Name: Seongil Heo
UID: u1527760
Date: February 11, 2026

Part 1: Tabular Q-Learning to Make Millions at Vegas?!

Answer

Figure 1 shows the learning curve of the Q-learning agent. The x-axis represents the number of episodes, and the y-axis represents the average episode return.

The blue line is the moving average of the training rewards. Since individual episode rewards are highly noisy, averaging over many recent episodes provides a smoother view of the overall learning trend. The upward trend of this line indicates that the agent is gradually improving its behavior over time.

The green dashed line represents periodic policy evaluation. At regular intervals, learning is paused and the current policy is run over multiple episodes without exploration. This curve reflects the actual performance of the learned policy.

The orange line shows the performance of a purely random policy, where actions are selected uniformly at random. This serves as a baseline for comparison.

Because both the training curve and the evaluation curve stay consistently above the random policy and the training curve shows an overall upward trend, the figure demonstrates that the agent is successfully learning an effective strategy.

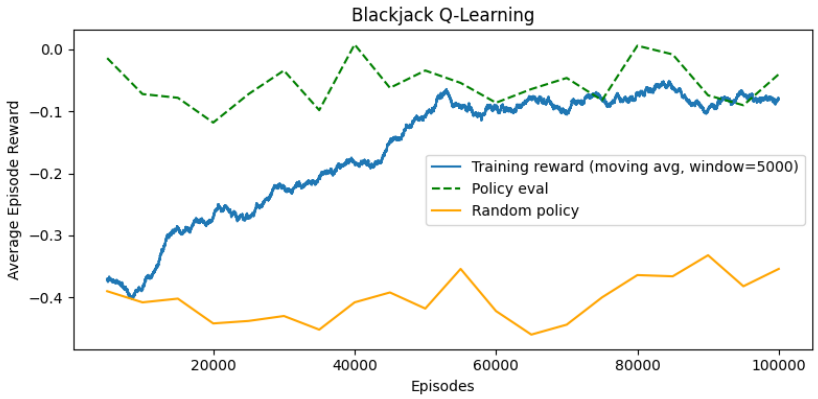


Figure 1: Training performance

Table 1: Tabular Q Hyperparameters

Parameter	Value
Discount factor	0.95
Episodes	100000
Epsilon decay step	50000
Epsilon final	0.05
Epsilon schedule	linear
Epsilon start	1.0
Eval episodes	500
Eval interval	5000
Learning rate	0.1

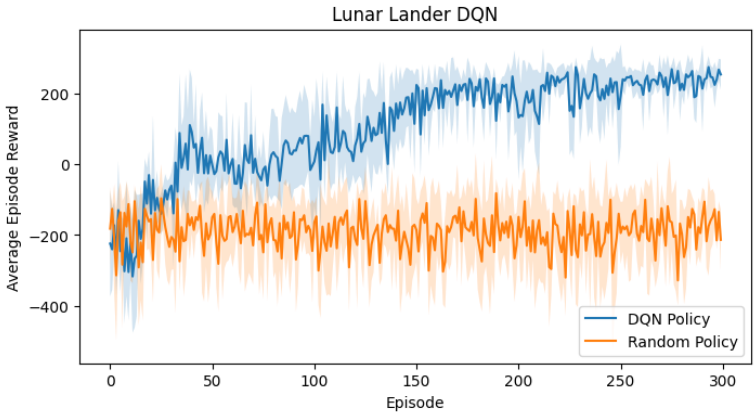
Part 2: Landing on the Moon using DQN!

Answer

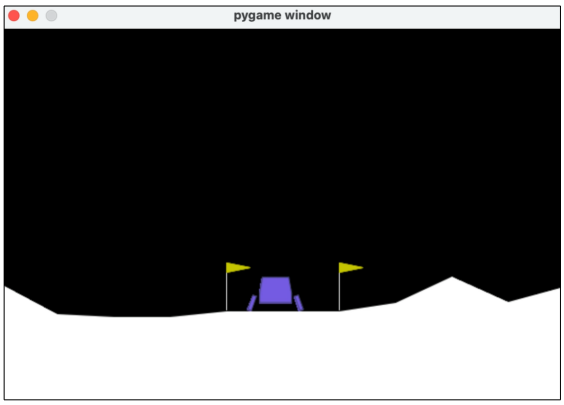
The DQN agent shows clear learning progress over training. Initially, performance is similar to a random policy with many crashes and negative rewards. However, the learning curve demonstrates a strong upward trend, and by the end of training the agent consistently achieves mean rewards between 200 and 260. This indicates successful, smooth, and stable landings with reduced fuel waste and fewer crashes.

Compared to a random policy, which remains around –200 reward, the learned policy performs dramatically better, showing that meaningful learning has occurred.

The trained agent can perform more consistently than a human player. While humans may occasionally achieve good landings, the agent demonstrates more stable control and higher average performance across episodes.



(a) DQN vs Random Learning Curve



(b) Visualization

Figure 2: Training performance and learned policy behavior

Table 2: DQN Hyperparameters

Parameter	**Value**
Batch size	128
Discount factor	0.99
Learning rate	3×10^{-4}
Replay buffer size	50,000
Warmup steps	1000
Epsilon start	1.0
Epsilon end	0.05
Epsilon decay	1500
Target update type	Soft update
Soft update rate (τ)	0.003
Loss function	Huber loss
Gradient clipping	$[-1, 1]$
Training episodes	300
Evaluation interval	10 episodes
Evaluation episodes	10
Optimizer	AdamW (AMSGrad)

Extra Credit 1:

Answer

The results show that the agent improves over time under all settings, confirming that Q-learning successfully learns a better policy. However, performance strongly depends on the exploration strategy and its intensity. In epsilon-greedy exploration, very small fixed ε values achieved the best performance, with $\varepsilon = 0.01$ producing the highest average reward. This suggests that minimal but persistent exploration allows the agent to quickly exploit good actions while still avoiding premature convergence. As ε increases, performance degrades because excessive randomness prevents the policy from stabilizing. Among decaying schedules, exponential decay outperformed linear decay, likely because it reduces exploration more rapidly early in training, which is suitable for the relatively small and simple Blackjack state space.

In addition, the linear epsilon-greedy curve, shown in blue, exhibits a clear overall upward trend over time. This steady increase in the moving-average reward illustrates the progression of learning, demonstrating that the agent gradually improves its policy as training proceeds.

For Boltzmann exploration, lower temperatures led to better performance, while higher temperatures caused significant degradation. As temperature increases, action selection becomes more uniform and approaches random behavior, which is detrimental in this task. Overall, epsilon-greedy exploration outperformed Boltzmann exploration. Since Blackjack has a small action space and a relatively deterministic optimal policy, limited exploration is sufficient, and more sophisticated probabilistic exploration does not provide additional benefits. These results indicate that maintaining a very small level of exploration is the most effective strategy in this environment.

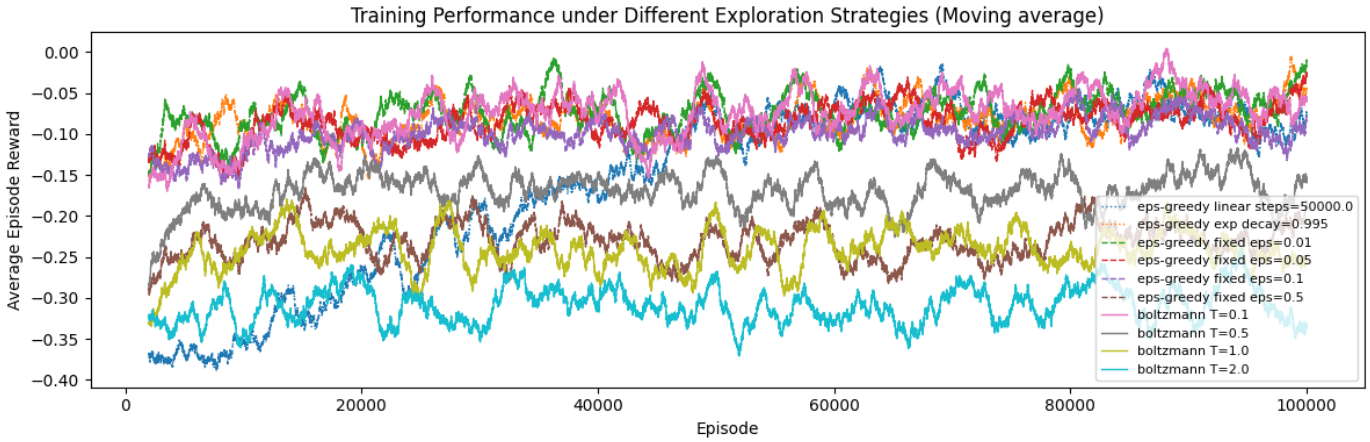


Figure 3: Training performance under different exploration strategies

Table 3: Exploration Strategies

Mode	decacy	epsilon	temperature	Last Average Reward
Epsilon-greedy	linear (steps: 50000)	[0.05-1.0]	-	-0.076
Epsilon-greedy	exp (ratio: 0.995)	[0.05-1.0]	-	-0.044
Epsilon-greedy	fixed	0.01	-	**-0.010**
Epsilon-greedy	fixed	0.05	-	-0.028
Epsilon-greedy	fixed	0.1	-	-0.079
Epsilon-greedy	fixed	0.5	-	-0.183
Boltzmann	-	-	0.1	-0.059
Boltzmann	-	-	0.5	-0.160
Boltzmann	-	-	1.0	-0.253
Boltzmann	-	-	2.0	**-0.334**

Extra Credit 2:

Answer

I referred to the official Gymnasium documentation to understand the observation format, action space, and environment settings of the CarRacing environment.

I convert the continuous action space into five discrete action in order to apply the DQN algorithm. For the CNN-DQN input preprocessing, RGB images were converted to grayscale, resized to 84×84, and normalized to the range [0,1]. Figure 5 shows the network architecture.

During training, a Replay Buffer was used to sample transitions randomly, and an epsilon-greedy exploration strategy was applied. The loss function used was Huber loss instead of MSE, and for training stability, soft target updates (τ) and gradient clipping were also applied.

As shown in Figure 4 although the learning curve itself was highly noisy, the moving average steadily increased, indicating that learning was occurring. Compared to a random policy, the learned policy achieved significantly higher performance, demonstrating the effectiveness of the training process.

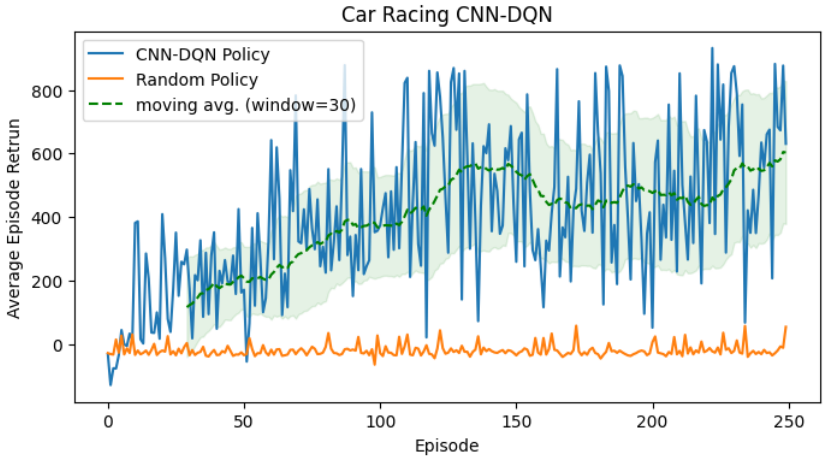


Figure 4: Training performance

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
CNNDQN                                   [1, 5]                    --
├─Sequential: 1-1                        [1, 64, 7, 7]             --
│    └─Conv2d: 2-1                       [1, 32, 20, 20]           2,080
│    └─ReLU: 2-2                         [1, 32, 20, 20]           --
│    └─Conv2d: 2-3                       [1, 64, 9, 9]             32,832
│    └─ReLU: 2-4                         [1, 64, 9, 9]             --
│    └─Conv2d: 2-5                       [1, 64, 7, 7]             36,928
│    └─ReLU: 2-6                         [1, 64, 7, 7]             --
├─Sequential: 1-2                        [1, 5]                    --
│    └─Flatten: 2-7                      [1, 3136]                 --
│    └─Linear: 2-8                       [1, 512]                  1,606,144
│    └─ReLU: 2-9                         [1, 512]                  --
│    └─Linear: 2-10                      [1, 5]                    2,565
==========================================================================================
Total params: 1,680,549
Trainable params: 1,680,549
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 6.91
==========================================================================================
Input size (MB): 0.03
Forward/backward pass size (MB): 0.17
Params size (MB): 6.72
Estimated Total Size (MB): 6.92
==========================================================================================
```

Figure 5: Model Architecture (torchinfo)