Multi-Label Neural Detection of Motifs in Chess Positions

Detecting tactical and positional chess patterns using neural networks

Project Description

Problem

- Modern chess engines excel at calculating optimal moves but often fail to explain *why* those moves are strong in terms that humans understand. This project aims to bridge that gap by building a neural network capable of detecting tactical and positional motifs in chess positions.

Solution

- Using a large dataset of annotated chess puzzles, the system will analyze board states and output the motifs present in each position as a multi-label classification problem.

Outcome and Impact

- The end goal is to create a fast, interpretable system that enhances chess learning by identifying patterns such as forks, pins, and discovered attacks.
- The tool is designed primarily as a research project, focusing on neural network design, dataset engineering, and motif detection accuracy rather than production deployment.

Features and Requirements

Motif Detection

Requirements:

- Detect well-established chess motifs (e.g., pins, forks, discovered attacks, weak squares, open files).
- Accept any valid FEN position as input.
- Target macro F1 ≥ 0.60 on the test set, balancing precision (correct positive predictions) and recall (ability to identify all true motifs).
- Support expansion as additional motifs are introduced.

Multi-label Output

Requirements:

- Allow multiple motifs per position.
- Represent outputs as a binary vector (~130 motifs).
- Use sigmoid activation for independent motif probabilities.
- Evaluate using multi-label metrics such as F1 score, precision, and recall.

Engine Analysis Integration

Requirements:

- Provide engine evaluations for positions.
- Optionally display best moves alongside detected motifs.
- Use engine output alongside neural network for contextual analysis

Readable UI

Requirements:

- Display chess boards in ASCII/terminal format.
- Output detected motifs in a clear list.
- Maintain fast response time for near real-time analysis.

Overall

- Features: 4
- 14 Requirements
- Projected Tests: 20

Schedule

Sprint 1

- Week 1: Finish research and architecture design
- Week 2: Data Processing
- Week 3: Construct Initial Neural Network
- Week 4: Basic UI and Testing

Sprint 2

- Week 5: Network Tuning & Training
- Week 6: Advanced Features and Engine Integration
- Week 7: UI improvements and Evaluation
- Week 8: Finish Research Paper and Documentation and Flex

Design Goal:

Start small for rapid experimentation, then scale dataset and model complexity in Sprint 2.

Learning With AI

Neural Network Architecture

Why I want to learn it

- Neural Networks are a core part of modern AI and Machine Learning
- Have lots of potential applications

How I will learn with AI

- Discovery of new terms
- Summarizing for subtopics
- Providing examples

Types of Neural Networks

- Perceptrons
- Feed Forward Networks (FNNs)
- Convolutional Neural Network (CNNs)
- And more

Activation Functions

- Decide how much signal the node sends
- Without them neural networks are basically just $ax + b = y$
- They introduce non-linearity by modifying the results of the linear transformation of each node
- This allows neural networks to approximate any complex function

Data Visualization

Why I want to learn it

- I want to be able to evaluate my neural network

How will I learn with AI

- Explain new concepts
- Generate guides
- Suggest libraries and tools to use