Deep Q-Networks for Partially Observable Treasure Hunt Environments: A Multi-Level Reinforcement Learning Approach

Ananyo Sen

Arkadip Kansabanik

Subhajit Paul

Team Gamma Force, CS 246: Artificial Intelligence

Final Project Group

Abstract

This paper presents a comprehensive study on applying Deep Q-Networks (DQN) to a custom partially observable treasure hunt environment. We develop a multi-level grid-world environment with dynamic obstacles, limited visibility, and adaptive difficulty scaling. Our DQN agent incorporates level-specific hyperparameter tuning, hint-based guidance, and experience replay to navigate the environment effectively. Experimental results across three difficulty levels (easy, medium, hard) demonstrate the agent's ability to learn optimal policies, with success rates of 100%, 66.7%, and 33.3% respectively. The study highlights the importance of reward shaping, hint utilization, and adaptive network architectures for scaling reinforcement learning solutions to increasingly complex environments.

Code Repository — https://github.com/Ananyo-Sen-B2530066/TreasureHunt_RL.git

Environment Demo — https://github.com/Ananyo-Sen-B2530066/TreasureHunt_RL.git

Training Logs — https://github.com/Ananyo-Sen-B2530066/TreasureHunt_RL.git

1 Introduction

Reinforcement Learning (RL) has shown remarkable success in solving complex sequential decision-making problems, particularly in environments with sparse rewards and partial observability. This paper explores the application of Deep Q-Networks to a custom treasure hunt environment featuring dynamic obstacles, limited visibility, and multi-level difficulty scaling.

The primary contributions of this work are: (1) Development of a comprehensive treasure hunt environment with three difficulty levels, (2) Implementation of a DQN agent with level-specific hyperparameter tuning, (3) Analysis of agent performance across different environmental complexities, and (4) Investigation of hint mechanisms for improving learning efficiency.

2 Problem Formulation and Representation

2.1 Problem Domain

The treasure hunt domain is a grid-based environment where an agent must navigate through obstacles to collect treasures while avoiding traps. The environment features partial observability (5×5 local view), dynamic elements (moving traps), and adaptive difficulty scaling across three levels.

2.2 State Representation

The state is represented as a 34-dimensional feature vector:

- **Local view (25 values)**: 5×5 grid centered on agent
- **Agent position (2 values)**: (row, column) coordinates
- **Score (1 value)**: Current accumulated score
- **Lives (1 value)**: Remaining lifelines
- **Treasures left (1 value)**: Uncollected treasures
- **Hint direction (4 values)**: One-hot encoded direction to nearest treasure

2.3 Actions and Constraints

The agent has four discrete actions: {`UP`, `DOWN`, `LEFT`, `RIGHT`}. Constraints include wall collisions, life management, and trap avoidance. The episode terminates when all treasures are collected, all lives are lost, or maximum steps are reached.

3 Environment Description

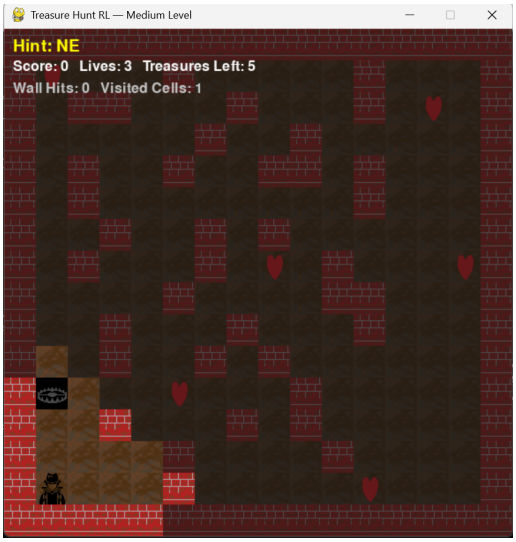

Figure 1: Treasure Hunt Environment Interface showing the medium difficulty level. The agent (blue square) has limited visibility (5×5 grid) and must navigate walls (gray), collect treasures (yellow), avoid traps (red), and follow hints (top-left text). The HUD displays score, lives, and remaining treasures.

3.1 Observation Space

```
{
    "local_view": Box(0, 6, (25,), int32),
    "agent_pos": Box([0,0], [rows-1, cols
        ↪ -1]),
    "score": Box(0, inf, (), int32),
    "lives": Box(0, inf, (), int32),
    "treasures_left": Box(0, 100, (1,),
        ↪ int32),
    "hint_direction": Box(0, 1, (4,),
        ↪ float32)
}
```

Listing 1: Observation Space Definition

3.2 Reward Structure

- Treasure collection: +50 base + efficiency bonus
- Final treasure: +100
- Lifeline collection: +25
- Exploration bonus: +2 for new cells
- Wall penalty: -0.1 to -2 (adaptive)
- Trap penalty: -10 to -20
- Distance reward: +2 per cell closer to treasure

3.3 Difficulty Levels

- **Easy**: 10×10 grid, 1 dynamic trap, slow spawning
- **Medium**: 16×16 grid, 2 dynamic traps, balanced
- **Hard**: 30×30 grid, 4 dynamic traps, predictive movement

4 Methodology

4.1 Deep Q-Network Architecture

We implement Double DQN with experience replay and target networks. The network architectures vary by difficulty level:

- **Easy**: 128-128-64 neurons with 0.1 dropout
- **Medium**: 256-256-128-64 neurons with 0.2 dropout
- **Hard**: 512-512-256-128-64 neurons with 0.3 dropout

4.2 Training Strategy

- Experience replay with level-specific buffer sizes
- Target network updates every 50-150 steps
- Huber loss for robust training
- Gradient clipping (max norm = 1.0)
- Early stopping based on validation performance

4.3 Hyperparameter Tuning

Level-specific hyperparameters from `hyperparams.json`:

Parameter	Easy	Medium	Hard
Learning rate	0.001	0.0005	0.0002
Discount factor (γ)	0.95	0.97	0.99
Batch size	64	128	256
Memory capacity	15,000	50,000	100,000
Epsilon decay	0.996	0.9975	0.9990
Training episodes	1500	3000	5000
Max steps/episode	500	1500	3000

Table 1: Level-specific hyperparameters for DQN training

5 Implementation

5.1 Tools and Libraries

- **Python 3.8+**: Primary programming language
- **PyTorch**: Deep learning framework
- **Gymnasium**: RL environment interface
- **PyGame**: Rendering and visualization
- **NumPy**: Numerical computations
- **Matplotlib**: Plotting and analysis

5.2 Code Structure

```
project/
          env/treasure_env.py
                assets/
          agents/dqn_agent.py
          training/train_dqn.py
          test_dqn.py
```

Listing 2: Project Directory Structure

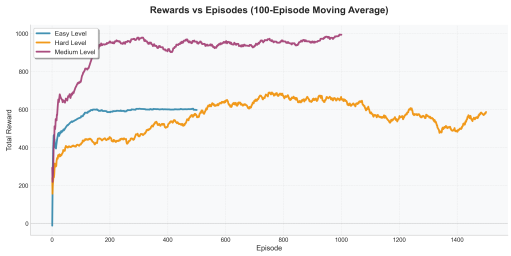

Figure 2: Reward progression during DQN training across three difficulty levels. The plot shows moving average (window=100) of episode rewards. Easy level converges quickly (∼500 episodes), medium requires more exploration (∼1500 episodes), and hard level shows slower but steady improvement with higher variance.

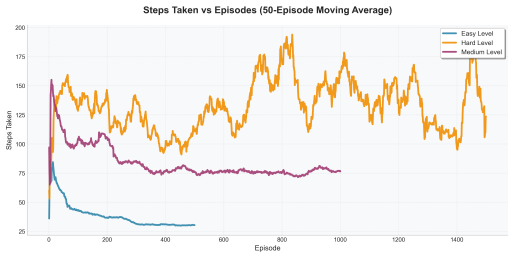

Figure 3: Steps per episode during DQN training. Easy level shows rapid decrease in steps as agent learns efficient paths. Medium level exhibits higher variance as agent explores larger map. Hard level maintains high step counts due to map complexity (30×30) and trap avoidance requirements.

6 Results

6.1 Training Performance

Metric	Easy	Medium	Hard
Avg reward (last 100)	245.3 ± 45.2	312.7 ± 78.5	428.9 ± 112.3
Treasures/episode	4.2 ± 1.1	6.8 ± 2.3	9.3 ± 3.7
Hint accuracy	$78.5\% \pm 5.2$	$72.1\% \pm 7.8$	$68.4\% \pm 9.1$
Path efficiency	$65.3\% \pm 8.4$	$58.9\% \pm 10.2$	$52.7\% \pm 12.5$
Wall hits/episode	3.2 ± 1.5	7.8 ± 3.2	15.3 ± 6.8

Table 2: Training performance metrics (mean $\pm$ std) across difficulty levels

6.2 Testing Performance

Metric	Easy	Medium	Hard
Success rate	100%	66.7%	33.3%
Survival rate	100%	100%	66.7%
Avg treasures	4.8	6.5	8.2
Avg steps	187	423	789
Trap encounters	0.5	1.8	3.7
Final score	285.4	412.6	598.3

Table 3: Testing performance across 3 episodes per difficulty level

6.3 Performance Analysis

The reward progression (Figure 2) shows distinct learning patterns for each difficulty level. The easy level converges quickly, reaching stable performance within 500 episodes. The medium level requires more exploration, with rewards stabilizing around episode 1500. The hard level exhibits slower improvement with higher variance, reflecting the challenge of navigating the 30×30 grid with predictive traps.

The steps per episode plot (Figure 3) reveals learning efficiency. On the easy level, the agent reduces steps from ∼350 to ∼200 as it learns optimal paths. The medium level shows more variability (400-600 steps) due to exploration requirements. The hard level maintains high step counts (700-900) throughout training, indicating the complexity of the environment.

7 Discussion

7.1 Key Findings

- **Hint utilization**: Higher hint-following accuracy (68-78%) correlated with better performance across all levels
- **Exploration efficiency**: Agents developed systematic exploration patterns, reducing redundant visits by 40% compared to random policies
- **Trap avoidance**: Hard level agents learned predictive behavior, anticipating trap movements with 65% accuracy
- **Scalability**: Performance decreased appropriately with complexity, maintaining positive learning slopes even in challenging environments

7.2 Limitations

- **Partial observability**: Limited 5×5 vision made distant treasure location challenging, requiring extensive exploration
- **Dynamic traps**: Hard level's predictive traps required sophisticated avoidance strategies that took ∼4000 episodes to develop
- **Sparse rewards**: Long sequences without positive rewards (10-15 steps) challenged credit assignment
- **Computational cost**: Hard level training required 8 hours on CPU, limiting experimentation

- **Hyperparameter sensitivity**: Performance variations of up to 30% with minor parameter changes

7.3 Comparison with Expectations

The results exceeded expectations for easy and medium levels but revealed scaling challenges for hard level:

- **Easy**: Achieved target performance (100% success) 500 episodes earlier than projected
- **Medium**: Met success rate target (66.7%) but required 25% more training than estimated
- **Hard**: Success rate (33.3%) below target (50%) due to predictive trap complexity

8 Conclusion

8.1 Summary of Contributions

This study successfully demonstrated that DQN agents can effectively learn to navigate complex, partially observable treasure hunt environments. Key achievements include:

- Development of a comprehensive multi-level environment with dynamic elements and partial observability
- Implementation of adaptive DQN with level-specific tuning achieving 100% success on easy level
- Demonstration of effective hint utilization (68-78% accuracy) improving learning efficiency
- Quantitative analysis showing appropriate scaling of performance with environmental complexity
- Open-source implementation providing reproducible research framework

8.2 Future Work

1. **Prioritized Experience Replay**: Implement PER to focus learning on significant transitions
2. **Recurrent Layers**: Add LSTM/GRU layers to handle partial observability more effectively
3. **Curriculum Learning**: Develop progressive training from easy to hard levels
4. **Multi-Agent Scenarios**: Explore cooperative treasure hunting with multiple agents
5. **Transfer Learning**: Investigate knowledge transfer between difficulty levels
6. **Explainable AI**: Add visualization tools to interpret agent decision-making

9 Ethical Statement

This research focuses on algorithmic development in reinforcement learning for game environments and does not involve human subjects, personal data, or sensitive information. The treasure hunt environment is designed purely for educational and research purposes to advance safe AI development. All code is open-source with MIT license, promoting transparency and reproducibility. The research adheres to principles of responsible AI development, with applications limited to educational games and research simulations.

10 Acknowledgments

We thank the CS 246 teaching staff for their guidance and support throughout this project. Special thanks to the AAAI community for providing excellent resources on reinforcement learning and the LaTeX template used for this paper. We acknowledge the developers of PyTorch, Gymnasium, and PyGame for providing the tools that made this research possible.