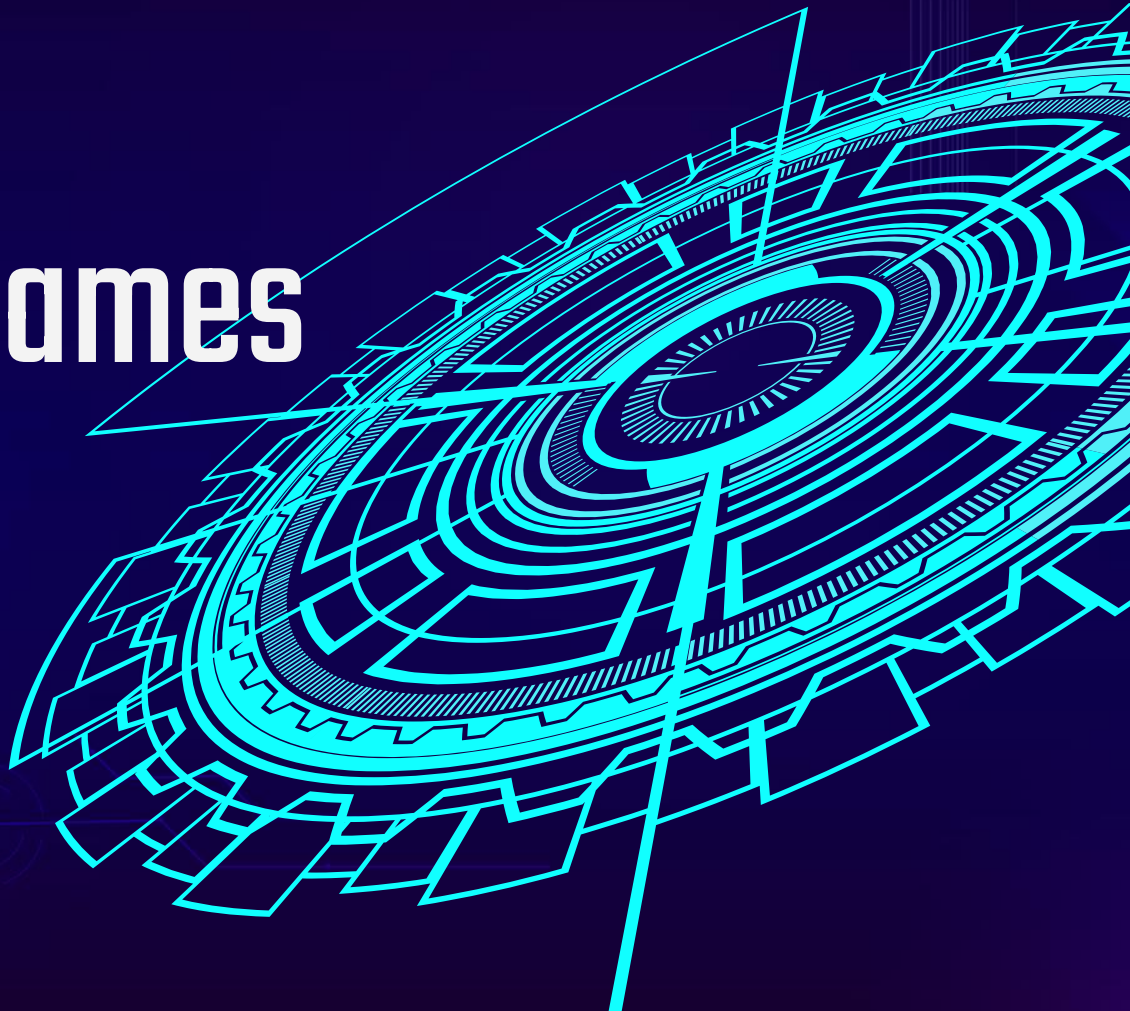
DQN: Atari Games

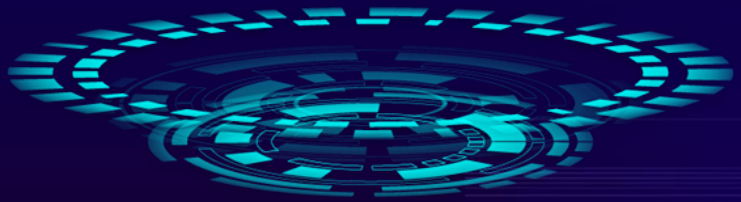


Presentation by Nikita Gaidamachenko

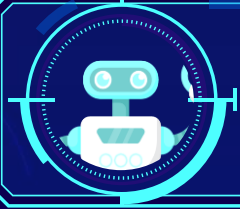
TABLE OF CONTENTS

INTRODUCTION

This project includes the implementation of three deep reinforcement learning models: CartPole, Space Invaders, and Pac-Man. These models utilize DQN to solve tasks related to balancing a pole, playing an arcade-style game, and controlling a character in a maze. Or, in other words, they “play” the games.



Atari Games



Cart Pole

The goal is to keep the pole balanced on the cart by applying left or right forces

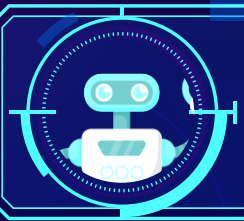
Space Invaders

The model is trained to control a spaceship, shoot down enemy invaders, and avoid projectiles.

Pacman

The goal is to play Pac-Man by learning to navigate the maze, eat pellets, and avoid ghosts

Models



Cart Pole

Algorithm: Deep Q-Learning Network (DQN)

Description:
The goal is to keep the pole balanced on the cart by applying left or right forces. The environment is solved by using a neural network to approximate the Q-value function, which maps states to action values.

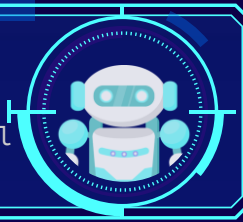
Key Hyperparameters

- Learning Rate: 0.001
- Gamma (Discount Factor): 0.99 (to balance immediate and future rewards)
- Epsilon (Exploration Rate): Starts at 1.0 and decays to 0.01 (using epsilon-greedy for exploration)
- Replay Buffer Size: 100,000 (stores previous experiences to break the correlation between sequential experiences)
- Batch Size: 64 (samples from the replay buffer for training)

Why this approach: DQN is suitable for discrete action spaces like CartPole. Using experience replay and target networks stabilizes the training process. The reward is maximized by learning the optimal way to keep the pole balanced.

Models

Space Invaders



Algorithm: Deep Q-Learning Network(DQN) with Convolutional Neural Networks (CNNs)

Description:
In the Space Invaders environment, the model is trained to control a spaceship, shoot down enemy invaders, and avoid incoming projectiles. A CNN is used to process the pixel-based game screen, and a DQN is used to learn the action-value function.

Key Hyperparameters

- Learning Rate: 0.00025 (lower than CartPole due to the complexity of the environment)
- Gamma (Discount Factor): 0.99 (to emphasize long-term rewards)
- Epsilon (Exploration Rate): Starts at 1.0 and decays to 0.1 over 1 million steps (slow decay to handle the complexity of the game)
- Replay Buffer Size: 1,000,000 (due to the high variance in gameplay)
- Batch Size: 32 (to balance memory usage and learning stability)
- Frame Skipping: Skips 4 out of every 5 frames to speed up learning and reduce computation.

Why this approach: Space Invaders has a large state space due to the pixel input, making CNNs necessary for feature extraction. The DQN algorithm, combined with techniques like frame skipping and experience replay, is effective in this environment.

Models

Pacman

Algorithm: Double DQN (DDQN) with CNNs and Prioritized Experience Replay

Description:
The goal is to play Pac-Man by learning to navigate the maze, eat pellets, and avoid ghosts. This model uses Double DQN to avoid overestimation bias in Q-learning and prioritized experience replay to focus on more important transitions.

Key Hyperparameters

- Learning Rate: 0.0001 (fine-tuned for the complexity of Pac-Man)
- Gamma (Discount Factor): 0.99 (to balance immediate pellet consumption with long-term survival)
- Epsilon (Exploration Rate): Starts at 1.0 and decays to 0.1 over 1 million steps (slow decay for more exploration early on)
- Replay Buffer Size: 1,000,000 (large buffer due to the complexity of the game)
- Batch Size: 32
- Frame Skipping: 4 (similar to Space Invaders to reduce computation)
- Prioritized Experience Replay: Ensures that more important experiences (those with higher TD error) are replayed more often, speeding up the learning process.

Why this approach: Pac-Man has a complex environment with many moving parts. Using Double DQN mitigates overestimation issues, while prioritized experience replay helps the model focus on learning from critical experiences faster. CNNs are essential for processing the pixel input and extracting relevant features. *Additionally*, multi-tier rewards system helps to recognize which parts are essential for achieving a high score.

DELTALIGHT GAMES

THANKS

Presentation prepared by Nikita Gaidamachenko

If you have any questions or comments, please reach out to me on Discord

CREDITS

This is where you give credit to the ones who are part of this project.

- Presentation template by Slidesgo
- Icons by Flaticon
- Infographics by Freepik
- Author introduction slide photo created by Freepik
- Text & Image slide photo created by Freepik.com