Human Chessbot

A Chess Application Suite with Machine Learning Capabilities

Project Overview

A comprehensive chess application suite featuring:

- **Interactive chess gameplay** with multiple engine support
- **Data processing pipeline** for chess game analysis
- **ML training infrastructure** for chess AI development

Built with Python 3.11+ and modern ML frameworks

Key Features

Play Package

- Interactive GUI and CLI interfaces
- Multiple chess engines (Stockfish, LCZero, Random bot)
- Time controls and automatic game recording

Convert Package

- PGN file processing and combination
- CSV conversion for ML training
- Batch processing capabilities

Architecture

```
human-chessbot/
├── packages/
│   ├── play/            # Chess game application
│   ├── convert/         # PGN conversion utilities
│   └── train/           # ML training pipeline
├── docs/                # Documentation
└── pyproject.toml       # Project configuration
```

Modular design enables independent package development and testing

Play Package - Interactive Chess

Features:

- Human vs Bot gameplay
- Bot vs Bot matches
- Multiple difficulty levels
- Real-time move validation

Supported Engines:

- Stockfish (advanced)
- LCZero (neural network-based)
- Random bot (testing)

Convert Package - Data Pipeline

Purpose: Transform raw PGN chess data into ML-ready format

Capabilities:

- Combine multiple PGN files
- Parse chess notation
- Extract game metadata
- Export to CSV format

Use Case: Prepare training datasets from online chess databases

Train Package - ML Infrastructure

Dataset Management:

- Automated Lichess data fetching
- Game snapshot processing
- Legal move generation
- Statistical analysis

Training Pipeline:

- Neural network architecture
- Custom dataset loaders
- Training configuration

Technical Implementation

Core Technologies:

- Python 3.11+
- Chess engine integration (Stockfish, LCZero)
- SQLite for data management
- PyTorch for ML training
- Pygame for GUI

Code Quality:

- Pre-commit hooks (ruff, black, isort, mypy)
- Comprehensive test coverage

Development Workflow

```
# Setup
python -m venv .venv
source .venv/bin/activate
pip install -e .
pre-commit install

# Run application
python -m packages.play.src.main

# Run tests
pytest packages/*/tests/ -v

# Code quality
pre-commit run --all-files
```

Challenges & Solutions

Challenge: Multiple chess engine integration
Solution: Abstract player interface with engine-specific implementations

Challenge: Large dataset processing
Solution: Incremental processing with SQLite backend

Challenge: Maintaining code quality across packages
Solution: Automated pre-commit hooks and comprehensive testing

Results & Achievements

- Fully functional chess game with multiple engines
- Automated data processing pipeline
- ML training infrastructure ready for model development
- Comprehensive test coverage
- Clean, maintainable codebase

Future Work:

- Complete neural network training
- Additional chess engines
- Enhanced UI features

Demo

Live demonstration of:

1. Chess gameplay (GUI/CLI)
2. PGN file processing
3. Dataset preparation
4. ML training pipeline

Questions?

Human Chessbot

A complete chess AI development platform

Thank you!