Project Proposal: Training a Deep Q-Network to Play Chess

Devin Williams

Goal

The goal of this project is to design and train a Deep Q-Network (DQN) capable of learning to play chess through reinforcement learning. By interacting with a simulated chess environment, the model will learn to evaluate board states and select actions that maximize its expected long-term reward. The primary motivation for this project is to explore how reinforcement learning, especially deep Q-learning, can manage an environment with a very large state and action space, delayed rewards, and the need for long-term planning. Ultimately, the goal is to produce an agent that can play full games of chess at a competitive beginner-to-intermediate level, demonstrating recognizable strategic behavior.

Approach

The environment will be implemented using the open-source *python-chess* library, which provides full game logic, move validation, and board representation. Each chessboard state will be encoded as an 8×8×12 tensor (one plane for each piece type and color), allowing the DQN to process spatial relationships between pieces. The neural network will output Q-values corresponding to possible legal moves. During training, the agent will play self-play matches to explore a wide range of board positions, gradually improving its decision-making through trial and error.

To stabilize learning, the model will incorporate key DQN techniques such as **experience replay**, **epsilon-greedy exploration**, and a **target network** to decouple learning targets from current predictions. Training will be implemented in PyTorch, leveraging GPU acceleration for faster convergence. The project will begin with a simplified setup (shorter games, limited move depth), then scale up in complexity as performance improves.

Measures of Success

Progress will be evaluated using several criteria:

1. **Quantitative Performance:** The agent's win rate against baseline opponents such as a random move generator or a shallow minimax engine.
2. **Learning Curve:** Increases in average episodic reward and reductions in random-move frequency across training epochs.
3. **Qualitative Strategy:** Observation of emergent chess principles, such as piece development, center control, or avoidance of blunders.
4. **Generalization:** The model's ability to play new positions it has not encountered during training.

Resources

- **Hardware:** Google Colab.
- **Software:** Python 3.12, PyTorch, *python-chess*, NumPy, Matplotlib, and optionally TensorBoard for tracking metrics.

- **References:** OpenAI's foundational DQN research, DeepMind's *AlphaZero* paper for conceptual guidance, and existing open-source implementations for move encoding and training loop design.

Expected Outcome

By the end of the project, I expect to have a functioning DQN model that can play complete chess games without human intervention and demonstrate steady improvement through self-play. The project will provide valuable hands-on experience in reinforcement learning, state representation, and the challenges of applying deep learning to complex sequential decision problems. I also want to be able to play against the model through a terminal based interface.