FYP Interim Report
Large Language Models for Formal Theorem Proving

Xijia Tao, 3035767762

January 2024

Abstract

In recent years, significant progress has been made in deep learning (DL), particularly with the development of large language models (LLMs) ChatGPT. These models have been successfully applied to tasks such as translation, summarization, and question-answering [1]. This reflects the strong capabilities of models in language understanding and generation. However, their potential in formal theorem proving (FTP), a field that involves rigorous reasoning and mathematical logic to establish the correctness of statements or prove theorems, remains relatively unexplored. FTP is traditionally time-consuming. Despite this, achievements in FTP hold greater significance due to the wide range of formal mathematics involved and the potential applications in the real world [2]. In 2023-24 Sem 1, we conducted a review of prevailing methods for this problem. In addition, 2 research questions (RQs) were identified and investigated. For RQ1, Code-Llama was finetuned on a domain dataset, with the evaluation pipeline built locally. The next steps of this project involve further training our model, proposing methods for RQ2, and conducting extensive evaluation.

1 Background

In this section, we introduce the background of FTP and then discuss some earlier work by the machine learning (ML) community in the field. Next, we provide a brief overview of the recent progress in LLMs and some recent work leveraging LLMs to conduct theorem proving. Finally, we elaborate on our motivation behind this FYP.

1.1 Formal Theorem Proving

In the past decade, many mathematicians have established the correctness of their proofs for theorems in a formal system. This not only automatically checks for any error in a written proof, but also allows others to trust in a proof's conclusion without going through it step by step. Such systems are also employed in other fields, such as software verification [2]. Currently, FTP is mainly performed using specialized proof assistants like Coq, Isabelle, and HOL Light [3]. These systems provide a framework for mechanically checking proofs, but they often require users to write detailed and explicit formal proofs, which can be laborious and error-prone. In addition, the learning curve associated with these proof assistants can be steep, posing a barrier to newcomers in the field.

1.2 Related Work before ChatGPT-like LLMs

Before the widespread use of foundation models, attempts by AI researchers in the field of FTP have centered around training a specialized model. In this section, we outline some representative work that involves training from scratch.

GPT-f [4] In 2020, GPT-f introduced the use of transformer networks for FTP. It leveraged pre-training and then iterative learning of two objectives that are specifically related to FTP, i.e., proofstep and outcome objective. This marked an important step in the application of DL techniques to automate aspects of mathematical proof.

Proof Artifact Co-Training [5] More recently, in the formal system Lean, extracting data from kernel-level problem statements and theorems derived during proof generation was found to benefit a model's performance. Specifically, it leverages these abundant self-supervised data for training for multiple curated tasks, alongside the proofstep objective proposed in the GPT-f paper. They verified the effectiveness of their method on a held-out suite of test theorems, outperforming the previous methods in the pass rate of generated proofs.

Expert iteration [2] In 2022, researchers proposed to leverage proofs generated by a model itself as training data. By interleaving proof search with learning, the method outperforms proof search only under the same compute budget. As a follow-up work from GPT-f, it proposed to replace the outcome objective with proofsize objective. The later achieves better performance and prefers goals leading to short proofs during proof search. Since the cost of reinforcement learning (RL) is significantly smaller than that of proof search, this approach combines the best of both worlds in a way that

heuristics learned from RL and rigorous reasoning performed in search are applied at the same time. Their model, in the same architecture as the previous work GPT-f, can be iteratively improved with proof search trajectories as its training data. Their method has achieved state-of-the-art (SOTA) on the *miniF2F* benchmark [6] which is widely acknowledged in the field of FTP to this date.

HyperTree Proof Search [7] In the same year as [2], a paper introduced a new search algorithm, inspired by AlphaZero [8]. It is an adaptation based on the Monte Carlo tree search for finding proofs in unbalanced hypergraphs. With a detailed ablation study, the paper suggested that online training works better than the expert iteration approach proposed in the previous work. When generalized from domains far from the training distribution, the method remains performant and outwits the previous SOTA GPT-f in the Metamath formal system. In another system Lean, it has also improved SOTA on the miniF2F benchmark.

1.3 Large Language Models

LLMs' development was largely motivated by OpenAI when they first invented ChatGPT. Now, many organizations have trained and open-sourced their own LLMs. Among them, Llama and its variants are one of the most popular choices for developers and users. Llamas are decoder-only transformers [9] [10]. Although it is not yet clear if ChatGPT and GPT-4 use the vanilla decoder-only architecture, many LLMs, including those fine-tuned from Llamas, are based on the same general architecture as previous GPTs (e.g., GPT-2 and GPT-3). More recently, Mixture of Experts (MoE) models have gained widespread attention. They are represented by Mixtral 7X8B, with a performance comparable or better than Llama 2 70B and ChatGPT on various benchmarks [11]. For such architecture, the number of parameters activated during inference is much less than the total number of parameters. It allows different experts to be employed for different inputs, thus maintaining comparable performance to models of large size.

While these language models have shown great potential, their application in the field of FTP remains relatively unexplored. In the following section, we introduce some recent work in this endeavor.

1.4 Related Work After ChatGPT

Subgoal-based Demonstration Learning [12] The work proposed to leverage existing LLMs to generate subgoal-based proofs from informal proofs. This is done via iterative interaction with ChatGPT and verification with the Isabelle prover to ensure that the constructed subgoals are easily provable by the LLM. The resulting subgoal-based proofs and their corresponding formal sketches are used as demonstration examples. In addition, the authors point out that the selection and order of in-context examples matter for the performance of demonstration learning. To achieve this, a diffusion model was trained to generate the most effective combination of examples for translating a subgoal-based proof to its corresponding formal sketch.

Lyra [13] Based on LLMs, a new framework employing two correction mechanisms has been proposed. On the one hand, in the post-processing stage, tool correction leverages prior knowledge of predefined prover tools to replace incorrectly used tools. This has been shown to mitigate the hallucination problem. On the other hand, conjecture correction involves interaction between an LLM and a prover to refine proof conjectures with error messages. With the two mechanisms, Lyra has improved the previous SOTA on miniF2F.

ReProver [14] Retrieval-augmented generation has developed as a technique to aid LLMs in integrating solid facts into their response. Recently, researchers open-sourced fine-grained annotation of premises in proofs. This contributes data with strong supervising signals for the task of premise selection, which is a key bottleneck in FTP. With this dataset, a retrieval-augmented prover was proposed. It consists of a retriever and a language model for tactic generation. Instead of larger language models like ChatGPT and LLaMA or models that have undergone domain-specific pretraining like Minerva [15], the researchers chose to continue training from a small (299-600M parameters) and generic. This serves as an orthogonal direction to the previously introduced works.

1.5 Motivation

Our motivation for investigating the use of LLMs for FTP is primarily the same as all the prior work in this field. On the one hand, LLMs currently lack sufficient reasoning ability, which many believe is the key to artificial general intelligence (AGI). An improved performance on FTP benchmarks implies an improved reasoning ability, which can potentially be extended to tasks other than theorem proving. It thus can be considered as a step towards AGI. On the other hand, FTP demands math expertise and is

time-consuming for both human experts and traditional search-based methods. LLMs can arise as a tool for mathematicians formulating their proofs. With automated proof generation, it is even possible to prove unproven conjectures. Among all the unproven conjectures, those which demand intensive calculations are the most suitable ones for computers. An example is the four-color theorem, which was proven with the aid of the Coq formal system.

The combination of LLMs and formal theorem provers holds great potential for advancing scientific discovery. However, current approaches for this problem have shown limited performance on relevant benchmarks. The miniF2F benchmark [6] comprises a few hundred problem statements sourced from high-school-level mathematics competitions and the International Mathematical Olympiad, as well as material from high school and undergraduate mathematics courses. The SOTA method has achieved only around 50% accuracy on this benchmark, indicating ample room for improvement. With stronger LLMs available every month, we see a great opportunity for standing on the shoulders of giants and introducing novel methods to improve SOTA.

2 Objective

We narrowed down the scope of our project to addressing two research questions (RQs). After experiments on both RQs, we will document our findings in a research paper. For RQ1, we will make our fine-tuned models publicly available. For RQ2, we will open-source the implementation of our method in Python.

2.1 RQ1 - Generalist vs Specialist

Which one is more suitable for FTP - generically capable models, exemplified by GPT-4, or smaller expert models, represented by Code-Llama fine-tuned on task data?

On the HumanEval benchmark [16], Code-Llama 34B achieves a 48.8% pass rate, outperforming ChatGPT by a small margin [17]. GPT-4 achieves 67% on the same benchmark, which is significantly higher. However, when sampling Code-Llama's response 100 times, it can reach a remarkable 93% pass rate. Therefore, we argue that Code-Llama possesses abundant knowledge in coding and a strong reasoning ability. FTP can also be viewed as a coding task. Therefore, it is sensible to compare Code-Llama and GPT-4's performance on FTP benchmarks. Extrapolating from their performance on HumanEval, we hypothesize that GPT-4 is better at FTP. However, after fine-tuning

on domain-related datasets, Code-Llama might in turn outperform GPT-4.

2.2 RQ2 - Inference Augmentation

How to elicit the full reasoning power of frozen LLMs when conducting FTP?

It is computationally expensive to train an LLM with billions of parameters. Hence, it is ideal to intervene in the inference process instead of training to improve FTP performance. We propose two ideas that might work out, namely incorporating LLM proof generation with proof search and multi-agent collaboration. We leave the investigation as follow-up work in Sem 2.

3 Methodology

In this section, we introduce our methodology for investigating RQ1. This includes environment setup, the training dataset, fine-tuning settings, and the evaluation benchmark. Then we elaborate on the potential methods to address RQ2.

3.1 RQ1

3.1.1 Experiment Settings

We selected Isabelle/HOL as the formal environment. It is widely used and has an active community contributing proofs to theorems. In addition, its formalization language is high-level with layers of abstraction, also incorporating multiple prover tools (e.g., Sledgehammer). This makes writing a proof much easier in comparison to other systems like Metamath.

For injecting formal math knowledge into Code-Llama, we identified a dataset contributed in a paper last year [18]. The dataset is a parallel corpus of formal-informal statement pairs. The informal data are diverse and flexible, making it suitable for LLMs to acquire generalization ability to different math domains. With more than 200K data points in Isabelle, LLMs can learn much about FTP without overfitting.

Apart from Isabelle, it also provides some data in Lean, another formal system. This makes multilingual fine-tuning of Code-Llama possible.

3.1.2 Experiment Procedure

The experiment procedure mainly involves two steps.

1. Fine-tune Code-Llama 7B using the train set:
    - At an initial trial run, we took 10% of the train set, i.e., 24K formal-informal pairs in Isabelle, and conducted supervised fine-tuning (SFT) with LoRA [19].
    - Loss was computed on both the input (an informal statement and instruction to translate) and the output (the formal statement).
    - Currently, we only have access to RTX 3090 cards with 24GB of GPU memory. Due to this memory constraint, LoRA was necessary to continue training the 7B LLM. To achieve parallelism in multiple GPUs, we employed ZeRO Stage 3 from DeepSpeed.
    - A cosine learning rate (lr) scheduler with a 5e-4 initial lr and a 0.1 ratio of linear warmup was employed. Batch size and gradient accumulation steps were set to 2 and 8, respectively, for each of the 4 GPUs. Code-Llama was trained for 2 epochs with bf16 precision. For LoRA settings, we set r=8, alpha=16, dropout=0.05. We only targeted the Q and V matrices from the attention layers.
2. Evaluate the fine-tuned model and GPT-4 on FTP benchmarks:
    - miniF2F is the most recognized benchmark in this field currently. We are also paying attention to other influencing benchmarks.
    - We set up the evaluation pipeline for Isabelle. Specifically, we extracted Isabelle proofs in the form of code from a model's response. Via the Portal-to-Isabelle API [20], the generated proofs can be sent to Isabelle to establish their correctness.
    - We call the percentage of proofs without any error message as the pass rate. In this way, pass rate@1, pass rate@10, etc., with @n meaning sampling a model n times, can be calculated. The resulting pass rates can be compared to those reported in previous papers in this field.

3.2 RQ2

Two tracks have been identified to improve the FTP performance of pre-trained LLMs without modification to parameters.

3.2.1 Integration of Proof Search

Before the emergence of foundation models like ChatGPT, proof search has been widely used in concurrence with DL models for FTP. However, in the last year, to the best of our knowledge, no work effectively integrates prior knowledge on proof search with LLMs. Therefore, we found this as a rewarding direction to investigate. On an initial thought, we would like to consider A* search and Monte Carlo tree search. LLMs can be viewed as a heuristic and prompted for the value of a specific state during proof search.

3.2.2 Multi-Agent Collaboration

In reinforcement learning, a multi-agent framework has been developed for various learning algorithms, such as DDPG and PPO. When viewing LLMs as agents, it is natural to formulate a multi-agent or multi-LLM paradigm for either collaboration or competition. There are some related work last year [21] [22]. For theorem proving, multiple LLMs working together towards solving a single problem might be more effective than a single LLM. Therefore, we would like to investigate this in Sem 2.

4 Peliminary Results for RQ1

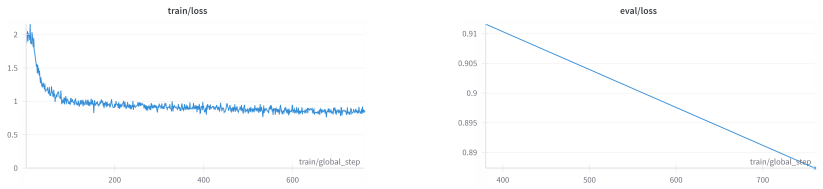

Figure 1: Language modeling loss during SFT on the train set (left) and at the end of SFT on the evaluation set (right) for the initial trial run

Figure 1 plots the change in standard language modeling loss during fine-tuning Code-Llama 7B. We observe that the loss first decreases and then plateaus at around 0.9. The loss remained constant before the end of the first epoch. This is not expected as all training data points should be diverse and challenging. Hence, the model should be able to improve from learning the remaining data in the dataset. We note that the loss for the evaluation set is also around 0.9, showing no sign of overfitting. We plan to investigate this odd phenomenon in Sem 2.

5 Tasks Accomplished

Below outlines a list of accomplished tasks in Sem 1, 2023-24.

1. Survey of topics in AI4Science
    - At the beginning, we were not certain about the specific science discipline that we would like to delve in and employ LLMs to solve its problems. We have read related articles in AI for Science in general, as well as research papers aiming at a specific domain problem, e.g., drug discovery and material property prediction.
    - With a general picture of AI4Science, we finally decided to focus on formal theorem proving, a branch of mathematics and logic. This decision was arrived considering the level of domain knowledge required, feasibility of implementation, and the impact of FTP.
2. Literature review
    - We decided to categorize the related work by their publish date - either before OpenAI introducing ChatGPT or after that. This is because at an initial glance, we discovered that the two categories have significant difference in the method employed.
    - We believe that reading papers in the two categories separately would give us a better understanding of how the problem is traditionally solved using DL, as well as the impact of LLMs in this science domain. With understanding of the traditional solutions, we could rediscover techniques that nowadays researchers might overlook because of the power of LLMs, hence combining the best of both worlds.
3. Proposal of research questions

- The two RQs serve as the main objective of our FYP. The first RQ is a binary classification problem (i.e., generalist vs specialist), while the second is open-ended (finding an improvement).

4. Experimenting for RQ1
    - The evaluation pipeline for Isabelle was built locally.
    - Available training datasets and benchmarks were surveyed.
    - A trial run of fine-tuning Code-Llama 7B on domain dataset was conducted.

6 Tasks to be Accomplished in Sem 2

1. **RQ1**
    - Finish training Code-Llama on the full dataset.
    - Investigate data augmentation techniques for generating more train data.
    - Collect GPT-4 responses in proof generation on miniF2F.
    - Evaluate the proofs generated by both models in the Isabelle formal environment.
2. **RQ2**
    - Formulate our method based on search algorithm, multi-agent, etc.
    - Evaluate on benchmarks
    - Compare its performance with other models and decoding strategies (e.g., Chain-of-Thought [23]).
3. General *TODO*s
    - Publish fine-tuned model weights for RQ1.
    - Organize and open-source implementations of our methods in Python.
    - Summarize findings in a paper.

7 Conclusion

In conclusion, our project centers around two research questions in the application of LLMs to FTP. In Sem 1, We completed literature review, setup of evaluation pipeline

and preliminary experiments. Our next steps include further experiments for RQ1, method formulation and implementation for RQ2, and evaluation on benchmarks for both RQs.

References

[1] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27730–27744, 2022.

[2] S. Polu, J. M. Han, K. Zheng, M. Baksys, I. Babuschkin, and I. Sutskever, "Formal mathematics statement curriculum learning," *arXiv preprint arXiv:2202.01344*, 2022.

[3] J. Harrison, J. Urban, and F. Wiedijk, "History of interactive theorem proving," in *Computational Logic*, 2014.

[4] S. Polu and I. Sutskever, "Generative language modeling for automated theorem proving," *arXiv preprint arXiv:2009.03393*, 2020.

[5] J. M. Han, J. Rute, Y. Wu, E. W. Ayers, and S. Polu, "Proof artifact co-training for theorem proving with language models," *arXiv preprint arXiv:2102.06203*, 2021.

[6] K. Zheng, J. M. Han, and S. Polu, "Minif2f: a cross-system benchmark for formal olympiad-level mathematics," *arXiv preprint arXiv:2109.00110*, 2021.

[7] G. Lample, T. Lacroix, M.-A. Lachaux, A. Rodriguez, A. Hayat, T. Lavril, G. Ebner, and X. Martinet, "Hypertree proof search for neural theorem proving," *Advances in Neural Information Processing Systems*, vol. 35, pp. 26337–26349, 2022.

[8] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[10] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[11] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mixtral of experts," 2024.

[12] X. Zhao, W. Li, and L. Kong, "Decomposing the enigma: Subgoal-based demonstration learning for formal theorem proving," *arXiv preprint arXiv:2305.16366*, 2023.

[13] C. Zheng, H. Wang, E. Xie, Z. Liu, J. Sun, H. Xin, J. Shen, Z. Li, and Y. Li, "Lyra: Orchestrating dual correction in automated theorem proving," *arXiv preprint arXiv:2309.15806*, 2023.

[14] K. Yang, A. M. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar, "Leandojo: Theorem proving with retrieval-augmented language models," *arXiv preprint arXiv:2306.15626*, 2023.

[15] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra, "Solving quantitative reasoning problems with language models," 2022.

[16] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[17] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.

[18] A. Q. Jiang, W. Li, and M. Jamnik, "Multilingual mathematical autoformalization," 2023.

[19] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[20] A. Q. Jiang, S. Welleck, J. P. Zhou, W. Li, J. Liu, M. Jamnik, T. Lacroix, Y. Wu, and G. Lample, "Draft, sketch, and prove: Guiding formal theorem provers with informal proofs," *arXiv preprint arXiv:2210.12283*, 2022.

[21] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, *et al.*, "Metagpt: Meta programming for multi-agent collaborative framework," *arXiv preprint arXiv:2308.00352*, 2023.

[22] D. Huang, Q. Bu, J. M. Zhang, M. Luck, and H. Cui, "Agentcoder: Multi-agent-based code generation with iterative testing and optimisation," *arXiv preprint arXiv:2312.13010*, 2023.

[23] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824–24837, 2022.