PSG College of Technology

Causal Robustness in LLM guided Reinforcement Learning Agents

7th November 2025

Agenda

Meet the team

Anandkumar NS
Dhakkshin S R
Kishoreadhith V
Raj Ragavender M
Rithvik K

What is the Problem ?

“How can we build LLM-guided RL agents that are safe, interpretable, and robust to adversarial inputs?”

Why is this problem important in the current technological landscape ?

Recent AI advancements integrate Large Language Models (LLMs) with Reinforcement Learning (RL) for intelligent decision-making.

LLMs act as strategic planners, but their advice can be unreliable under corrupted or ambiguous input states.

LLM-guided agents often trust language advice blindly. Under corrupted state conditions, the LLM may give misleading guidance.

Why LLMs Fail?

High stress levels overwhelm the model and impair its cognitive functioning, leading to declines in performance across tasks...

Context Overload in High-speed and Changing Environments...

arXiv:2409.17167v1 [cs.HC] 14 Sep 2024

StressPrompt: Does Stress Impact Large Language Models and Human Performance Similarly?

Guobin Shen[1,2,3,4]\equalcontrib, Dongcheng Zhao[1,2,3]\equalcontrib, Aorigele Bao[1,2,3,5], Xiang He[1,2,3], Yiting Dong[1,2,3,4], Yi Zeng[1,2,3,5][1]

Our Unique Solution

Our Solution

Our unique solution is to approach this solution from a mathematical point of view to achieve efficiency, reliability and speed.

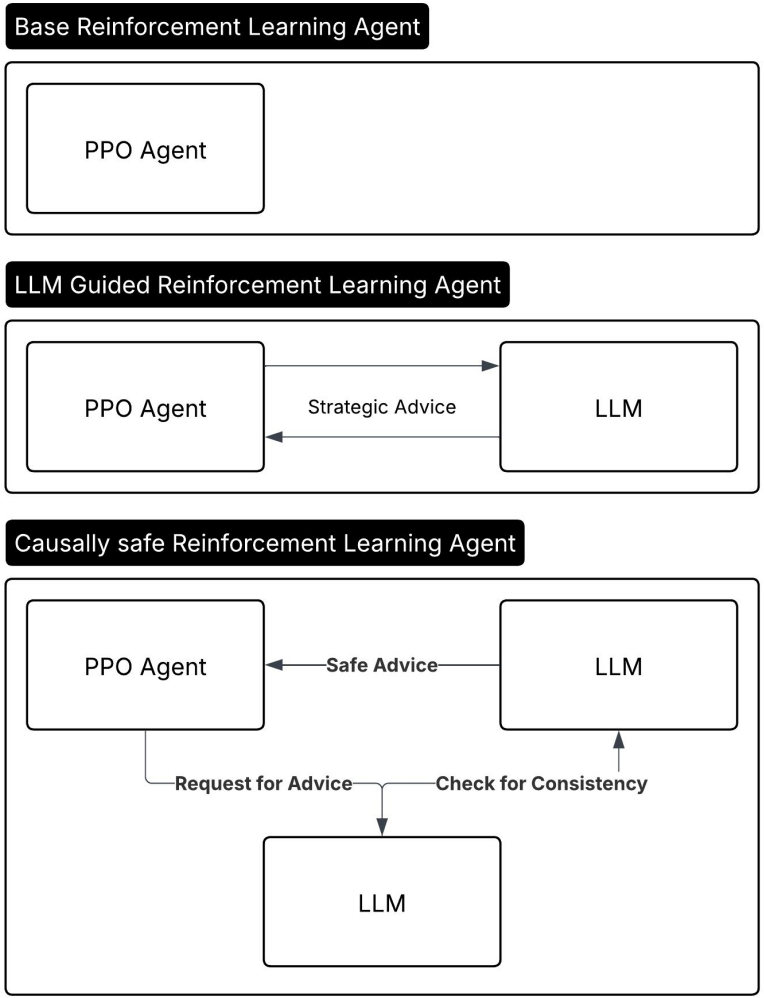
A causal safety system ***has been developed*** using the data from the LLM guided RL agent.

This involves monitoring the prior and posterior conditions of each step to study how decisions are made and how they affect the environment.

Different Systems to Build

There are 3 central systems that are needed to be built for a strong and meaningful analysis of the project.

The Base RL Agent and LLM guided RL Agent are there to establish a logically sound baseline and demonstrate overall improvement of the created work product.



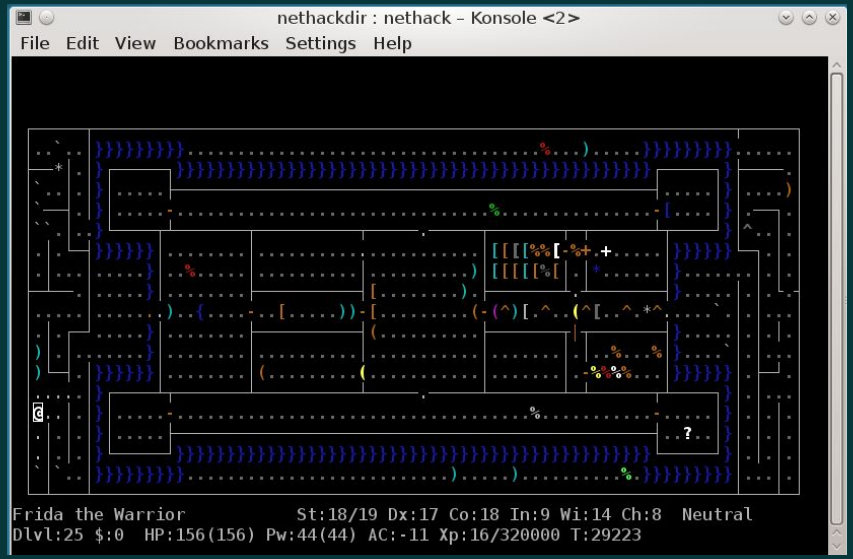
NetHack Environment

What is NetHack ?

NetHack is a classic, open-source single-player roguelike game that was first released in 1987.

The best features of NetHack: -

1. Procedural Generation
2. Extreme Complexity
3. Ascension
4. Item Identification



Fundamental Background Papers

The NetHack Learning Environment

Heinrich Küttler[+] **Nantas Nardelli**[=] **Alexander H. Miller**[+]
Roberta Raileanu[*] **Marco Selvatici**[#] **Edward Grefenstette**[+!] **Tim Rocktäschel**[+!]

[+]Facebook AI Research [=]University of Oxford [*]New York University
[#]Imperial College London [!]University College London

{hnr,rockt}@fb.com

Top Projects that use NetHack

NetPlay

AutoAscend

LuckyMera (Hybrid AI Agents)

NeurIPS Conference

Hosted the prominent NetHack Challenge in 2021, with papers and talks on AI agents for NetHack. It remains a key venue for cutting-edge AI research involving reinforcement learning and symbolic AI on NetHack

Fundamental Background Papers

This is a background paper that also highlights problems with using LLMs in use cases like NetHack Environment.

arXiv:2403.00690v1 [cs.AI] 01 Mar 2024

Playing NetHack with LLMs: Potential & Limitations as Zero-Shot Agents

Dominik Jeurissen, Diego Perez-Liebana, Jeremy Gow

Queen Mary University of London
{d.jeurissen, diego.perez, jeremy.gow}@qmul.ac.uk

Duygu Çakmak, James Kwan

Creative Assembly
{duygu.cakmak, james.kwan}@creative-assembly.com

(RL and LLM+RL Agents)

Baseline Models

Proximal Policy Optimisation Model

Reinforcement Learning Agent

Fundamental Background Papers

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

Multimodal preprocessing

1. GLYPHS (Visual Information) [21 × 79 grid]

```
┌──────────────────────────────────┐
│ @ = Player    # = Wall           │
│ . = Floor     d = Dog            │
│ ) = Weapon    % = Food           │
└──────────────────────────────────┘
```

This is like a "screenshot" of the game world

2. STATS (Game Statistics) [26 numbers]

```
┌──────────────────────────────────┐
│ Health: 15/15   Level: 3         │
│ Strength: 18    Experience: 145  │
│ Position: (5,10) etc...          │
└──────────────────────────────────┘
```

3. MESSAGES (Text Information) [256 characters]

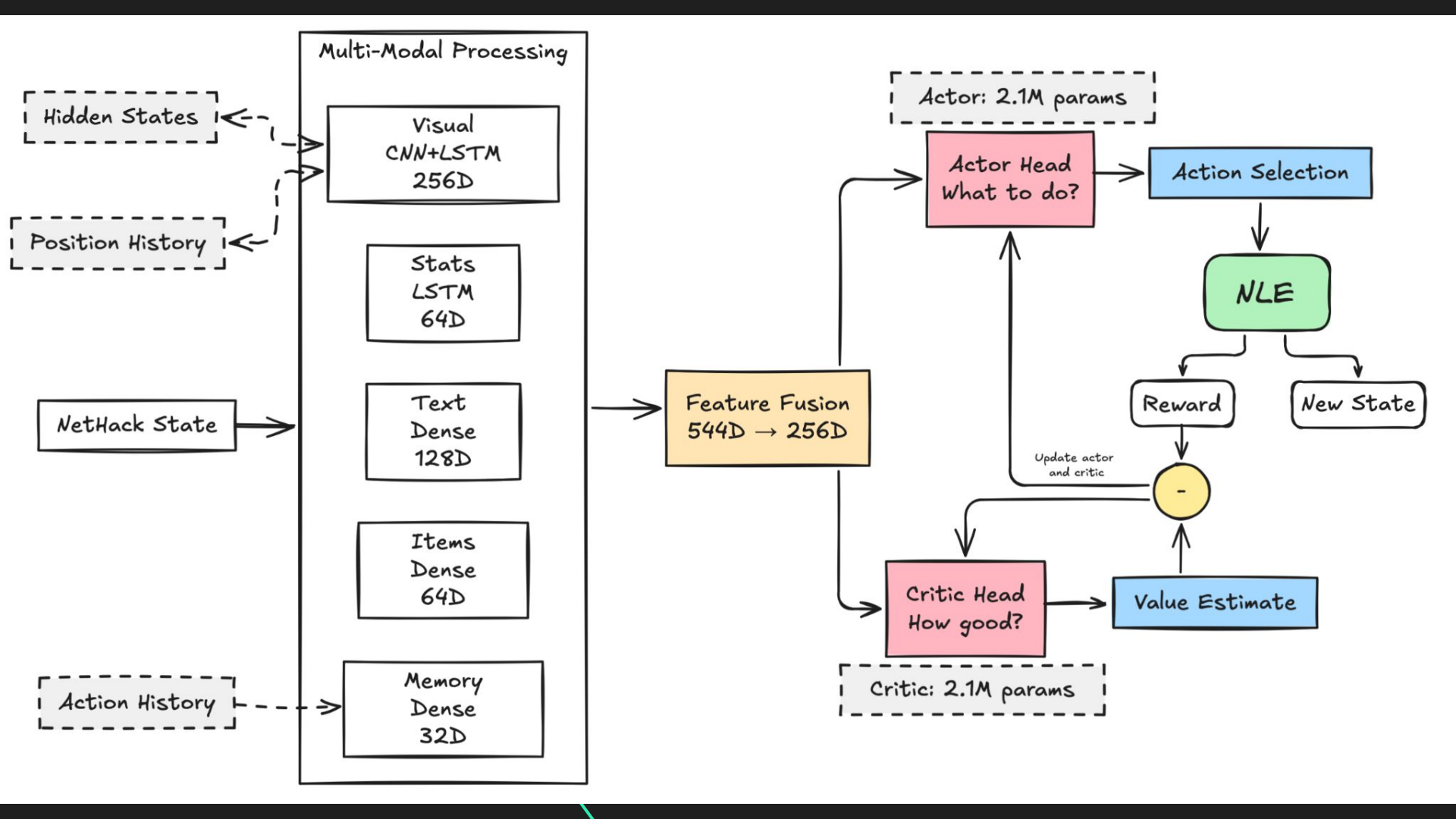
```
┌──────────────────────────────────┐
│ "You kill the goblin!"           │
│ "You feel hungry."               │
└──────────────────────────────────┘
```

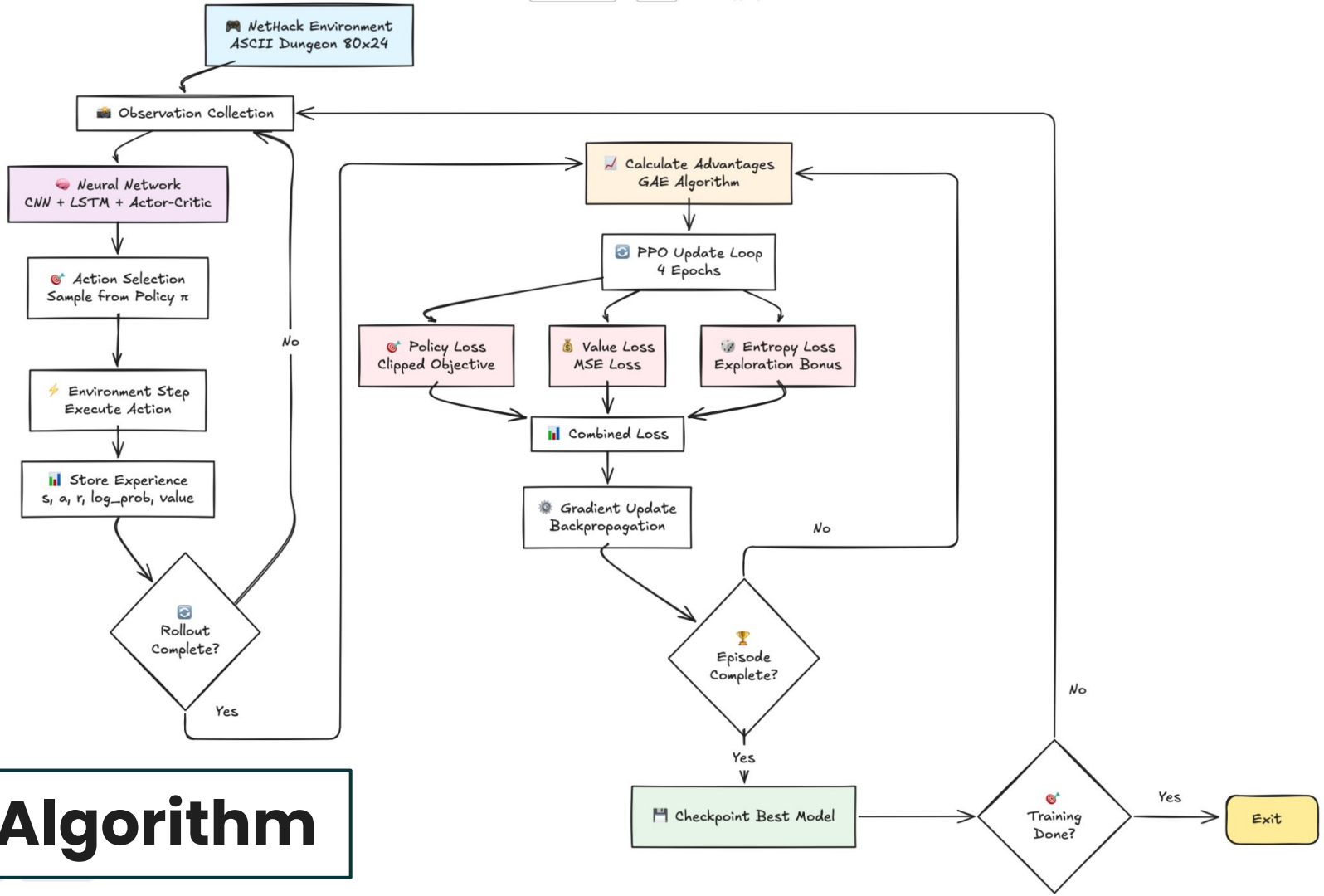
4. INVENTORY (Items You Have) [55 slots]

```
┌──────────────────────────────────┐
│ Sword: Yes    Potion: No         │
│ Shield: Yes   Food: Yes          │
└──────────────────────────────────┘
```

5. ACTION HISTORY (Recent Actions) [50 actions]

```
┌──────────────────────────────────┐
│ Last 50 moves: North, North,     │
│ Attack, South, Pickup, etc.      │
└──────────────────────────────────┘
```


Multi-Modal Processing
Hidden States
Position History
NetHack State
Action History
Visual
CNN+LSTM
256D
Stats
LSTM
64D
Text
Dense
128D
Items
Dense
64D
Memory
Dense
32D
Feature Fusion
544D → 256D
Actor: 2.1M params
Actor Head
What to do?
Action Selection
NLE
Reward
New State
Update actor
and critic
-
Critic Head
How good?
Value Estimate
Critic: 2.1M params



PPO Algorithm

Reward Shaping

- The rewards from NLE are sparse and delayed. This makes learning hard

Total Reward = Base Game Score + Shaped Components

```
class NetHackRewardShaper:
    def __init__(self):
        # Reward weights - these control the magnitude of each signal
        self.exploration_reward = 0.01      # Small reward for visiting new areas
        self.health_reward = 0.001          # Reward/penalty for health changes
        self.level_reward = 1.0             # Big reward for leveling up
        self.experience_reward = 0.0001     # Tiny reward for gaining XP
        self.death_penalty = -1.0           # Penalty for dying
        self.stuck_penalty = -0.01          # Penalty for not moving
        self.item_pickup_reward = 0.05      # Reward for collecting items
        self.monster_kill_reward = 0.1      # Reward for killing monsters
```

LLM Guided

Reinforcement Learning Agent

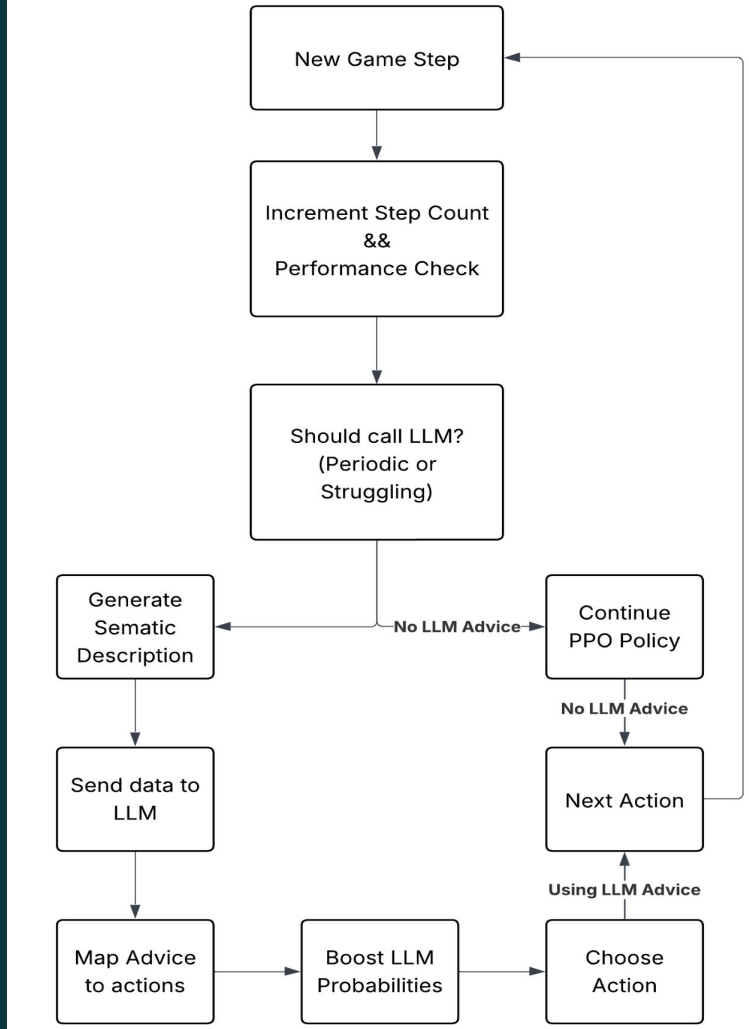
Fundamental Background Papers

VOYAGER: An Open-Ended Embodied Agent with Large Language Models

Guanzhi Wang[1 2]✉, **Yuqi Xie**[3], **Yunfan Jiang**[4*], **Ajay Mandlekar**[1*],
Chaowei Xiao[1 5], **Yuke Zhu**[1 3], **Linxi "Jim" Fan**[1†]✉, **Anima Anandkumar**[1 2†]
[1]NVIDIA, [2]Caltech, [3]UT Austin, [4]Stanford, [5]UW Madison
*Equal contribution †Equal advising ✉ Corresponding authors
https://voyager.minedojo.org

System Design

1. The system is built using multi-level checks and balances design.
2. A choice is made dynamically if the LLM is required based on change in reward in the last few steps.
3. But, for high-level planning's sake, the LLM is consulted every 50 steps regardless of performance.



Prompt

```
prompt = f"""You are an expert NetHack strategic advisor. {description}
RECENT PERFORMANCE: -
Average Reward: {performance['avg_reward']:.2f} -
Average Survival: {performance['avg_length']:.0f} steps

STRATEGIC ANALYSIS:
Based on the game state above, choose ONE primary strategy:

1. "explore" - No immediate threats, safe to move and search
2. "combat" - Monster nearby AND health good (>60%)
3. "retreat" - Monster nearby BUT health low (<40%)
4. "collect" - Items nearby and safe
5. "wait" - Critical health or need recovery

CRITICAL RULES:
- If threat distance 1-2 AND health <40%: Choose "retreat"
- If threat distance 1-2 AND health >60%: Choose "combat"
- If NO IMMEDIATE THREATS: Choose "explore" or "collect"
- If health critical: Choose "wait" or "retreat"

Respond with ONLY ONE WORD: explore, combat, retreat, collect, wait

Your strategic choice:"""
```

Semantic Description

```
NETHACK GAME STATE:
Status: Level 3, Health: 28/45 (low), XP: 234, Depth:
4, Gold: 87
Surroundings: CLOSEST THREAT: orc south (dist:1);
Other threats: kobold west (dist:3)
Recent Message: You are hit by the orc!
Inventory: Carrying 7 items (moderate load)
Recent Actions: move_south → kick → move_south → wait

STRATEGIC ANALYSIS:
Based on game state, choose ONE primary strategy:
1. "explore" - No immediate threats
2. "combat" - Monster nearby AND health good (>60%)
3. "retreat" - Monster nearby BUT health low (<40%)
4. "collect" - Items nearby and safe
5. "wait" - Critical health or need recovery

Your strategic choice:
```

Key Design Principles in AI Implementation

A few key design choices that were implemented in this module were arrived upon through extensive trial and error.

1. **Sparse LLM Calling:** Every 50 steps (automatically or when struggling) which represents about ~5% of the recorded timestamps to minimize the overload of querying the LLM
2. **Structured Prompts**: This allows us to provide clear prompts and decision rules to the LLM. Even if a LLM without knowledge of the NetHack Environment is picked, it can respond purely on the basis of the prompt's context.
3. **Low Temperature for the LLM**: 0.2 for consistent and deterministic strategy decisions.
4. **Soft Hints**: A 20% probability boost is added to the LLM advice, it avoids the use of hard constraints on the choice of actions.
5. **Async API**: It employs non-blocking LLM calls which avoids blocking the system.

Math behind the LLM's Choice Inclusion

$$\pi_\theta(a \mid s, h) \propto \exp\left(\text{logits}_\theta(s) + \lambda \cdot W_g h\right) \tag{4.2.3.1}$$

Equation 4.2.3.1: LLM-Guided Policy with Additive Bias
Where:

- $\pi_\theta(a \mid s, h)$ is the guided policy distribution over actions given state s and hints h.
- a is the action taken from the action space A.
- s is the current state of the environment.
- θ represents the learnable parameters of the policy network.
- $\text{logits}_\theta(s) \in \mathbb{R}^{|A|}$ are the base policy logits generated by the PPO actor.
- $h \in \mathbb{R}^{|A|}$ are the LLM-derived action hints (e.g., 0.2 for suggested actions, 0 otherwise).
- $W_g \in \mathbb{R}^{|A| \times |A|}$ is the trainable guidance transformation layer (implemented as `llm_guidance_fc`).
- λ is a small scalar guidance weight controlling the influence of the LLM (typically $0 < \lambda \leq 0.1$).
- $|A|$ denotes the cardinality (size) of the action space.

The additive bias mechanism can be expressed as:

$$\text{guided\_logits} = \text{base\_logits} + \lambda \cdot W_g h$$

and the resulting policy distribution is:

$$\pi_\theta(a \mid s, h) = \frac{\exp(\text{guided\_logits}_a)}{\sum_{a'} \exp(\text{guided\_logits}_{a'})} \tag{4.2.3.2}$$

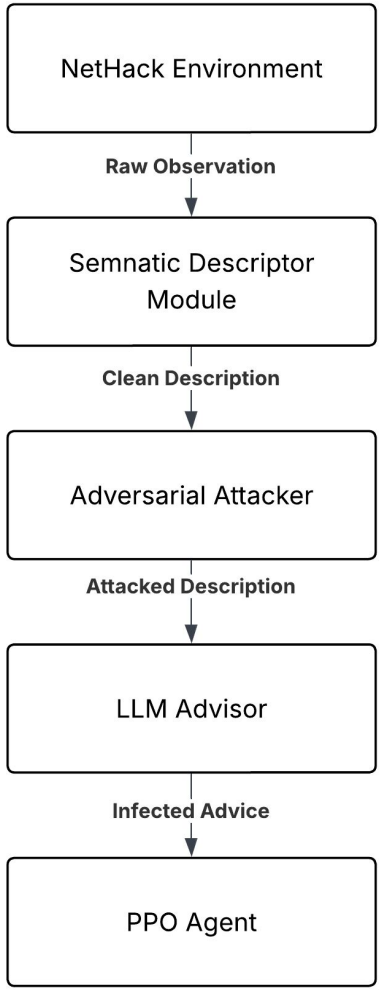
Equation 4.2.3.2: Softmax Normalization of Guided Policy
where a' ranges over all possible actions in the action space A.

LLM+RL Under Attack

System Design

This is the design for the ***Adversarial Attacker*** that has been developed specifically for creating infecting Semantic Descriptions that are passed to the LLM.

1. **Insertion** Point: Attacks occur **AFTER** semantic description generation but **BEFORE** the LLM gets the incoming data for processing.
2. **Transparency**: The agent receives attack input without awareness of manipulation. This ensures that the LLM does not bias itself against the data if it is marked as breached.
3. **Controllability**: Attack strength parameter (0.0-1.0) controls the intensity of the breach in the semantic description.
4. **Measurability**: Real-time monitoring tracks performance degradation.



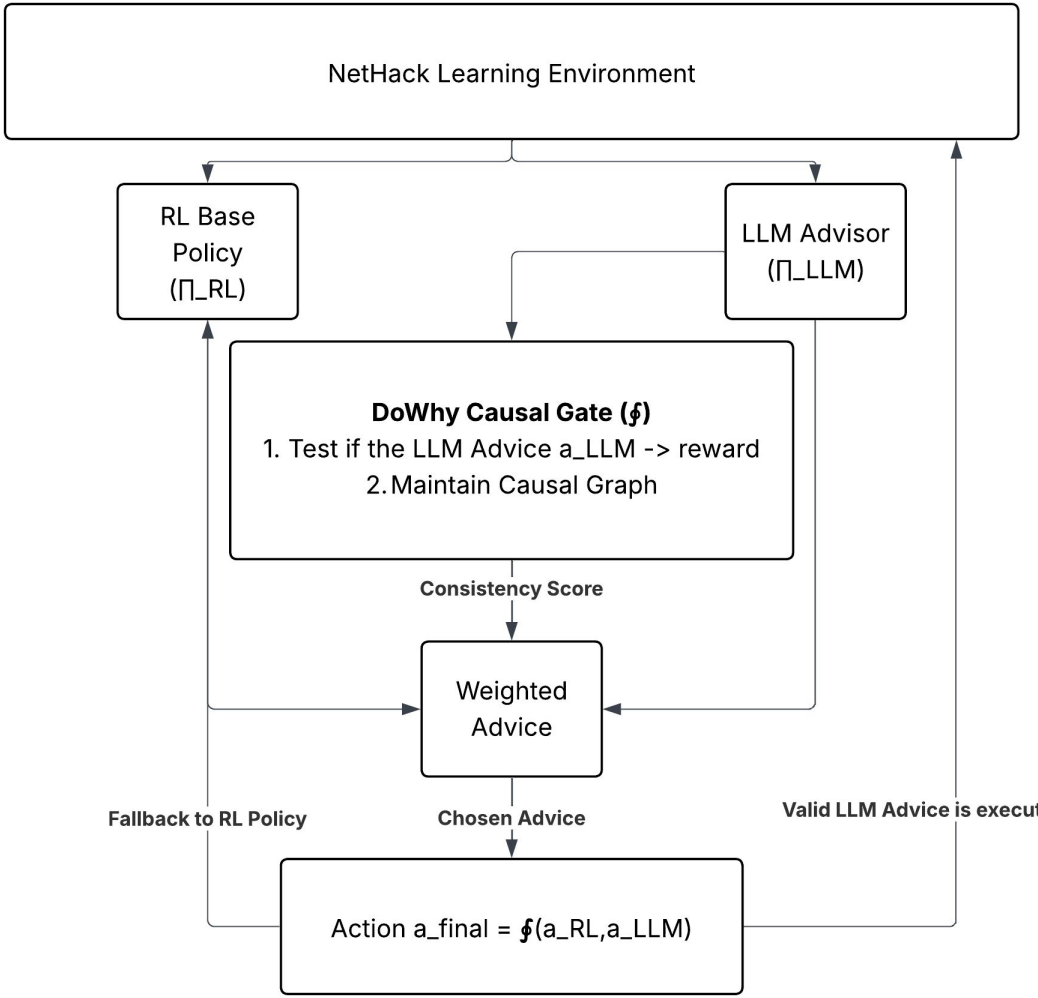
Types of Attacks Designed

- None(Baseline)
- Noise Injection
- State Inversion
- Random Corruption
- Misleading Context
- Contradictory Information
- Critical Information Removal
- Strategic Poisoning

Attack Strength by Type

Configuration Name	Strength	Expected Impact
Baseline	0.0	None (Control)
Noise (Mild)	0.3	Low-Moderate
Noise (Severe)	0.8	High
Inversion (Mild)	0.3	Moderate
Inversion (Severe)	0.8	Very High
Misleading	0.7	High
Poisoning	0.8	Very High
Info Removal	0.6	High

Causal Protection



Design for Causal Gate

The Causal Gate was constructed using the DoWhy Library.

The causal gate uses “Effect Estimators” using the Doubly Robust Estimator technique.

It measures effect of changing policy on cumulative reward.

Why Traditional Defenses fail ?

1. **Input Sanitisation**: Cannot distinguish between semantically valid but strategically harmful advice.

2. **Adversarial Training**: Requires knowing the attack patterns in advance.

3. **Ensemble Methods**: All LLMs may be fooled by well-crafted semantic manipulations.

4. **Output Filtering**: Cannot detect subtle strategic errors without understanding causality.

Doubly Robust Estimators

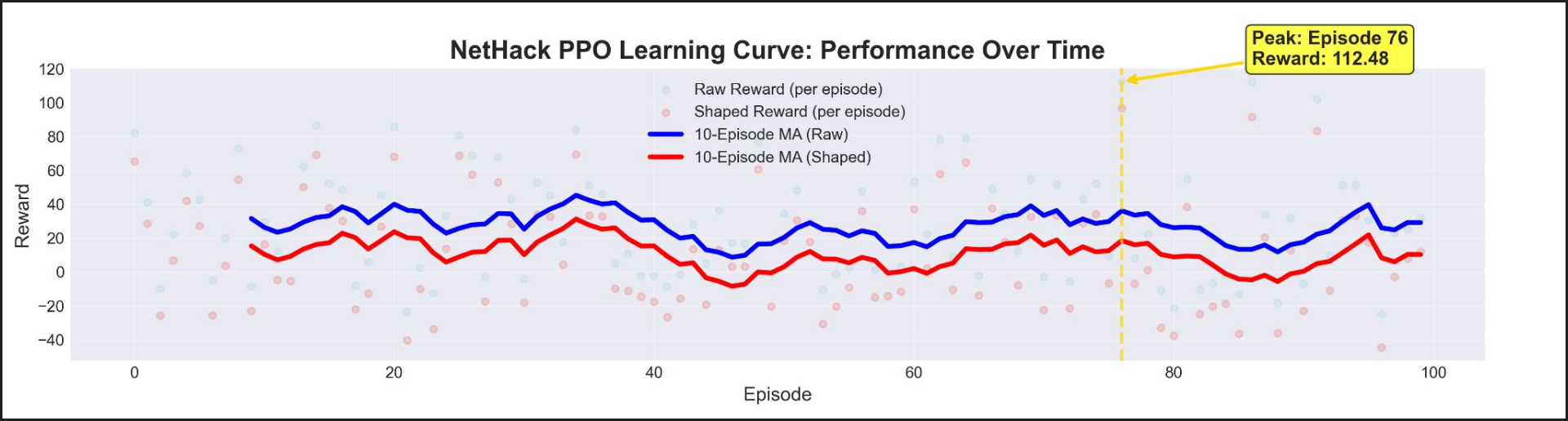
Doubly robust estimators act as uniquely advantageous options in this case as they are consistent estimators as long as ***EITHER*** the propensity score model or the outcome models are correct.

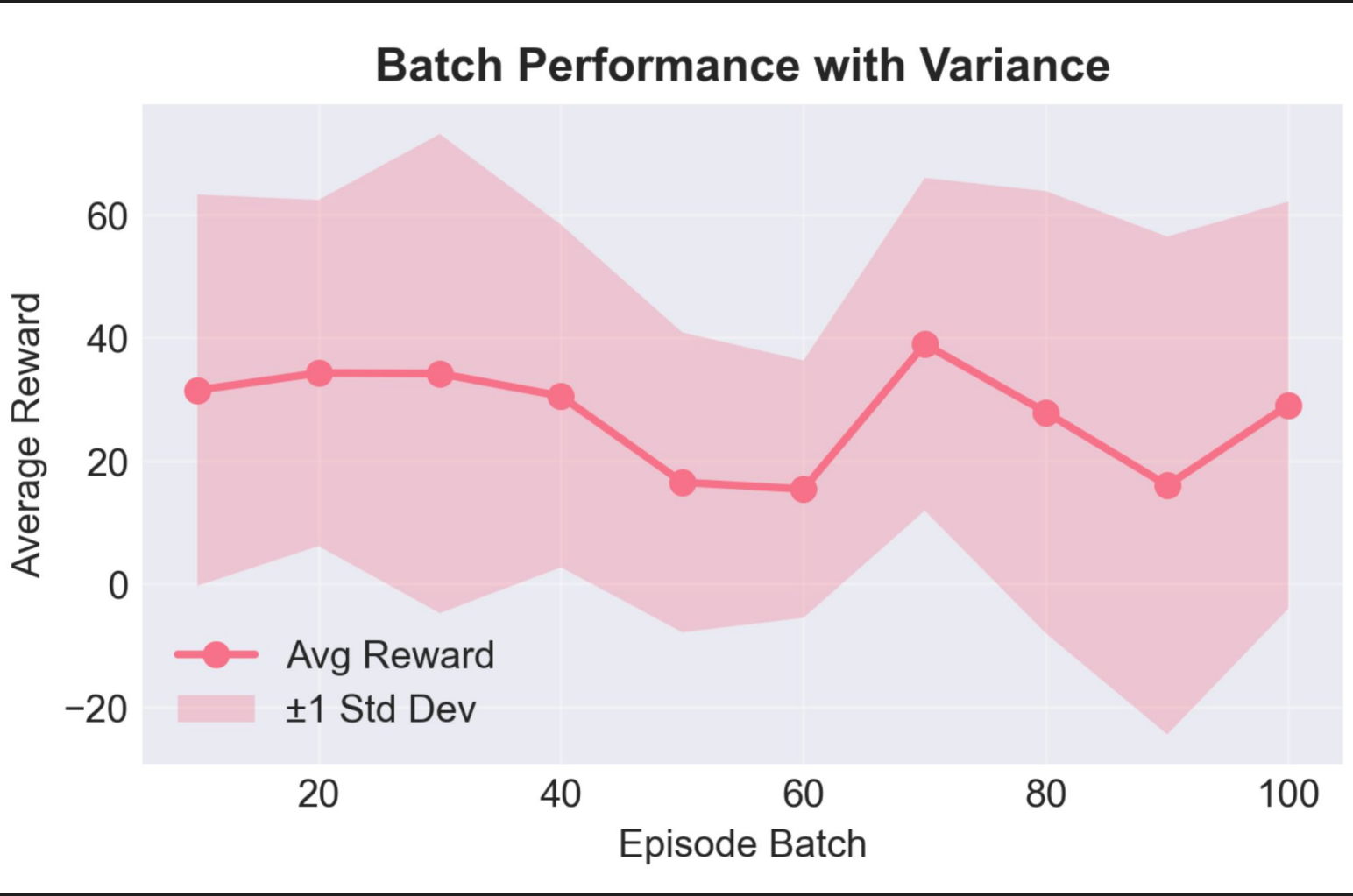
This provides us with protection against model misspecification

RESULTS

Proximal Policy Optimisation Model

Reinforcement Learning Agent

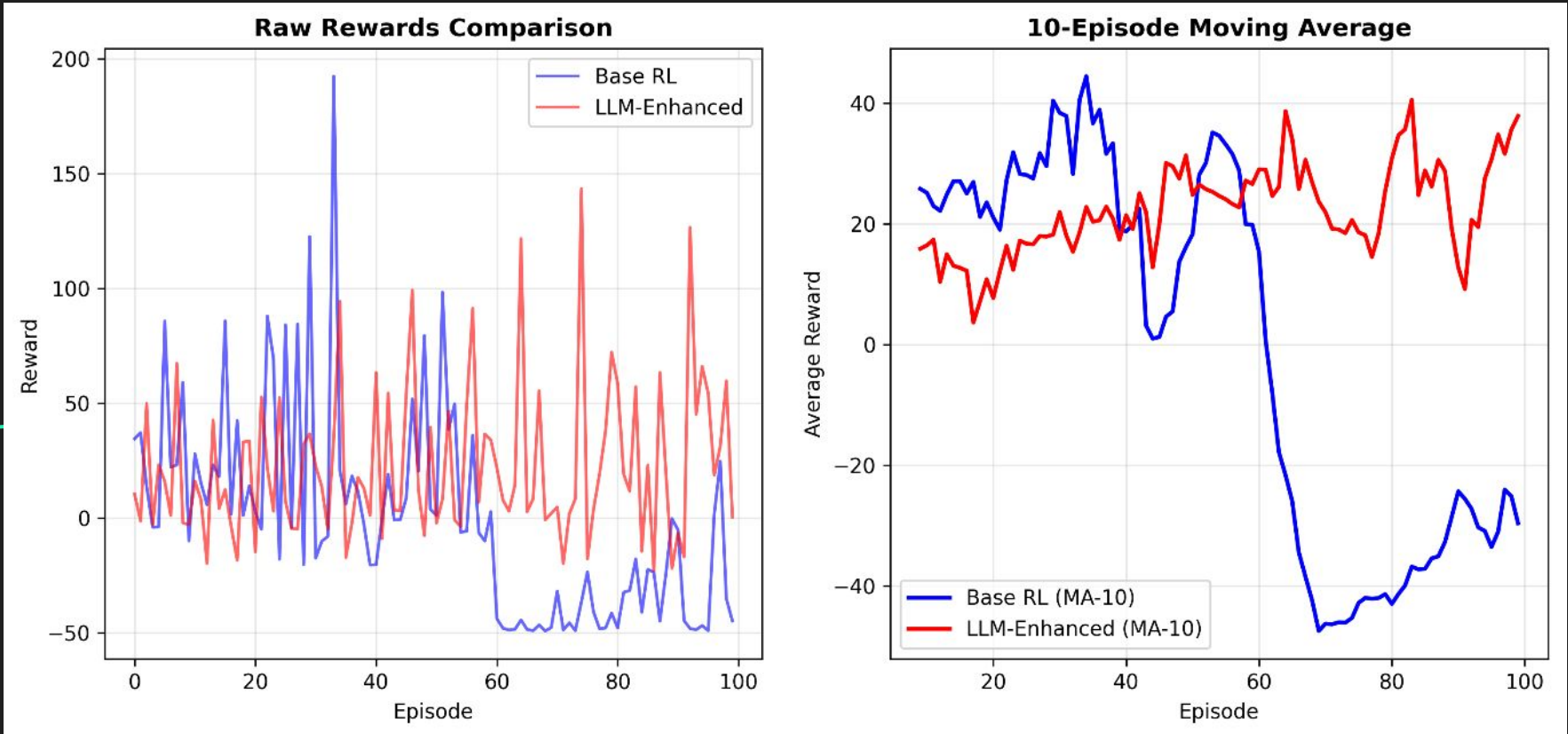

NetHack PPO Learning Curve: Performance Over Time
Peak: Episode 76
Reward: 112.48
Raw Reward (per episode)
Shaped Reward (per episode)
10-Episode MA (Raw)
10-Episode MA (Shaped)
Reward
Episode
120
100
80
60
40
20
0
−20
−40
0
20
40
60
80
100


Batch Performance with Variance
Average Reward
60
40
20
0
−20
Avg Reward
±1 Std Dev
20
40
60
80
100
Episode Batch

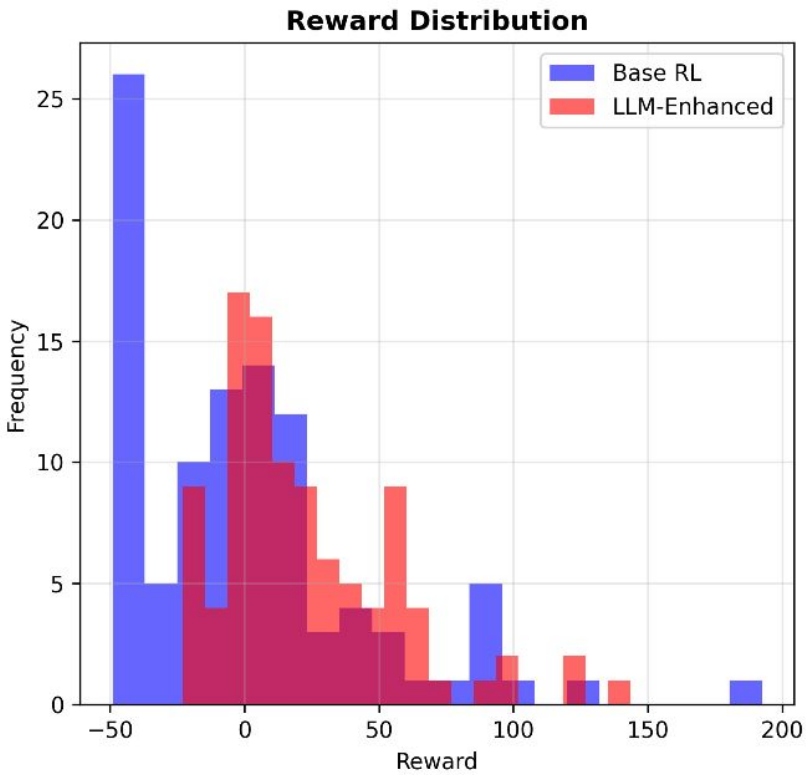
LLM Guided

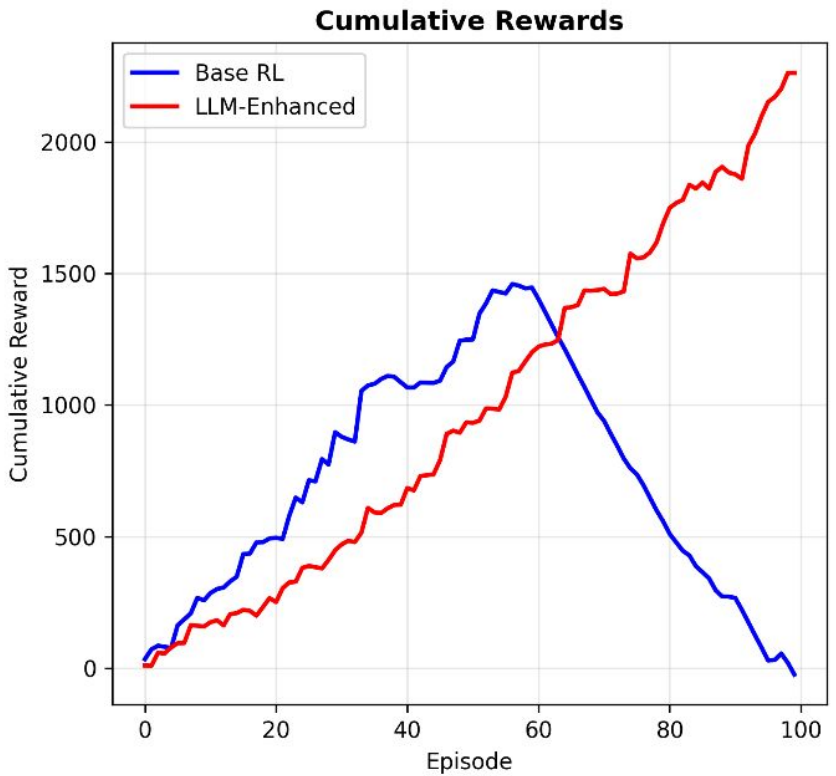
Reinforcement Learning Agent

Comparison of LLM+RL against BaseRL



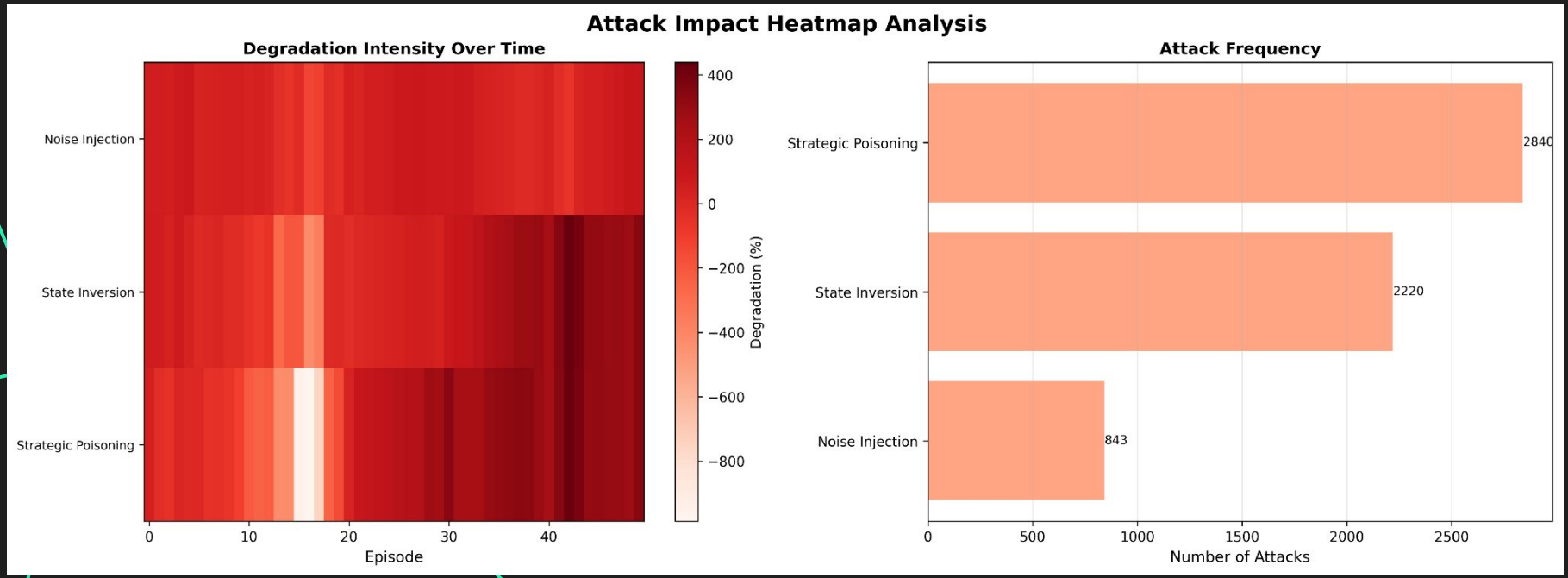
Comparison of LLM+RL against BaseRL



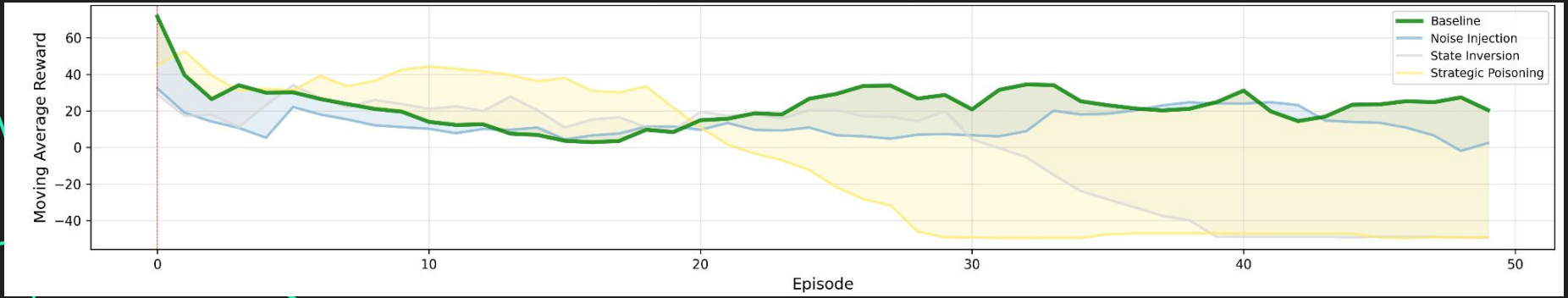


LLM Guided
Reinforcement Learning Agent
Under Attack

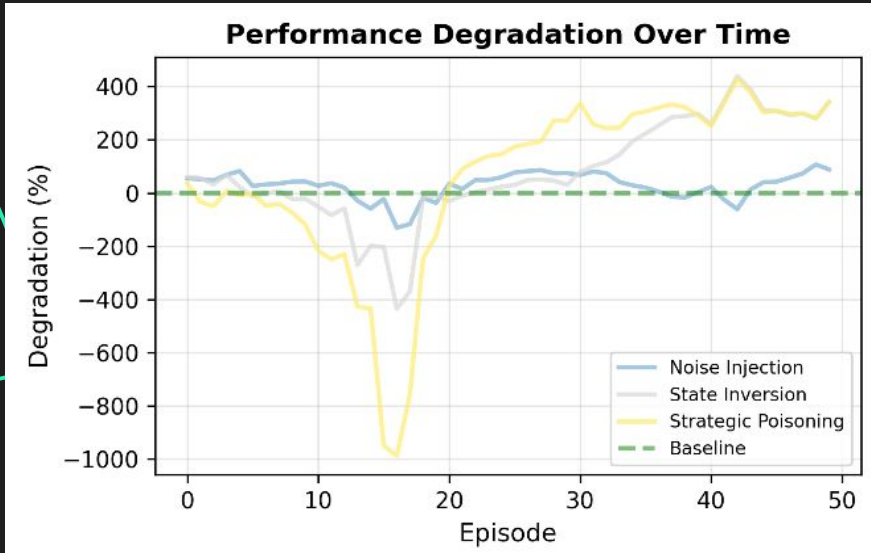
LLM+RL Under Attack

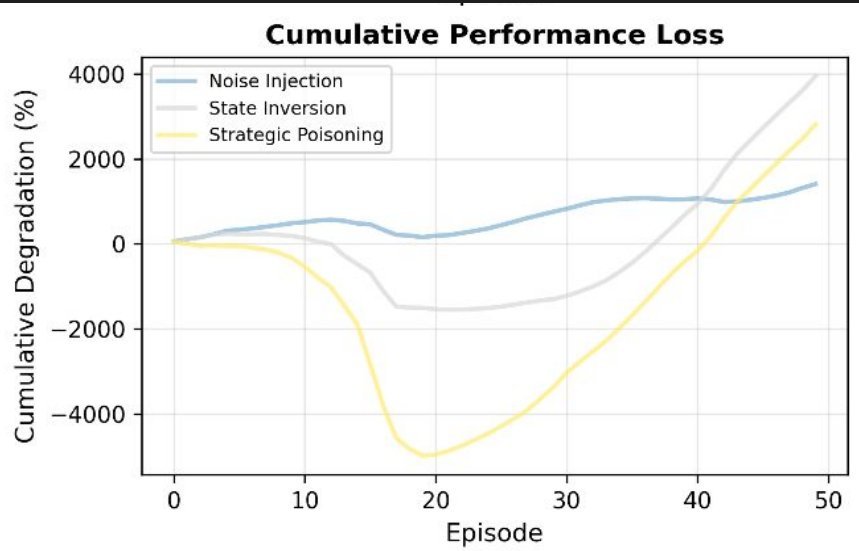


LLM+RL Under Attack



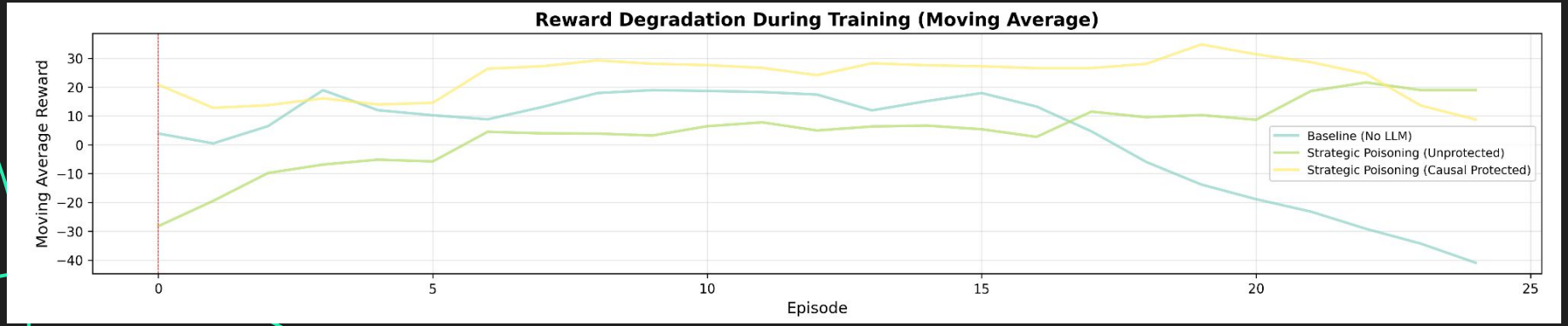
LLM+RL Under Attack



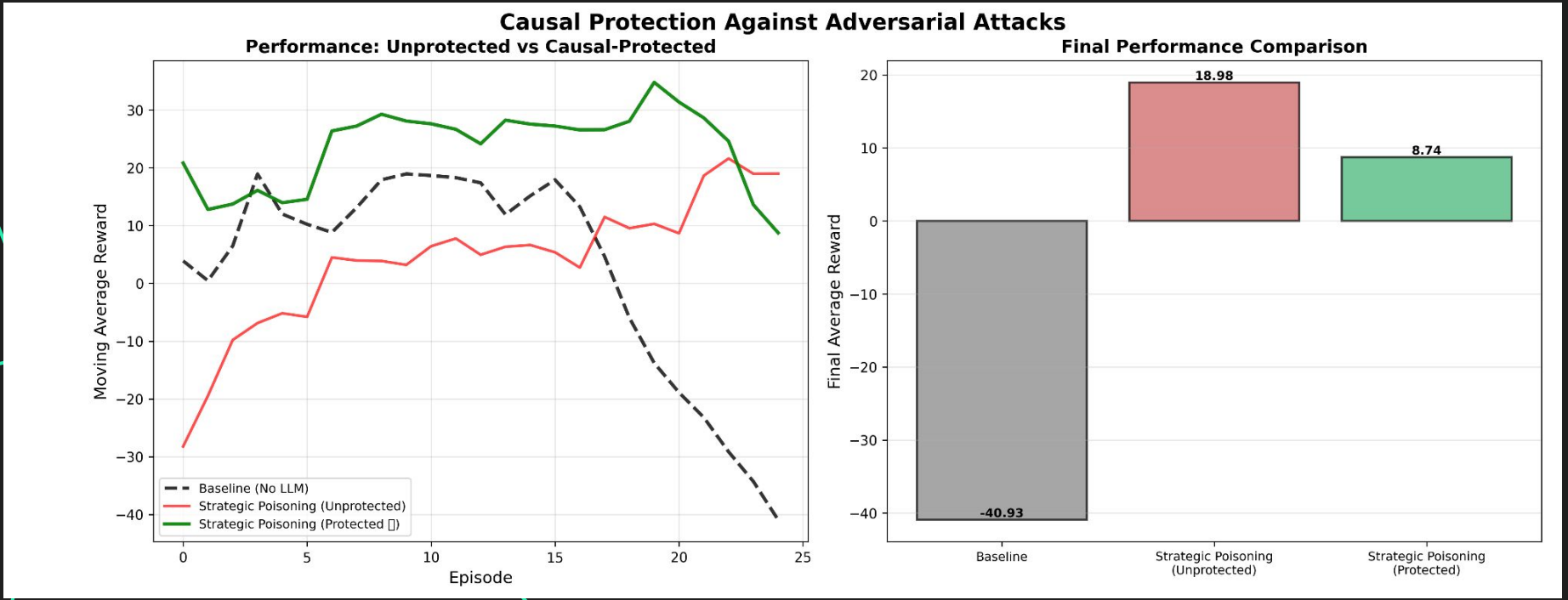


Causally Protected LLM Guided
Reinforcement Learning Agent

Causally Protected LLM+RL Under Attack



Causally Protected LLM+RL Under Attack



Thank you!