

# Google-Tunix Checklist Reasoning with GRPO

## Project Report

---

### Authors

Name	Role	GitHub	LinkedIn
<b>Kanishk Surwade</b>	Lead Developer & Research	<a href="https://github.com/Kanishksurwade">https://github.com/Kanishksurwade</a>	<a href="http://www.linkedin.com/in/kanishk-surwade">http://www.linkedin.com/in/kanishk-surwade</a>
<b>Sarthak</b>	Research & Training Support	<a href="https://github.com/sarthak3877">https://github.com/sarthak3877</a>	<a href="http://www.linkedin.com/in/sarthak-sudarshani">www.linkedin.com/in/sarthak-sudarshani</a>

---

### 1. Introduction

Large Language Models (LLMs) are capable of producing correct answers across a wide range of tasks. However, most models do not explicitly explain the reasoning steps used to reach those answers. This lack of transparency limits trust, interpretability, and reliability—especially for reasoning-intensive tasks.

This project was developed as part of the **Google Tunix Hackathon**, with the goal of training a language model that not only provides answers but also **shows its step-by-step reasoning** in a structured and consistent manner.

---

### 2. Problem Statement

**Key Challenges:** - LLMs often skip intermediate reasoning steps - Reasoning is inconsistent across prompts - Outputs are difficult to audit or verify

**Objective:** To fine-tune an open-weight LLM so that it explicitly produces a reasoning trace before generating a final answer.

---

### 3. Tools and Technologies Used

Component	Description
Base Model	Google Gemma (Gemma3 1B)
Training Library	Tunix (JAX-native LLM post-training)
Optimization Method	GRPO (Group Relative Policy Optimization)
Compute Platform	Kaggle TPU
Language	Python

---

### 4. Project Workflow Overview

The project followed a structured, end-to-end workflow:

1. Understanding the hackathon requirements
  2. Studying Tunix and GRPO methodology
  3. Designing a checklist-based reasoning format
  4. Preparing prompts and reward strategy
  5. Fine-tuning the model on Kaggle TPU
  6. Evaluating outputs and formatting results
  7. Documenting and submitting the project
- 

### 5. Reasoning Strategy Design

Instead of allowing the model to directly generate an answer, a **checklist-based reasoning approach** was introduced.

#### Output Format Enforced:

```
<reasoning>
Model thinking trace
</reasoning>

<answer>
Final answer
</answer>
```

This structure encourages the model to: - Follow logical steps - Avoid skipping reasoning - Separate explanation from conclusion

---

## 6. Training Methodology

### 6.1 GRPO (Group Relative Policy Optimization)

GRPO improves reasoning by: - Generating multiple responses for a single prompt - Comparing them within a group - Optimizing the model based on relative performance

This method eliminates the need for a separate value model, making it efficient under limited compute constraints.

### 6.2 Training Setup

Parameter	Description
Session Type	Single Kaggle TPU session
Max Output Tokens	< 1000
Language	English
Multimodality	Not used
Tool Use	Not required

---

## 7. Implementation Details

The complete implementation is provided in a **public Kaggle notebook**, which includes: - Environment setup - Prompt design - Reward composition - Hyperparameters - Training loop - Model output examples

The notebook is designed to be reproducible and runnable within Kaggle's compute limits.

---

## 8. Results and Observations

### Key Improvements Observed:

- Clear step-by-step reasoning traces
- Better logical consistency
- Improved transparency of model decisions

### Applicable Task Domains:

- Mathematical reasoning
  - Coding problems
  - Summarization
  - Creative reasoning
-

## 9. Limitations

- Training limited to a single TPU session
- English-only evaluation
- No multimodal reasoning

These limitations were intentionally accepted to align with hackathon constraints.

---

## 10. Conclusion

This project demonstrates that structured reasoning can be effectively taught to language models using open tools like Tunix and GRPO. By enforcing a checklist-based reasoning format, the model becomes more transparent, interpretable, and reliable.

The approach lowers the barrier for building explainable AI systems and contributes reusable training recipes for the open-source community.

---

## 11. References and Links

- Kaggle Notebook: <https://www.kaggle.com/code/kanishksurwade123/google-tunix-checklist-reasoning-with-grpo/edit>
  - GitHub Repository: <https://github.com/Kanishksurwade>
  - Hackathon: Google Tunix – Train a Model to Show Its Work
- 

**End of Report**