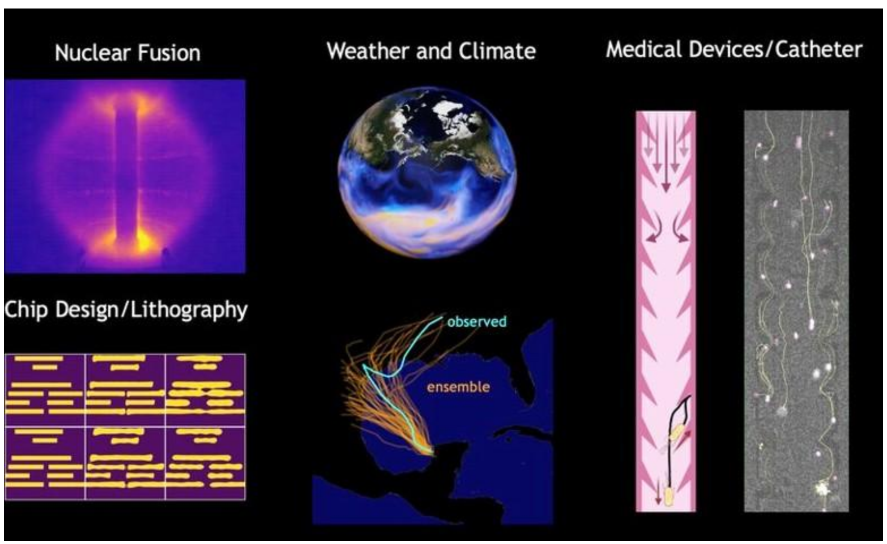
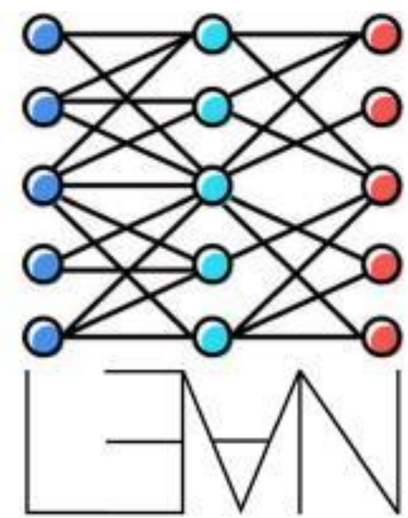
Verified Mathematical Reasoning with Lean and Large Language Models

Robert Joseph George, PhD Student Math & CS

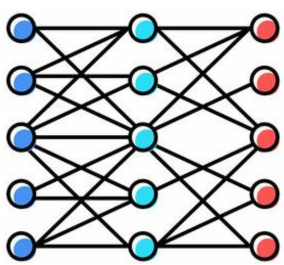




Neural Theorem Proving (NTP)

- Build ML models to prove theorems!

Large Language Models



Interactive Theorem Provers

Large Language Models (LLMs) for math

- LLMs achieve great performance in elementary, High school and now even in Olympiad/Competition Mathematics.
- **Data contamination**: The proofs may have appeared in the training data and were memorized by the model
- **Proof checking**: Difficult to check if the proofs are correct **(especially for research-level mathematics)**

Question: For every $a, b, b \neq a$ prove that

$$\frac{a^2+b^2}{2} > \left(\frac{a+b}{2}\right)^2.$$

Model output:

$$\frac{a^2+b^2}{2} > \left(\frac{a+b}{2}\right)^2$$

$$\iff \frac{a^2+b^2}{2} > \frac{a^2+b^2+2ab}{4}$$

$$\iff a^2+b^2 > \frac{a^2+b^2+2ab}{2}$$

$$\iff 2a^2+2b^2 > a^2+b^2+2ab$$

$$\iff a^2+b^2 > 2ab$$

$$\iff a^2+b^2-2ab > 0$$

$$\iff (a-b)^2 > 0$$

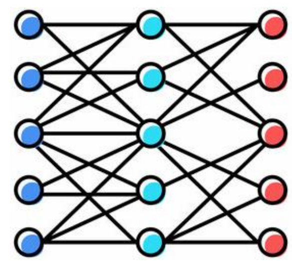
which is true, because the square of a real number is positive.

Neural Theorem Proving

Automated Theorem Provers

- SMT solvers, model checkers ATP systems in first-order logic, etc.
- Minimal efforts from humans.
- Limited expressiveness.

Large Language Models



- Automatic learning capability.
- Proof checking for research - level mathematics is very hard.
- Combine it with formal languages.

Interactive Theorem Provers

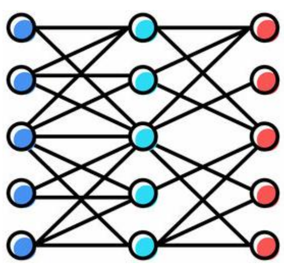
- Rigorous proof checking.
- Expressive logic, e.g., Dependent type theory.
- Lots of efforts from humans to write proofs.
- Human efforts to pick premises.

Neural Theorem Proving

Interactive Theorem Provers

Large Language Models

Run models locally or on a server
Automatically suggest premises/tactics

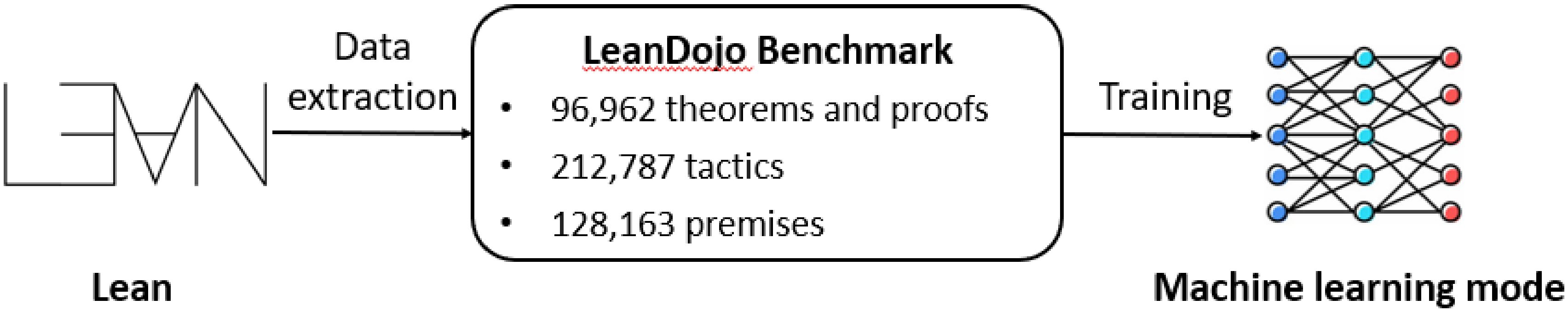


Extract enriched data as “learning materials”
Train a tactic & premise selector

LeanDojo [Yang et al. 2023]

LeanDojo: Theorem Proving with Retrieval-Augmented Language Models

Kaiyu Yang[1], Aidan M. Swope[2], Alex Gu[3], Rahul Chalamala[1], Peiyang Song[4], Shixing Yu[5], Saad Godil*, Ryan Prenger[2], Anima Anandkumar[1,2]
[1]Caltech, [2]NVIDIA, [3]MIT, [4] UC Santa Barbara, [5]UT Austin
`https://leandojo.org`



Caltech

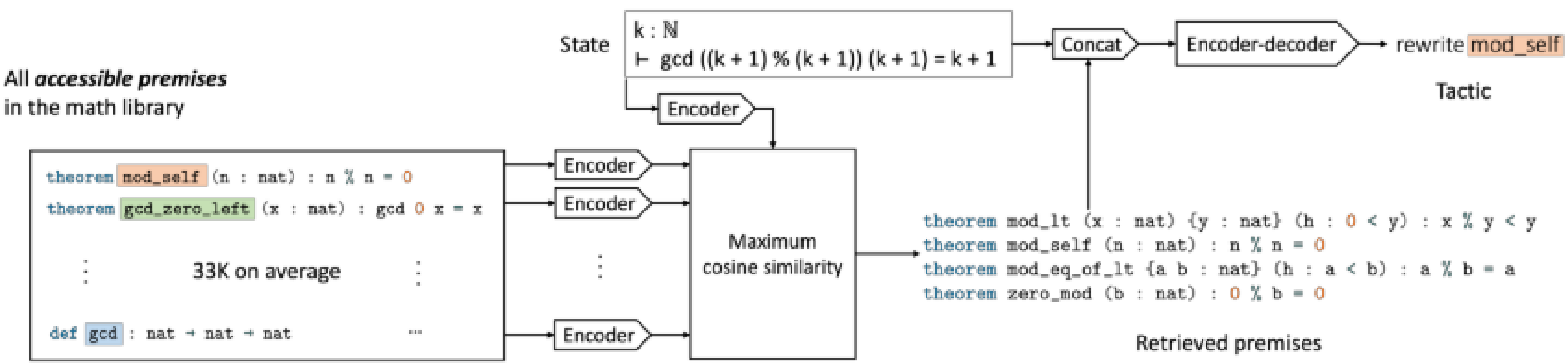
LeanDojo

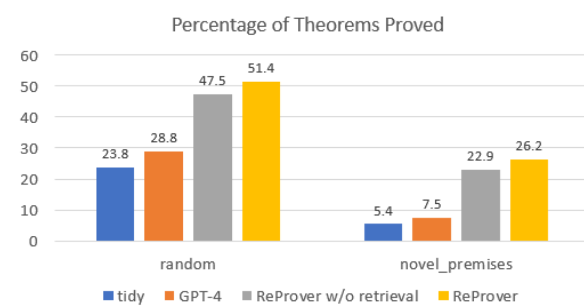
	Dataset available	Model available	Code available	Interaction tool available	Model size (# params)	Compute (hours)
Jiang et al., LISA, 2021	✓	✗	✗	✓	163M	-
Jiang et al., Thor, 2022	✓	✗	✗	✓	700M	1K on TPU
First et al., Baldur, 2023	✗	✗	✗	✓	62,000M	-
Polu and Sutskever, GPT-f, 2020	✗	✗	✗	✗	774M	40K on GPU
Han et al., PACT, 2022	✗	✗	✗	✓	837M	1.5K on GPU
Polu et al., 2023	✗	✗	✗	✓	774M	48K on GPU
Lample et al., HTPS 2022	✗	✗	✗	✗	600M	34K on GPU
Wang et al., DT-Solver, 2023	✓	✗	✗	✗	774M	1K on GPU
LeanDojo (ours)	✓	✓	✓	✓	**517M**	**120 on GPU**

[Yang et al., "LeanDojo: Theorem Proving with Retrieval-Augmented Language Models", NeurIPS 2023]

ReProver: Retrieval-Augmented Prover

- Given a state, we retrieve premises from the set of **all accessible premises**
- Retrieved premises are concatenated with the state and used for tactic generation



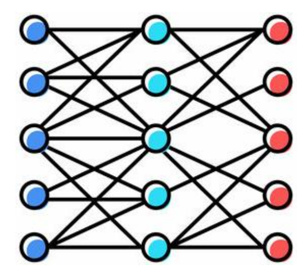


Neural Theorem Proving

Interactive Theorem Provers

Large Language Models

Run models locally or on a server
Automatically suggest premises/tactics/proofs



Extract enriched data as "learning materials"
Train a tactic & premise selector

Lean Copilot [Song et al. 2024]

Towards Large Language Models as Copilots for Theorem Proving in Lean

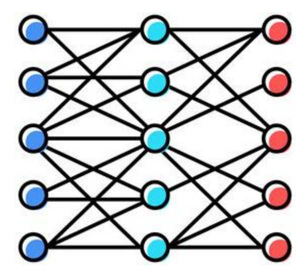
Peiyang Song[1]
UC Santa Barbara, U.S.A.
California Institute of Technology, U.S.A.

Kaiyu Yang
California Institute of Technology, U.S.A.

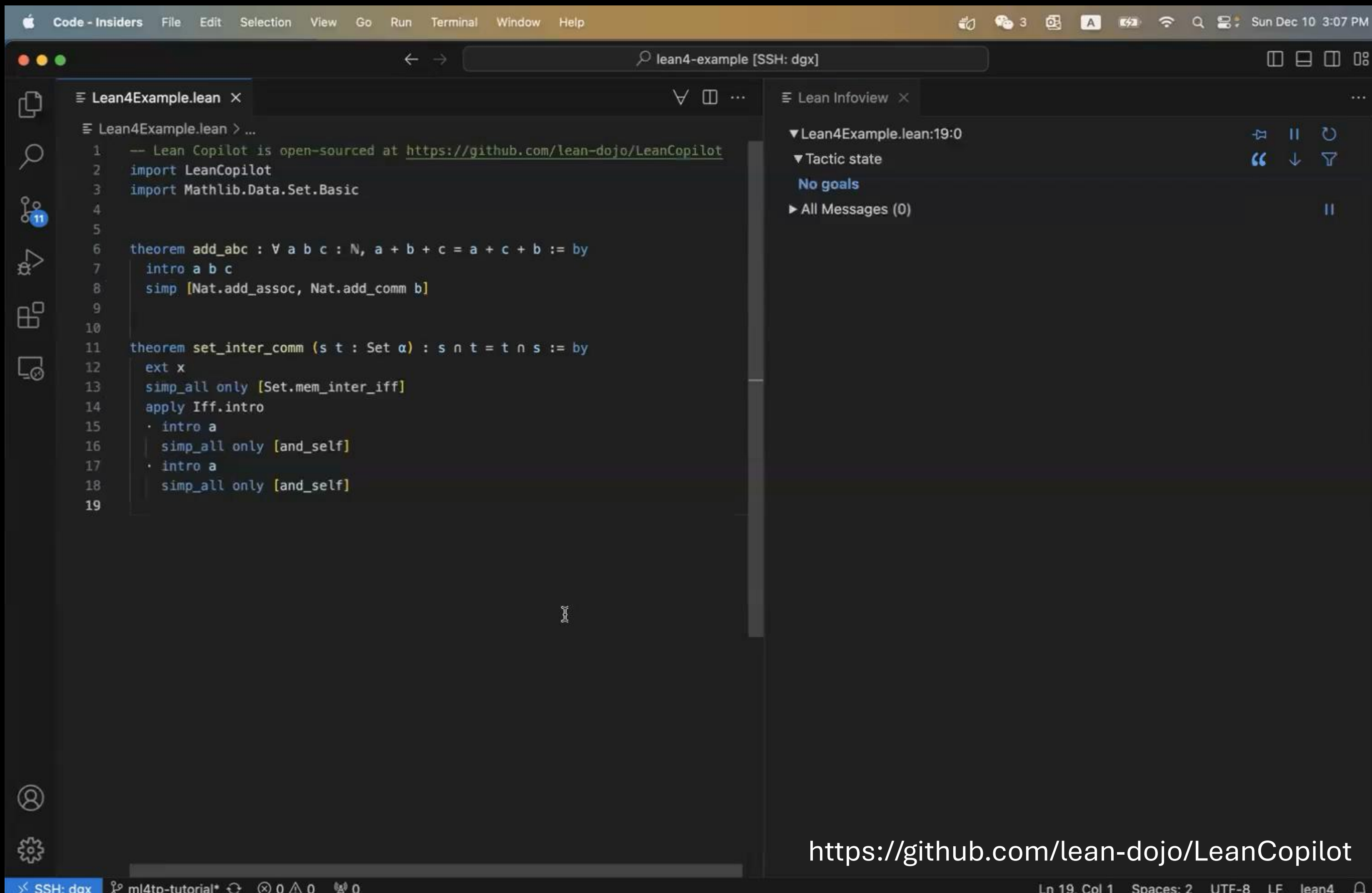
Anima Anandkumar
California Institute of Technology, U.S.A.

Run models locally or on a server

Automatically suggest premises/tactics



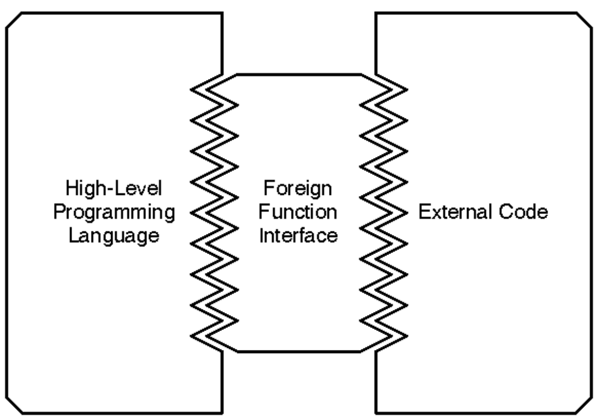
Lean Copilot



Also actual demo after this talk:)! If we have time

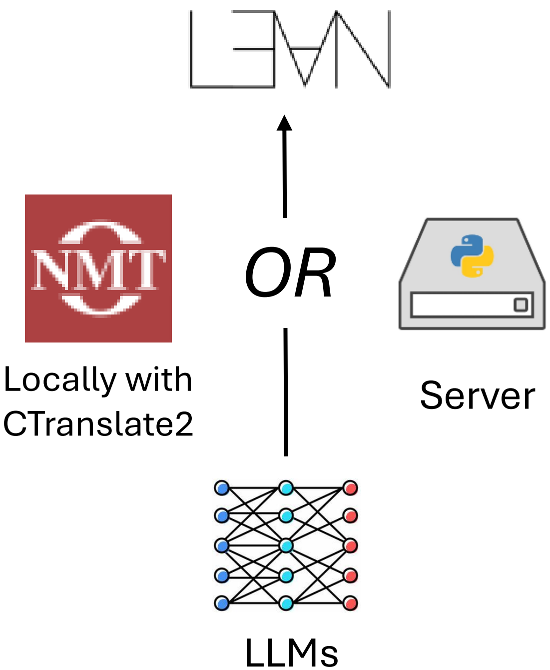
Lean Copilot [Song et al. 2024]

- Framework for neural network inference natively in Lean
  - **Fast** and **cheap** – accessible to all
  - Platform-independent (We have support for windows now natively!)
  - Always open to suggestions on how to improve it for mathematicians and end users.



Lean 4

C++



Tactic Suggestion

```
import LeanCopilot


theorem add_abc (a b c : Nat) : a + b + c = a + c + b := by
  suggest_tactics
```

▼ Tactic state

No goals

▼ Suggestions

```
Try these:

 • apply  Nat.add_right_comm
 • rw [Nat.add_assoc ]
   Remaining subgoals:
   ⊢ a + (b + c) = a + c + b
 • rw [Nat.add_comm ]
   Remaining subgoals:
   ⊢ c + (a + b) = a + c + b
 • simp [Nat.add_assoc ]
   Remaining subgoals:
   ⊢ a + (b + c) = a + (c + b)
```

Proof Search

```
import LeanCopilot

theorem add_abc (a b c : Nat) : a + b + c = c + b + a := by
  search_proof
```

▼ Tactic state

No goals

▼ Suggestions

```
Try this:
  simp [Nat.add_comm,
    Nat.add_left_comm]
```

- Generate a complete proof rather than the next tactic
- With autonomous check – correctness guaranteed w\ no hallucination!

Premise Selection

```
import LeanCopilot

theorem add_abc (a b c : Nat) : a + b + c = a + c + b := by
  select_premises
```

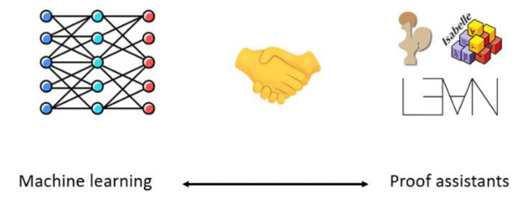
```
Nat.add_assoc : ∀ (n m k : Nat), n + m + k = n + (m + k)

Nat.add_comm : ∀ (n m : Nat), n + m = m + n

Nat.add_left_comm : ∀ (n m k : Nat), n + (m + k) = m + (n + k)

Nat.add_right_comm : ∀ (n m k : Nat), n + m + k = n + k + m
```

With rich annotations!

- In-scope premises: provide type information and doc strings
- Out-of-scope premises: provide complete definition + instruction on usage

Neuro-Symbolic Theorem Proving with Lean

- LeanDojo: Theorem Proving with Retrieval-Augmented Language Models
  - LeanDojo: Data Extraction & Interaction Tool for Theorem Proving in Lean
  - ReProver: Retrieval-Augmented Language Model as Theorem Prover



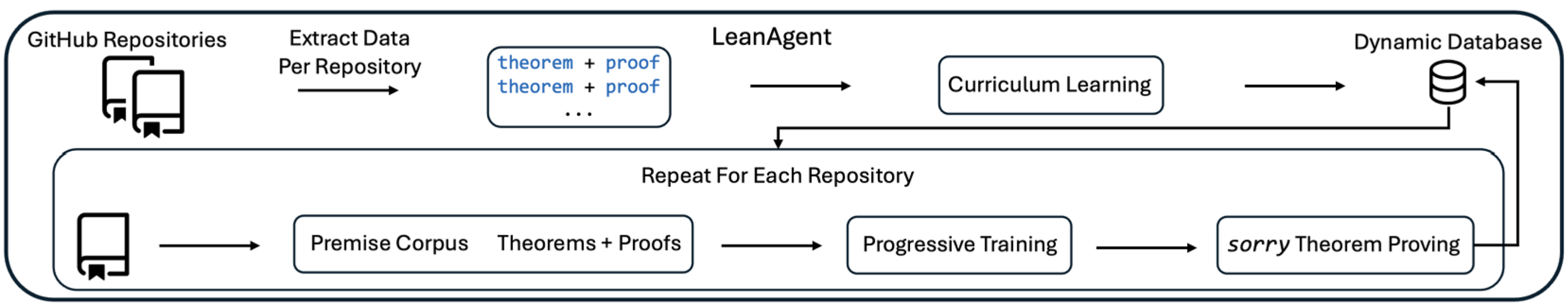
- Towards Large Language Models as Copilots for Theorem Proving in Lean
  - Lean Copilot: Native Machine Learning Toolkit in Lean
  - LLM-Powered Tools for Tactic Suggestion, Proof Search & Premise Selection

Lean Agent [Kumarappan et al. 2025]

Published as a conference paper at ICLR 2025

LEANAGENT: LIFELONG LEARNING FOR FORMAL THEOREM PROVING

Adarsh Kumarappan[*,1], **Mo Tiwari**[*,2], **Peiyang Song**[1], **Robert Joseph George**[1], **Chaowei Xiao**[3], **Anima Anandkumar**[1]
[1]California Institute of Technology, [2]Stanford University, [3]University of Wisconsin, Madison
{adarsh, psong, rgeorge, anima}@caltech.edu, motiwari@stanford.edu, cxiao34@wisc.edu



Problem and Motivation

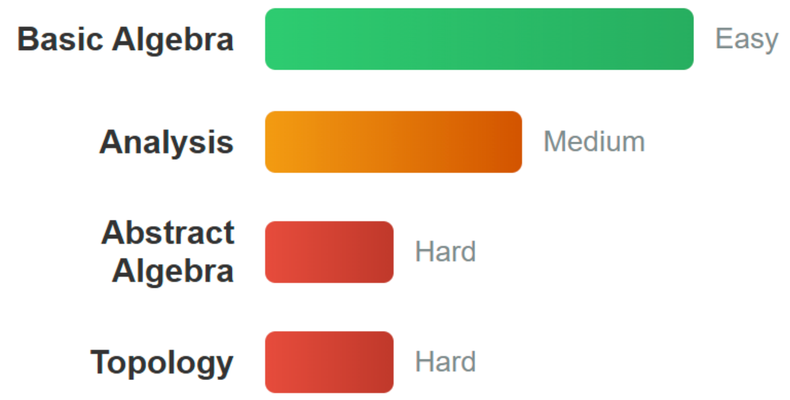
The Problem

→ Existing LLM-based theorem provers are trained on specific datasets and struggle to generalize to advanced mathematics

→ Data scarcity in formal theorem proving limits model capabilities

→ Current approaches operate on static domains, failing to mimic how mathematicians work across multiple domains simultaneously

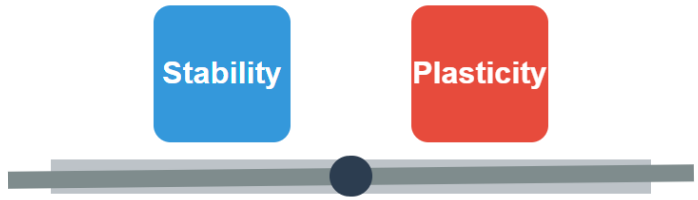
→ Need for a system that can continuously adapt and improve across diverse mathematical domains without forgetting



Mathematicians work across a spectrum of difficulty, but AI systems often fail to generalize to harder domains

The Stability-Plasticity Dilemma

The Core Challenge in Lifelong Theorem Proving



No Previous Lifelong Learning Frameworks

Prior to LeanAgent, no framework existed for continuous learning across mathematical domains without forgetting. Traditional lifelong learning methods struggle with the unique challenges of theorem proving.

Too Much Stability

Static models fail to adapt to new mathematical domains and complex theorems

Balancing Act

Finding the right balance between retaining knowledge and learning new concepts

Too Much Plasticity

Models forget previously learned mathematical techniques (catastrophic forgetting)

Why This Matters: Mathematicians work across multiple domains simultaneously, building on foundational knowledge while exploring advanced concepts

LeanAgent: Key Components

Curriculum Learning Strategy

Measures theorem complexity as e^S (where S = proof steps) and sorts repositories by easy theorem count. This optimizes the learning trajectory, allowing LeanAgent to build foundational knowledge before tackling advanced concepts.

Dynamic Database

Manages evolving mathematical knowledge efficiently. Stores repository metadata, theorems, premises, and traced tactics to track progress and enable knowledge transfer between domains.

Progressive Training

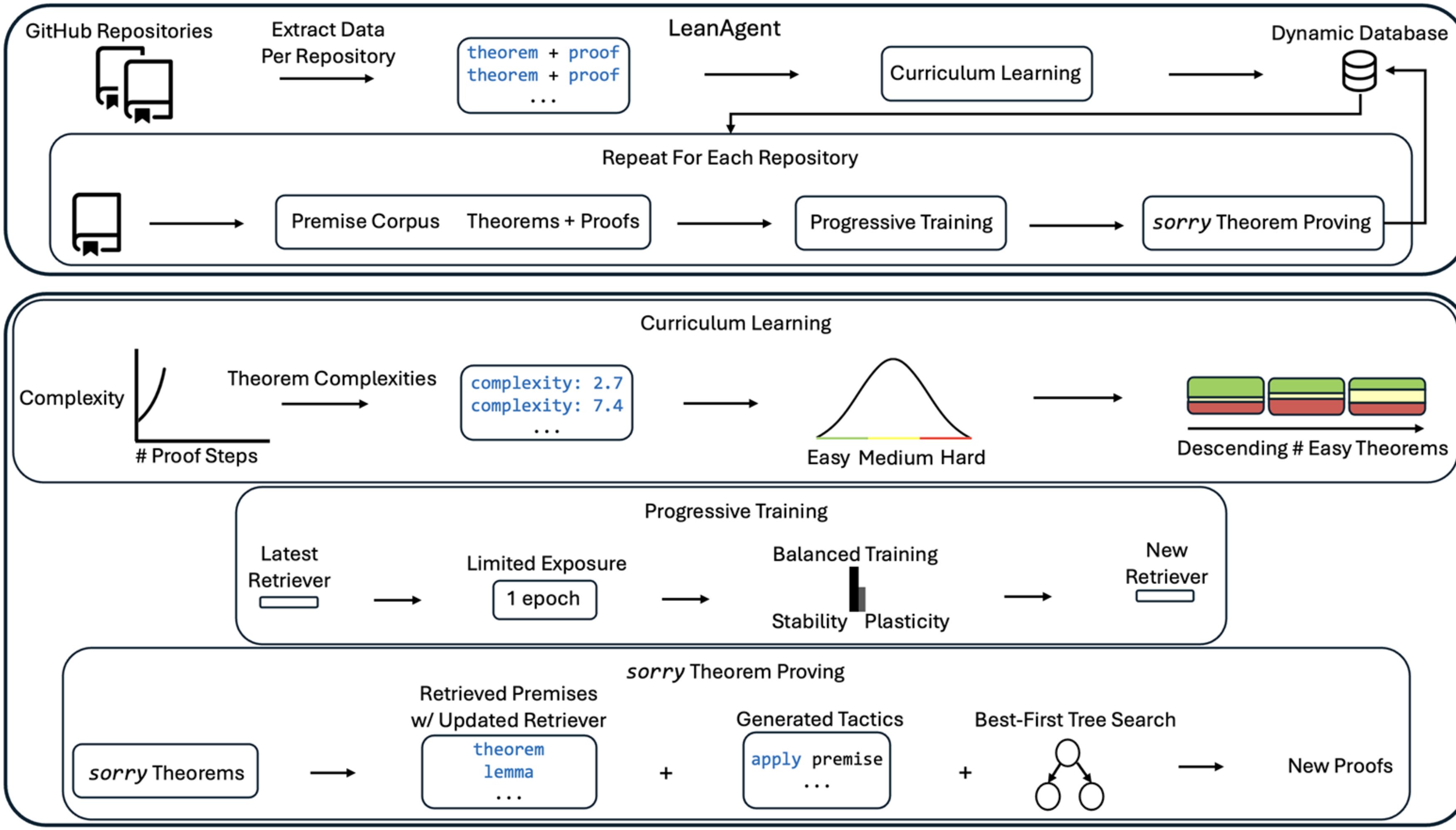
Balances stability (retaining previous knowledge) and plasticity (acquiring new skills). Incrementally trains the retriever on new datasets for one epoch per repository, preserving previously learned information while adapting to new domains.

Best-First Tree Search

Generates formal proofs for theorems where proofs were previously missing (sorry theorems). Uses a search-based method with an updated retriever to retrieve relevant premises, generate tactic candidates, and find proof paths.

These components work together to enable lifelong learning in theorem proving, allowing continuous adaptation across mathematical domains without forgetting.

Framework



LeanAgent
GitHub Repositories
Extract Data Per Repository
theorem + proof
theorem + proof
...
Curriculum Learning
Dynamic Database
Repeat For Each Repository
Premise Corpus
Theorems + Proofs
Progressive Training
sorry Theorem Proving
Curriculum Learning
Complexity
Proof Steps
Theorem Complexities
complexity: 2.7
complexity: 7.4
...
Easy
Medium
Hard
Descending # Easy Theorems
Progressive Training
Latest Retriever
Limited Exposure
1 epoch
Balanced Training
Stability
Plasticity
New Retriever
sorry Theorem Proving
sorry Theorems
Retrieved Premises w/ Updated Retriever
theorem
lemma
...
+
Generated Tactics
apply premise
...
+
Best-First Tree Search
New Proofs

Progressive Training in LeanAgent

A Simple but Effective Approach

1 — Start

Begin with pre-trained retriever based on mathlib4

2 — Train

Train on new repository for exactly one epoch

3 — Prove

Use updated retriever to prove sorry theorems

4 — Repeat

Add new proofs to database and continue to next repository

Why One Epoch?

Limited exposure prevents overfitting while allowing essential learning of new concepts

Avoiding EWC

Complex regularizers fail as theorem complexity increases across repositories

Continuous Learning

Adds new premises and expands the space of possible proof states incrementally

Knowledge Transfer

Enables backward transfer where advanced concepts improve basic theorem proving

LeanAgent: Key Results

155
New Theorems Proven
Across 23 diverse repositories

75%
Lower Forgetting
Reduced catastrophic forgetting

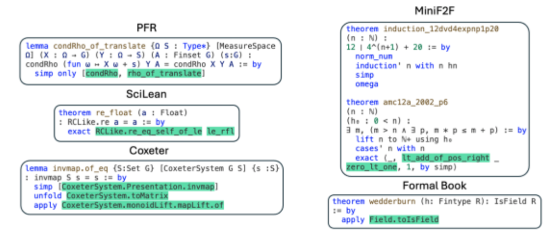
16%
Better Transfer
Improved backward transfer

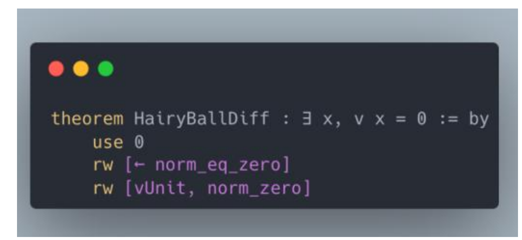
Number of Sorry Theorems Proven

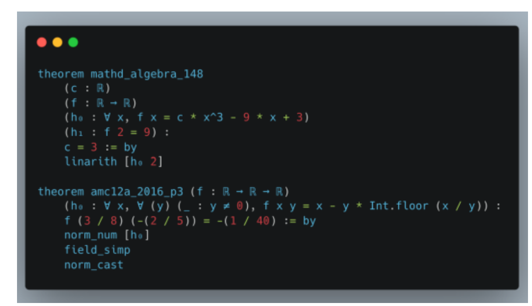
Total Sorry Theorems Across All Repositories: 878

Repositories Where Theorems Were Proven

Repository	Sorry Theorems	Proven by LeanAgent	% Success
Mathematics in Lean	29	21	72.4%
MiniF2F	406	99	24.4%
Formal Book	29	3	10.3%
SciLean	294	27	9.2%
Hairy Ball Theorem	14	1	7.1%
Coxeter	15	1	6.7%
Other Repositories (17)	91	3	3.3%







LeanAgent: Open-Source Framework & Pipeline

Streamlined Workflow for Automated Theorem Proving

1. **Repository Discovery & Extraction**
   Automatically searches for and clones Lean repositories from GitHub. Extracts theorems, proofs, and identifies "sorry" theorems using LeanDojo.

2. **Curriculum Construction**
   Builds a learning curriculum by calculating theorem complexities and sorting repositories by easy theorem count.

3. **Progressive Training & Proof Generation**
   Incrementally trains on each repository and uses best-first tree search to prove sorry theorems.

4. **Automatic Pull Request Generation**
   Integrates newly generated proofs into original Lean files and automatically creates pull requests to repository owners.

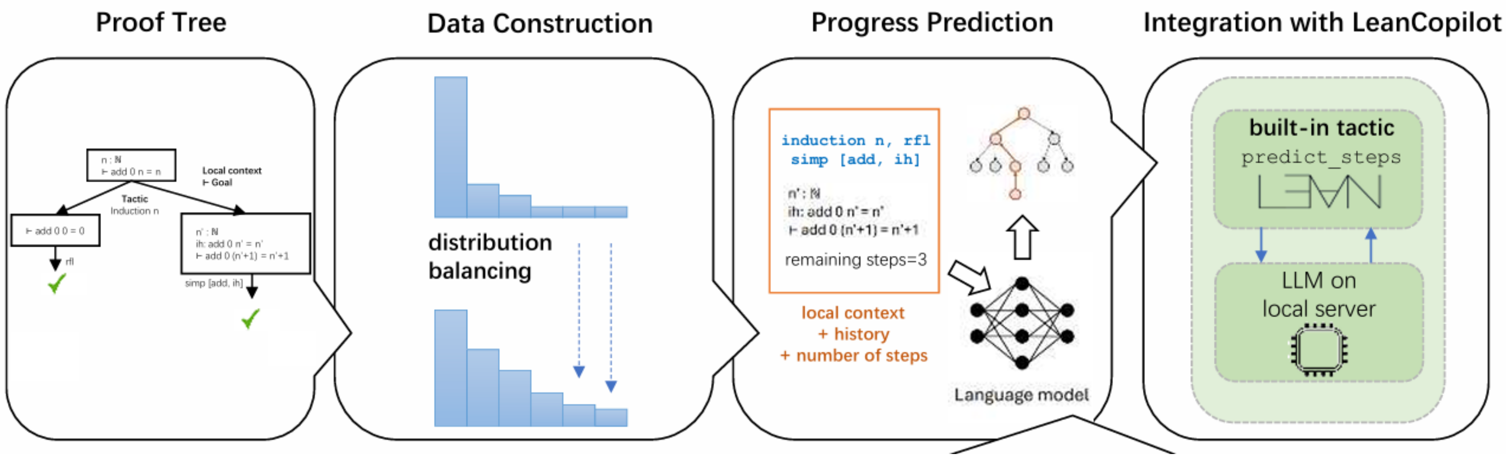
- **Fully Automated Pipeline** from repository discovery to pull request generation
- **Local Repository Support** for working with private mathematical libraries
- **Modular Design** allows using specific components (curriculum builder, prover) independently
- **Integrated Analytics** to track proof progress and identify challenging theorems
- **Pluggable Architecture** supports any LLM as the proving engine
- **Dynamic Database** for managing evolving mathematical knowledge

Available at: github.com/lean-dojo/LeanAgent

Lean Progress [Huang et al. 2025]

LeanProgress: Guiding Search for Neural Theorem Proving via Proof Progress Prediction

Suozhi Huang [1] **Peiyang Song** [2] **Robert Joseph George** [2] **Anima Anandkumar** [2]



Problem and Motivation

Research Challenge

Formal theorem proving presents a critical challenge in computational mathematics and artificial intelligence. **Large Language Models (LLMs)** struggle to generate reliable mathematical proofs due to inherent limitations:

- Persistent hallucinations in mathematical reasoning
- Inability to navigate complex proof spaces effectively
- Limited understanding of global proof trajectory
- Difficulty in generating verifiable proof steps

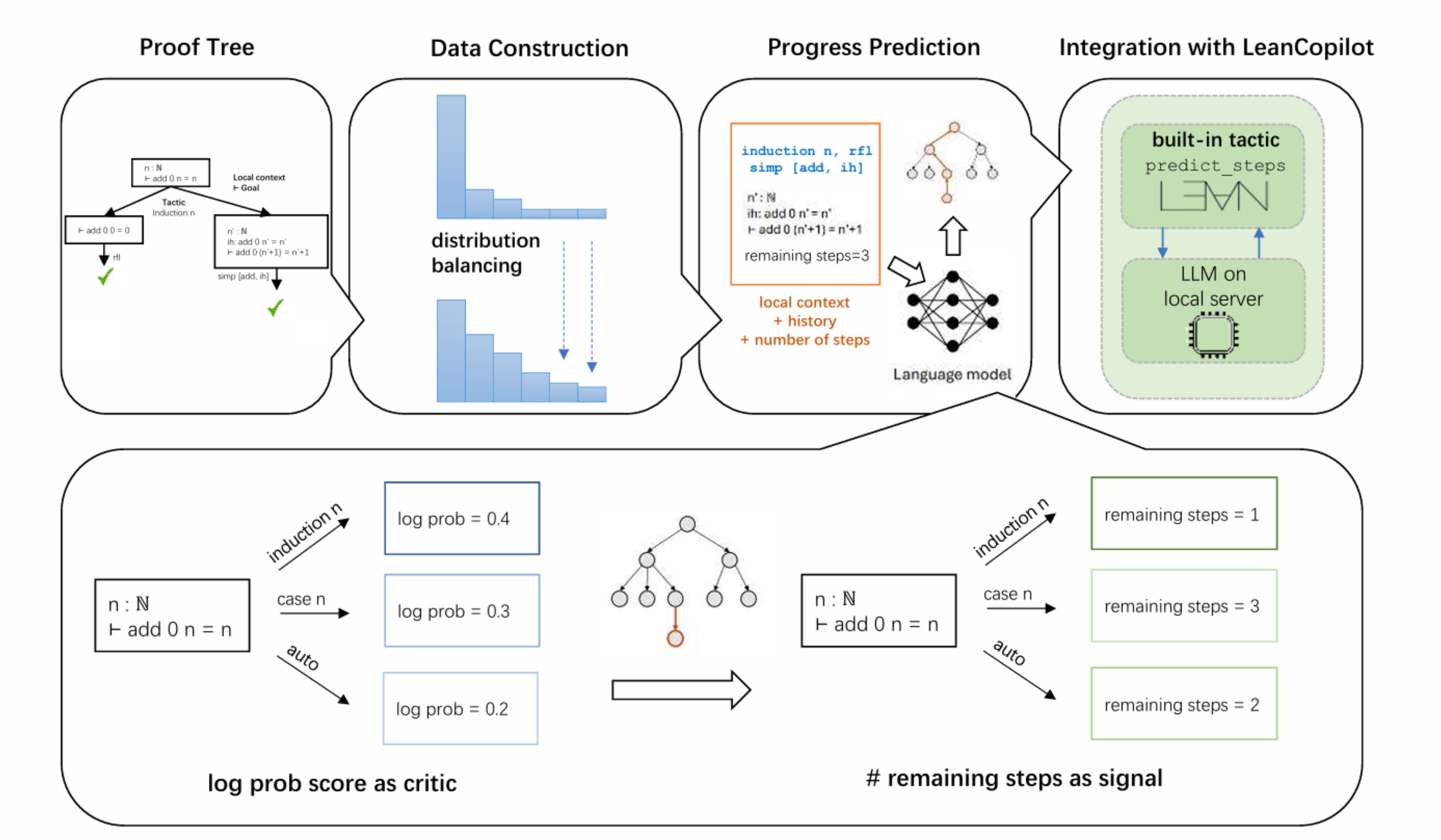
Research Motivation

The proliferation of large-scale formalization projects, such as those in the Lean mathematical library, underscores the urgent need for intelligent theorem proving assistance. Our research aims to bridge the critical gap between:

Local Tactic Prediction ↔ Global Proof Understanding

- Develop a method to predict proof progress accurately
- Provide actionable insights during theorem proving
- Enhance the efficiency of formal mathematical reasoning

Pipeline


Proof Tree
Data Construction
Progress Prediction
Integration with LeanCopilot
Local context
⊢ Goal
Tactic
Induction n
distribution
balancing
induction n, rfl
simp [add, ih]
remaining steps=3
local context
+ history
+ number of steps
Language model
built-in tactic
predict_steps
LLM on
local server
n : ℕ
⊢ add 0 n = n
induction n
case n
auto
log prob = 0.4
log prob = 0.3
log prob = 0.2
remaining steps = 1
remaining steps = 3
remaining steps = 2
log prob score as critic
remaining steps as signal

Key Points

Pipeline

- Generate proof trees using best-first search.
- Collect proof trajectories from Lean Workbook Plus and Mathlib4.
- Assign sampling ratios to proof length ranges.
- Fine-tune DeepSeek Coder 1.3B model
- Predict remaining proof steps

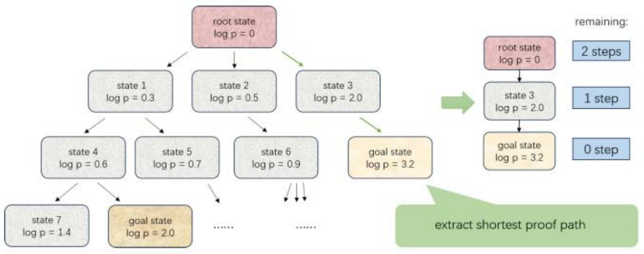
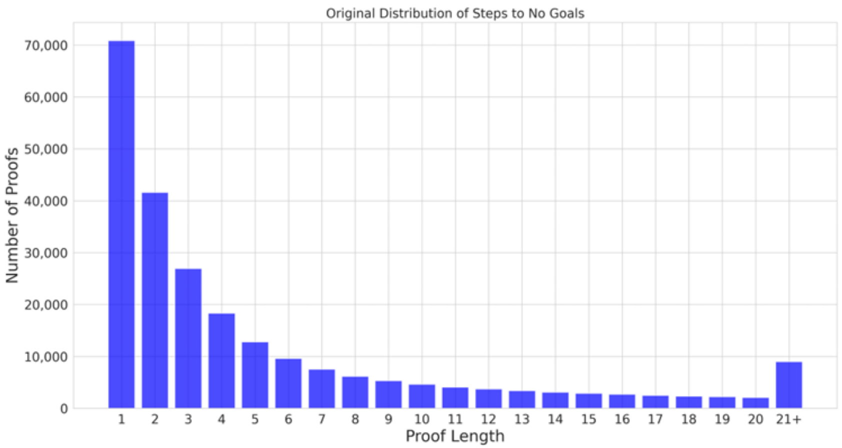
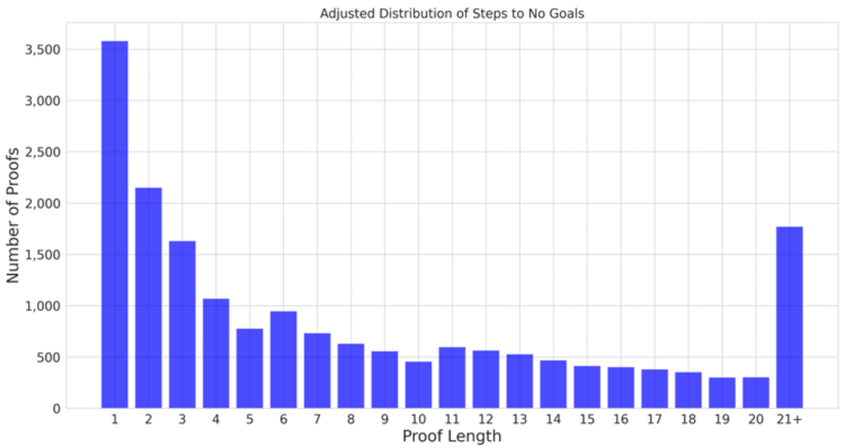


Figure 2. **The visualization of extract proof tree** in theorem proving.



(a) Original dataset of steps distribution



(b) Adjusted dataset of steps distribution by balancing different ranges

Results and Conclusion

Step Prediction Accuracy

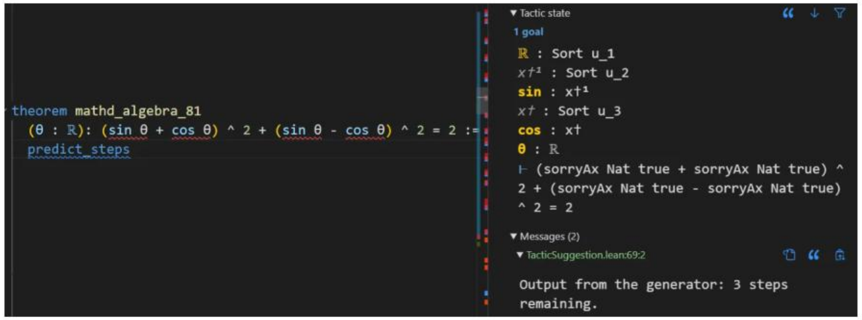
- Overall Prediction Accuracy: 75.1%
- Mean Absolute Error (MAE): 3.29
- Performance across proof length ranges:
  - 1-5 steps: 79.0% accuracy
  - 6-10 steps: 61.5% accuracy
  - 11-15 steps: 68.3% accuracy
  - 16-20 steps: 77.1% accuracy
  - 21+ steps: 76.7% accuracy

Proof Search Performance

- Baseline Reprover Performance: 41.2%
- LeanProgress Enhanced Performance: 45.0%
- Improvement on Mathlib4 test dataset: 3.8%
- Key Enhancement:
  - More effective for longer, complex proofs
  - Provides global proof progress view
  - Guides search beyond log-probability

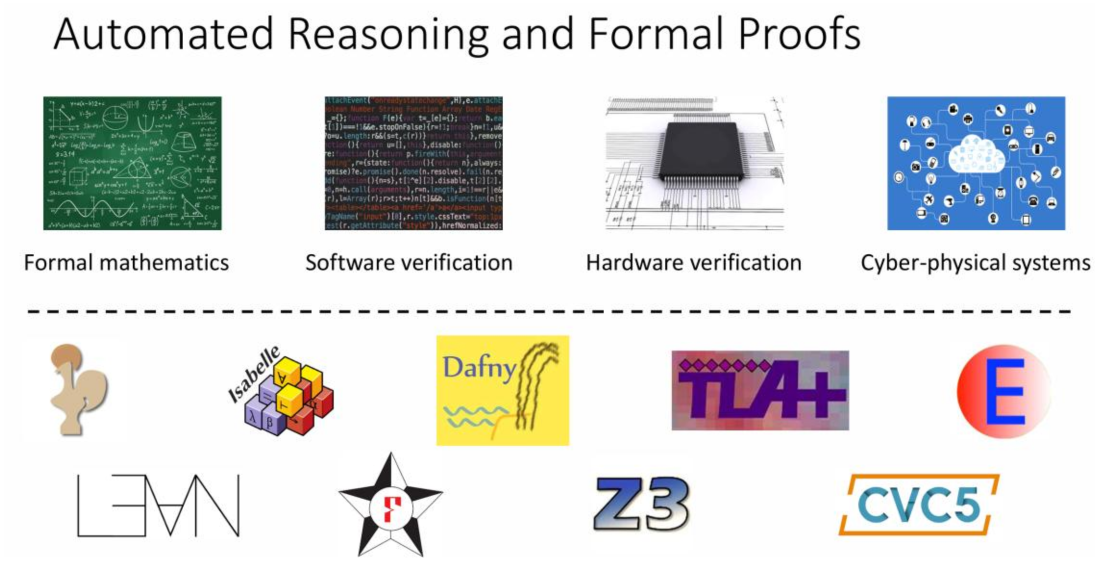
Bridges the gap between local tactic prediction and global proof trajectory understanding

Opens new possibilities for reinforcement learning in automated theorem proving



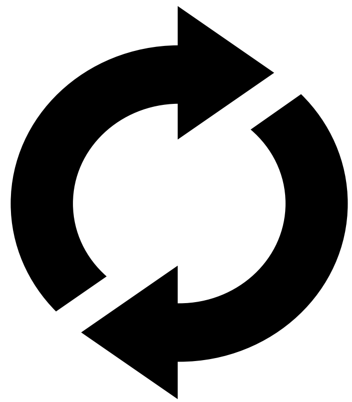
Future works and Open Questions

- Incorporate more ML techniques into the neural theorem proving pipeline
- Generalize to other advanced mathematics, or even scientific domains
- Explore other challenges in AI for mathematics
    - Adaptability across domains
    - Generalization and self-improvement
    - ... and much more!



How is Engineering and Scientific Research done Today?

Human Intuition

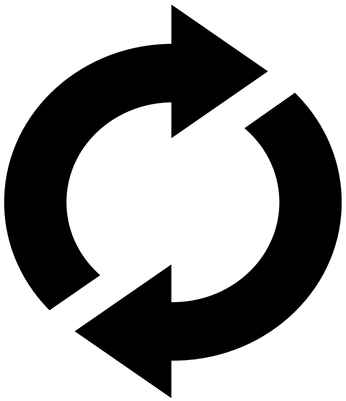


Bottleneck: simulation/physical experiments

(weeks - months)

The future of Engineering and Scientific Research

Human Intuition + AI

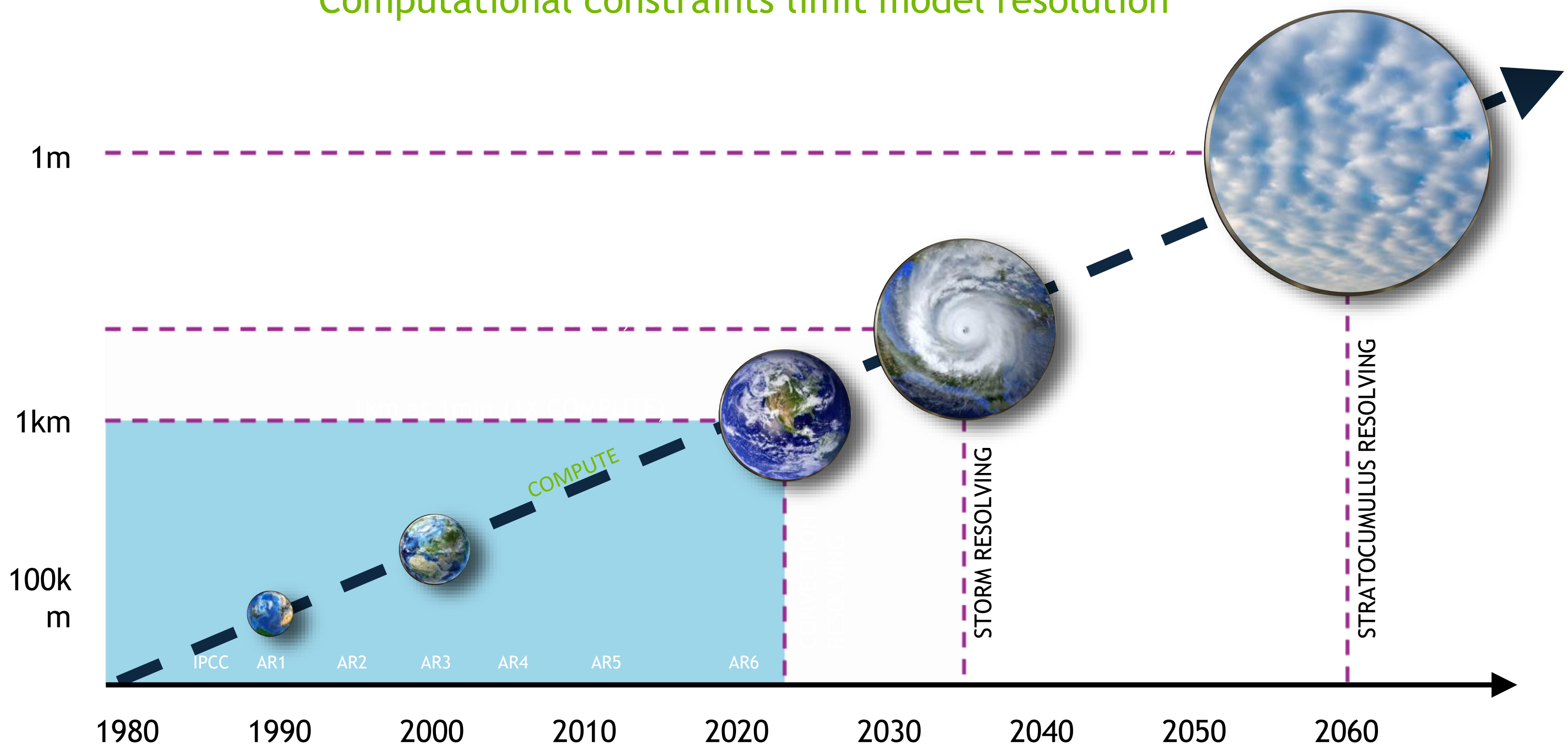


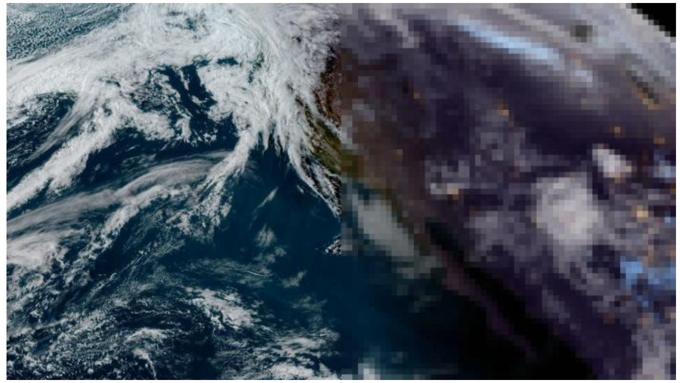
AI simulates, designs and experiments + Verification with Theorem Provers.

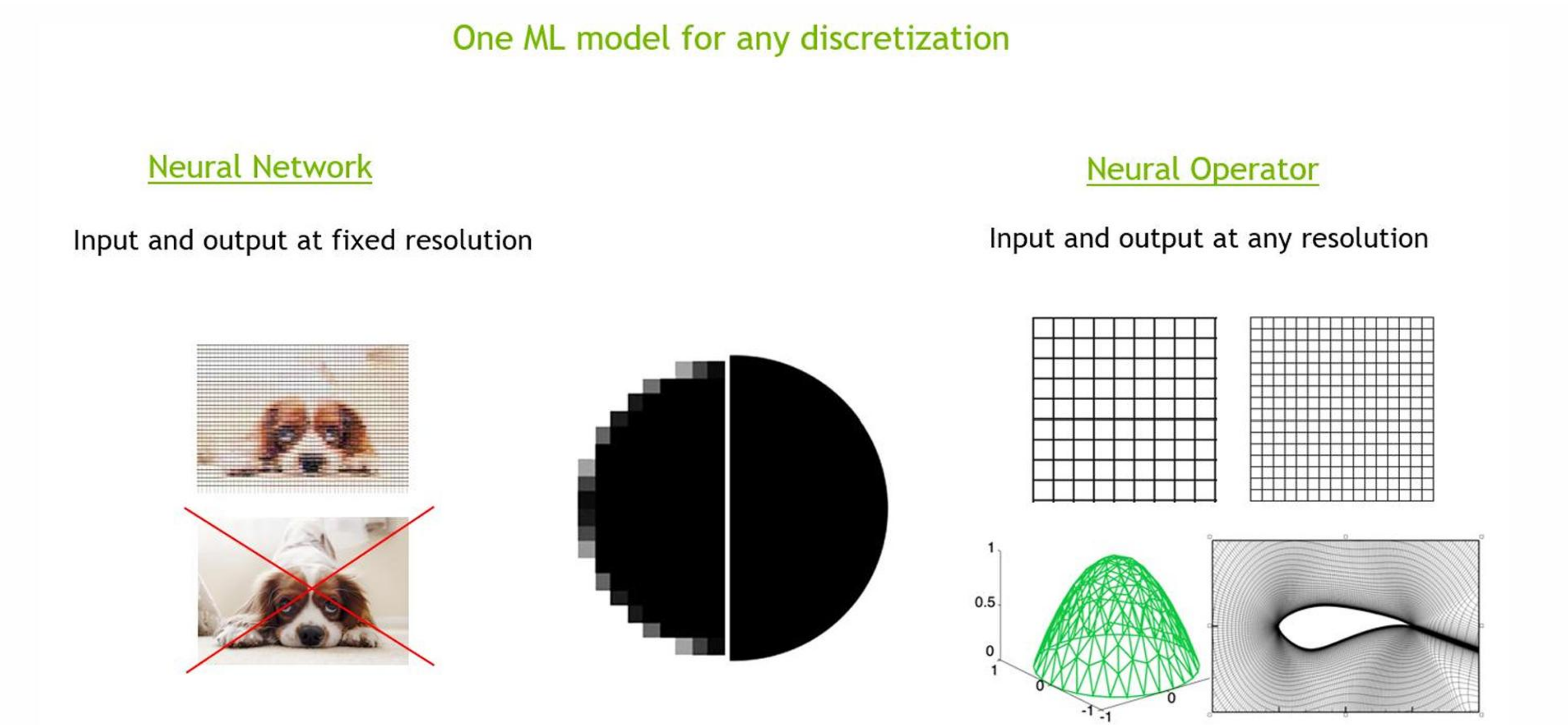
Lightning speed!

Example: Climate Modeling requires fine-scale modeling

Computational constraints limit model resolution

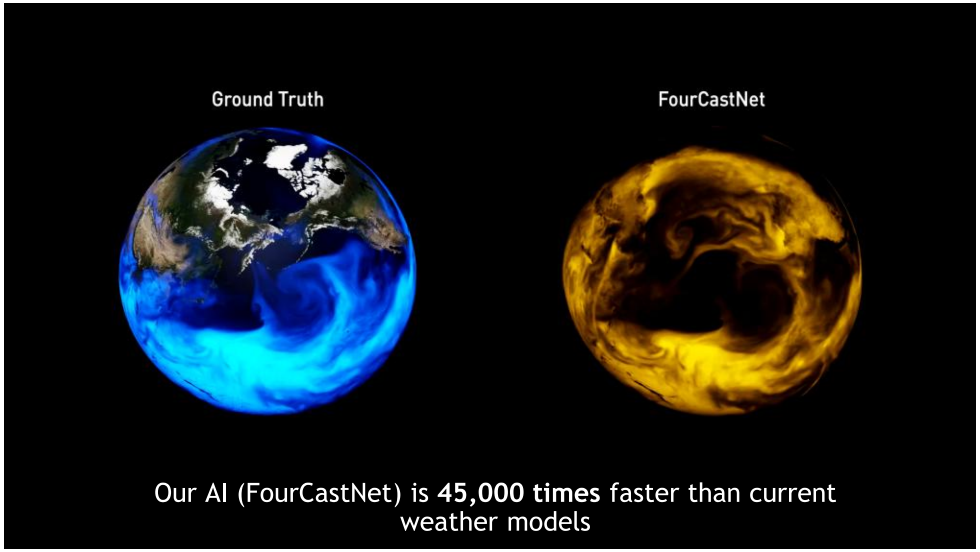


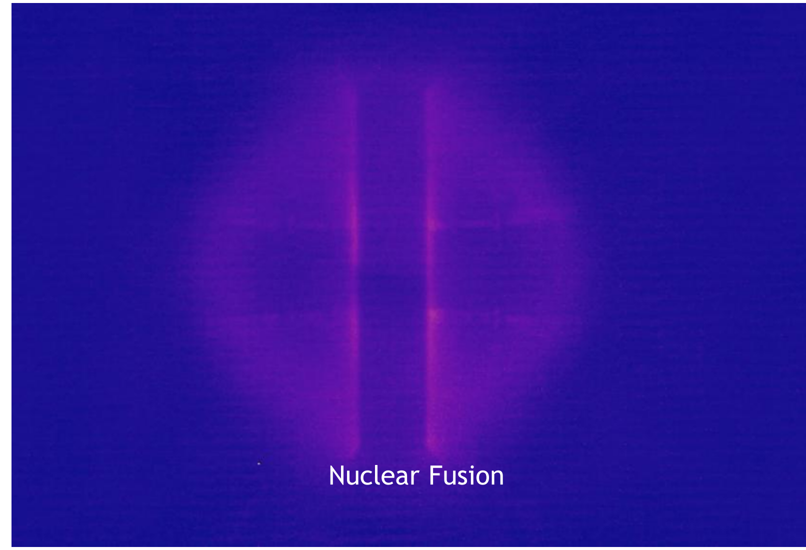




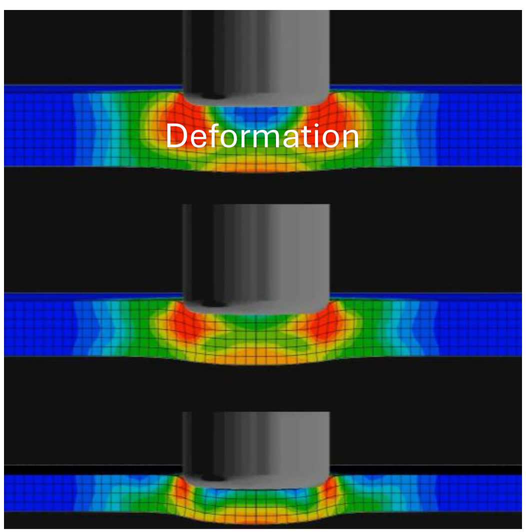
Discretization Agnostic Learning

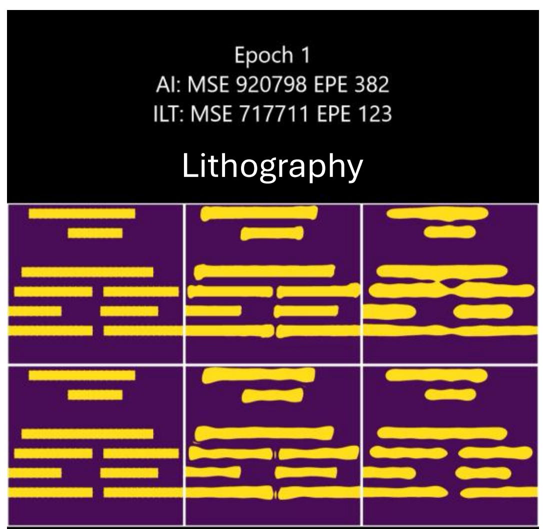
AI4SCIENCE: BIG APPLICATIONS + VERIFICATION = FUTURE

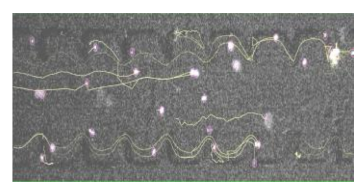


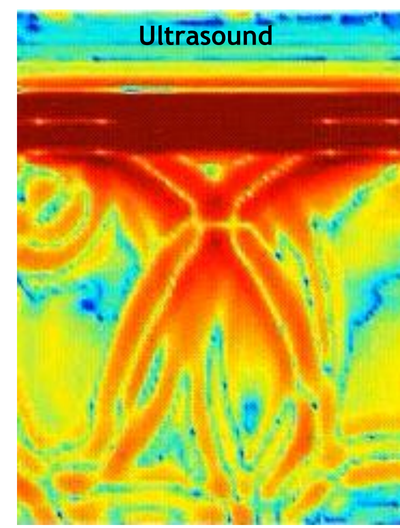


Our AI (FNO) is **differentiable** and can do **inverse design**









Acknowledgments

Thank you to the Organizers of the ICERM workshop for giving me a chance to speak here on our work.

All this work is mostly done by the team here at Caltech and Nvidia (For most of the AI4Science Projects) under Anima Anandkumar. For the AI4Math projects, special thanks to all the students in our lab who have been involved in it and led some of them!

I thank the rest of the contributors and the other members of the teams for their feedback and help on the projects.

Thank you!