Enhancing LLM's Mathematical and Theorem Proving Abilities

Naga Rohan Kumar Bayya

Agenda

01. **The Problem: LLMs vs. Formal Reasoning**

02. **Project Goal & Objectives**

03. **Background: Key Concepts**

04. **Preliminary Work**

05. **Main Experiments & Results**

06. **Infrastructure & Demo**

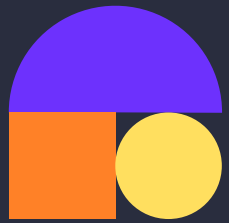
The Problem: LLMs vs. Formal Reasoning

The Problem:
Formal Reasoning

- Large Language Models (LLMs) are trained on linguistic patterns, not formal reasoning.
- They produce plausible-sounding but often incorrect answers for math and logic.
- Struggle with multi-step deductions and symbolic manipulation.
- Lack the ability to verify the correctness of their own steps.

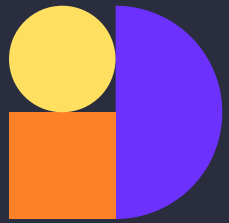
Project Goal & Objectives

To bridge the gap between language models and formal mathematical reasoning.



Primary Focus

- Enhance LLM capabilities in formal theorem proving, a comparatively under-studied domain.



Preliminary Objectives

- Improve mathematical problem-solving through specialized fine-tuning.
- Integrate symbolic tools (Mathics, Lean) to ensure correctness.

Experiments

- Investigate advanced architectures (MoE) and training methods (OBT, GRPO, SFT, etc).

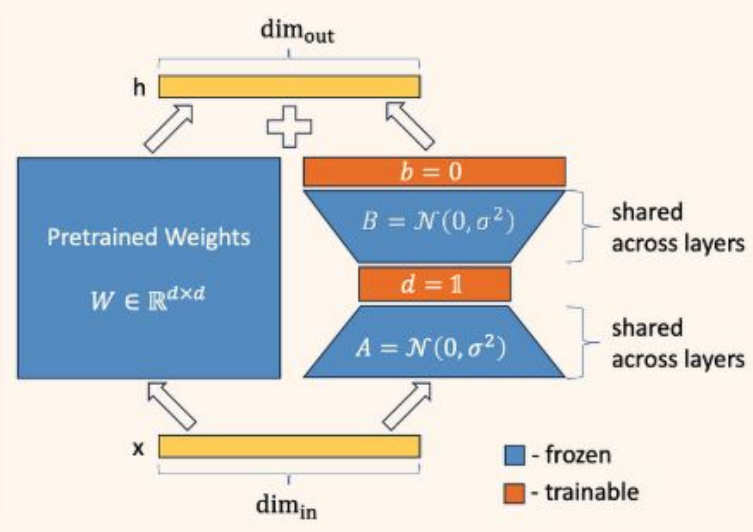
Background: Key Concepts

Parameter-Efficient Fine-Tuning (PEFT)

- Fine-tune models efficiently without retraining all parameters.
  - LoRA (Low-Rank Adaptation): Freezes original weights and trains small "adapter" matrices. Drastically reduces memory and compute.
  - DoRA (Decomposed Low-Rank Adaptation): An improvement upon LoRA to achieve better accuracy..

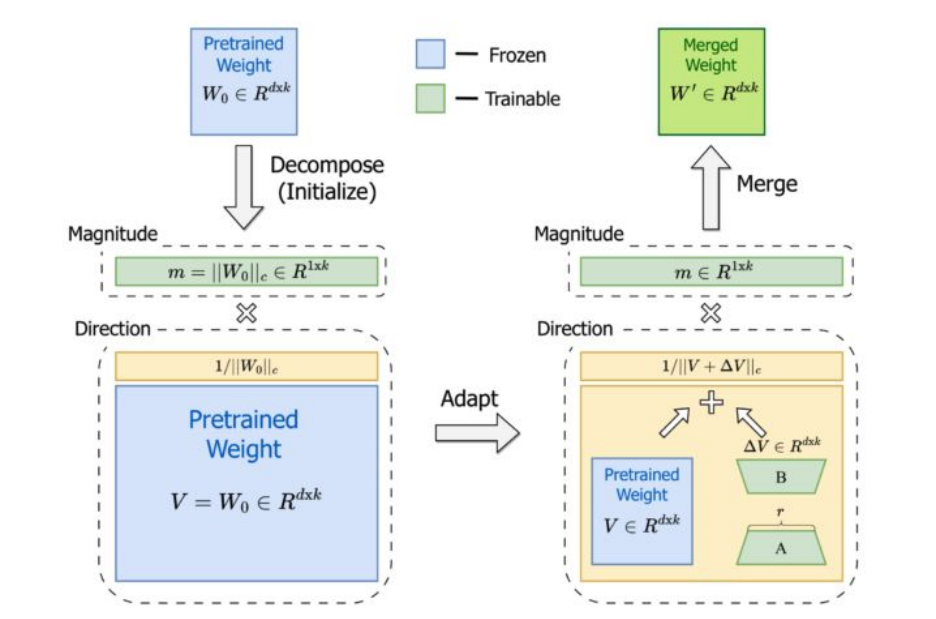
LoRA

- Freezes the original, large pre-trained weights (W)
- Injects two small, trainable "low-rank" matrices (A and B)
- W_new = W + B.A



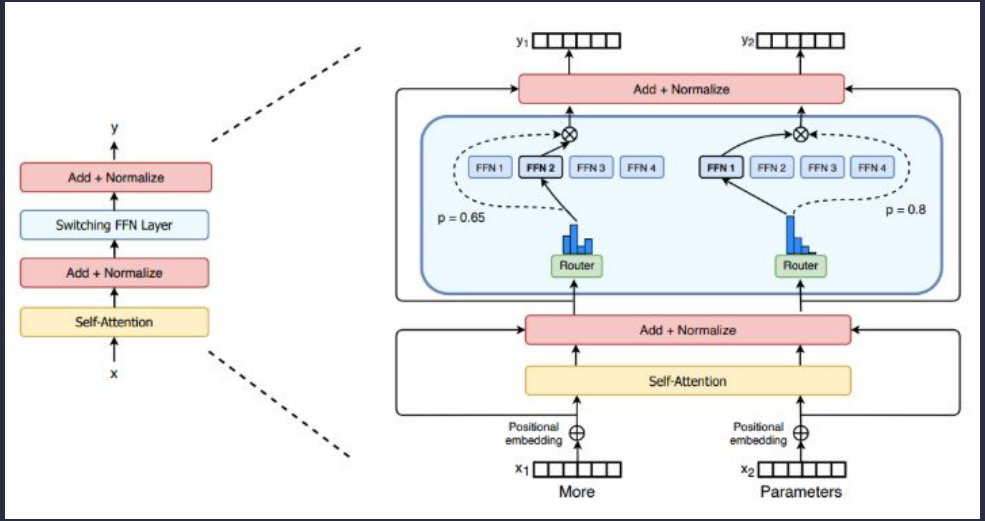
DoRA

- Achieves performance closer to full fine-tuning
- Splits original weight (W) in 2 parts: magnitude(m) and direction (D)
- LoRA is applied only to D

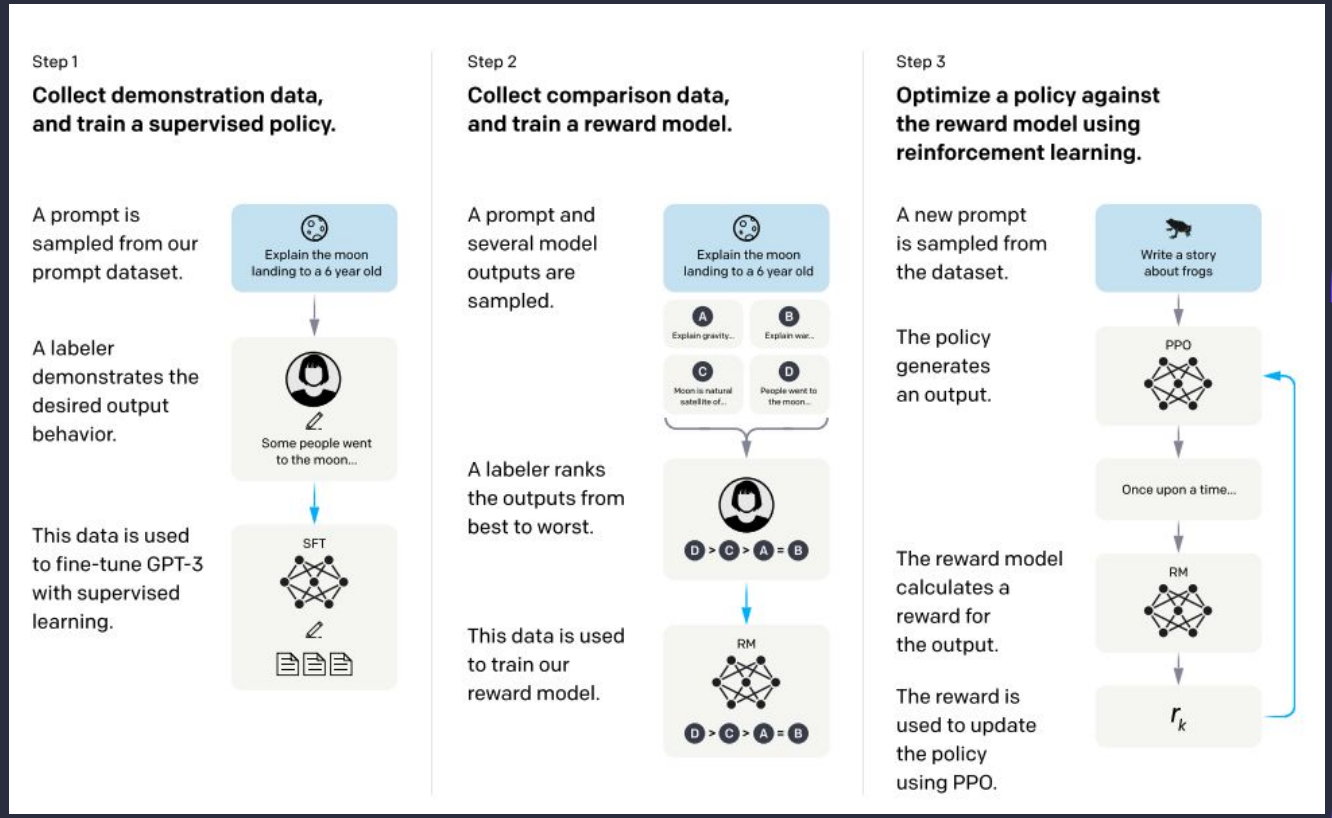


Mixture-of-Experts (MoE) Architecture

- Analogy: A team of specialists, not a single generalist.
- A "gating network" routes each input token to a few specialized "expert" subnetworks.
- Allows for a massive number of total parameters with low computational cost.
- Theoretically more resistant to catastrophic forgetting.

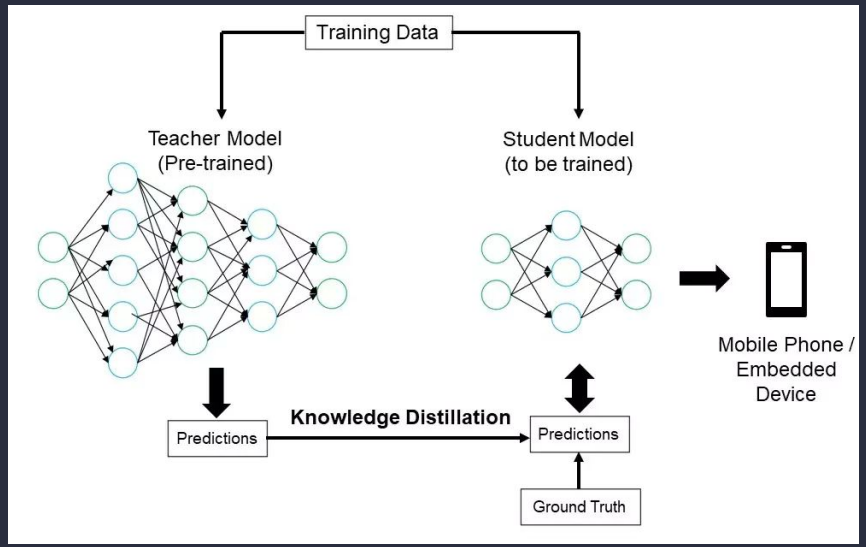


Post-Training Techniques



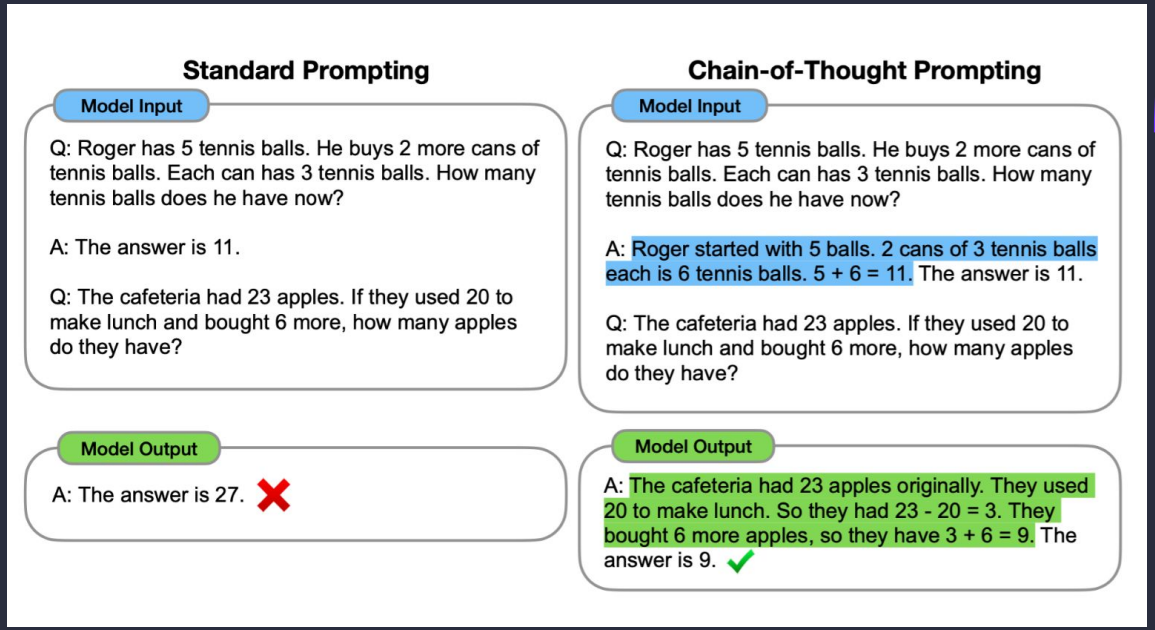
Model Distillation

- Student-teacher technique for transferring knowledge to smaller model
- Working
  - Teacher Model: A large, highly capable (often proprietary) LLM
  - Teacher generates a massive, high-quality dataset with CoT
  - Student is fine-tued on dataset
- Efficient, cost effective, and fast



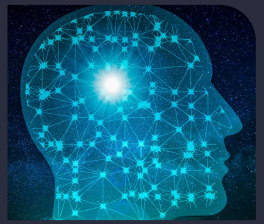
Chain-of-Thought (CoT) Prompting

- Break down complex problems into intermediate steps
- Improved reasoning and transparency



Tool-Augmented Reasoning

An LLM acts as an orchestrator, offloading tasks to reliable external tools.

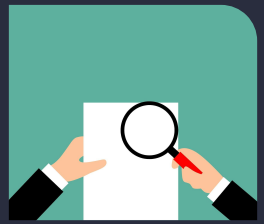


LLM (Brain)

Parses the problem

Mathics (Calculator)

For precise symbolic math



Lean (Verifier)

For ensuring logical correctness of proofs.

Mathics Tool

```python
@tool
def perform_math(expression: str) -> int:
  """ Use this tool when you need to calculate a complex mathematical expression.
  from mathics.session import MathicsSession
  session = MathicsSession(add_builtin=True, catch_interrupt=True)
  result = session.evaluate(expression).to_python()
  return result
```

```python
tools = [perform_math]
augmented_agent = create_tool_calling_agent(
    llm=model,
    tools=tools,
    prompt=prompt
)

agent_executor = AgentExecutor(agent=augmented_agent, tools=tools, verbose=True)
result = agent_executor.invoke({
    "input": ("Integrate sin^2(x) from 0 to pi/2 ")
})
```

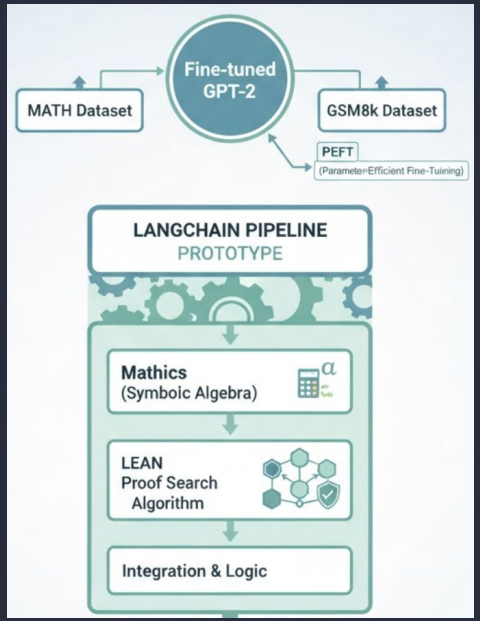
```
> Entering new AgentExecutor chain...

Invoking: `perform_math` with `{'expression': 'Integrate[Sin[x]^2, {x, 0, Pi/2}]'}`

System`Times[1/4, System`Pi] The integral of sin(x) from 0 to pi/2 is equal to 1/4 * pi.

> Finished chain.
```

Preliminary Work

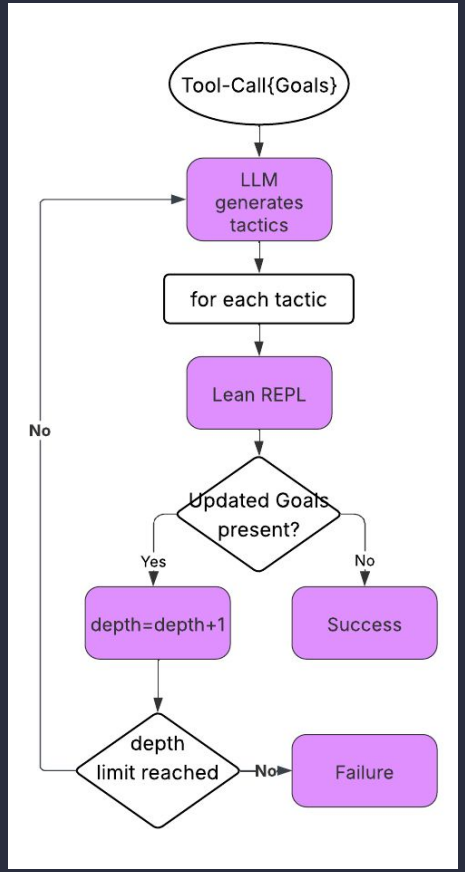
Small-Scale Validation

Validate core hypotheses using a small model

- Fine-tuned GPT-2 on MATH and GSM8k datasets using PEFT
- Implemented LEAN Proof search algorithm
- Prototyped a LangChain pipeline to integrate Mathics and Lean proof search



Proof Search



Key Findings & Learnings

Promising Results, Clear Limitations

- PEFT successfully improved mathematical formatting and basic reasoning. ✅
- LLMs show learning capabilities when finetuned on different modalities ✅
- Basic tool integration was not robust enough/feasible for conversational use. ❌
- GPT-2's small size was a major bottleneck for complex problems. ❌

Transition to Main Experiments

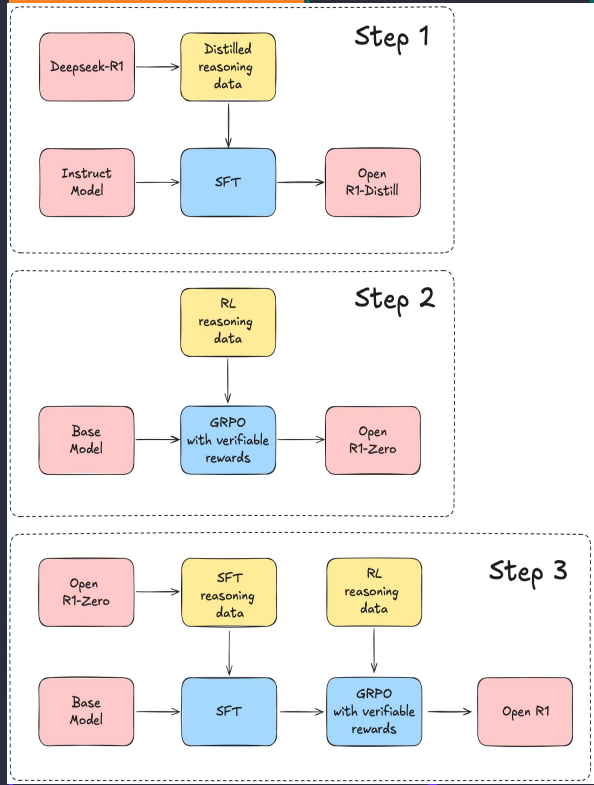
- Move from GPT-2 to more capable models like Llama-3.2-3B and Phi-tiny-MoE
- Focus on theorem proving domain
- Implement more sophisticated training pipelines inspired by TheoremLlama and open-r1.

Experiment-1

Open-r1 + data supervision

SFT and GRPO

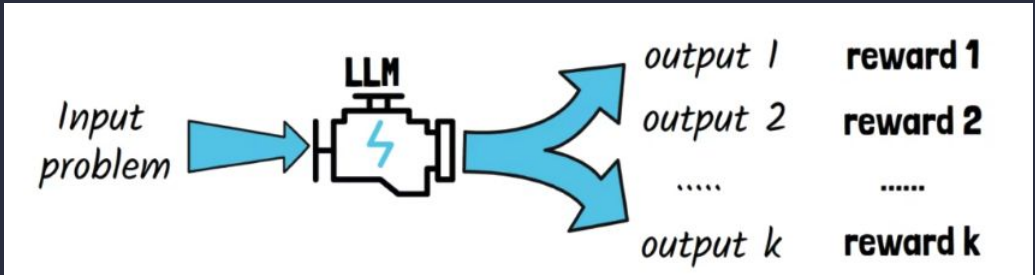
- Checkpoint-1 SFT (Supervised Fine-Tuning) with Distillation: Teach the model the basic task on text-guided data.
- Checkpoint-2 GRPO: Refine the model. It generates multiple proofs, a custom reward function picks the "best" one, and the model is trained only on that winner.



GRPO

- Group Relative Policy Optimization
- Samples a group of outputs <reasoning, answer>
- Assigns reward to outputs
  - Accuracy
  - Format

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: prompt. Assistant:



GRPO - Custom Reward Fn

- Normalization
  - Extracts code block from model's completion and gold-standard proof
  - Removes comments and normalizes whitespace
- Jaccard similarity between the tactics
- Penalties: sorry, by skip, etc
- Format bonus: <think> tags

The Impact of Data Supervision Style

Does the style of training data matter for theorem proving?

- Regime 1 (Formal-Only): Fine-tuned on DeepSeek-Prover-V1 (Lean code only).
- Regime 2 (Text-Guided): Fine-tuned on DeepSeek-Prover-V2-generation (NL rationale + Lean code).

```
let total_students := 900
let selected_students := 10
let A_students := 8
let B_students := 2
let A_correct := 7
let B_correct := 8
let total_correct := A_correct + B_correct
let total_samples := Nat.choose
selected_students total_correct
let A_samples := Nat.choose A_correct
selected_students
let B_samples := Nat.choose B_correct
selected_students
let prob_A := A_samples / total_samples
let prob_B := B_samples / total_samples
simp [prob_A, prob_B, Nat.choose, factorial,
Nat.sub_zero, Nat.mul_zero, Nat.mul_one]
```

First, we need to prove that \(5^2 + 12^2 = 13^2\). This is a straightforward arithmetic problem:
1. Calculate \(5^2 = 25\).
2. Calculate \(12^2 = 144\).
3. Calculate \(13^2 = 169\).
4. Add \(25 + 144 = 169\), which is \(13^2\).

Thus, the equation holds.

Step 1: Abstract Plan

1. Calculate \(5^2 = 25\).
2. Calculate \(12^2 = 144\).
3. Calculate \(13^2 = 169\).
4. Add \(25 + 144 = 169\) to verify that \(5^2 + 12^2 = 13^2\).

Step 2: Lean 4 `have` Statements

```lean4
theorem
have pythag : 5 ^ 2 + 12 ^ 2 = 13 ^ 2 := by
sorry
```

Complete Lean 4 Proof

```lean4
theorem
have pythag : 5 ^ 2 + 12 ^ 2 = 13 ^ 2 := by
norm_num
<;> rfl
<;> simp [pow_two]
<;> norm_num
<;> rfl
```

Results

Theorem	Base Model	SFT-Non CoT dataset	SFT-CoT dataset	GRPO+SFT
imp_chain	5	10	93	93
succ25_mem_of_zero_and_succ_closed	40	10	55	80
add_zero_eq_self	40	10	98	98
succ_add_one_eq_add_two	40	5	40	40
succ_pred_self	5	10	40	45
two_divides_double (Prompt 1)	10	5	5	94
two_divides_double (Prompt 2)	40	10	45	93

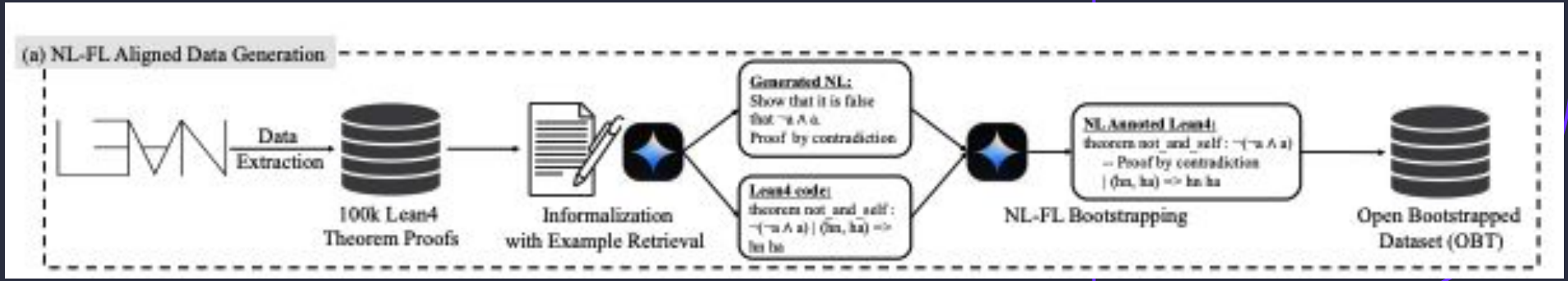
Main Category	Max Points
Proof Score	60
Explanation Score	25
Format and Integrity Score	15

Experiment-2

TheoremLlama style Fine-tuning

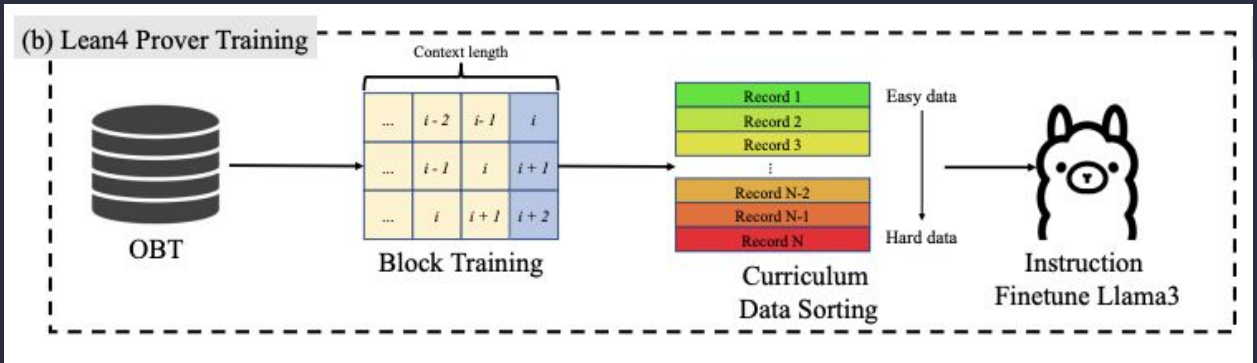
Open Bootstrapped Theorems Dataset

- Lean 4 code with NL reasoning
- Extraction & Retrieval
- Informalization
- Bootstrapping



Advanced Training with OBT Data

- OBT: Open Bootstrapped Theorems
- Curriculum Sorting
  - Train on examples from easy to hard.
  - Prevents the model from being overwhelmed
- Few-Shot Block Training
  - Provide the model with 3 solved examples in the prompt before asking it to solve the current problem.
  - The model learns to generate more concise and human-like proofs.



Results

Theorem Statement (Simplified)	Base Model Score	OBT-Finetuned Model Score
theorem imp_chain	35	50
theorem succ25_mem_of_zero...	20	40
theorem add_zero_eq_self	5	55
theorem succ_add_one_eq_add...	10	100
theorem succ_pred_self	5	20
theorem two_divides_double	5	25

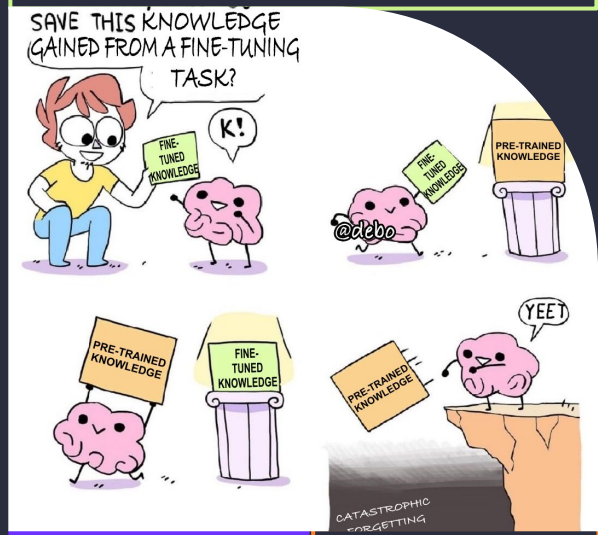
Main Category	Max Points
Proof Score	60
Explanation Score	25
Format and Integrity Score	15

Experiment-3

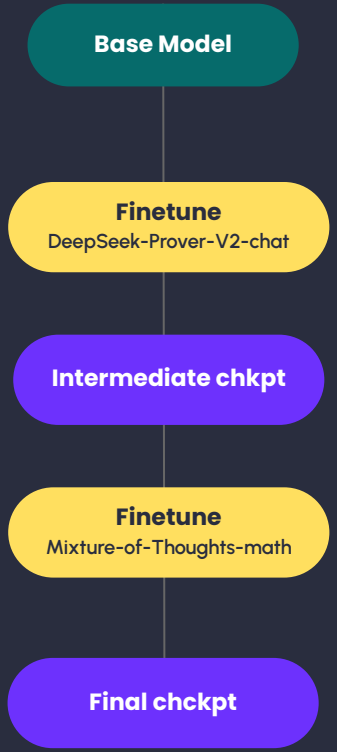
MoE and Catastrophic Forgetting

Sequential vs Mixed

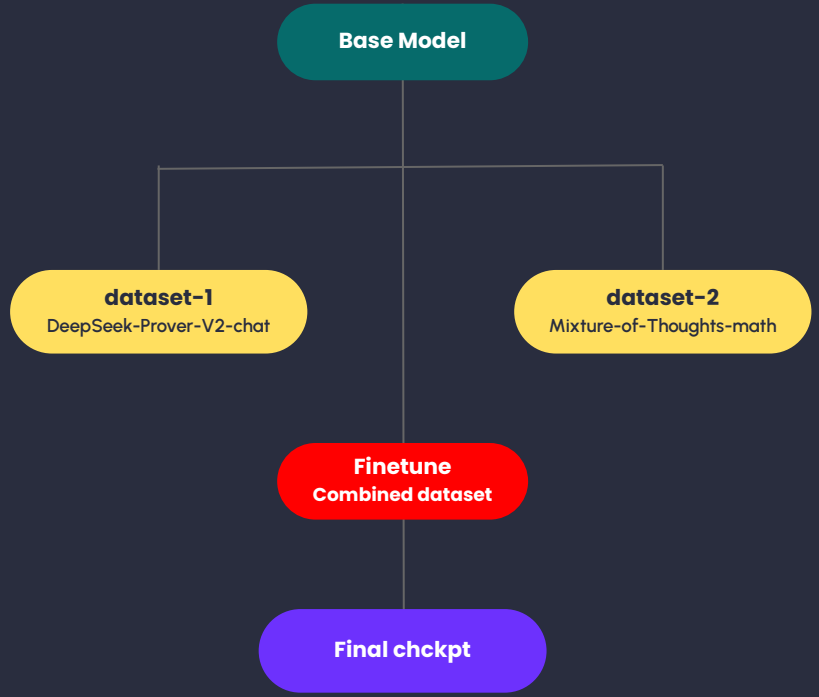
- Backbone: microsoft/phi-mini-moe-instruct
- Two fine tuning regimes
  - Sequential
  - Mixed
- Catastrophic forgetting



Sequential



Mixed



Results

Theorem	Base Model	Mixed FT	Sequential FT
imp_chain	35	84	5
succ25_mem_of_zero_and_succ_closed	20	60	5
add_zero_eq_self	5	97	0
succ_add_one_eq_add_two	0	96	30
succ_pred_self	5	92	5

Main Category	Max Points
Proof Score	60
Explanation Score	25
Format and Integrity Score	15

Infrastructure: Evaluation Framework

Infrastructure for Systematic Evaluation

Tech Stack

01. **Frontend (React)**
chat UI for side-by-side model comparison

02. **Backend (FastAPI)**
An orchestration layer to route requests to the correct model

03. **Inference Layer (TGI on H200 GPUs)**
High-performance model serving using Text Generation Inference.

The UI in Action

Conclusion

- Data-centric fine-tuning is a powerful strategy.
- Training pipelines like OBT and GRPO are key in niche domains like formal theorem proving.
- Training implications on catastrophic forgetting

Any questions? Ask away!

Thank you