

# Assignment-3

Kunal Kumar Sahoo (2025AIZ8459)  
AIL7022

November 2, 2025

## 1 Deep Q Networks During Distribution Shift

1. The environment has been trained on a linear network and non-linear network. Refer to Table 1 for architecture details of both the neural networks. For each architecture two identical network were instantiated, one which acts as the predictor (current network) and other as a ground truth (target network). The target network was updated using Polyak Averaging Rule. For experience replay, Prioritized Experience Replay Buffer was used.

Table 1: Network Architectures

Architecture	Layer Name	# Neurons	Layer Type	Activation
Linear Network	Input Layer	800	Fully Connected	Linear
	Output Layer	4	Fully Connected	Linear
Non-Linear Network	Input Layer	800	Linear	ReLU
	Hidden Layer	1024	Fully Connected	ReLU
	Hidden Layer	1024	Fully Connected	ReLU
	Output Layer	4	Fully Connected	Linear

Table 2: Hyperparameter Configuration for DQN Training

Hyperparameter	Value
Number of Seeds	10
Max Episode Steps	100
Batch Size	64
Discount Factor ( $\gamma$ )	0.99
Learning Rate	$1 \times 10^{-3}$
Replay Buffer Capacity	100,000
Start Training After	1,000 transitions
Target Network Update Frequency	Every 1,000 steps
PER Priority Exponent ( $\alpha$ )	0.6
PER IS Bias Start ( $\beta_0$ )	0.4
PER $\beta$ Annealing Frames	100,000
PER TD Error Epsilon	$1 \times 10^{-6}$
Boltzmann Temperature Start	1.0
Boltzmann Temperature End	0.1
Temperature Decay Frames	200,000
Training Episodes per Seed	10,000
Optimizer	Adam
Gradient Clipping (norm)	10.0
Loss Function	MSE (element-wise, weighted by IS)

Table 3: Prioritized Experience Replay Buffer Configuration

Feature	Configuration
Storage	Circular buffer (fixed capacity)
Capacity	100,000 transitions
Priority Assignment	$p_i =  \delta_i  + \epsilon$
Sampling Probability	$P(i) \propto p_i^\alpha$
Importance Sampling Weights	$w_i = (N \cdot P(i))^{-\beta} / \max w_i$
$\beta$ Annealing	Linear: $0.4 \rightarrow 1.0$ over 100,000 frames
Update Trigger	After each gradient step (using TD error)

Table 4: Exploration Policy: Boltzmann (Softmax) Action Selection

Component	Configuration
Temperature Schedule	$T(t) = T_{\text{end}} + (T_{\text{start}} - T_{\text{end}}) \cdot \exp\left(-\frac{t}{D}\right)$
Initial Temperature ( $T_{\text{start}}$ )	1.0
Final Temperature ( $T_{\text{end}}$ )	0.1
Decay Time Constant ( $D$ )	200,000 steps
Annealing Trigger	Per training step ( <code>self.steps_done</code> )

- For both networks, the best models have been saved as `models/best_linear.pt` and `models/best_nonlinear.pt`, respectively.
- Training rewards for both networks can be referred to in Figure 1.

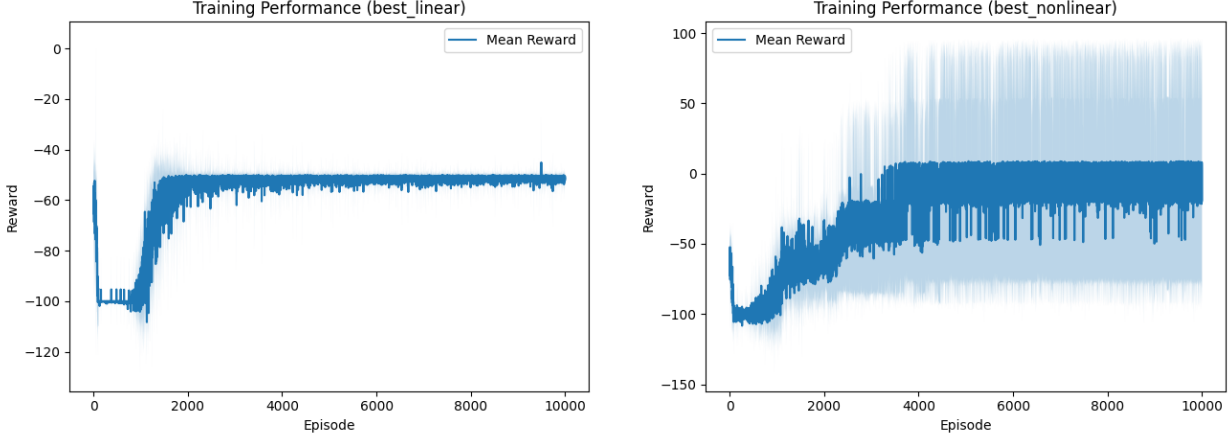


Figure 1: Training Rewards of Linear and Non-Linear DQN respectively

- Evaluation results of the policy extracted from both the learned Deep Q-Networks are mentioned in Table 5

Table 5: Evaluation results of the Linear and Non-Linear Deep Q-Networks

Architecture	Mean Reward	Std. Dev. Reward
Linear	-50.0	0.0
Non-Linear	147.9	69.59

- The linear DQN exhibits poor generalization, converging quickly during training to a stable but suboptimal policy (mean reward approximately  $-50$ ) that fails completely in the high-slip test environment, consistently falling off the cliff. This reflects high bias due to its limited capacity to model the logical dependency between checkpoints (`checkA` `checkB`) required for the risky goal. In contrast, the non-linear DQN shows strong generalization, achieving high training rewards ( $\approx 180$ ) and robust test performance ( $147.9 \pm 69.6$ ), successfully reaching the risky goal despite increased stochasticity and shifted whiskey positions. Though it displays higher training variance (noisy reward curve), its lower bias from deeper ReLU layers enables it to learn complex state interactions, demonstrating that reducing bias through expressive capacity outweighs increased variance in structured MDPs with distribution shift, yielding a favorable bias-variance tradeoff.

## 2 Deep SARSA

1. The Deep SARSA algorithm has been implemented in the same way as non-linear Deep Q-Networks in Section 1 with the change in the TD error calculation where it is calculated as  $R_{t+1} + \gamma Q(S_{t+1}, a') - Q(S_t, t)$  ( $a'$  is chosen from the current policy) instead of  $R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, t)$  in Q-Learning.

Table 6: Q-Network Architecture (Deep SARSA)

Layer Name	# Neurons	Layer Type	Activation
Input Layer	8	Input	—
Hidden Layer 1	1024	Fully Connected	SiLU
Hidden Layer 2	1024	Fully Connected	SiLU
Hidden Layer 3	1024	Fully Connected	SiLU
Output Layer	4	Fully Connected	Linear

Table 7: Hyperparameter Configuration

Hyperparameter	Value
Episodes	10,000
Max Steps per Episode	1,000
Discount Factor ( $\gamma$ )	0.99
Learning Rate	$5 \times 10^{-4}$
Optimizer	AdamW
Batch Size	64
Polyak Coefficient ( $\tau$ )	0.01
Replay Buffer Size	100,000
Random Seed	42

Table 8: Experience Replay Buffer Configuration

Feature	Configuration
Storage	Circular Deque
Capacity	100,000 transitions
Update Trigger	Every 4 steps
Batch Size	64

2. The best Deep SARSA model trained is saved at `models/deep_sarsa.pt`.
3. During the evaluation of 100 episodes, the mean reward reported is  $286.30 \pm 17.24$ .
4. The GIF of the evaluation of the learned policy can be found here: [https://drive.google.com/file/d/1s\\_v\\_s19PCef2A0j3w\\_\\_zqz9jbfvHS\\_w4/view?usp=sharing](https://drive.google.com/file/d/1s_v_s19PCef2A0j3w__zqz9jbfvHS_w4/view?usp=sharing).

Table 9: Exploration Policy: Temperature-Annealed Softmax

Component	Configuration
Initial Temperature	5.0
Final Temperature	0.1
Decay Type	Exponential (per episode)
Decay Rate	0.999
Update Rule	$T \leftarrow \max(0.1, 0.999 \cdot T)$

### 3 DQN for Portfolio Management

Both tasks mentioned in this question require different rewards. The reward design for both tasks have been provided in Table 10.

Table 10: Reward Structure for Task 1 and Task 2

Task	Reward Timing	Reward Definition
<b>Task 1</b> (Sparse)	Only at final step ( $t = T$ ) (All prior steps: $r_t = 0$ )	$r_T = \text{cash} + \sum_{i=1}^5 p_T^{(i)} \cdot h_T^{(i)}$
<b>Task 2</b> (Dense)	At <b>every</b> step $t$ (Including $t = T$ )	$r_t = \text{cash}_t + \sum_{i=1}^5 p_t^{(i)} \cdot h_t^{(i)}$

Refer to Table 11 for the DQN used in this environment.

Table 11: DQN Network Architecture

Model Specification	
Algorithm	Deep Q-Network (DQN)
Network Architecture	3-layer MLP
Input Dimension	11
Hidden Layers	$2 \times 1024$ neurons
Output Dimension	3125 (one per action)
Activation (Hidden)	ReLU
Activation (Output)	Linear
Loss Function	Mean Squared Error (MSE)
Optimizer	AdamW

Hyperparameters	
Batch Size	256
Learning Rate	$1 \times 10^{-4}$
Discount Factor ( $\gamma$ )	1.0
Replay Memory Capacity	10,000
Target Update	Soft update
Soft Update ( $\tau$ )	0.005
Exploration	Temperature-based softmax
Initial Temperature	5.0
Final Temperature	0.1
Temperature Decay	$\exp(-t/20000)$
Training Episodes	10,000
Evaluation Seeds	100
Gradient Clipping	100 (by value)
Random Seed	42

Table 12: Exploration Policy: Temperature-Annealed Boltzmann

Component	Configuration
Initial Temperature	5.0
Final Temperature	0.1
Decay Schedule	$T(t) = T_{\text{end}} + (T_{\text{start}} - T_{\text{end}}) \cdot \exp(-t/D)$
Decay Constant ( $D$ )	20,000 steps

Refer to Table 13 for evaluation results across 100 seeds at the end of 10 steps:

Table 13: Evaluation results of different tasks

Task	Mean Total Wealth	Mean:Std. Dev.
Maximize Total Wealth	74.18	4.32
Maximize Wealth Throughout	74.25	4.55

Refer to Figure 2 for portfolio wealth across all the timesteps.

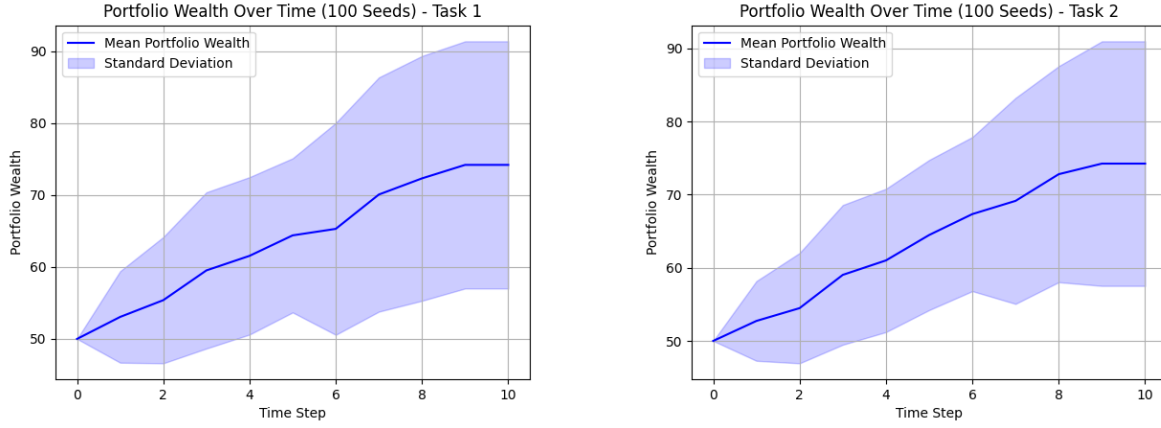


Figure 2: Mean wealth collected across all timesteps for different tasks.

Refer to Figure 3 for training losses.

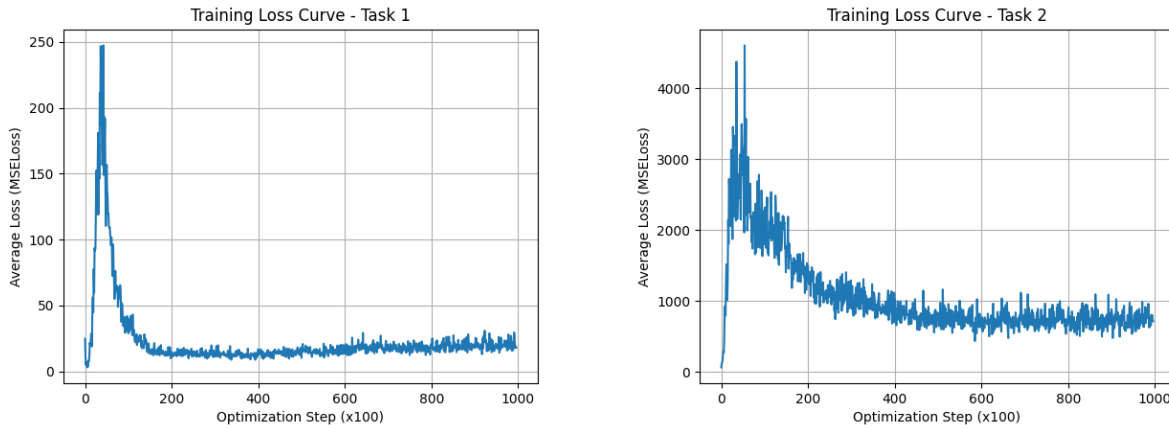


Figure 3: Mean wealth collected across all timesteps for different tasks.

## 4 TreasureHunt-v2

1. The neural network architecture mentioned in the question has been implemented.
2. A Prioritized Experience Replay has been implemented for this question (Table 14).

Table 14: Prioritized Experience Replay (PER) Hyperparameters

Parameter	Value
Replay Buffer Capacity	10,000
Priority Exponent ( $\alpha$ )	0.6
Initial Importance Sampling Bias ( $\beta_0$ )	0.4
Final $\beta$	1.0
$\beta$ Annealing Frames	$1500 \times 500 = 750,000$
TD Error $\epsilon$ (PER_EPS)	$1 \times 10^{-6}$
Priority Update	$p_i = ( \delta_i  + \epsilon)^\alpha$
IS Weight	$w_i = (N \cdot P(i))^{-\beta} / \max w$

3. Instead of the  $\epsilon$ -greedy policy suggested for this task, I have adopted a Boltzmann exploration policy with hyperparameters (Table 15):

Table 15: Softmax Exploration Policy with Temperature Annealing

Parameter	Value
Action Selection	$\pi(a s) = \frac{\exp(Q(s,a)/T)}{\sum_{a'} \exp(Q(s,a')/T)}$
Initial Temperature ( $T_{\text{start}}$ )	5.0
Final Temperature ( $T_{\text{end}}$ )	0.1
Decay Type	Exponential (per episode)
Decay Rate	0.995
Update Rule	$T \leftarrow \max(T_{\text{end}}, 0.995 \cdot T)$

4. Refer to Figure 4 for training rewards over 15000 episodes.

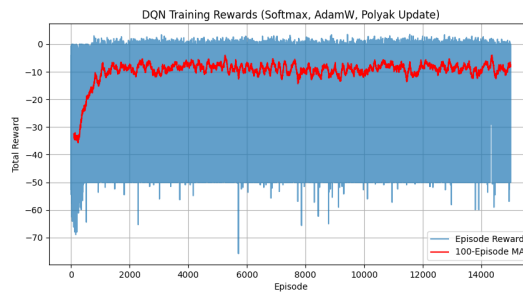


Figure 4: Training Reward for TreasureHunt-v2 Environment

5. • Random Agent GIF: [https://drive.google.com/file/d/1QLBPac0mjM64HlR7e3r0csrx\\_ZT4CGNr/view?usp=sharing](https://drive.google.com/file/d/1QLBPac0mjM64HlR7e3r0csrx_ZT4CGNr/view?usp=sharing)
- Trained Agent GIF: [https://drive.google.com/file/d/1kx4PUE5R\\_8YDwc4aLZJFBmISeSsNmG17/view?usp=sharing](https://drive.google.com/file/d/1kx4PUE5R_8YDwc4aLZJFBmISeSsNmG17/view?usp=sharing)