Developing a Chess AI with Convolutional Neural Networks

A deep learning approach to predicting optimal chess moves using modern AI techniques

Why Chess is Perfect for AI Learning

Well-defined Rules

Chess has clear rules and a finite state space, making it easier to model computationally

Historical AI Benchmark

From Deep Blue to AlphaZero, chess has rich literature and frameworks for AI development

Scalable Complexity

Simple moves to deep strategic planning provide a natural learning curve

Data Availability

Vast databases of chess games make training AI models more accessible

Our goal: Develop a CNN that can play with either colour, at any point in the game, predicting legal moves with special attention to openings and endgames.

What is a CNN?

A **Convolutional Neural Network (CNN)** is a specialised computer program that helps machines recognise patterns in visual data. It works similarly to how humans process images:

- Scans images in small parts to identify important details
- Recognises shapes, edges, and patterns
- Combines these features to understand the complete image

CNNs are widely used in facial recognition, self-driving cars, and medical imaging—helping computers "see" like humans do.

Why Use CNN for Chess Move Prediction?

Chessboard as a Spatial Grid

An 8×8 grid similar to image pixels, perfect for CNN's spatial pattern recognition

Pattern Recognition

CNNs excel at identifying chess patterns like forks, pins, and checkmate threats

Local Feature Extraction

Detects piece clusters, threats, and control over squares to predict optimal moves

Efficient Processing

Processes the board like an image, reducing complexity and focusing on important regions

Top chess engines like AlphaZero use CNNs to evaluate positions and choose moves like grandmasters.

Our CNN Approach

We're creating a **Convolutional Neural Network** to predict chess moves, similar to AlphaZero and Leela Chess Zero, but with key differences:

- Instead of self-play learning, we'll train using human-played games
- Need to represent the chessboard and moves in AI-friendly format
- Must create a diverse dataset of chess positions
- Ensure suggested moves are both valid and strong

This approach presents unique challenges but allows us to leverage human expertise in training our model.

Representing the Chessboard for AI

Single Matrix Problem

Using one matrix with different numbers for pieces (e.g., 6=king, 1=pawn) causes the CNN to misinterpret piece importance

Better Solution

Use separate binary matrices (0s and 1s) for each piece type—one for white pawns, one for black knights, etc.

AlphaZero Approach

This multi-matrix approach treats all pieces equally and improves move predictions by focusing on positions

Preparing Moves Data

AlphaZero uses a 119-channel matrix to track all possible moves. We'll use a similar approach:

Representing the Best Move (Target)

- Use matrix representation instead of simple notation
- Helps AI learn patterns of strong moves

Representing All Legal Moves

- Show all possible legal moves for every piece
- AI knows both piece positions and where they can move

Data Format: 77-Layer Matrix

1

Layers 1-12: Piece Positions

Separate binary matrices for each piece type and colour (white pawns, black knights, etc.)

2

Layer 13: Turn Indicator

8×8 matrix of all 1s (black's turn) or all 0s (white's turn)

3

Layers 14-69: Queen Moves

56 matrices representing possible queen moves in 8 directions (7 squares in each direction)

4

Layers 70-77: Knight Moves

8 matrices representing the knight's unique L-shaped movement patterns

This format is stored efficiently using int64 compression, reducing dataset size from potential 10GB to about 1.5GB.

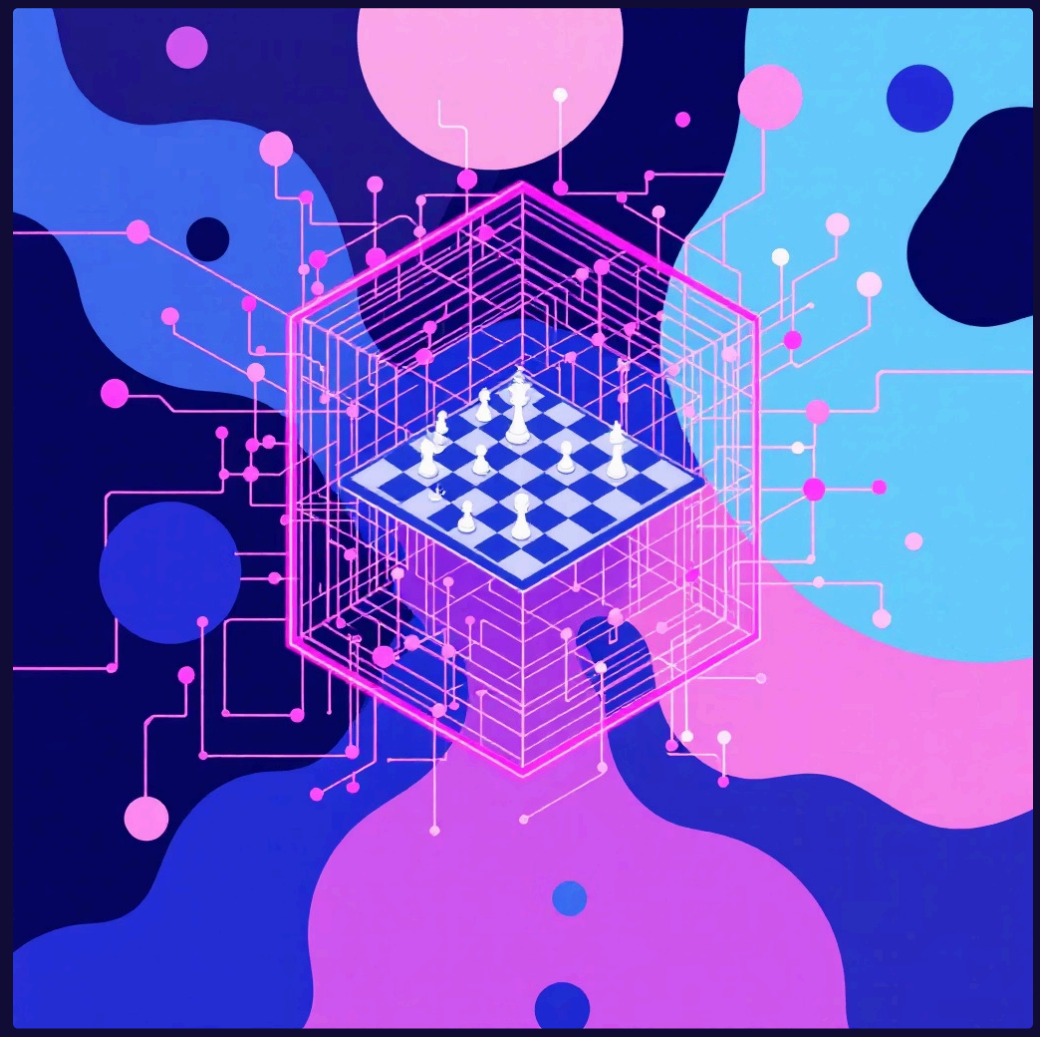
Why This Structure Works

Benefits for CNN Processing

- Hierarchical Features: Learn distinctive patterns for each piece type
- Spatial Information: Preserves relative positions crucial for chess
- Discriminative Learning: Recognizes unique patterns for each piece
- Flexibility: Easily adapts to changing board configurations
- Interpretability: Each matrix corresponds to a specific aspect

Move Representation Benefits

- Alignment with Board: Natural 8×8 grid structure
- Complete Information: Captures all possible moves
- Spatial Relationships: Maintains positional context
- CNN Compatibility: Ideal for convolutional processing



Implementation Summary

77

Input Layers

8×8 matrices representing board state and possible moves

4

CNN Layers

Convolutional layers with batch normalization

4096

Output Moves

Possible move combinations the model can predict

1M

Training Positions

Chess positions available for training (starting with 200k)

Our implementation combines CNN with optional Monte Carlo Tree Search for look-ahead capability, deployable via Streamlit, Gradio, or Hugging Face Spaces with proper model saving techniques.

Training strategy: 80/20 train/test split, 64-128 batch size, 10-30 epochs with early stopping to prevent overfitting.