[Results] Enhancing Reinforcement Learning with Large Language Models

Vaibhav Jain
MS - Data Science and AI
Universität des Saarlandes
vaja00001@stud.uni-saarland.de

1 Experiments

We perform experiments to answer our research questions *RQ(1-4)*. We choose a variety of environments which require strategic, long-term planning and where real-world knowledge priors would help in faster learning.

1.1 Minigrid - BabyAI

Minigrid library contains a collection of 2D grid-world environments with goal-oriented tasks. The agent in these environments is a triangle-like agent with a discrete action space. The tasks involve solving different maze maps and interacting with different objects such as doors, keys, or boxes. The design of the library is meant to be simple, fast, and easily customizable.

Implementation Details [Add any relevant details here.]

1.1.1 Phase I: Evaluating LLM Suggestions for Minigrid

Research Questions (RQ):

- **RQ1:** Can we integrate Minigrid game dynamics using text descriptions into an LLM through prompt engineering?
- **RQ2:** How do different representations of the Minigrid observation (input) and output format affect the quality, interpretability, and utility of LLM-suggested actions?

We systematically explore two ablation dimensions:

1. **Input Representation (State Encoding)**
2. **Output Representation (Action Abstraction Level)**

We evaluate each Input-Output combination using multiple prompting techniques (Zero-Shot, Few-Shot, Chain-of-Thought).

1. Input Representation (State Encoding) We experiment with four methods to encode Minigrid observations as text:

1. **Structured Natural Language Description** "In front of you is a red key. To your left is a closed blue door."
2. **Grid-Style ASCII Map with Legend** A text-based grid showing the agent's view with a legend.

Preprint. Under review.

3. **Tuple List (Structured Text)** "Visible objects: [(red key, (1,2)), (blue door (closed), (1,3))]"
4. **Relative Object Descriptions (Agent-Centric)** "There is a red key 1 tile ahead. A blue door is 2 tiles to your right."

Representation Method	Description	Example
Natural Language Description	Free-text description of nearby objects	"In front of you is a red key."
ASCII Grid Map	Text grid with symbols and legend	See Appendix-X
Tuple List	Structured list of objects + coordinates	"[(red key, (1,2))]"
Relative Descriptions	Object locations relative to agent	"Red key 1 tile ahead"

Table 1: Summary of different state representations for Minigrid (Input Ablations)

2. Output Representation (Action Abstraction Level) We experiment with three formats for LLM-suggested actions:

1. **Primitive Action Output (Low-level control)** Atomic action. Examples: "0", "move forward", "turn left"
2. **Subgoal Output (Mid-level intent)** Short-term subgoal. Examples: "GoNextToSubgoal(red key)", "OpenSubgoal(blue door)"
3. **Instruction Output (High-level instruction)** Temporally-structured instruction. Examples: "Go to the red key and pick it up."

Output Type	Description	Example	Abstraction Level
Primitive Action	Single atomic action	"move forward" / "0"	Low
Subgoal	Short-term objective	"PickupSubgoal(red key)"	Medium
Instruction	Full temporal instruction	"Go to red key and pick it up."	High

Table 2: Summary of different LLM output representations for Minigrid (Output Ablations)

3. Prompting Strategy Analysis **Prompting Methods Evaluated:** We systematically compared three prompting strategies across all Input-Output configurations:

- Zero-Shot Prompting
- Few-Shot Prompting (1, 3, 5 examples)
- Chain-of-Thought Prompting

Key Finding: Chain-of-Thought (CoT) prompting consistently outperformed both zero-shot and few-shot approaches, delivering the most reliable and contextually appropriate suggestions across all input-output combinations. CoT prompting enabled the LLM to provide explicit reasoning steps, leading to more consistent and higher-quality action suggestions. `TODO: provide sample examples here`

4. Experimental Results and Analysis **LLM Model:** Llama3-70b was used for all evaluations with Chain-of-Thought prompting.

Key Findings from Input Representation Ablations:

- **Grid-Style ASCII Map Superior:** The grid-based encoding achieved the best performance across all evaluation metrics. Possibly because it provides the LLM with spatial context necessary for effective reasoning.
- **Previous Actions Critical:** Including previous action history as input was essential for guiding the LLM through circular dead-ends and preventing repetitive behaviors.

- **Spatial Context Matters:** Representations that preserved spatial relationships (Grid ASCII, Relative Descriptions) outperformed purely textual descriptions on average.

Key Findings from Output Representation Ablations:

- **Subgoal Output Most Sensible:** LLM-generated subgoals (e.g., *GoNextToSubgoal*) demonstrated high interpretability and alignment with task objectives
- **Primitive Actions Show Strong Bias:** The LLM exhibited a slight bias toward action "2" (move forward). This bias was more pronounced when not using Chain-of-Thought reasoning, for example, with Zero-Shot prediction, it predicted the "2" action in approximately 80% of cases.
- **Sequential Decision-Making Success:** Despite action bias in individual predictions, when the LLM played out entire episodes, albeit sub-optimally, it demonstrated successful sequential decision-making capabilities.

Ground Truth (Oracle) Comparison Results:

- **Sub-optimal but Functional:** LLM performance was consistently sub-optimal compared to oracle policies, but maintained task-relevant reasoning.

5. Example LLM Response The following example demonstrates the best-performing configuration (Grid ASCII input + Subgoal output + Chain-of-Thought prompting):

Input:

```
Grid:
W W W W W W W W
W . . . . . . W
W . . . . . . W
W O . . . . . W
W . > . . . . W
W . . . . . . W
W . . . . . . W
W W W W W W W W

Legend:
> - agent facing east
? - unseen
. - empty
W - wall
O - grey ball

Mission: go to the grey ball
Previous actions: []
```

LLM Response (Chain-of-Thought):

```
Prediction(
    reasoning="The agent's mission is to go to the grey ball. The grey ball is
               located to the north of the agent. Since the agent is facing east,
               it first needs to turn north to face the grey ball. The action to
               turn the agent to the left (north) would be the most efficient
               first step.",
    primitive_action=0,
    subgoal=GoNextToSubgoal
)
```

6. Integration with RL Training **Subgoal Limitation:** While subgoal outputs demonstrated high interpretability and sensible reasoning, direct integration as RL training hints did not yield performance improvements. This suggests a gap between LLM reasoning quality and RL training utility that requires further investigation. For further details on the integration process and additional experimental results, see Appendix **??** (A.3).

7. Phase I Conclusions **RQ1 Answer:** Yes, Minigrid game dynamics can be successfully integrated into LLMs through prompt engineering. Grid-based ASCII representations with spatial legends prove most effective for encoding game states, and Chain-of-Thought prompting is essential for consistent, high-quality suggestions.

RQ2 Answer: Input representation significantly impacts LLM performance, with grid-based formats enabling superior spatial reasoning. Output format affects performance, with primitive actions providing the best overall performance for RL integration, as we will discuss in the Phase II section. Chain-of-Thought prompting consistently outperforms zero-shot and few-shot approaches across all configurations.

1.1.2 Phase II: Evaluating RL Training

Research Questions (RQ):

- **RQ3:** Do the hints provided by the LLM help the RL agent improve its performance?
- **RQ4:** Can we successfully use LLM reasoning for RL training?

Experimental Setup

- **RL Algorithm:** PPO with 3M frames (GoToObj, OpenDoor) and 5M frames (PickupLoc)
- **Hint Configurations:** Oracle (ground truth), LLM-hints (enhanced-observations), Baseline (no hints)
- **Hint Frequencies (f):** 1, 5, 10 actions per hint (only f=5, f=10 for LLM-hints)
- **Text-Conditioned Policy:** With/without mission text in policy input
- **Environments:** GoToObj (easy), OpenDoor (medium), PickupLoc (hard)

Environment	Baseline	Oracle f=1	Oracle f=5	LLM-hints f=5	LLM-hints f=10
GoToObj	100.0%	100.0%	100.0%	100.0%	100.0%
OpenDoor	74.2%	100.0%	96.4%	75.0%	74.3%
PickupLoc	17.2%	99.4%	42.1%	29.5%	18.9%

Table 3: Final rolling win rates across environments and configurations (no text)

LLM-Hints Performance Results **Key Findings (Table 3):**

- **Task complexity drives LLM-hints benefits:** LLM-hints improvements scale dramatically with environment difficulty
  - PickupLoc: **Significant improvement** (17.2% → 29.5% with f=5)
  - OpenDoor: **Marginal gains** (74.2% → 75.0% with f=5)
  - GoToObj: **Ceiling effect** (already at 100% baseline performance)
- **Frequency sensitivity:** f=5 consistently outperforms f=10 for LLM-hints guidance
- **Oracle comparison:** Perfect guidance (Oracle f=1) establishes theoretical upper bounds, achieving near-perfect performance across all environments

Environment	50% Success		75% Success		90% Success	
	Baseline	**Oracle f=1**	**Baseline**	**Oracle f=1**	**Baseline**	**Oracle f=1**
GoToObj	1.08M	57K (**19×**)	1.58M	57K (**28×**)	1.83M	66K (**28×**)
OpenDoor	475K	57K (**8×**)	950K	57K (**17×**)	Never	57K (∞)
PickupLoc	Never	66K (∞)	Never	74K (∞)	Never	82K (∞)

Table 4: Sample complexity reduction: Frames required to reach success thresholds (see Figure 2)

LLM-Hints Sample Efficiency Analysis **Critical Insights (Table 4):**

- **LLM-hints enable impossible tasks:** In PickupLoc, baseline *never reaches 25%* success, while LLM-hints f=5 achieves 29.5% final performance
- **Dramatic acceleration potential:** Oracle f=1 demonstrates up to **28× faster learning**, establishing upper bounds for LLM-hints improvements
- **LLM-hints sample efficiency:** LLM-hints achieves **9× acceleration** at 50% threshold in GoToObj compared to baseline
- **Threshold-dependent benefits:** Higher success thresholds (75%, 90%) show exponentially larger improvements with guidance (Figure 2)

LLM-Hints Without Goal Instructions **Research Question:** *Do LLM-hints work without explicit mission text?*

Environment	Baseline	Baseline +Text	LLM-hints f=5	LLM-hints f=5 +Text
GoToObj	100.0%	99.6%	100.0%	100.0%
OpenDoor	74.2%	75.9%	75.0%	77.9%
PickupLoc	17.2%	15.9%	29.5%	20.4%

Table 5: LLM-hints performance with/without mission text conditioning (see Figure 1)

Answer: Yes, LLM-hints are highly effective without goal instructions (Table 5):

- **LLM-hints core capability:** Achieves substantial improvements without mission text
  - PickupLoc: **71% boost** over baseline (17.2% $\rightarrow$ 29.5%)
  - OpenDoor: Small but consistent improvement (74.2% $\rightarrow$ 75.0%)
- **Mixed text effects:** Adding mission text to LLM-hints shows environment-dependent results
  - OpenDoor: **+2.9%** improvement (75.0% $\rightarrow$ 77.9%)
  - PickupLoc: **-9.1%** degradation (29.5% $\rightarrow$ 20.4%)
- **Oracle benchmark:** Perfect guidance works excellently both with and without mission text, providing a comparison baseline

Figures and Visualizations

Research Question Conclusions **RQ3: Do the hints provided by the LLM help the RL agent improve its performance?**

- **Yes - Dramatic performance improvements:** LLM-hints achieve **9× faster learning** at 50% threshold (Table 4), with substantial gains across all environments
- **Enables new performance heights:** *Transforms complete failures into successes* – PickupLoc baseline never reaches 25% success, while LLM-hints achieve 29.5% (Figure 1)
- **Scales with complexity:** Benefits increase dramatically with environment difficulty and sparsity, proving most valuable where agents struggle most (Table 3)

RQ4: Can we successfully use LLM reasoning for RL training?

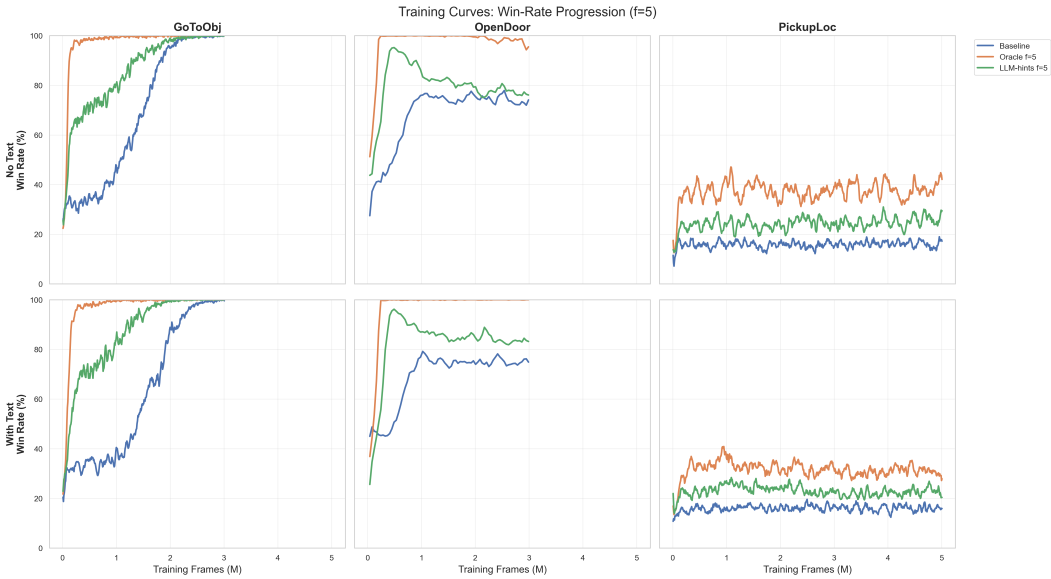

Figure 1: Win-rate progression during training for frequency f=5. Rows: no text vs with text configurations. Columns: GoToObj, OpenDoor, PickupLoc. Each subplot compares Baseline vs Oracle f=5 vs LLM-hints f=5.

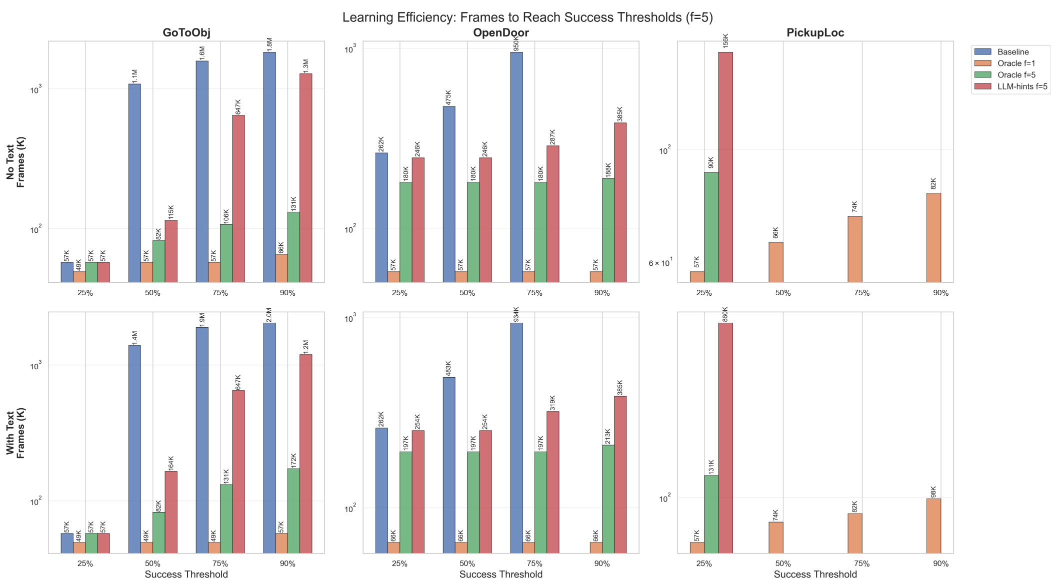

Figure 2: Learning efficiency comparison showing frames required to reach 25%, 50%, 75%, and 90% success thresholds. Grouped bars for Baseline, Oracle f=1, Oracle f=5, LLM-hints f=5. Rows: no text vs with text. Columns: GoToObj, OpenDoor, PickupLoc.

- **Yes - Highly successful integration:** LLM reasoning provides **71% improvement** in PickupLoc without any goal instruction, demonstrating effective knowledge transfer (Table 5)
- **Robust across conditions:** LLM reasoning works effectively both with and without mission text, showing flexible integration capabilities
- **Frequency-dependent effectiveness:** LLM reasoning shows optimal performance at specific hint frequencies, indicating successful integration of LLM knowledge with RL exploration (Figure 3)

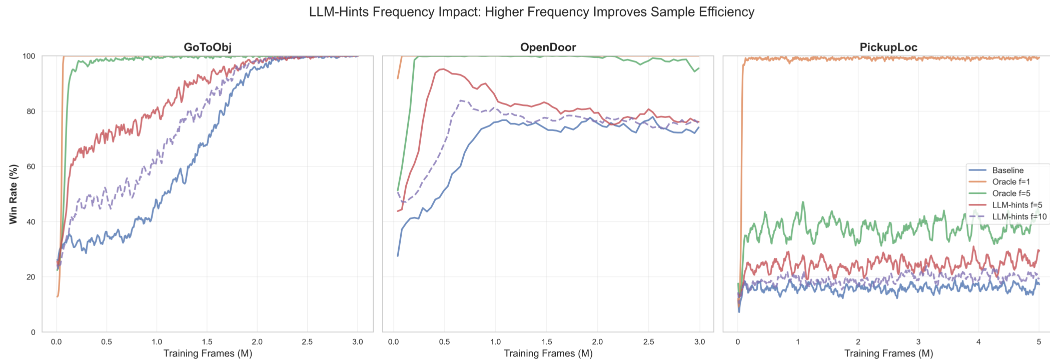

Figure 3: Impact of hint frequency on learning curves. Training progression comparing Baseline, Oracle (f=1, f=5), and LLM-hints (f=5, f=10) across environments. Higher hint frequencies demonstrate improved sample efficiency and faster convergence.

TODO: Performance when LLM-Hints are Stopped at X% of Training Steps

Research Question: *How does the timing of LLM-hints withdrawal affect final performance and learning retention?*

Experimental Setup:

- **Withdrawal Schedule:** LLM-hints provided for first X% of training steps, then stopped
- **Testing Points:** X = {10%, 20%} of total training duration

[Results to be added]

Key Questions to Address:

- Does early LLM-hints guidance create lasting learning benefits?
- How does performance degradation after LLM-hints withdrawal compare across environments?

Key Insight: *LLM-hints successfully transform RL learning from a sample-inefficient trial-and-error process into an accelerated, guided exploration paradigm.* LLM reasoning proves highly effective for RL training, with benefits scaling with task complexity. The most dramatic improvements occur in sparse-reward, long-horizon environments where baseline methods fail entirely. The approach works effectively without explicit goal instructions, demonstrating successful integration of LLM reasoning capabilities with RL training.

A Additional Experimental Results

A.1 Training Curves for Frequency f=10

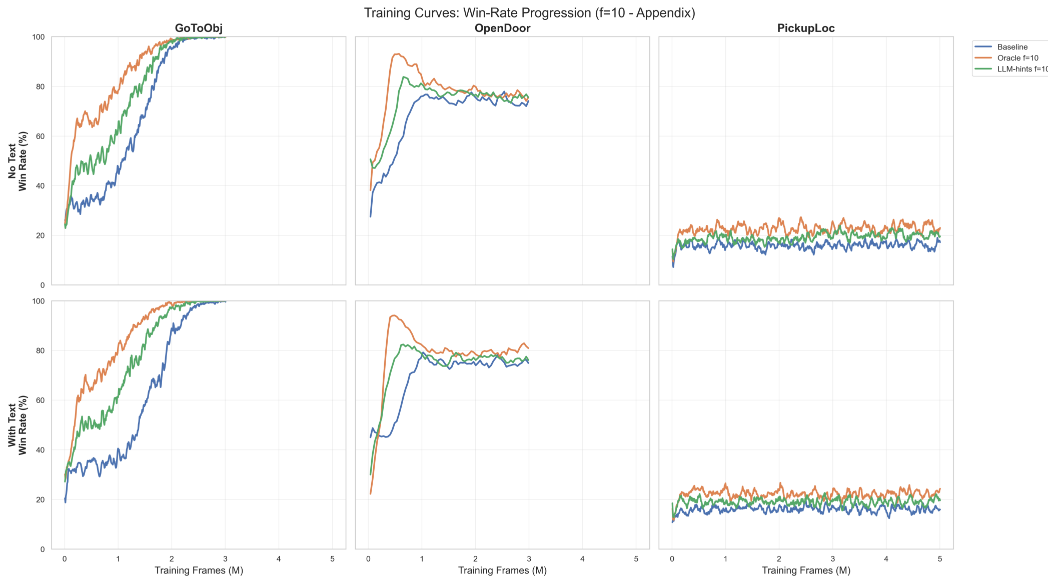

Figure 4: Win-rate progression during training for frequency f=10. Rows: no text vs with text configurations. Columns: GoToObj, OpenDoor, PickupLoc. Each subplot compares Baseline vs Oracle f=10 vs LLM-hints f=10. Shows similar patterns to f=5 but with slightly different convergence dynamics.

A.2 Complete Learning Efficiency Analysis

A.3 Subgoal-based LLM Hints RL Training Results

Summary: RL training results using subgoal-based LLM hints showed no performance improvements compared to baseline and primitive action hints.

Notably, as shown in Figure 6, even with perfect subgoal hints from the Oracle, the RL agent's performance does not improve over the baseline. Despite high interpretability and sensible reasoning in LLM-generated subgoals, direct integration as RL training hints did not yield performance improvements across environments.

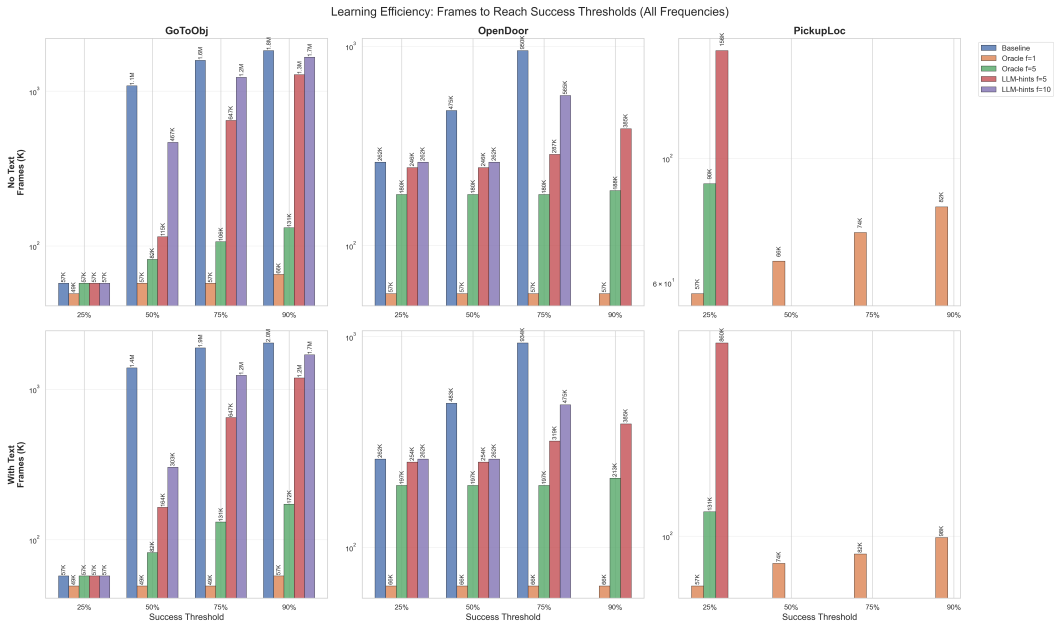

Figure 5: Complete learning efficiency comparison showing frames required to reach 25%, 50%, 75%, and 90% success thresholds. Grouped bars for Baseline, Oracle f=1, Oracle f=5, LLM-hints f=5, LLM-hints f=10. Rows: no text vs with text. Columns: GoToObj, OpenDoor, PickupLoc. This comprehensive view includes all frequency configurations studied.

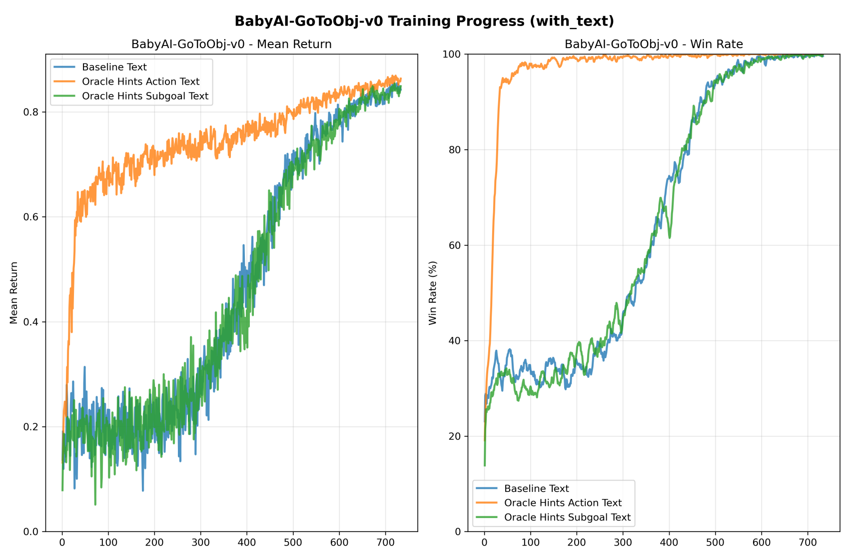

Figure 6: RL training curves for BabyAI-GoToObj-v0 with text, comparing Baseline Text, Oracle Hints Action Text, and Oracle Hints Subgoal Text.