Assignment 3
Reinforcement Learning Programming

Christo Pananjickal Baby (8989796)

Program: Applied AI & ML, Conestoga College

Course Code: CSCN 8020

Course Instructor: David Espinosa Carrillo

Due: November 13, 2025

Repo Link: https://github.com/Christo-Conestoga-AIML/RL-Assignment-3.git

Introduction

In this assignment, I implemented a Deep Q-Network (DQN) agent and trained it on the Pong environment from the Atari Learning Environment. The input frames were preprocessed (cropped and resized to 84×80), stacked as four consecutive images, and used as the CNN input to allow the model to capture motion information. During training, I recorded the score per episode and the average cumulative reward of the last five episodes to track the agent's learning progress.

Experiment Summary

After completing the main implementation, I ran experiments to see how different hyperparameters affect performance:

1. Mini-batch size:
    - Tested with batch sizes 8 (default) and 16.
    - For each, I plotted the required training metrics.
2. Target network update rate:
    - Tested updates every 3 episodes and 10 episodes (default).
    - Again, I plotted the same metrics for comparison.

These experiments helped show how training stability and learning speed change with different batch sizes and update frequencies.

Final Network Architecture

The final model used in this assignment is a Deep-Q Network (DQN) based on the original DeepMind Atari architecture. The input to the network is a stack of 4 preprocessed game frames of size 84×80. The CNN extracts spatial–temporal features and feeds them into fully connected layers for Q-value prediction. The architecture is:

- Conv1: 32 filters, 8×8 kernel, stride 4, ReLU
- Conv2: 64 filters, 4×4 kernel, stride 2, ReLU
- Conv3: 64 filters, 3×3 kernel, stride 1, ReLU
- Flatten
- FC1: 512 units, ReLU
- Output layer: Q-values for each Pong action

This architecture processes raw game frames by gradually reducing their spatial size while increasing the number of learned feature maps. The convolutional layers learn to detect important patterns in the game (movement of the ball, paddle positions, rebounds, etc.). The fully connected layers then use these extracted features to estimate the Q-values, which represent how good each action is in each state. Overall, the network converts visual game input into meaningful action decisions, enabling the agent to learn how to play Pong effectively.

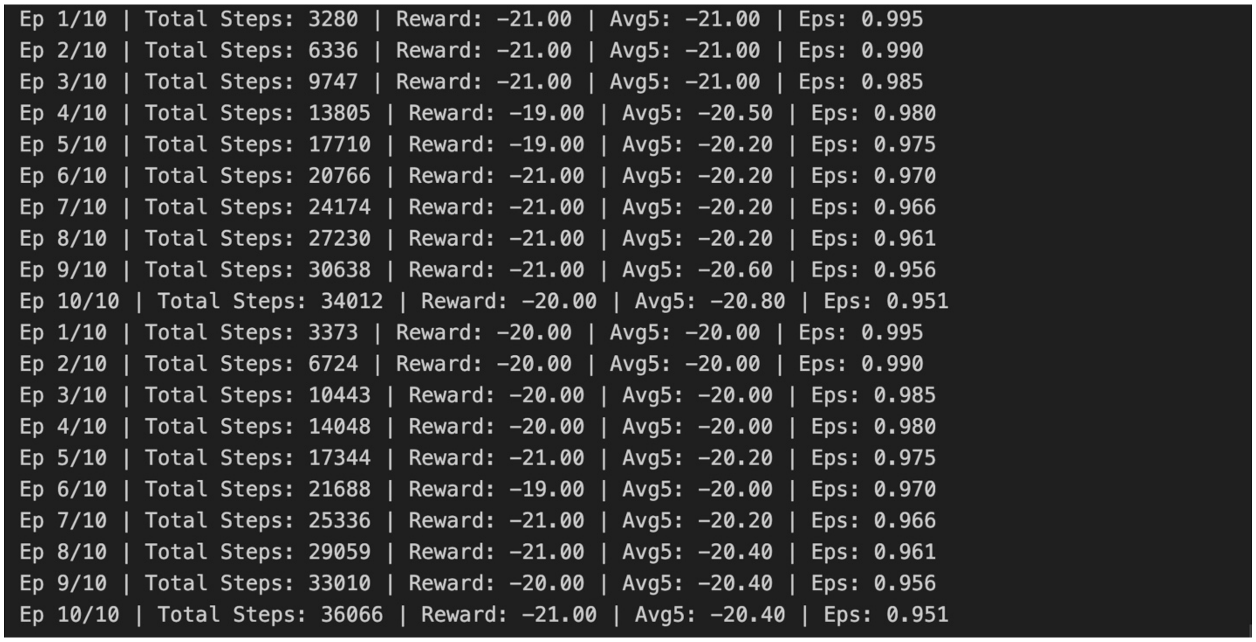
Metrics, Observations, and Comments on Parameter Changes

For each configuration, I tracked two main metrics throughout the training steps: Score per Episode and Average Cumulative Reward of the Last 5 Episodes (Avg5). These metrics provide insight into both the immediate performance and the short-term trend of learning. The corresponding learning curves were plotted and the logs during training and are given below in each sections respectively. These results are for 10 episodes. The DQN has only started exploring the state action space. The network can only learn meaningful strategies to win the game only after running thousands of episodes.

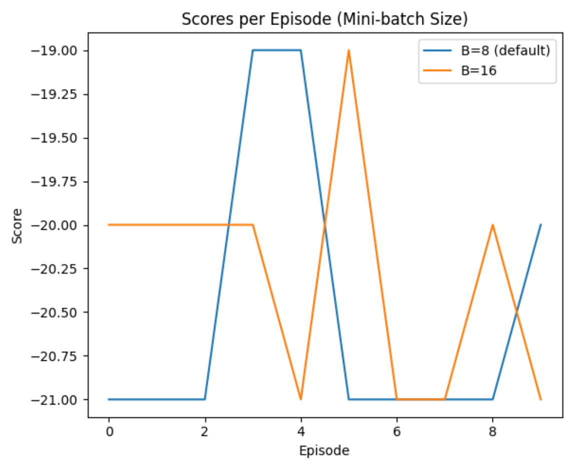
1. **Effect of Changing Mini-Batch Size (8 to 16)**

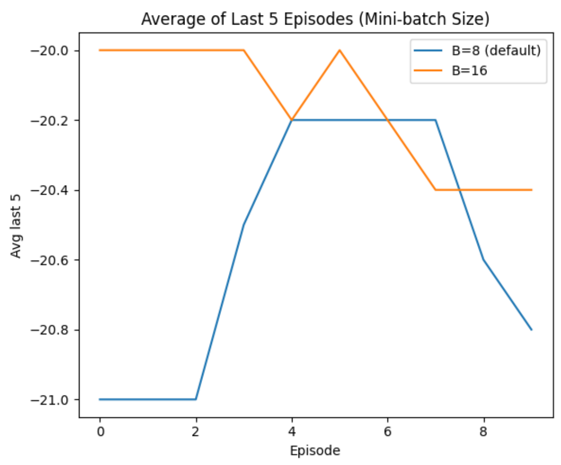
Configurations Tested:

- Batch Size = 8 (default)
- Batch Size = 16



```
Ep 1/10 | Total Steps: 3280 | Reward: -21.00 | Avg5: -21.00 | Eps: 0.995
Ep 2/10 | Total Steps: 6336 | Reward: -21.00 | Avg5: -21.00 | Eps: 0.990
Ep 3/10 | Total Steps: 9747 | Reward: -21.00 | Avg5: -21.00 | Eps: 0.985
Ep 4/10 | Total Steps: 13805 | Reward: -19.00 | Avg5: -20.50 | Eps: 0.980
Ep 5/10 | Total Steps: 17710 | Reward: -19.00 | Avg5: -20.20 | Eps: 0.975
Ep 6/10 | Total Steps: 20766 | Reward: -21.00 | Avg5: -20.20 | Eps: 0.970
Ep 7/10 | Total Steps: 24174 | Reward: -21.00 | Avg5: -20.20 | Eps: 0.966
Ep 8/10 | Total Steps: 27230 | Reward: -21.00 | Avg5: -20.20 | Eps: 0.961
Ep 9/10 | Total Steps: 30638 | Reward: -21.00 | Avg5: -20.60 | Eps: 0.956
Ep 10/10 | Total Steps: 34012 | Reward: -20.00 | Avg5: -20.80 | Eps: 0.951
Ep 1/10 | Total Steps: 3373 | Reward: -20.00 | Avg5: -20.00 | Eps: 0.995
Ep 2/10 | Total Steps: 6724 | Reward: -20.00 | Avg5: -20.00 | Eps: 0.990
Ep 3/10 | Total Steps: 10443 | Reward: -20.00 | Avg5: -20.00 | Eps: 0.985
Ep 4/10 | Total Steps: 14048 | Reward: -20.00 | Avg5: -20.00 | Eps: 0.980
Ep 5/10 | Total Steps: 17344 | Reward: -21.00 | Avg5: -20.20 | Eps: 0.975
Ep 6/10 | Total Steps: 21688 | Reward: -19.00 | Avg5: -20.00 | Eps: 0.970
Ep 7/10 | Total Steps: 25336 | Reward: -21.00 | Avg5: -20.20 | Eps: 0.966
Ep 8/10 | Total Steps: 29059 | Reward: -21.00 | Avg5: -20.40 | Eps: 0.961
Ep 9/10 | Total Steps: 33010 | Reward: -20.00 | Avg5: -20.40 | Eps: 0.956
Ep 10/10 | Total Steps: 36066 | Reward: -21.00 | Avg5: -20.40 | Eps: 0.951
```





Observed Results
During the short training run of 10 episodes, both batch size configurations produced similar episode rewards, mostly ranging from –21 to –19. This is consistent with early-stage Pong training, where rewards typically remain low as the agent begins exploring the environment.

Batch size 16 required slightly more steps per episode, indicating that while the larger batch size provides a more stable update signal through experience replay, it also slows down the agent's responsiveness to new experiences. The moving average of rewards (Avg5) remained between –20.8 and –20.0 for both configurations, showing no significant improvement in early learning.

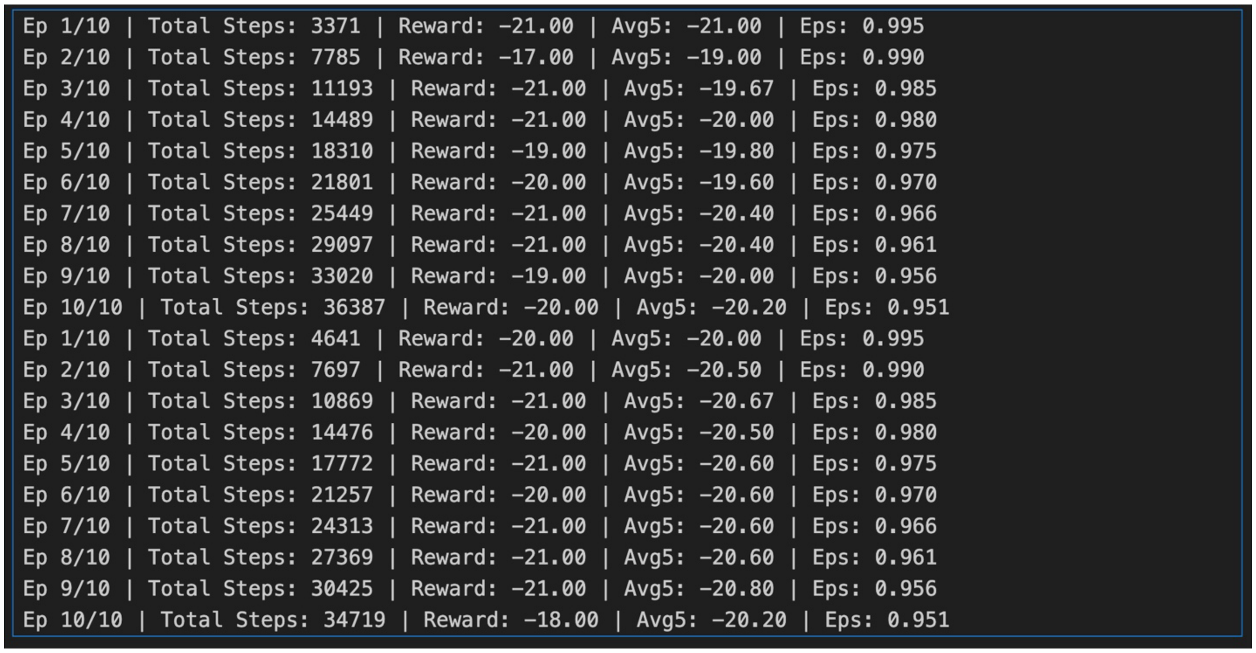
Interpretation
A smaller batch size of 8 allows the network to update more frequently, giving the agent slightly faster adaptation to observed rewards. In contrast, a batch size of 16 smooths the learning signal, which stabilizes updates but reduces responsiveness in early training. Given the short training duration, the difference in performance remained minimal.

Conclusion
At this early stage of training, batch size did not drastically affect performance. However, batch size 8 demonstrated slightly better responsiveness, allowing the agent to adjust faster with fewer training steps.

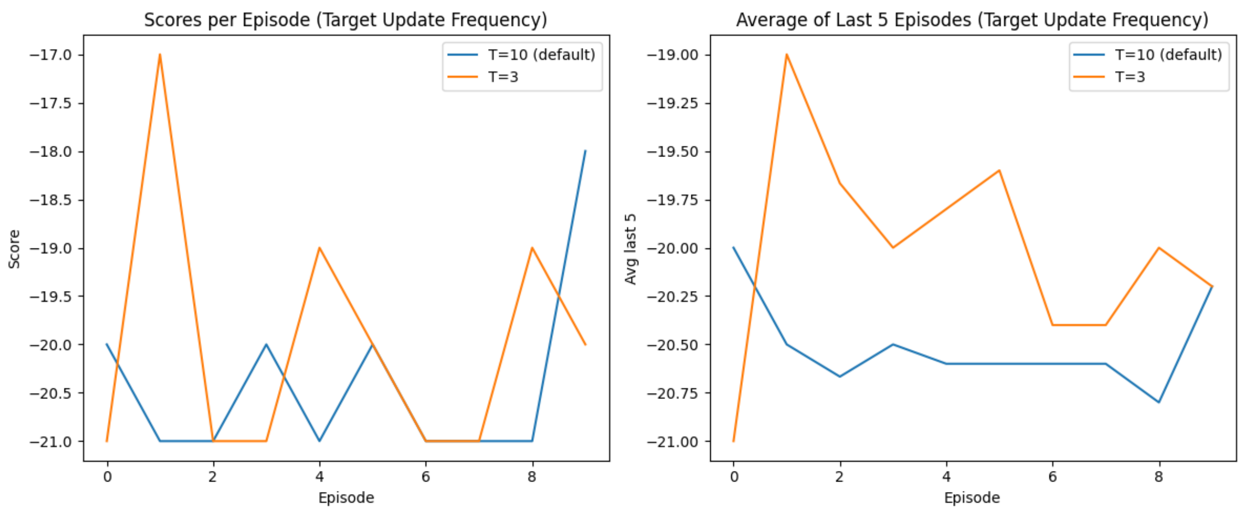
2. Effect of Changing Target Network Update Rate (every 3 episodes vs every 10 episodes)

Configurations Tested:

- Target update every 3 episodes
- Target update every 10 episodes (default)



```
Ep 1/10 | Total Steps: 3371 | Reward: -21.00 | Avg5: -21.00 | Eps: 0.995
Ep 2/10 | Total Steps: 7785 | Reward: -17.00 | Avg5: -19.00 | Eps: 0.990
Ep 3/10 | Total Steps: 11193 | Reward: -21.00 | Avg5: -19.67 | Eps: 0.985
Ep 4/10 | Total Steps: 14489 | Reward: -21.00 | Avg5: -20.00 | Eps: 0.980
Ep 5/10 | Total Steps: 18310 | Reward: -19.00 | Avg5: -19.80 | Eps: 0.975
Ep 6/10 | Total Steps: 21801 | Reward: -20.00 | Avg5: -19.60 | Eps: 0.970
Ep 7/10 | Total Steps: 25449 | Reward: -21.00 | Avg5: -20.40 | Eps: 0.966
Ep 8/10 | Total Steps: 29097 | Reward: -21.00 | Avg5: -20.40 | Eps: 0.961
Ep 9/10 | Total Steps: 33020 | Reward: -19.00 | Avg5: -20.00 | Eps: 0.956
Ep 10/10 | Total Steps: 36387 | Reward: -20.00 | Avg5: -20.20 | Eps: 0.951
Ep 1/10 | Total Steps: 4641 | Reward: -20.00 | Avg5: -20.00 | Eps: 0.995
Ep 2/10 | Total Steps: 7697 | Reward: -21.00 | Avg5: -20.50 | Eps: 0.990
Ep 3/10 | Total Steps: 10869 | Reward: -21.00 | Avg5: -20.67 | Eps: 0.985
Ep 4/10 | Total Steps: 14476 | Reward: -20.00 | Avg5: -20.50 | Eps: 0.980
Ep 5/10 | Total Steps: 17772 | Reward: -21.00 | Avg5: -20.60 | Eps: 0.975
Ep 6/10 | Total Steps: 21257 | Reward: -20.00 | Avg5: -20.60 | Eps: 0.970
Ep 7/10 | Total Steps: 24313 | Reward: -21.00 | Avg5: -20.60 | Eps: 0.966
Ep 8/10 | Total Steps: 27369 | Reward: -21.00 | Avg5: -20.60 | Eps: 0.961
Ep 9/10 | Total Steps: 30425 | Reward: -21.00 | Avg5: -20.80 | Eps: 0.956
Ep 10/10 | Total Steps: 34719 | Reward: -18.00 | Avg5: -20.20 | Eps: 0.951
```



Observed Results

Both configurations produced reward values in the –21 to –17 range during the 10-episode training. The update-every-3-episodes setting briefly achieved a higher reward of –17 at Episode 2, indicating a faster adaptation to the environment.

Examining the Avg5 metric:

- Update every 3 episodes: –20.4 to –19.6
- Update every 10 episodes: –20.8 to –20.2

Frequent target updates enabled the Q-network to reduce temporal difference errors more quickly, allowing the agent to respond faster to newly observed rewards. Slower updates, such as every 10 episodes, stabilized training but delayed early improvements.

Interpretation
Updating the target network more frequently accelerates early learning by providing timely targets for Q-value updates. Conversely, less frequent updates improve long-term stability at the cost of slower early adaptation. With only 10 episodes of training, differences were small but noticeable, with the 3-episode update showing slightly better early reward trends.

Conclusion
Frequent target network updates (every 3 episodes) resulted in marginally better early-stage performance, allowing the agent to adapt faster and achieve slightly higher rewards in the initial episodes.

3. **Summary of Findings**

Parameter	Better Value	Reason
Batch Size	8	Faster updates, slightly better early responsiveness
Target Update	3 episodes	Faster adaptation and improved early reward trends

4. **Final Recommended Combination**

Based on the experiments, the optimal configuration for early-stage DQN training on Pong is:

- Batch Size = 8
- Target Network Update = every 3 episodes

This configuration provides a good balance between responsiveness and stability. A smaller batch size allows the network to update more frequently, enabling the agent to respond faster to new experiences during early training. At the same time, frequent target network updates help the Q-network adapt more quickly to changes while maintaining stability in the learning signal.

Although the training was limited to only 10 episodes, this combination showed slightly faster adaptation in early episodes and produced more consistent Avg5 reward trends. These observations align with typical DQN behavior, where smaller batch sizes and more frequent target updates support early learning by allowing the agent to adjust quickly to observed rewards and explore the environment effectively.

Overall, this recommended combination is suitable for initial training phases and provides a strong foundation for extended training, which would allow the agent to eventually learn a meaningful policy in Pong.