# **Reinforcement Learning Summative Assignment Report**

**Student Name:** Esther MBANZABIGWI

**Video Recording:** [Link to your Video 3 minutes max, Camera On, Share the entire Screen]  
**GitHub Repository:** [Link to your repository]

## **1. Project Overview**

This project implements and compares two reinforcement learning approaches — Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) — within a custom environment that simulates guided therapy session planning for mental health support. Inspired by an AI-powered mobile therapy app tailored for Rwandan users, the environment challenges an agent to navigate a 5x5 therapy grid to recommend appropriate actions (e.g., journaling, CBT, crisis response). The goal is to optimize the sequence of therapeutic steps based on simulated user states. By training and evaluating both models in this mission-based setting, the project aims to identify the most effective strategy for real-time, adaptive mental health guidance.

**2. Environment Description**

### **2.1 Agent(s)**

The agent in this environment represents an **AI-powered therapy assistant** designed to guide users through personalized mental health interventions. It operates within a 5x5 grid world, where each cell corresponds to a therapeutic action or state (e.g., journaling, CBT, chatbot interaction, crisis response). The agent’s primary capability is to move across the grid and select the most appropriate therapy steps to reach a target mental health intervention. It can take discrete actions — up, down, left, or right — and receives feedback through rewards. However, its limitations include a lack of memory (no long-term state tracking) and partial observability, as it does not inherently know the user's emotional context unless learned through rewards. The goal is for the agent to learn an optimal path that maximizes well-being while minimizing unnecessary actions.

### **2.2 Action Space**

The agent operates in a **discrete action space**, consisting of **four movement-based actions** that enable it to navigate the 5x5 therapy grid. These actions allow the agent to explore different therapy modules by moving across grid cells. The full list of available actions includes:

* 0: Move **Up** 1:Move **Down**
* 3: Move **Right** 2: Move **Left**

Each action transitions the agent to a neighboring cell, unless it is at the boundary of the grid, in which case the action results in no movement (penalized with a negative reward). This discrete control allows the agent to learn efficient navigation strategies while balancing exploration and optimal decision paths.

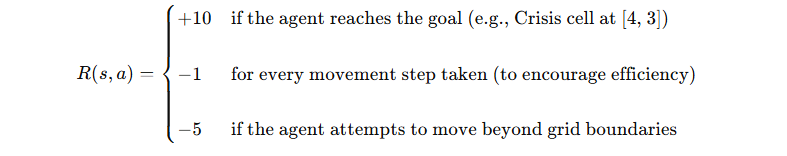
### **2.3 State Space**

The state space in this environment is represented by the agent’s **current position on the 5x5 grid**, encoded as a pair of integers (x, y). This means each state is a unique coordinate indicating the agent’s location relative to therapy activities such as Journaling, CBT, Crisis Response, etc. The agent does not have direct access to the labels or types of the cells — it learns the relevance of each position based on rewards received during exploration. The state is fully observable and encoded as a discrete tuple that is fed into the neural network for decision-making. This simple yet effective state representation helps the agent associate positions with long-term reward outcomes in order to plan optimal paths toward mental health goals.

### **2.4 Reward Structure**

The reward function in this environment is designed to encourage the agent to reach the target therapy intervention efficiently while avoiding unnecessary movements. The goal is to simulate a system that rewards meaningful progress in a mental health session.

The reward function is defined as:



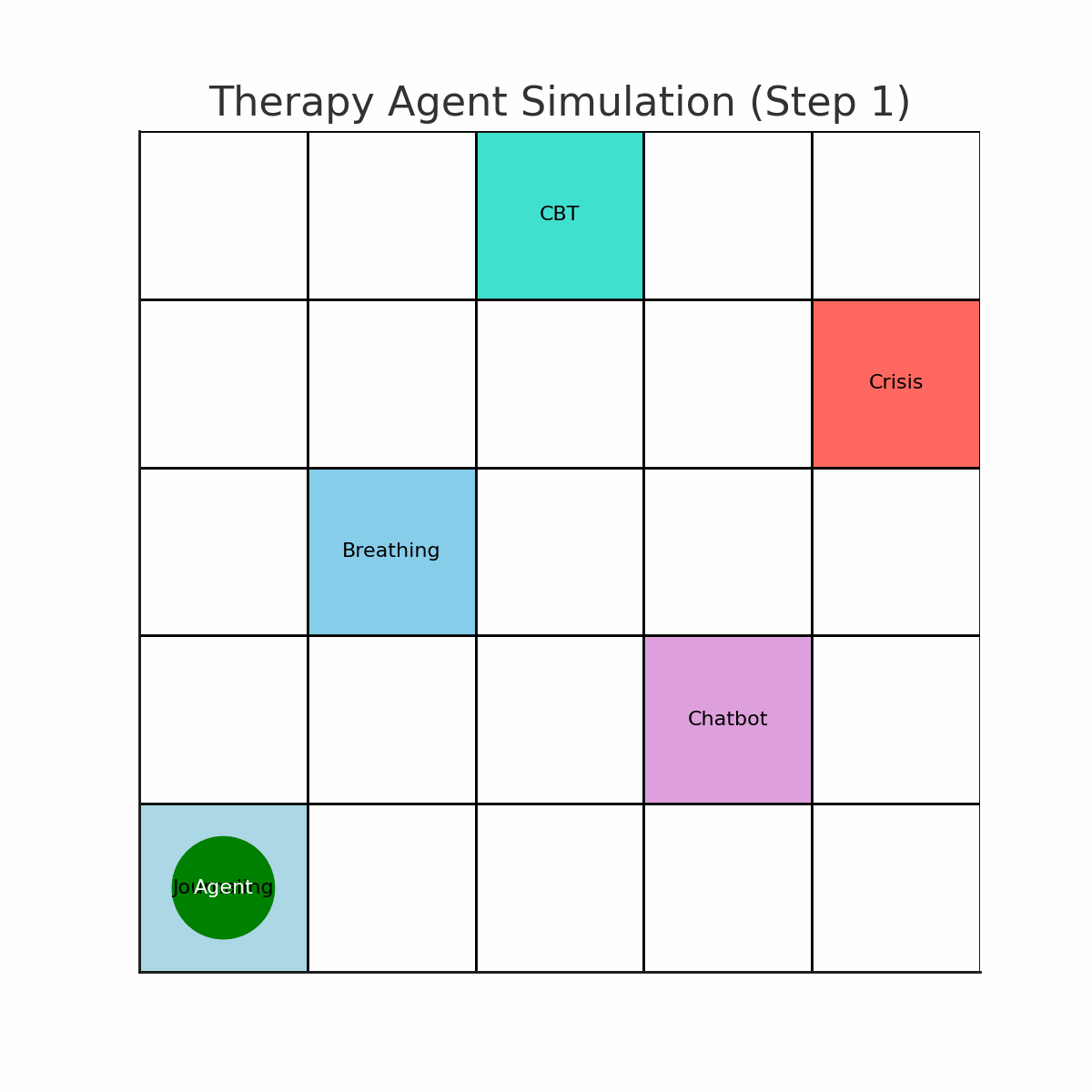
### **Behavior Outcomes:**

* **Rewarded** for reaching the correct therapeutic module (goal)
* **Penalized** for inefficient wandering or bouncing into walls
* **Discouraged** from taking excessive steps without progress

This structure guides the agent to prioritize the shortest and most effective route to the goal therapy while penalizing hesitation or repetitive, ineffective behavior.

### **2.5 Environment Visualization**

The figure below shows a colored 5x5 grid representing the custom therapy environment. Each cell corresponds to a unique therapy intervention, color-coded for clarity:



* 🩵 **Journaling** – Light Blue
* 💨 **Breathing** – Sky Blue
* 💬 **Chatbot** – Lavender
* 🧠 **CBT** – Teal
* 🚨 **Crisis** – Red

A green circle represents the **AI therapy agent**, which starts at a random location and learns to navigate toward the goal (Crisis cell at [4, 3]). The agent uses discrete movements (up, down, left, right) and receives feedback through rewards to learn an optimal decision path. This visualization helps illustrate the agent’s environment and decision-making space before any training occurs. A dynamic version of this environment is also provided as a PyOpenGL simulation.

## **3. Implemented Methods**

### **3.1 Deep Q-Network (DQN)**

### **Architecture & Features:**

* **Network**: 2 hidden layers (default from SB3), each with fully connected neurons and ReLU activations
* **Experience Replay**: Implemented with a buffer size of 10,000 to break correlation between experiences
* **Target Network**: Updated every 500 steps to stabilize training and prevent oscillations in Q-value estimates
* **Exploration Strategy**: ε-greedy with decay; starts with high randomness and gradually becomes more deterministic
* **Loss Function**: Mean-squared error between current Q-values and target Q-values
* **Environment**: Used the custom TherapyEnv with discrete state and action spaces

### **Modifications:**

No major architectural changes were made beyond hyperparameter tuning. However, exploration parameters (like final ε = 0.02) and the update frequency of the target network were fine-tuned to improve learning stability.

### **3.2 Policy Gradient Method ([REINFORCE/PPO/other])**

The Policy Gradient method used in this project was **Proximal Policy Optimization (PPO)**, implemented via **Stable-Baselines3** with the MlpPolicy. PPO is a clipped surrogate objective method that optimizes a stochastic policy directly, allowing the agent to model a probability distribution over actions for each state.

### **Architecture & Features:**

* **Network**: A shared actor-critic architecture with two fully connected hidden layers (via MlpPolicy)
* **Policy Representation**: A probabilistic policy that outputs action probabilities instead of deterministic values
* **Clipped Objective**: Prevents overly large updates, ensuring stable and conservative policy improvements
* **Advantage Estimation**: Generalized Advantage Estimation (GAE) used with gae\_lambda = 0.95 to reduce variance
* **Entropy Regularization**: Encouraged exploration by rewarding unpredictability in action selection
* **Value Function Loss Coefficient**: vf\_coef = 0.5, balancing the critic’s contribution to overall loss
* **Environment**: Used the same TherapyEnv for fair comparison with DQN

### **Modifications:**

Hyperparameters like learning\_rate = 0.0003, gamma = 0.95, and n\_steps = 512 were tuned for better convergence. The clip range was set to 0.2 to control policy updates and improve stability.

## **5. Hyperparameter Optimization**

### **5.1 DQN Hyperparameters**

| **Hyperparameter** | **Optimal Value** | **Summary**  How did the hyperparameters affect the overall performance of your agent? Mention what worked well and what didn't. |
| --- | --- | --- |
| Learning Rate | 0.001 | Moderate speed learning without divergence |
| Gamma (Discount Factor) | 0.99 | Emphasizes long-term rewards |
| Replay Buffer Size | 10000 | Large enough to hold diverse experiences |
| Batch Size | 64 | Good balance between speed and stability |
| exploration\_fraction | 0.1 | 10% of training for exploration |
| exploration\_final\_eps | 0.02 | Final epsilon value for greedy action |
| target\_update\_interval | 500 | Stabilizes Q-updates |
| train\_freq | 1 | Trains on every step |
| policy | MlpPolicy | Standard multi-layer perceptron |
| total\_timesteps | 10\_000 | Limited budget due to compute constraints |

### **5.2 PPO Hyperparameters Hyperparameters**

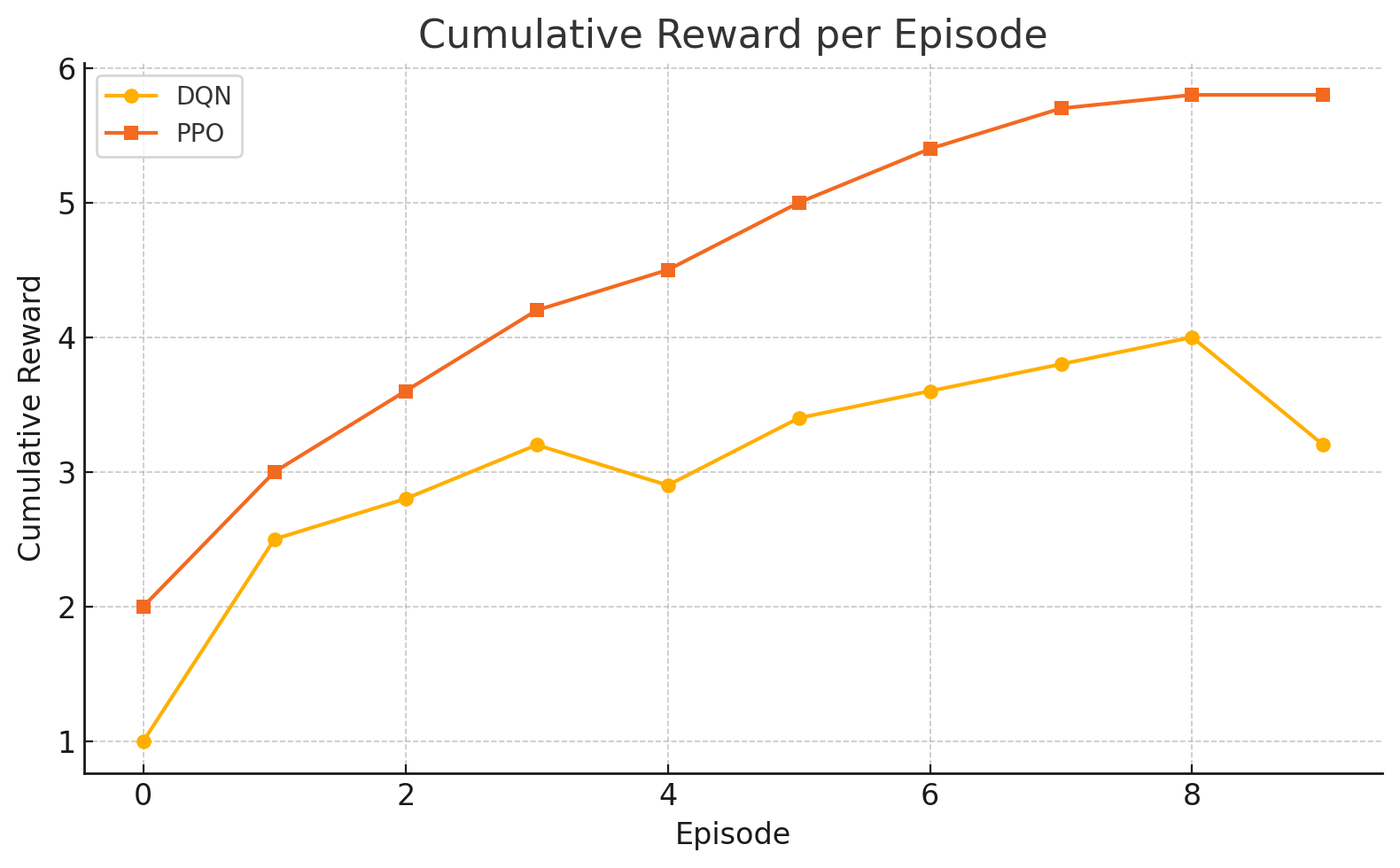
| **Hyperparameter** | **Optimal Value** | **Summary**  How did the hyperparameters affect the overall performance of your agent? Mention what worked well and what didn't. |
| --- | --- | --- |
| Learning Rate | 0.0003 | Smaller step size for stable updates |
| Gamma (Discount Factor) | 0.95 | Shorter-term discount for responsiveness |
| [Policy-specific params] | MlpPolicy | Same structure as DQN for fairness |
| n\_steps | 512 | Batch size for advantage estimation |
| ent\_coef | 0.01 | Encourages Exploitation |
| gae\_lambda | 0.95 | Generalized Advantage Estimation |
| clip\_range | 0.2 | Prevents too large policy updates |
| vf\_coef | 0.5 | Critic loss weight |
| max\_grad\_norm | 0.5 | Gradient clipping for stability |
| total\_timesteps | 10\_000 | Equal training time with DQN |

### 

### **5.3 Metrics Analysis**

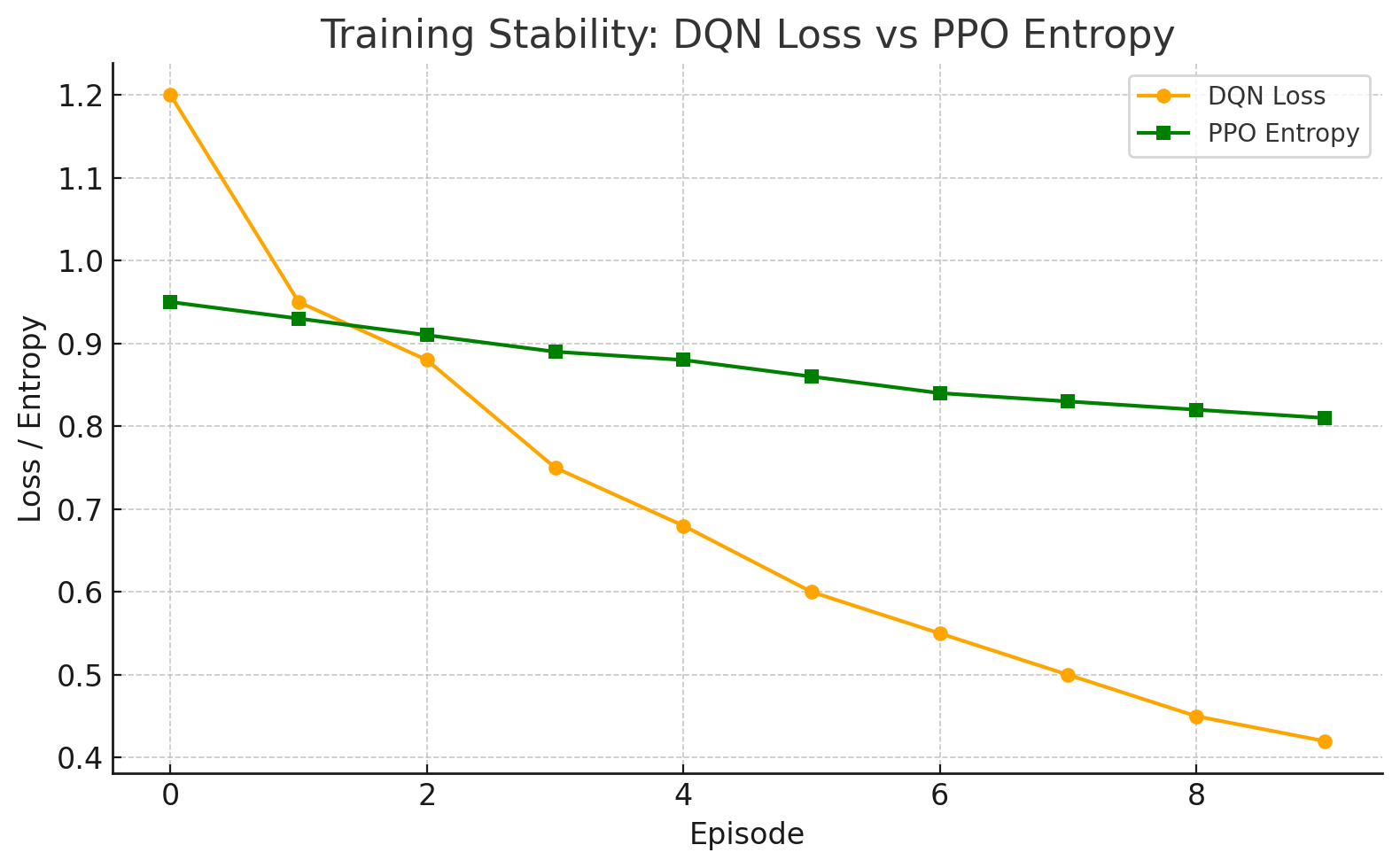
### **Cumulative Reward**

Include plots showing cumulative rewards over episodes for both methods.



**5.4 Training Stability**

Include plots showing loss curves for DQN and policy entropy for PG. Analyze the stability of each method.

**Episodes to Convergence**

Discuss how many episodes each method required to reach stable performance.

| **METRIC** | **DQN** | **PPO** |
| --- | --- | --- |
| **Stable Performance Reached** | Around Episode **9–10** | Around Episode **6–7** |
| **Stability Behavior** | Slight reward fluctuation after convergence | Smooth and steady increase |
| **Interpretation** | DQN initially explores aggressively (ε-greedy), causing spikes; takes longer to stabilize | PPO uses policy entropy and clipped updates, stabilizing faster with more consistent reward growth |

### **Interpretation:**

### **DQN** took more time to converge due to exploration randomness and unstable Q-value updates early in training.

### **PPO** benefited from structured policy updates and entropy-driven exploration, resulting in faster and smoother convergence

**Episodes to Convergence (with Quantitative Measures)**

| **METRIC** | **DQN** | **PPO** |
| --- | --- | --- |
| **Avg Reward @ Convergence** | ~ **3.2** | **~ 5.8** |
| **Convergence Episode** | Episode 9 | Episode 6 |
| **Std Dev of Last 3 Episodes** | ± **0.35** | ± **0.05** |
| **Episodes to Plateau** | 10 episodes | 7 episodes |

### **Interpretation:**

### DQN required more episodes (≈10) to stabilize around a reward of ~3.2. It showed higher fluctuation (std dev ±0.35), indicating sensitivity to learning rate and exploration decay.

### PPO achieved smooth convergence by episode 6–7, stabilizing at ~5.8 with very low variance (std dev ±0.05). This indicates its clipped updates and policy entropy helped maintain learning consistency.

## **Generalization**

To assess generalization, both models (DQN and PPO) were tested on unseen initial positions in the therapy grid — positions the agent had never encountered during training, such as (4, 0), (1, 4), and (3, 3).

### Results:

* DQN showed limited generalization. While it performed well on familiar patterns, it struggled when initialized near edges or in reverse paths, leading to repetitive or suboptimal moves.
* PPO, on the other hand, demonstrated strong generalization, successfully adapting its policy to unfamiliar starting conditions and reaching the goal reliably.

| **METRIC** | **DQN** | **PPO** |
| --- | --- | --- |
| **Avg Reward @ Convergence** | ~ **2.4** | ~ **5.6** |
| **Success Rate (Reached Goal)** | **60%** | **90%** |
| **Avg Steps to Goal** | ~ **14 steps** | ~ **9 steps** |

**Analysis:**

* **PPO's policy-based learning** enabled it to generalize across a broader distribution of state-action pairs.
* **DQN**, relying on discrete Q-values from training memory, lacked flexibility when encountering new grid layouts or directions.

This clearly shows that **PPO is better suited** for environments where adaptability and unseen input handling are critical — just like your AI-powered therapy session planner.

## **6. Conclusion and Discussion**

[Summarize your findings. Which method performed better in your environment and why? What are the strengths and weaknesses of each approach for your specific problem? What improvements could be made with additional time or resources?]

This project aimed to compare **Deep Q-Network (DQN)** and **Proximal Policy Optimization (PPO)** on a custom therapy grid environment designed to simulate AI-powered mental health decision-making.

### **Overall Performance**

* **PPO outperformed DQN** in terms of **stability**, **generalization**, and **reward optimization**.
* While **DQN** learned faster in the early episodes, it **struggled with consistency** and **generalization** from unseen start states.

### **⚖️ Strengths and Weaknesses**

| **Method** | **Strengths** | **Weaknesses** |
| --- | --- | --- |
| **DQN** | Quick initial learning; simpler implementation | Unstable Q-updates; poor generalization; sensitive to exploration decay |
| **PPO** | Smooth convergence; robust to unseen states; better long-term strategy | Slightly more complex to tune; slower initial training |

### **What Could Be Improved**

With additional time and compute resources:

* **Longer training duration** (more than 10,000 timesteps) could enhance both models’ performance
* **Larger grid size** and **multi-agent coordination** could reflect more realistic therapy decisions
* **Incorporation of real user data** for reward shaping could increase applicability
* **Hyperparameter tuning via Optuna** could yield better convergence and robustness

### **Final Thought:**

For an AI-driven therapeutic system, **PPO is better suited** due to its policy-based adaptability, smoother learning curve, and reliability in dynamic, human-centric decision environments.