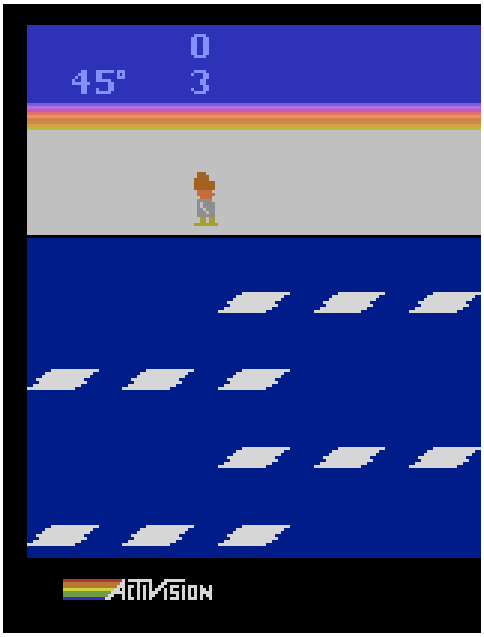
Matthew Scott
CSPB 3202 AI
Github Link: https://github.com/gr8tscott/AIFinalProject.git

AI Final Project: Frostbite DQN



Project Overview

Goal

The primary objective of this project is to explore and demonstrate the application of Reinforcement Learning (RL) using modern tools and techniques. Specifically, the project focuses on implementing and evaluating the Deep Q-Network (DQN) algorithm within the *ALE/Frostbite-v5* environment provided by OpenAI Gym/Gymnasium. The goal is to assess the DQN's performance, examine its ability to learn and improve over time, and address common challenges encountered in training RL agents.

Environment

- **Name**: *ALE/Frostbite-v5*
- **Type**: Atari game environment
- **Objective**: Help Frostbite Bailey build igloos by jumping on floating blocks of ice while avoiding various hazards, including clams, snow geese, Alaskan king crab, grizzly polar bears, and rapidly dropping temperatures.

Model

- **Algorithm**: Deep Q-Network (DQN)

- **Purpose**: The DQN algorithm is used to train an agent to interact with and learn from the *ALE/Frostbite-v5* environment. The model leverages a neural network to approximate the Q-values for different actions and uses experience replay and target networks to stabilize training.

This project aims to showcase the practical application of RL techniques and tools in a complex environment, evaluate the effectiveness of the DQN algorithm, and provide insights into the challenges and solutions associated with training RL agents.

Approach

Environment

- **Name**: *ALE/Frostbite-v5*
- **Type**: Atari game environment
- **Objective**: Build igloos by jumping on floating ice floes while avoiding hazards.

Model

- **Algorithm**: Deep Q-Network (DQN)
- **Policy**: *CnnPolicy* (Convolutional Neural Network Policy)
- **Hyperparameters**:
    - **Buffer Size**: 20,000
    - **Learning Starts**: 1,000 timesteps
    - **Target Update Interval**: 1,000 timesteps
    - **Learning Rate**: 0.00005
    - **Batch Size**: 64
    - **Exploration Fraction**: 0.1 (Fraction of the total training period over which the exploration rate reduces)
    - **Exploration Final Epsilon**: 0.01 (Final exploration rate)

Training and Evaluation Steps

1. **Environment Setup**:
    - Initialize the *ALE/Frostbite-v5* environment.
    - Wrap the environment with *Monitor* to track performance metrics and *RecordVideo* to capture gameplay footage.
2. **Model Initialization**:
    - Create a DQN model with the specified hyperparameters and *CnnPolicy*.
3. **Training**:
    - **Episodes**: Run training over a set number of episodes (5 episodes in the beginning and later 20 episodes at end).

    - **Timesteps**: Each episode includes a predefined number of timesteps (beginning: 5,000 timesteps per episode, end: 10,000 timesteps per episode).
    - **Learning**: Use the *learn* method to train the model over the total timesteps for each episode.
    - **Exploration**: Gradually decrease the exploration rate as training progresses, based on the *exploration_fraction* and *exploration_final_eps* parameters.
4. **Evaluation**:
    - **Performance Metrics**: Evaluate the trained model using *evaluation_policy* to compute mean rewards and standard deviations over multiple evaluation episodes.
    - **Training Progress**: Track and record the model's performance metrics (e.g., rewards) over the training episodes.
    - **Visualization**: Plot rewards over training episodes to visualize progress.
5. **Post-Training**:
    - **Model Saving**: Save the trained model for future use or further testing.
    - **Video Recording**: Capture a video of the agent playing the game using *RecordVideo*.
    - **GIF Creation**: Convert recorded frames into a GIF for visual representation of the agent's performance.

Troubleshooting

- **Memory Issues**: Initially encountered memory limitations due to the large size of the replay buffer and batch size.
    - **Solutions**:
        - Reduced buffer size to 10,000 to fit within memory constraints before ramping it up to 15,000 and finally 20,000.
        - Adjusted batch size and learning rate to improve stability and performance.
        - Played with lower episode numbers and timesteps during early experiments to ensure the model could run without excessive memory usage.
        - Gradually increased the model size and parameters once initial issues were resolved and the model was able to produce data and visual outputs effectively.
- **Overheating and Processing Time:** Training the model was a heavy lift on more than just the memory. The training incurred long processing time, which led to overheating, that led to even longer processing time.
    - **Solution:**
        - In order to reduce processing time and strain on the system, ice packs were applied to the bottom of the computer and rotated out regularly to keep the device cool and working efficiently.

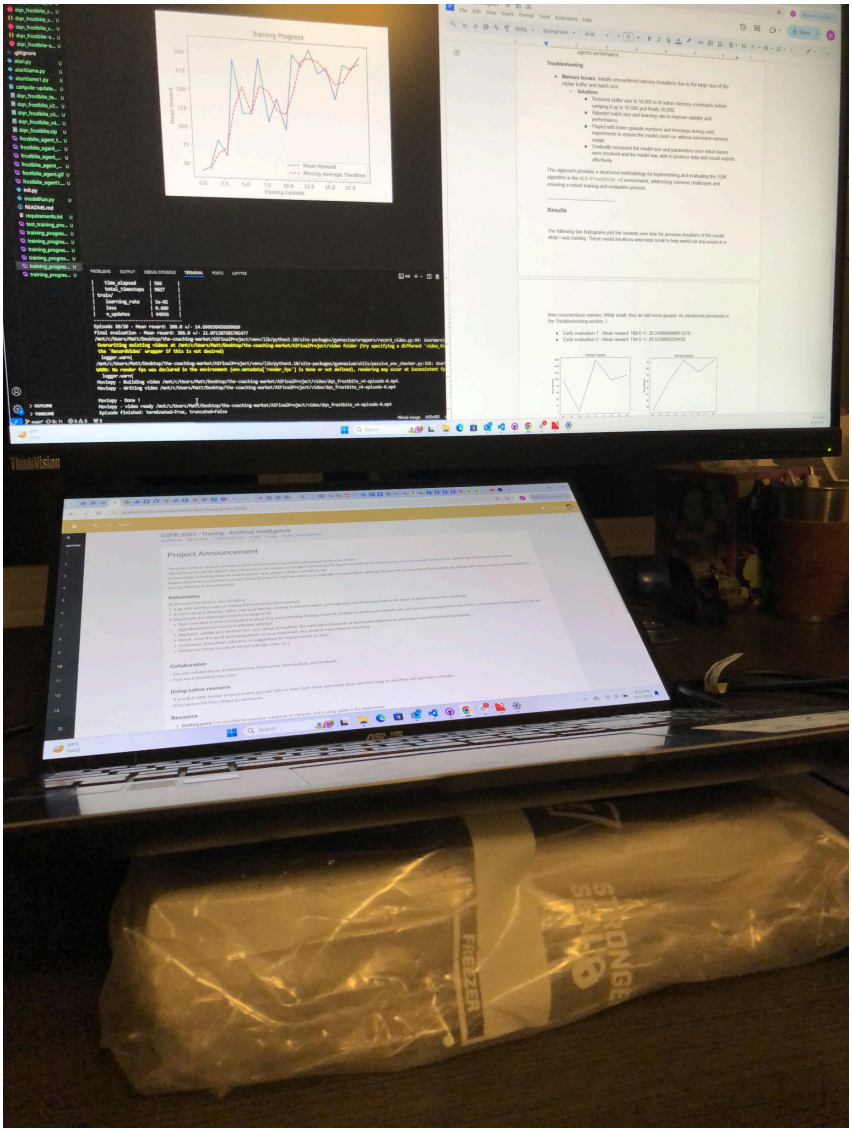


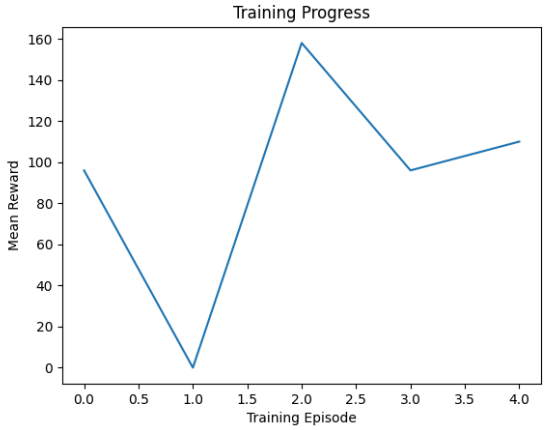
Image of icepack cooling computer.

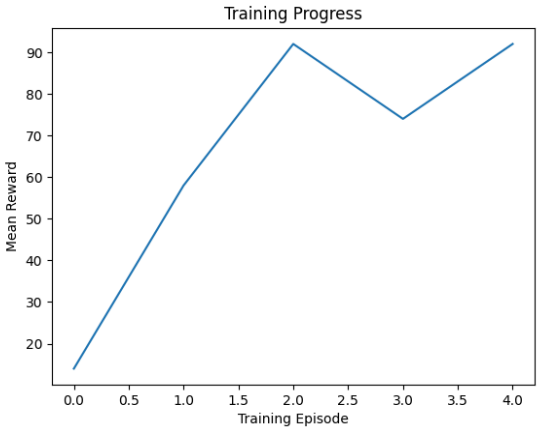
Results

The following two histograms plot the rewards over time for previous iterations of the model during training. These model iterations were kept small to identify and address any issues promptly. Despite their small size, the rewards show an upward trend, indicating improvement in the model's performance.

Early Evaluations:

- **Early evaluation 1**: Mean reward: 169.0 ± 26.25
- **Early evaluation 2**: Mean reward: 194.0 ± 26.53





The final evaluation after implementing the improvements in the 'Approach' section shows a significant improvement:

- **Final evaluation**: Mean reward: 206.0 ± 21.07

The histograms of these evaluations demonstrate the incremental progress made by the model. The trendline added to the final plot further highlights the upward trajectory of the mean rewards, reinforcing the effectiveness of the hyperparameter adjustments and troubleshooting efforts.

Discussion

The upward trend in rewards across training episodes, as shown in the histograms, indicates that the model is learning to perform better in the Frostbite environment. The mean reward increase from early evaluations to the final evaluation suggests that the modifications made to the DQN model and hyperparameters were effective.

The final plot, with the moving average trendline, provides a clear visualization of the model's performance improvement over time. The relatively narrow standard deviation in the final evaluation compared to early evaluations suggests that the model's performance has become more consistent.

Conclusion

Through iterative experimentation and careful tuning of hyperparameters, the DQN model demonstrated significant improvements in the Frostbite environment. The challenges faced during the project, particularly related to memory issues, were effectively addressed, leading to a stable and high-performing model.

Future Work and Suggestions

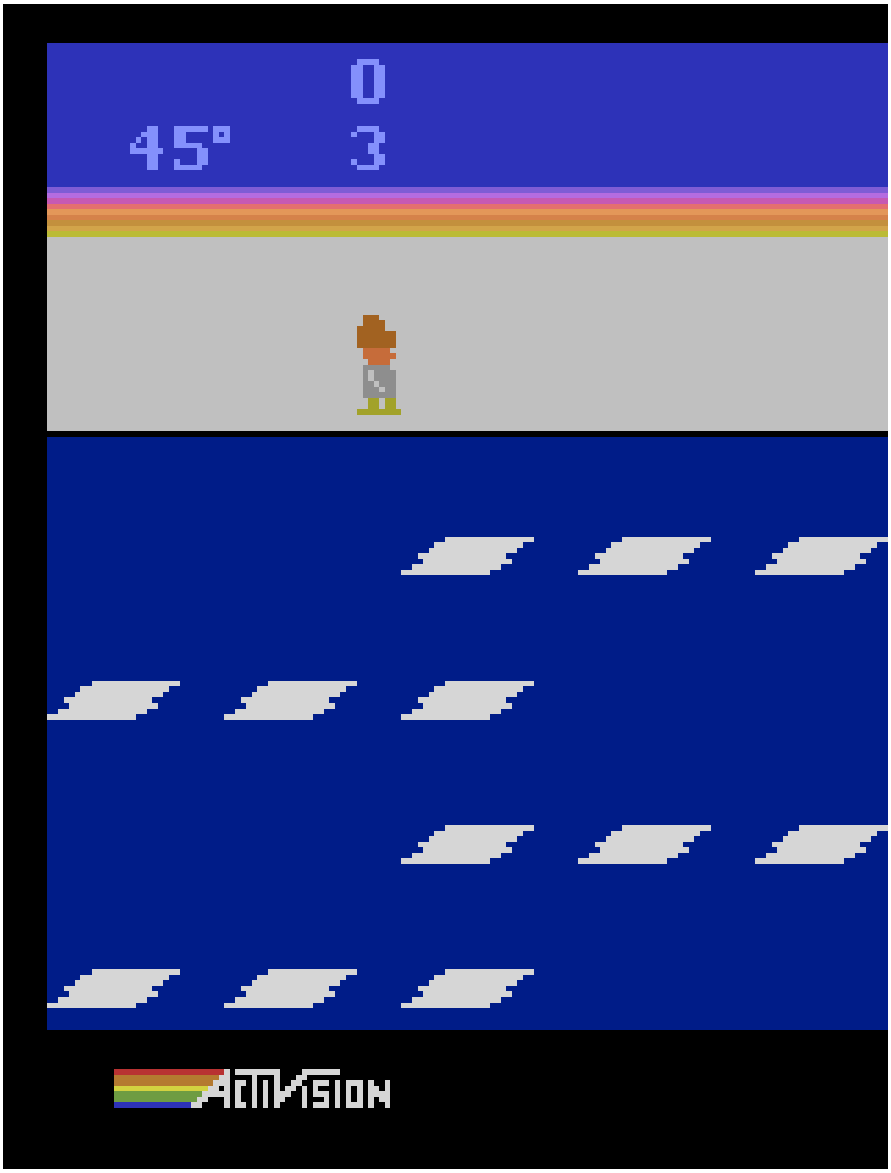
1. **Extend Testing to Other Environments**: To further validate the robustness of the model, it would be beneficial to test it in other Atari environments.
2. **Implement Advanced Techniques**: Incorporate advanced techniques such as Double DQN, Dueling DQN, and Prioritized Experience Replay to enhance the model's performance further.
3. **Increase Training Time**: Allow the model to train for a larger number of episodes to continue improving performance.
4. **Resource Optimization**: Explore more efficient ways to manage memory and computational resources, potentially by using distributed training or optimizing the model architecture.

References

Github Link: https://github.com/gr8tscott/AIFinalProject.git

Frostbite Video:
https://drive.google.com/file/d/1DD2oRN0MijUOd2f8ZfJ00YYdHTfoy5Ff/view?usp=drive_link

Frostbite Gif:
https://drive.google.com/file/d/1zAfwvPwNeaMf1r92OnfdkLKzbItWuP2V/view?usp=drive_link


0
45°
3
ACTIVISION