Neural Network-Based Chess Motif Detection

Project Description

- 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.

- 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.
- The tool is designed primarily as a research project, focusing on neural network design, dataset engineering, and motif detection accuracy rather than production deployment.

- The end goal is to create a fast, interpretable system that enhances chess learning by identifying patterns such as forks, pins, and discovered attacks.

Features and Requirements

Core Features

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.
- Achieve performance significantly better than random classification.
- 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 (Optional)

Requirements:

- Provide engine evaluations for positions.
- Optionally display best moves alongside detected motifs.
- Use engine output as a validation mechanism for predictions.

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.

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

Spring 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 Documention and Flex

Dataset

- Source: Lichess puzzle dataset (via Kaggle).
- File size: ~460MB CSV containing hundreds of thousands of puzzles, with millions available.
- Sprint 1 subset: **10,000 puzzles** for proof-of-concept.

Key Fields Used:

- FEN (board state)

- Themes (motif labels)

- Moves (optional auxiliary data)

- Rating / Popularity (optional filtering for data quality)

Design Goal:

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