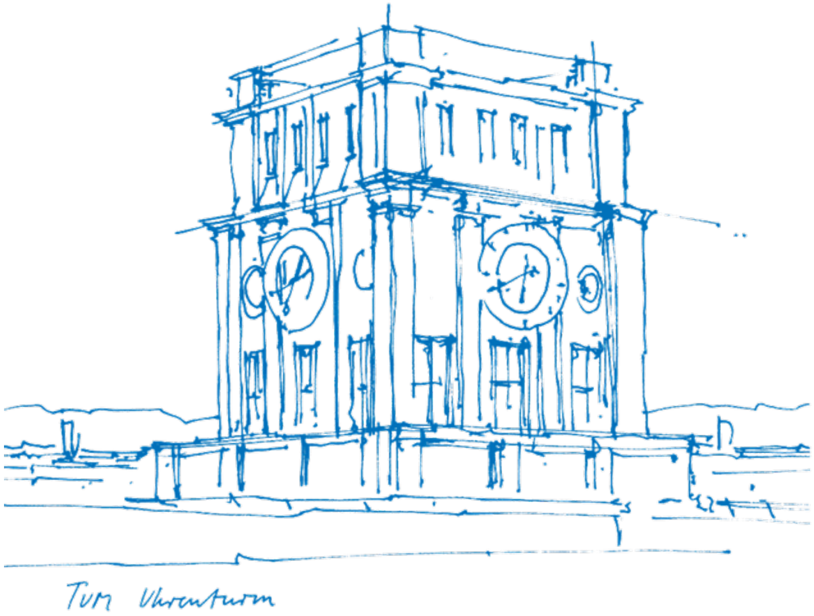
Title

Alexandros Stathakopoulos, Mohanad Kandil
Technische Universität München
Heilbronn, 09. Dezember 2024



Introduction

- Checkers is a strategy game involving diagonal moves of pieces and captures by jumping over opponent pieces.
- The objective of the project is to create an AI player using reinforcement learning.
- Techniques: Q-learning and Deep Q-Networks (DQN).

What is Reinforcement Learning?

- A machine learning paradigm focused on training agents to make sequences of decisions.
- Key components:
  - **Agent**: Learns to act in an environment.
  - **Environment**: The system with which the agent interacts.
  - **Reward**: Feedback signal indicating the success of an action.
- Goal: Maximize cumulative rewards over time.

Deep Q-Learning

- reinforcement learning technique to find the optimal action-value function `Q(s,a)`
- It maps state-action pairs to their expected future rewards
- Q-Update rule:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma\ max_{a'}\ Q(s',a') - Q(s,a)]$$

Deep Q-Learning

- reinforcement learning technique to find the optimal action-value function `Q(s,a)`
- It maps state-action pairs to their expected future rewards
- Q-Update rule:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma\ max_{a'}\ Q(s',a') - Q(s,a)]$$

Challenge: Traditional Q-learning fails for large or continuous state spaces

Deep Q-Learning: Solution

Use a **Deep Neural Network** (DNN) as a function approximator for $Q(s, a)$ instead of a table.

- **Input**: State s, **Output**: Q-values for all actions a
- **Target Network**: stabilizes training by holding fixed weights for a few updates
- **Experience Replay**: samples random batches of past experiences (s, a, r, s') to break correlation in training data

Deep Q-Learning Workflow

Use a **Deep Neural Network** (DNN) as a function approximator for $Q(s, a)$ instead of a table.

Deep Q-Learning Workflow

Use a **Deep Neural Network** (DNN) as a function approximator for $Q(s, a)$ instead of a table.

1. Initialize:
   Q-Network $Q(s, a; \theta)$ and Target Network $Q'(s, a; \theta^-)$

Deep Q-Learning Workflow

Use a **Deep Neural Network** (DNN) as a function approximator for $Q(s,a)$ instead of a table.

1. Initialize:
   Q-Network $Q(s,a;\theta)$ and Target Network $Q'(s,a;\theta^{-})$
2. Action Selection:
   Use an $\varepsilon-$ greedy policy to balance exploration vs. exploitation.

Deep Q-Learning Workflow

Use a **Deep Neural Network** (DNN) as a function approximator for $Q(s, a)$ instead of a table.

1. Initialize:
   Q-Network $Q(s, a; \theta)$ and Target Network $Q'(s, a; \theta^{-})$
2. Action Selection:
   Use an $\varepsilon-$ greedy policy to balance exploration vs. exploitation.
3. Store Experience:
   Add (s, a, r, s') to a replay buffer

Deep Q-Learning Workflow

Use a **Deep Neural Network** (DNN) as a function approximator for $Q(s,a)$ instead of a table.

1. Initialize:

   Q-Network $Q(s,a;\theta)$ and Target Network $Q'(s,a;\theta^-)$
2. Action Selection:

   Use an $\varepsilon-$ greedy policy to balance exploration vs. exploitation.
3. Store Experience:

   Add (s,a,r,s') to a replay buffer
4. Train the Network:
   - sample a minibatch from the buffer
   - Compute target Q-values: $y = r + \gamma\, max_{a'}\; Q'(s',a';\theta^-)$
   - Minimize loss: $L(\theta) = (y - Q(s,a,;\theta))^2$

Deep Q-Learning Workflow

Use a **Deep Neural Network** (DNN) as a function approximator for $Q(s,a)$ instead of a table.

1. Initialize:

   Q-Network $Q(s,a;\theta)$ and Target Network $Q'(s,a;\theta^{-})$

2. Action Selection:

   Use an $\varepsilon-$ greedy policy to balance exploration vs. exploitation.

3. Store Experience:

   Add (s,a,r,s') to a replay buffer

4. Train the Network:
   - sample a minibatch from the buffer
   - Compute target Q-values: $y = r + \gamma\, max_{a'}\; Q'(s',a';\theta^{-})$
   - Minimize loss: $L(\theta) = (y - Q(s,a,;\theta))^2$

5. Update Target Network:

   Periodically copy weights: $\theta^{-} \leftarrow \theta$

Key Improvements and Applications

Improvements:

- **Double DQN**: Reduces overestimation of Q-values.
- **Dueling DQN**: Separates value and advantage functions.
- **Prioritized Experience Replay**: Samples important experiences more frequently.

Key Improvements and Applications

Improvements:

- **Double DQN**: Reduces overestimation of Q-values.
- **Dueling DQN**: Separates value and advantage functions.
- **Prioritized Experience Replay**: Samples important experiences more frequently.

Applications:

- Games
- robotics
- autonomous systems

Future Work

- Fine-tuning the DQN model for improved decision-making.
- Incorporating advanced techniques like Double DQN and Dueling DQN.
- Testing against human players to evaluate real-world performance.
- Extending the approach to other board games or strategy games (some side-projects).

Project Planning & Tiimeline

- Iterative Development: 4 sprints $\rightarrow$ 4 weeks
- TODO: insert image here

1.

Introduction

title

lorem ipsum