PROBING STRUCTURAL SIGNALS IN LEAN4 PROOF GRAPHS WITH GNNS

Jiayi Wu

1 INTRODUCTION

Formal theorem proving is a branch of computer science where proofs are constructed in a fully formalized, machine-checkable language Yang et al. (2024). Systems such as Lean allow mathematicians and computer scientists to write proofs whose correctness can be verified automatically by a proof assistant.

While large language models (LLMs) has been widely adopted to generate both informal and formal proofs, most of the existing research focus either on post-training optimization tweaks on LLMs or reinforcement-learning methods Li et al. (2024). These approaches are impressively strong at generating informal proofs, but formal theorem proving offers a very different regime:

- Proofs have fully explicit syntax and semantics
- Proof states have unambiguous structure
- The correctness of every proof step is machine checkable/verifiable

Therefore, formal proofs naturally provide a cleaner environment than natural-language informal proofs for studying models' compositional, structured reasoning processes. That said, a fundamental question yet to be answered - or even asked - is that, before adopting complex model architectures, whether we're able to capture certain latent structures in mathematical proofs through simple representation learning.

Problem Setup. In this project, I translate each Lean proof into a state-tactic graph, and frame next-tactic prediction as predicting the next tactic being used given a proof-state graph. A couple of key questions arise:

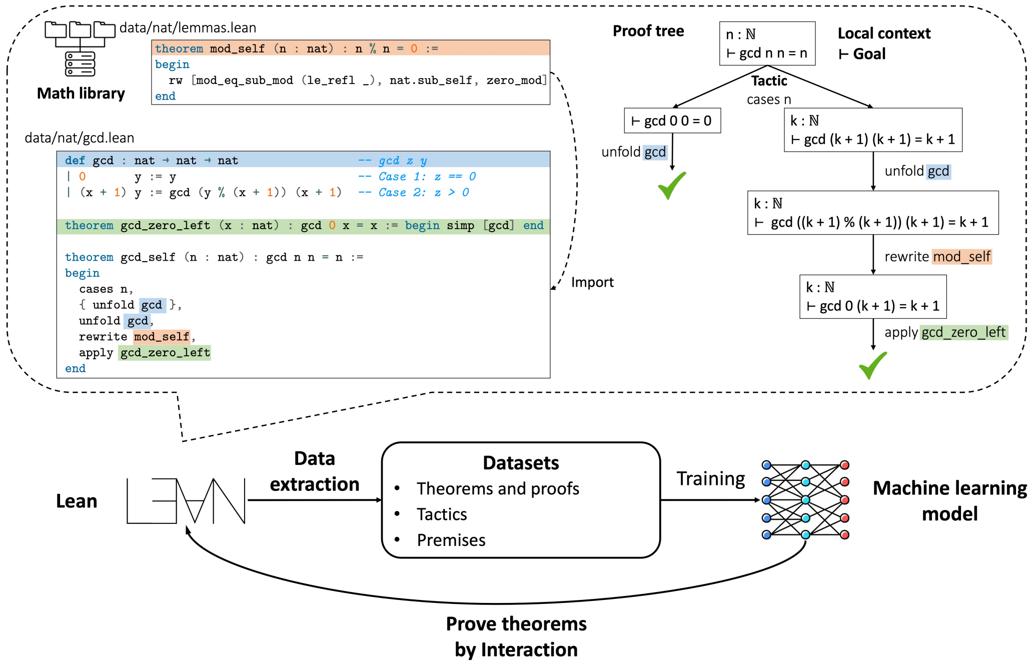
- How predictable is the next tactic from proof structures?
- In particular, is prediction driven graph topology, tactic identities, or simple frequency biases? If proof graphs encode certain recurring structural patterns, a GNN should outperform simple bag-of-tactics baselines and meaningfully degrade when structure is removed.

1.1 RELATED WORK: NEURAL THEOREM PROVING

Recent efforts in automated reasoning have increasingly explored neural approaches to guiding proof search in interactive theorem provers such as Lean, Isabelle, and Coq. Systems such as GPT-f Polu & Sutskever (2020), DeepSeek-Prover Ren et al. (2025), and other LLM-guided provers treat proof generation as a sequence prediction problem: models are trained to produce the next tactic or subgoal transformation given a textual or structured encoding of the current proof state. These approaches demonstrate impressive empirical performance Yu et al. (2025) but often rely heavily on strong priors, large-scale pretraining, and extensive retrieval augmentation rather than on explicit structural understanding of proofs.

2 DATA

I built a graph dataset based on the LeanDojo dataset Yang et al. (2023), which contains thousands of Lean proof traces from the Mathlib: 98,734 theorems from 3,384 Lean files; definitions of 130,262 premises; 217,776 tactics, 129,243 of them with at least one premise.



data/nat/lemmas.lean

```
theorem mod_self (n : nat) : n % n = 0 :=
begin
  rw [mod_eq_sub_mod (le_refl _), nat.sub_self, zero_mod]
end
```

data/nat/gcd.lean

```
def gcd : nat → nat → nat                       -- gcd z y
| 0       y := y                                -- Case 1: z == 0
| (x + 1) y := gcd (y % (x + 1)) (x + 1)        -- Case 2: z > 0

theorem gcd_zero_left (x : nat) : gcd 0 x = x := begin simp [gcd] end

theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
  apply gcd_zero_left
end
```

Moreover, LeanDojo provides the proof traces of Mathlib theorems, where each proof is a sequence of state-tactic transitions. An illustrative example is provided below, where s_i represents proof states, and t_i represents tactics applied:

$$s_0 \to t_0 \to s_1 \to t_1 \to \cdots \to s_n$$

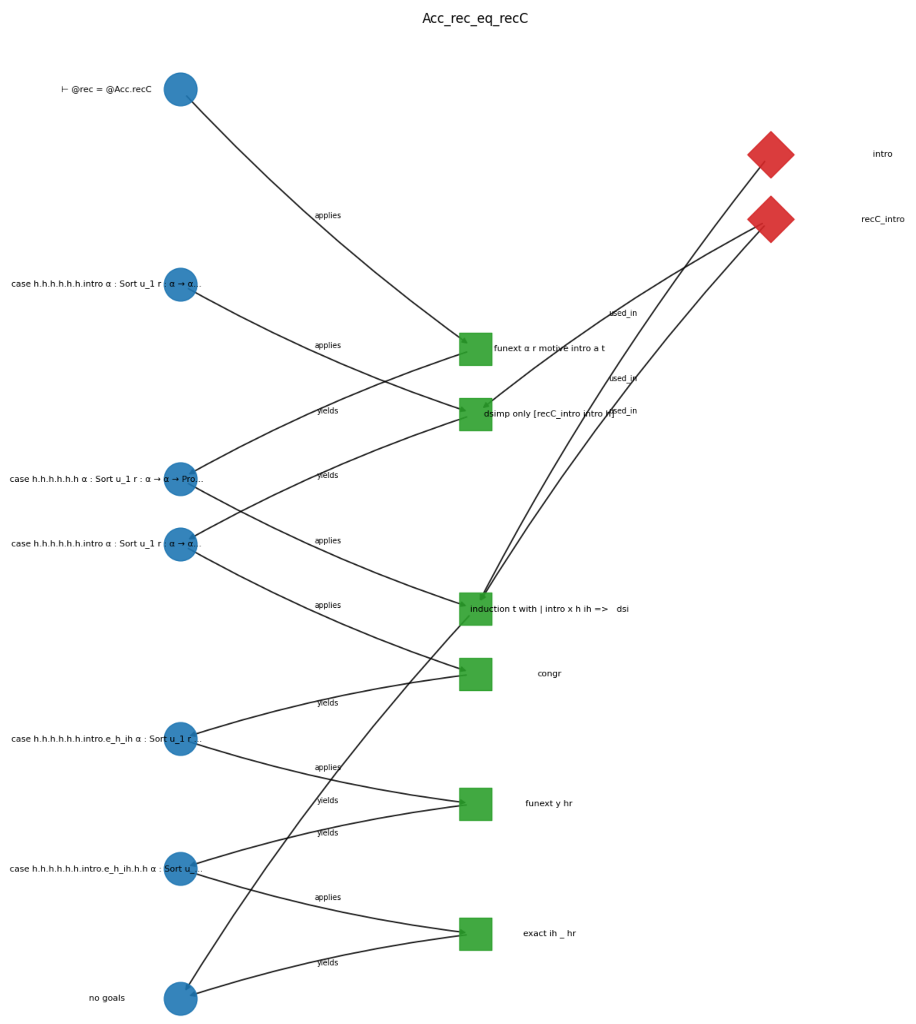
Graph Construction. For each of such proofs in LeanDojo, I generate a directed acyclic graph $G(V, E)$, where each node $v_i \in V$ can represent:

- $v_i \in V_s \subseteq V$, a state (proof goal in Lean)
- $v_i \in V_t \subseteq V$, a tactic
- $v_i \in V_p \subseteq V$, a premise (lemma references)

and each edge $(v_i, v_j) \in E$ represents a relation between nodes, which includes:

- $v_i \in V_t, v_j \in V_s$, applying a tactic node yields a state node
- $v_i \in V_s, v_j \in V_t$, a tactic node is applied to a state node
- $v_i \in V_p, v_j \in V_t$, a premise node is referenced in a tactic node

With this proof-graph translation setup, each graph corresponds to a proof step, and the graph label is the tactic used at the next state. An illustrative example of the proof graph for Acc_rec_eq_recC is below:


Acc_rec_eq_recC
⊢ @rec = @Acc.recC
case h.h.h.h.h.h.intro α : Sort u_1 r : α → α...
case h.h.h.h.h.h α : Sort u_1 r : α → α → Pro...
case h.h.h.h.h.h.intro α : Sort u_1 r : α → α...
case h.h.h.h.h.h.intro.e_h_ih α : Sort u_1 r ...
case h.h.h.h.h.h.intro.e_h_ih.h.h α : Sort u_...
no goals
funext α r motive intro a t
dsimp only [recC_intro intro H]
induction t with | intro x h ih =>   dsi
congr
funext y hr
exact ih _ hr
intro
recC_intro
applies
yields
used_in

Tactics. For tactics, I scraped a fixed set of 499 canonical tactics from this online collection lea (2025), with each tactic assigned a unique ID.

States. For proof states, I tried two different representations:

- Using symbolic IDs (which means almost every state is unique)
- Encoding state-node strings using a SentenceTransformer model, which produces matrix of LM embeddings, where every string is indexed into the embedding bank. This allowed the model to recognize semantically similar goals.

3 Methodology & Results

3.1 Setup

The task is graph-level classification:

$$G \mapsto \text{tactic} \in \{1, \dots, 499\}$$

Node features contains:

- Node type embeddings: state, tactic, or premise

- Tactic ID embeddings: the categorical ID for tactic nodes; absent for state or premise nodes
- Semantci embedding: projected LM embeddings for state nodes; absent for tactic or premise nodes

The final node feature is therefore:

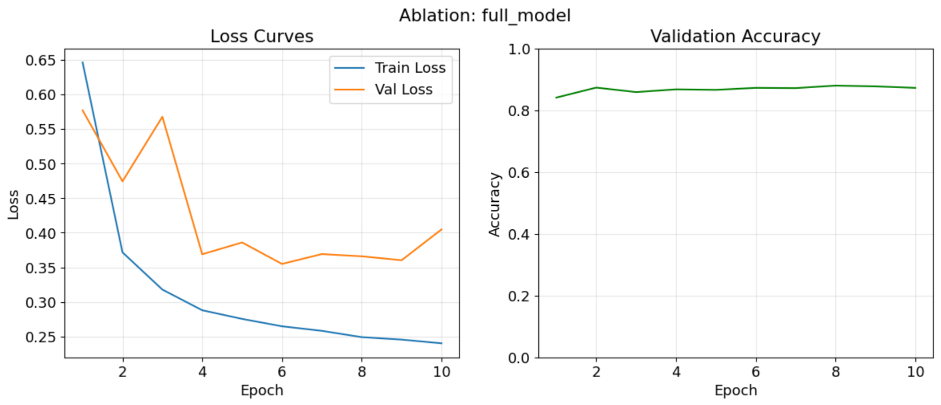
$$x_v = \text{type\_emb}(v) || \text{tactic\_emb}(v) || \text{semantic\_emb}(v)$$

The model setup is very simple since I want to keep everything interpretable:

- 2-layer GraphSAGE encoder
- Cosine-similarity–gated message passing over semantic channels
- Global Attention Pooling to aggregate graph-level representation
- 2-layer MLP classifier predicting the next tactic

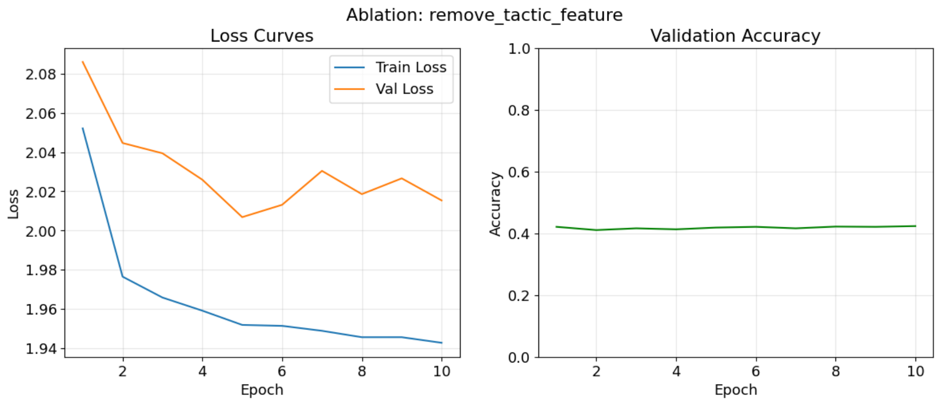
3.2 Experiments & Ablations

A key part of this project was a thorough ablation study to understand what signal the model is actually using. The success metric is primarily defined by the accuracy of next-tactic prediction, along with expectation for a significant accuracy drop under edge-shuffling ablation, which indicates the significant role of graph topology in the learning processes. For each ablation, I used the format of Hypothesis-Result-Conclusion to make sure that the analysis is clear. Below is the Result of the full model and an overview of ablation results.



Condition	Final Val Accuracy
Full model	**0.875**
No tactic embedding	0.423
No node type	0.874
Shuffle edges	0.873
Shuffle tactic IDs	0.777
Shuffle targets (sanity check)	0.000
MLP baseline (bag-of-tactics)	0.836
Add semantic embeddings	0.888

3.2.1 Ablation: No tactic embedding

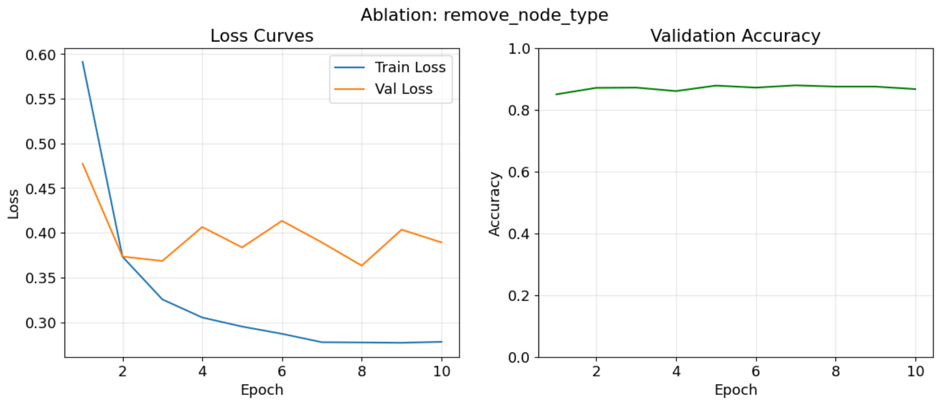


Hypothesis. If the GNN relies heavily on tactic identity, then removing tactic ID embeddings should cause validation accuracy to collapse.

Result. Performance drops from the full model's 0.875 to 0.423.

Conclusion. Tactic identity is the dominant predictive signal in the dataset; the model uses tactic labels far more than any structural or semantic property of the proof graph.

3.2.2 Ablation: No node type

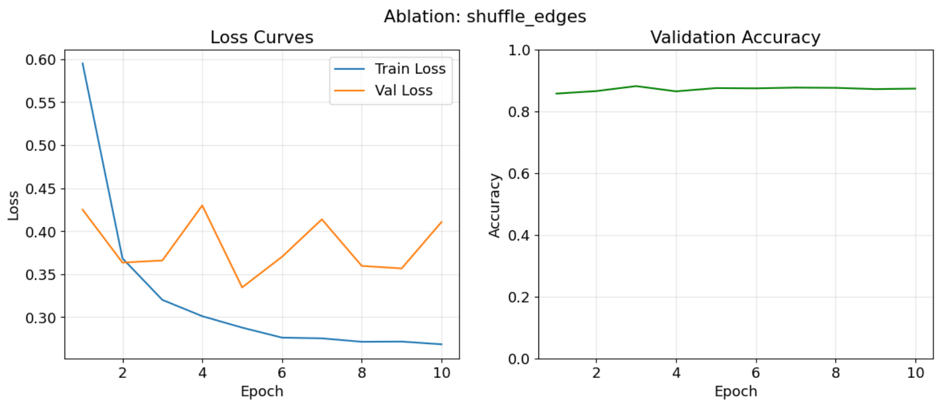


Hypothesis. Since proof states and tactic nodes play different functional roles, removing node-type features should degrade performance.

Result. Accuracy remains effectively unchanged at 0.874.

Conclusion. Graph connectivity already implies node role, so explicit node-type features are redundant. The model infers "state vs. tactic" positionally from the topology rather than the feature.

3.2.3 Ablation: Shuffle edges

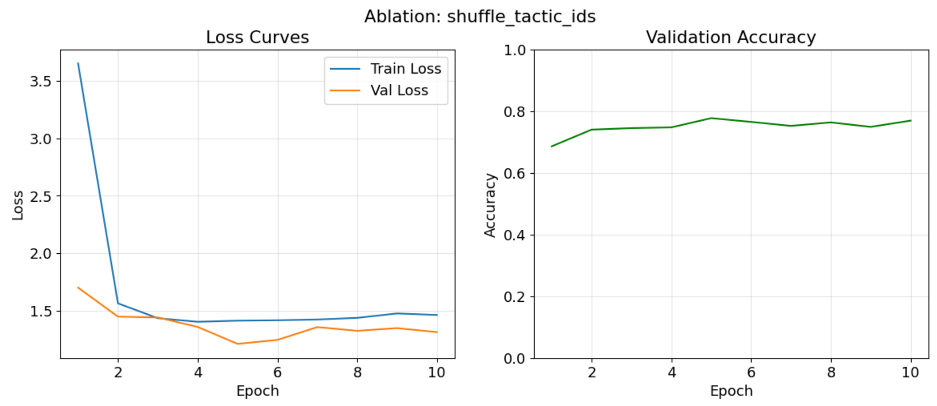


Hypothesis. If the proof-graph structure is meaningful to prediction, shuffling the edges should cause a substantial accuracy drop.

Result. Accuracy remains essentially unchanged at 0.873.

Conclusion. The GNN is not exploiting structural information. It behaves like a permutation-invariant set encoder rather than a graph encoder, indicating that topology contributes almost no useful signal under the current representation.

3.2.4 Ablation: Shuffle tactic IDs

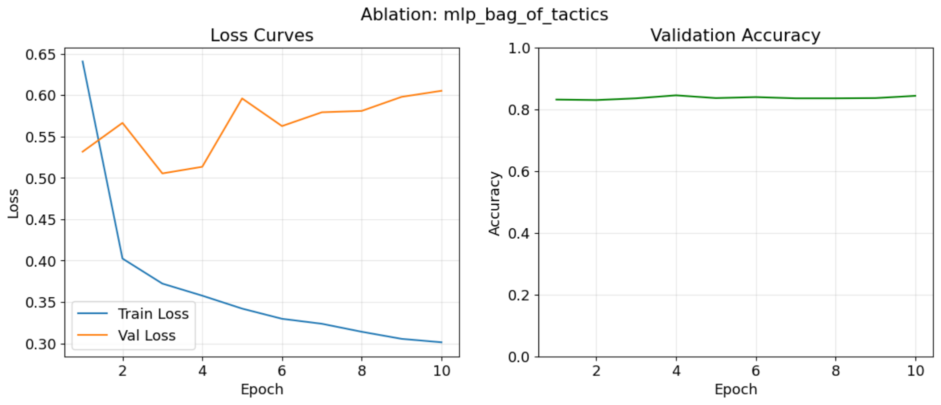


Hypothesis. If the model is memorizing specific tactic names, shuffling tactic IDs should destroy accuracy.

Result. Accuracy decreases to 0.777.

Conclusion. A meaningful drop occurs, but not a collapse. This indicates that while the model indeed relies on tactic identity, some higher-level semantic regularities remain detectable even after shuffling, though the signal is substantially weakened.

3.2.5 Ablation: MLP baseline



Hypothesis. If graph structure contributes little, then a simple MLP operating on tactic-frequency histograms should achieve strong performance.

Result. The MLP baseline achieves 0.836.

Conclusion. Its performance approaches the GNN, confirming that most predictive power arises from tactic-distribution statistics rather than relational structure within the proof graph.

3.2.6 Ablation: Semantic embeddings

Hypothesis. Augmenting nodes with language-model embeddings of proof states should allow the model to capture semantic similarity and thus improve generalization.

Result. Accuracy increases from 0.875 to 0.888.

Conclusion. Semantic augmentation provides a small but measurable improvement. However, the modest gain suggests that the model still lacks a strong structural inductive bias and does not yet extract deep relational information from proof graphs.

4 Discussions

While the overall accuracy of 87% for next-tactic prediction task seems satisfactory, the ablation results indicated that the GNN model did not actually learn graph structures from the proofs. Instead, it behaved almost entirely like a bag-of-tactics frequency classifier plus a weak semantic encoder of local states.

In response to our research questions, the latent structural regularity across Lean proofs is much weaker than expected, and certainly not captured by our current graph encodings and the basic GNN message passing over raw proof graphs. However, this somewhat supports a broader hypothesis: proof structures is not surface syntactic, and the recurring patterns may appear at a level of higher-order substructures rather than raw proof graphs.

4.1 Limitations & Challenges

LM embeddings for proof states. One key limitation of our current setup is that LM embeddings of proof states might be fundamentally limited: proof states that are logically equivalent but syntatically permuted may have very different text embeddings; conversely, two states with similar syntax but totally different logical roles may be embedded as very close. Therefore, in general, LM embeddings probably do not respect proof-theoretic equivalence but only introduce metric noise, blur rather than sharpen the proofs' logical structures.

Non-canonical proof graphs. The proof graph extracted from Lean is not a canonical structural object: small syntactic or procedural changes in how a tactic decomposes a goal can lead to disproportionately large changes in the resulting graph. Two proofs that are semantically identical may differ in the number of intermediate nodes, the branching pattern induced by automation calls, or the order in which subgoals are discharged. As a result, the graph representation is highly sensitive to low-level proof engineering choices rather than the underlying mathematical structure. This sensitivity produces a representation space where graphs that look drastically different may encode essentially the same reasoning step, making it difficult for a GNN to learn stable inductive patterns.

Sparse recurrence of similar states. Proof states seldom recur in a way that provides dense training signal for learning generalizable reasoning patterns. Even when two states are logically equivalent, their textual or syntactic forms may differ significantly, preventing them from being recognized as instances of the same conceptual configuration. This sparsity of repeated local structure limits the model's opportunity to learn reusable abstractions, motifs, or higher-level schemas. Therefore in effect, the dataset provides little redundancy and therefore weak support for robust structural learning.

4.2 Ethical Implications

Broader societal issues relevant to this problem space. Although proof-graph learning may appear at first glance to be an abstract and purely technical investigation, it intersects with several important societal discussions about the future of automated reasoning systems. Contemporary discourse around artificial general intelligence (AGI) often takes mathematical reasoning as a proxy for "general intelligence," therefore improvements in automated theorem proving are sometimes interpreted as signals of progress toward highly capable general reasoning systems Hendrycks et al. (2025). This framing can generate misplaced expectations or anxieties if the limitations of current methods are not well understood, and a more fundamental question would be: do mathematical reasoning capabilities necessarily indicate growing capabilities for other domains of learning and reasoning? Yang et al. (2024)

Another more concrete social domain is that trustworthy and interpretable reasoning systems have direct implications for technical AI governance Reuel et al. (2025). Formal verification, certified robustness, and specification-guided model behavior all rely on machine-checkable reasoning pipelines. Understanding when neural networks learn authentic inferential structure, and when they fail to, matters for safety-critical contexts such as autonomous systems, medical decision support, and software verification. Thus, even negative results about structure learning in proofs contribute valuable knowledge about the limits of model interpretability and the conditions under which automated reasoning can be meaningfully trusted.

Why Deep Learning is an appropriate approach. Deep Learning is a natural approach for this problem because modern neural architectures excel at learning distributed representations from high-dimensional, irregular data such as graphs, trees, and symbolic structures. If proofs contain reusable patterns, a GNN should, in principle, be able to discover them without manual feature engineering Li et al. (2024). Moreover, neural encoders can integrate semantic information, such as embeddings of proof states or natural-language descriptions of theorems, in ways that classical search-based provers cannot, making Deep Learning an appealing tool for bridging symbolic and semantic reasoning.

4.3 Future Directions

Beyond representation: library learning. A key open direction is to determine whether meaningful higher-level regularities exist in proofs that are obscured by surface-level statistical noise. Instead of relying on raw adjacency structures, the goal is to extract recurring patterns or reusable abstractions such as common subgoal transformations, canonical rewrite patterns, or tactical schemas that recur across proofs and domains. Techniques from library learning Ellis et al. (2020), program synthesis, and structure discovery Xin et al. (2025); Wang et al. (2023) may provide a principled way to identify invariant substructures even when the underlying graph representations differ syntactically. If successful, this approach would yield a more stable and semantically grounded basis for learning proof strategies than current node-edge encodings.

Alternative graph granularities My current approach to constructing graph representation may be too fine-grained to reflect the conceptual structure of reasoning. Exploring coarser or differently structured graph abstractions could provide more informative inductive biases (while that might require more manual labeling as well–there's a trade-off).

- **Tactic dependency graphs** Xin et al. (2025) explicitly represent how tactics depend on earlier subgoals, emphasizing strategic flow rather than syntactic decomposition.
- **CoT-derived DAGs** Zhang et al. (2025), inspired by chain-of-thought reasoning work, capture high-level inference steps without committing to Lean's proof-term structure.

In general, these alternative graph constructions may generate more stable and canonical representations that reveal true reasoning patterns rather than procedural artifacts.

Improved Structural Metrics Besides, a central limitation of my current GNN setup is the lack of a meaningful notion of ”logical closeness” between proof states. Inspired by Maene & Tsamoura (2025), developing richer similarity metrics based on theorem embeddings, historical frequency of co-occurrence, or similarity of proof terms could yield structural features that reflect the actual mathematical relationships rather than statistical signals or surface-level graph topology.

4.4 Reflection

Did the project achieve its goals? I think the core objective of the project was *partially* achieved. I successfully implemented a full end-to-end GNN pipeline for handling Lean proof graphs, including data parsing, graph construction, model design, training, and evaluation; but the more ambitious goal of discovering reusable, generalizable structural patterns across proofs was not quite met. The experimental results strongly suggest that raw proof graphs do not contain stable or learnable cross-proof structure at the granularity provided by the current dataset.

What changed over time? Given limited time, the trajectory of the project shifted slightly over time. Initially, I was planning to focus on a baseline GNN architecture applied to the raw extracted graph, while ablation studies revealed that the model was effectively not using structural information, with performance unchanged when edges were randomized. This prompted experiments with semantic embeddings, but despite these modifications, the model continued to rely predominantly on tactic-frequency signals. Therefore I'd say over time, it became clear that the dataset itself limited structural generalization.

What would I do differently? With hindsight, starting from low-level graph representations was not the ideal setup. A more fruitful approach would begin at a higher level of abstraction, rewrite patterns, or canonicalized subgoal transitions before committing to a GNN architecture.

What I learned. This project highlighted several key insights about the nature of proof data and the limits of graph learning:

- Proof states (features) are extremely sparse, with minimal recurrence of semantically equivalent configurations across different theorems.
- GNNs depend critically on structural patterns to generalize, but such patterns are largely absent in the current proof dataset and representation.
- Ablation studies are essential for understanding whether a model is learning structures or just statistical co-occurrence correlations; in my project, they revealed the dominance of tactic-level statistical signals.

Overall, the experience underscored that progress on learning-based theorem proving requires understanding of representation and data curation: the structural inductive biases must match the underlying mathematical semantics, rather than any syntactic artifacts of proof construction.

REFERENCES

Tactics, 2025. URL https://seasawher.github.io/mathlib4-help/tactics/.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B. Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning, 2020. URL https://arxiv.org/abs/2006.08381.

Dan Hendrycks, Dawn Song, Christian Szegedy, Honglak Lee, Yarin Gal, Erik Brynjolfsson, Sharon Li, Andy Zou, Lionel Levine, Bo Han, Jie Fu, Ziwei Liu, Jinwoo Shin, Kimin Lee, Mantas Mazeika, Long Phan, George Ingebretsen, Adam Khoja, Cihang Xie, Olawale Salaudeen, Matthias Hein, Kevin Zhao, Alexander Pan, David Duvenaud, Bo Li, Steve Omohundro, Gabriel Alfour, Max Tegmark, Kevin McGrew, Gary Marcus, Jaan Tallinn, Eric Schmidt, and Yoshua Bengio. A definition of agi, 2025. URL https://arxiv.org/abs/2510.18212.

Zhaoyu Li, Jialiang Sun, Logan Murphy, Qidong Su, Zenan Li, Xian Zhang, Kaiyu Yang, and Xujie Si. A survey on deep learning for theorem proving, 2024. URL https://arxiv.org/abs/2404.09939.

Jaron Maene and Efthymia Tsamoura. Embeddings as probabilistic equivalence in logic programs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=rEUbDhWaXh.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving, 2020. URL https://arxiv.org/abs/2009.03393.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition, 2025. URL https://arxiv.org/abs/2504.21801.

Anka Reuel, Ben Bucknall, Stephen Casper, Tim Fist, Lisa Soder, Onni Aarne, Lewis Hammond, Lujain Ibrahim, Alan Chan, Peter Wills, Markus Anderljung, Ben Garfinkel, Lennart Heim, Andrew Trask, Gabriel Mukobi, Rylan Schaeffer, Mauricio Baker, Sara Hooker, Irene Solaiman, Alexandra Sasha Luccioni, Nitarshan Rajkumar, Nicolas Moës, Jeffrey Ladish, David Bau, Paul Bricman, Neel Guha, Jessica Newman, Yoshua Bengio, Tobin South, Alex Pentland, Sanmi Koyejo, Mykel J. Kochenderfer, and Robert Trager. Open problems in technical ai governance, 2025. URL https://arxiv.org/abs/2407.14981.

Haiming Wang, Huajian Xin, Chuanyang Zheng, Lin Li, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, Jian Yin, Zhenguo Li, Heng Liao, and Xiaodan Liang. Lego-prover: Neural theorem proving with growing libraries, 2023. URL https://arxiv.org/abs/2310.00656.

Yutong Xin, Jimmy Xin, Gabriel Poesia, Noah Goodman, Qiaochu Chen, and Isil Dillig. Automated discovery of tactic libraries for interactive theorem proving, 2025. URL https://arxiv.org/abs/2503.24036.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models, 2023. URL https://arxiv.org/abs/2306.15626.

Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin Lauter, Swarat Chaudhuri, and Dawn Song. Formal mathematical reasoning: A new frontier in ai, 2024. URL https://arxiv.org/abs/2412.16075.

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, Ge Zhang, and Weiyang Liu. Formalmath: Benchmarking formal mathematical reasoning of large language models, 2025. URL https://arxiv.org/abs/2505.02735.

Yuanhe Zhang, Ilja Kuzborskij, Jason D. Lee, Chenlei Leng, and Fanghui Liu. Dag-math: Graph-guided mathematical reasoning in llms, 2025. URL https://arxiv.org/abs/2510.19842.