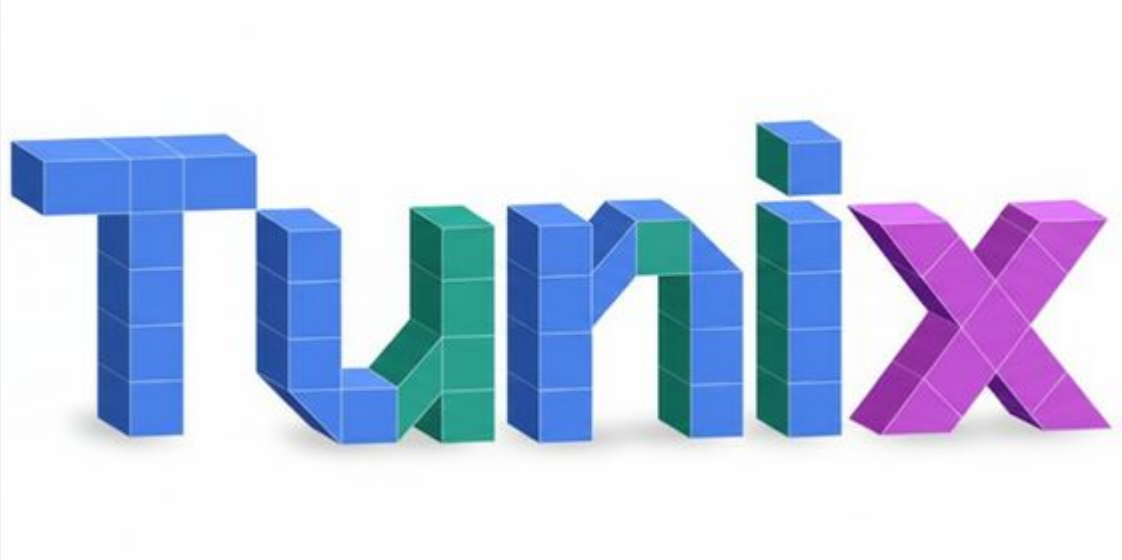
Training an LLM to Reason Using Tunix Supported RL

CJ Jones &
Adam Stein



Overview

- Most LLMs give an answer to a query - but they don’t show you how they got to their answer
- Training an LLM to explain its reasoning and ‘show its work’ is critical to fostering trust in AI tools
- Tunix - Google’s JAX-native library for LLM post-training - can be used to to train a model to show its work by laying out a reasoning trace before landing on an answer

Why Tunix?

- Tunix is a powerful library designed to streamline the post-training of LLMs

- It provides efficient and scalable support for:
  - Supervised Learning
  - **Reinforcement Learning**
  - Knowledge Distillation

- Tunix supports current RL algorithms:
  - Proximal Policy Optimization (PPO)
  - Group Relative Policy Optimization (GRPO)
  - Token-level Group Sequence Policy Optimization (GSPO-token)

Our Approach

- Why reinforcement learning?
  - LLM is already trained to get the right answer - it is not trained to show the steps of its reasoning
  - This is a behavior shaping problem - where a reward is earned for each correct step of the reasoning trace

- Start with an open-weight base model (Gemma3 1B) and use supervised fine-tuning to teach the correct format for reasoning trace

- Construct Two Step Pipeline with Planner and Solver Steps

- Improve the reasoning trace using RL algorithms by providing rewards for quality responses

Reward Function Design

- Reward function needs to carefully define what “good reasoning” looks like

- Examples of good reasoning include:
  - Answer correctness
  - Proper format
  - Response length
  - Response clarity
  - Coherence

Optimize Reward with Tunix

- Tunix supports GRPO:

1. Model samples a reasoning trace + answer
2. Reward function scores the entire output based on format and correctness
3. Tunix adjusts the model to increase probability of high-reward sequences

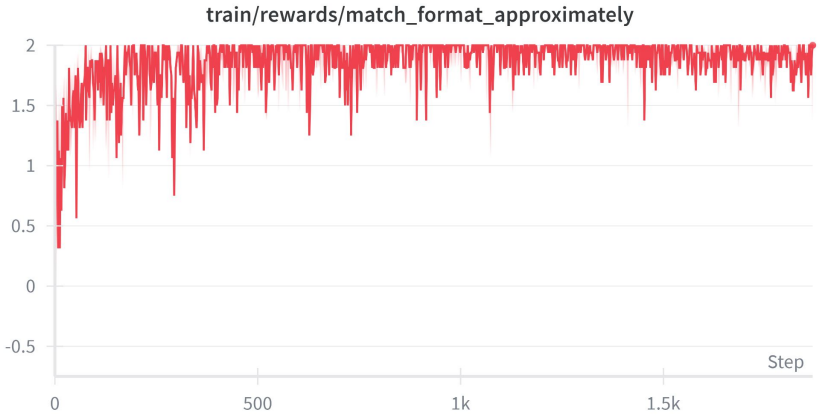
- Our Planner Reward Function:
  - reward if the format of the output **approximately** matches the instruction given +0.5 points for each correct piece

- Our Solver Reward Function:
  - reward if the format of the output **exactly** matches the instruction given in prompt: +3 points
  - reward if the format of the output **approximately** matches the instruction given in: +0.5 points for each correct piece
  - reward if the answer is **correct/partially correct**:
    - +3 points if fully correct
    - 0.25 - 0.50 points for partial correctness
    - -1 point for wrong answer

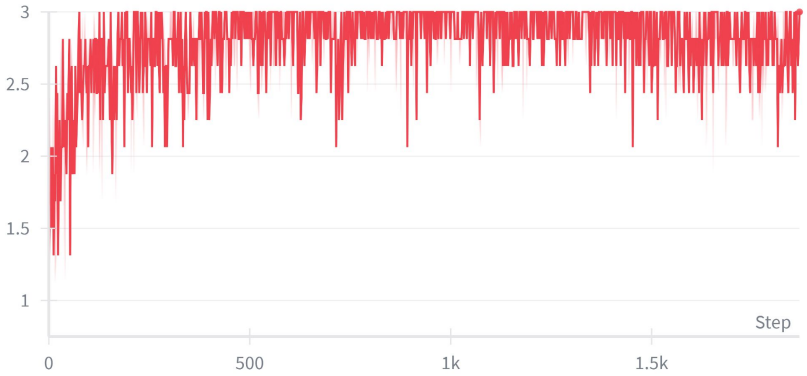
Planner + Solver with RL

- A Planner model (Gemma3-1B + LoRA, trained with GRPO) **generates a plan**
- A Solver model (Gemma3-1B + LoRA, trained with GRPO) receives (question + plan) and **generates the final reasoning + answer**
- A Two Step Pipeline where two models are trained independently
  - The trained planner is frozen after training and used to create plans for the solver
- All existing reward functions (format + correctness) are preserved

Training Results - Demo



Training Results - Demo



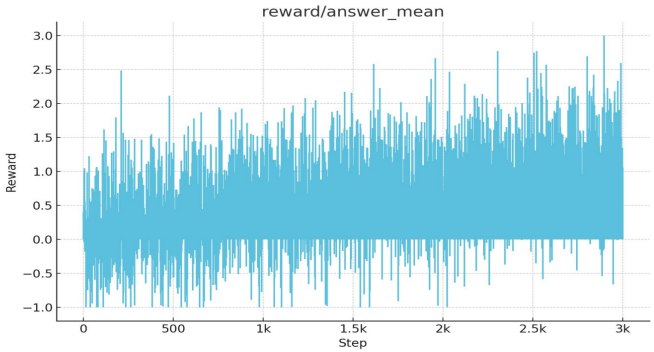
Baseline Reasoning Example

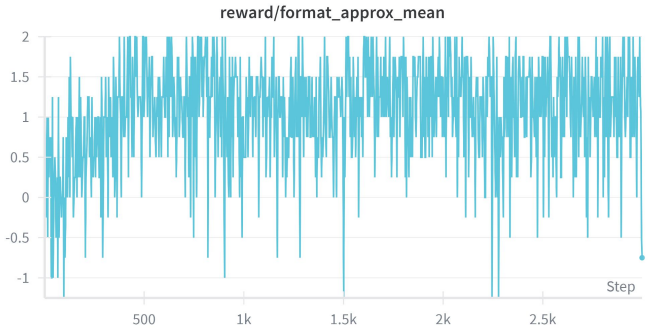
- Question: In Johnstown, the population is 80 and every single person drives by themselves to work. Each car on a road pollutes 10 pounds of carbon a year. A single bus pollutes 100 pounds of carbon a year. A bus can hold 40 people. The town decides to start running a bus and 25% of the people who used to drive now take the bus. How many fewer pounds of carbon are now emitted per year?

- Answer: 100

- Reasoning: The initial population of Johnstown is 80 people. The initial amount of carbon pollution per person is 10 pounds per car, and the initial amount of carbon pollution per bus is 100 pounds per bus. We need to figure out how many new buses are running and how many people are taking the bus. The number of cars driven by the initial population is 80. The number of buses driven by the initial population is 80 * 0.25 = 20 buses. The total carbon pollution from the initial cars is 80 * 10 = 800 pounds. The total carbon pollution from the initial buses is 20 * 100 = 2000 pounds. The total carbon pollution from the initial population is 800 + 2000 = 2800 pounds. Now, with the bus, the total carbon pollution is 20 * 100 + 20 * 10 = 2000 + 200 = 2200 pounds. The decrease in carbon pollution is 2800 - 2200 = 600 pounds.

Training Results - P + S





Two Step Reasoning Run

```
Planner Input:
<start_of_turn>user
You are a planning assistant. Produce a short numbered plan (3-5
steps)
for solving the problem. Do NOT solve the problem.
Problem:
Mr Hezekiah had 20 trucks from his store supplying fertiliser to
different farmers in his hometown dispatched for delivery on a
particular day. Each truck was carrying 20 tons of fertiliser packed
in bags. Two hours after the trucks had departed for delivery, Mr
Hezekiah got the news that a quarter of the number of lorries
dispatched for delivery had mechanical failures on the road and
could not deliver the fertilisers to the farmers. Calculate the
total number of tons of fertiliser that reached the farmers that
day?
<end_of_turn>
<start_of_turn>planner
<plan>

Planner Output:

Generated Plan:
<plan>
1.  Determine the initial number of trucks dispatched.
2.  Calculate the number of trucks that failed.
3.  Calculate the remaining number of trucks.
4.  Determine the total amount of fertiliser carried by each truck.
5.  Calculate the total amount of fertiliser that reached the
farmers.

</plan>

Planner Format Match Score: 2.5

Solver Format Match (Exact): 0.0
Solver Format Match (Approx): 1.0

Check Answer Score: 3.0

--- End Tracing Pipeline ---
```

```
Solver Input:
Your task:
1. Follow the provided plan EXACTLY.
2. Write detailed reasoning in the <reasoning>...</reasoning> block.
3. Place ONLY the final numeric answer in <answer>...</answer>.
4. After </answer>, STOP immediately.
Response format:
<solution>
<reasoning>
[step-by-step reasoning following the plan; do NOT skip steps]
</reasoning>
<answer>
[FINAL NUMERIC ANSWER ONLY]
</answer>
</solution>
Problem:
Mr Hezekiah had 20 trucks from his store supplying fertiliser to different farmers in his hometown
dispatched for delivery on a particular day. Each truck was carrying 20 tons of fertiliser packed i
bags. Two hours after the trucks had departed for delivery, Mr Hezekiah got the news that a quarter
of the number of lorries dispatched for delivery had mechanical failures on the road and could not
deliver the fertilisers to the farmers. Calculate the total number of tons of fertiliser that reach
the farmers that day?
Plan:
1.  Determine the initial number of trucks dispatched.
2.  Calculate the number of trucks that failed.
3.  Calculate the remaining number of trucks.
4.  Determine the total amount of fertiliser carried by each truck.
5.  Calculate the total amount of fertiliser that reached the farmers.

Solver Output:
<solution>
1.  The initial number of trucks dispatched for delivery was 20.
2.  The number of trucks that failed is $\frac{1}{4} \times 20 = 5$.
3.  The remaining number of trucks is $20 - 5 = 15$.
4.  Each truck carried 20 tons of fertiliser.
5.  The total amount of fertiliser carried by the remaining 15 trucks is $15 \times 20 = 300$ tons.
6.  The total amount of fertiliser that reached the farmers that day is 300 tons.
</reasoning>
<answer>300</answer>
</solution>
```

Anecdotal Comparison

Question: Ian has a board that is 40 feet long. He decides to make a cut so he can have two pieces. The longer piece is 4 times longer than the shorter piece. How long is the longer piece?

Answer: 32

=== BASELINE PLANNER ===
1. Define the variables: Let 'x' represent the length of the shorter piece.
2. Express the longer piece in terms of 'x': The longer piece is 4 times longer than the shorter piece, so its length is 4x.
3. Set up an equation: The total length of both pieces is 40 feet, so x + 4x = 40.
4. Solve for x: Combine like terms and simplify the equation to find the length of the shorter piece.

=== BASELINE SOLVER ===
x = 40/2 = 20
The longer piece is 4x = 4 * 20 = 80 feet.
<answer>80</answer>

=== TRAINED PLANNER ===
1. Define the variables: Let 'x' represent the length of the shorter piece.
2. Express the longer piece in terms of 'x': The longer piece is 4 times longer than the shorter piece, so its length is 4x.
3. Set up an equation: The total length of the two pieces is 40 feet, so x + 4x = 40.
4. Solve for x: Combine like terms and simplify the equation to find the length of the shorter piece.

=== TRAINED SOLVER ===
<solution>
x = 40/2 = 20 feet.
<reasoning>
The problem states that the longer piece is 4 times longer than the shorter piece.
We are given that the total length of the two pieces is 40 feet.
Let 'x' be the length of the shorter piece.
Then the length of the longer piece is 4x.
We can set up the equation: x + 4x = 40.
Simplifying, we get 5x = 40. Dividing both sides by 5, we get x = 8

Results

On a hold out test set of 100 questions the models were evaluated on their reward outputs

Two-Step GRPO: Model Performance Summary

Model Version	Correct	Total	Answer Accuracy	Partial Accuracy	Format Accuracy
Baseline Single-Step	29	200	14.5%	16.0%	94.25%
Two-Step (Planner + Solver) – Pre-Training	62	200	31.0%	34.5%	18.5%
Two-Step – Post-GRPO Solver Training	77	200	38.5%	42.0%	47.5%

Limitations & Future Work

Limitations

- **Format-based rewards dominate** the learning signal, causing the model to over-optimize for structure rather than reasoning
- **Correctness rewards are sparse and noisy**, leading to unstable improvement in the answer-level reward curve
- No **mutual feedback loop between Planner and Solver** (one-way communication only)

Future Work

- **Longer RL training** to evaluate whether correctness reward stabilizes or continues improving
- Introduce a **reward for reasoning quality**, e.g., LLM-grader or chain-of-thought coherence scoring
- Train the Planner with RL, not just the Solver, enabling **coordinated multi-agent reasoning**

Works Cited

- https://www.kaggle.com/competitions/google-tunix-hackathon/data
- https://github.com/google/tunix
- https://pypi.org/project/google-tunix/

- Saha, Swarnadeep, et al. "Learning to plan & reason for evaluation with thinking-llm-as-a-judge." arXiv preprint arXiv:2501.18099 (2025).

- DeepSeek-AI et al., "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning" (2025), which explores using reinforcement learning to enhance the reasoning capabilities of Large Language Models (LLMs).