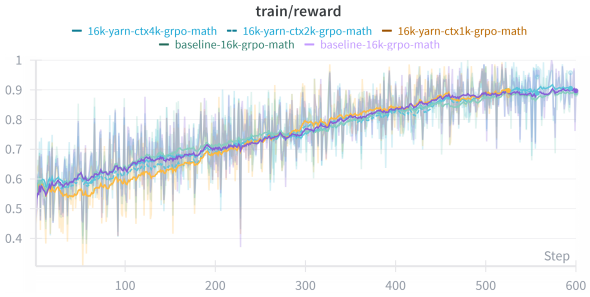

図 5　コンテキストウインドウ別の GRPO 訓練報酬

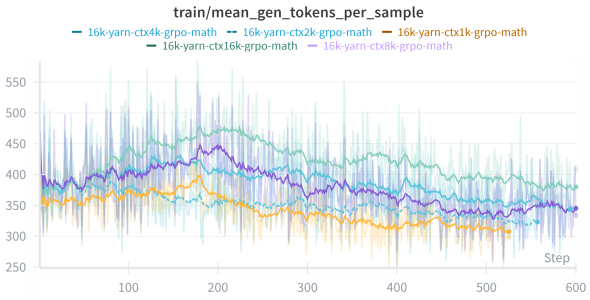

図 6　コンテキストウインドウ別の平均生成トークン数

6）https://github.com/mcleish7/retrofitting-recurrence