Chess2vec: Learning Vector Representations for Chess

Berk Kapicioglu *OccamzRazor*

Ramiz Iqbal *MD Anderson*

Tarik Koc *OccamzRazor*

Louis Nicolas Andre *OccamzRazor*

Katharina Sophia Volz *OccamzRazor*

Computer Chess

Computer chess studies algorithms that play chess.

Deep Blue, developed by IBM, defeated the world chess champion in 1996.

Historically, chess programs, such as Deep Blue, did not leverage **machine learning (ML)**.

Instead, they leveraged tree search, minimax, alpha-beta pruning, and hand-crafted position evaluation algorithms.

Chess and Machine Learning

Modern computer chess programs, such as **AlphaZero** and **DeepChess**, are fundamentally based on machine learning.

How is machine learning used in computer chess?
One way is to learn a function that maps chess positions to board moves.

Modern computer chess programs use machine learning to learn this function or one of its variants.

Our Contribution

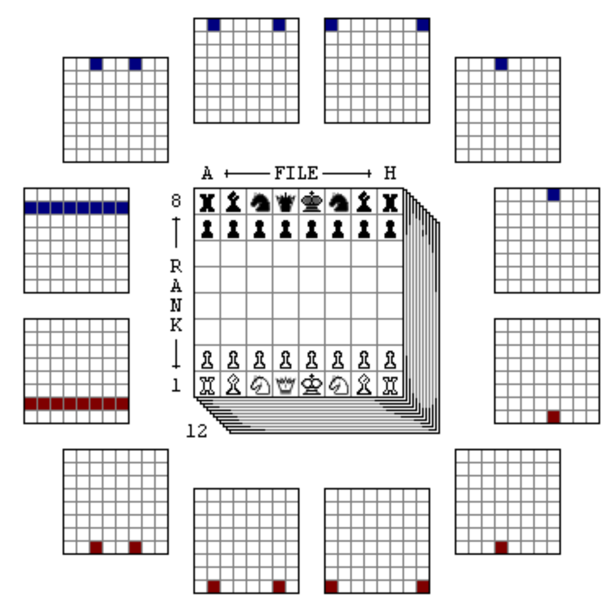
In order to train models and make inferences on chess positions, one needs a computational way to represent them.

In modern applications, chess positions are typically represented via the bitboard representation.

In this work, we conduct the first qualitative and quantitative study about learning alternative representations for chess pieces and positions.

Bitboard Representation

The standard representation of chess positions is the **bitboard representation**. It is a $8 \times 8 \times 12$ binary tensor.



Key insight: The bitboard representation can be specified by assigning a 12-dimensional indicator vector to each piece type and color.

Key question: Are there better vector representations that can be assigned to each piece type and color?

Datasets

We generate two datasets using Stockfish, the leading computer chess engine in the world.

In each dataset, we log the number of times a white piece has been observed to make a specific board move.

1) Legal Moves Dataset
Generate legal moves for each piece type on an empty board.

2) Expert Moves Dataset
Generate expert moves by pitting a white Stockfish agent play against a weaker black Stockfish agent.

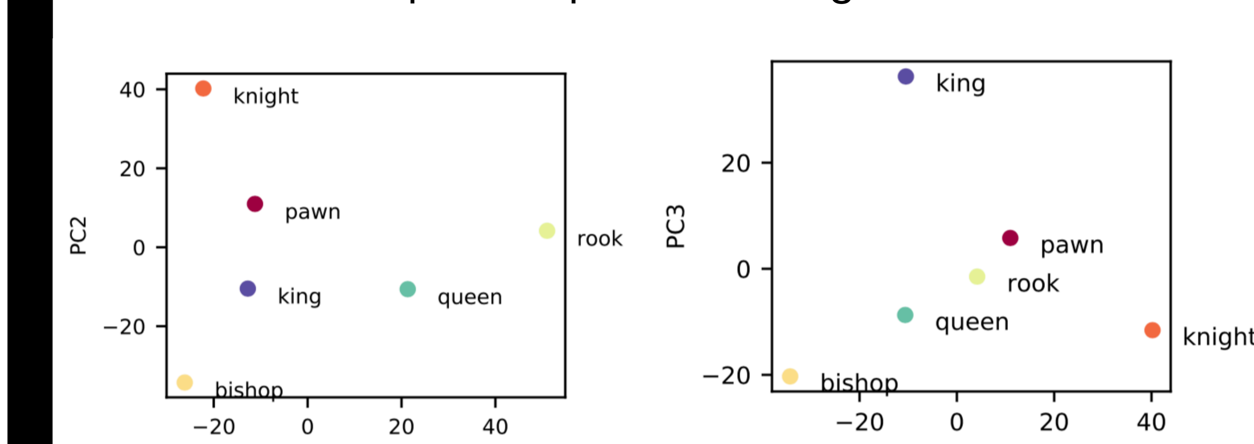
Each dataset yields $X \in \mathbb{R}^{16 \times 4096}$.

Key idea: Pieces with similar movement patterns have similar features.

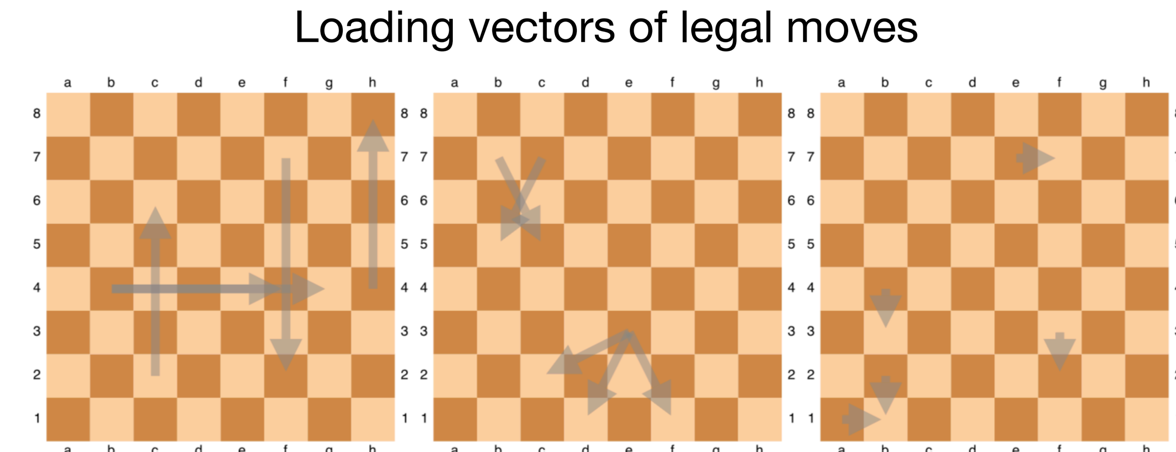
PCA

We decompose the matrix and obtain both chess piece embeddings and a qualitative understanding of the fundamental building blocks of chess moves.

Principal components of legal moves

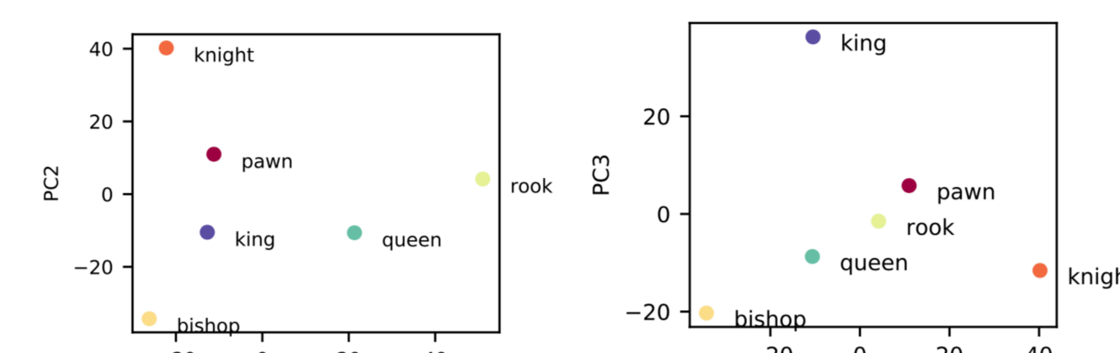


Loading vectors of legal moves

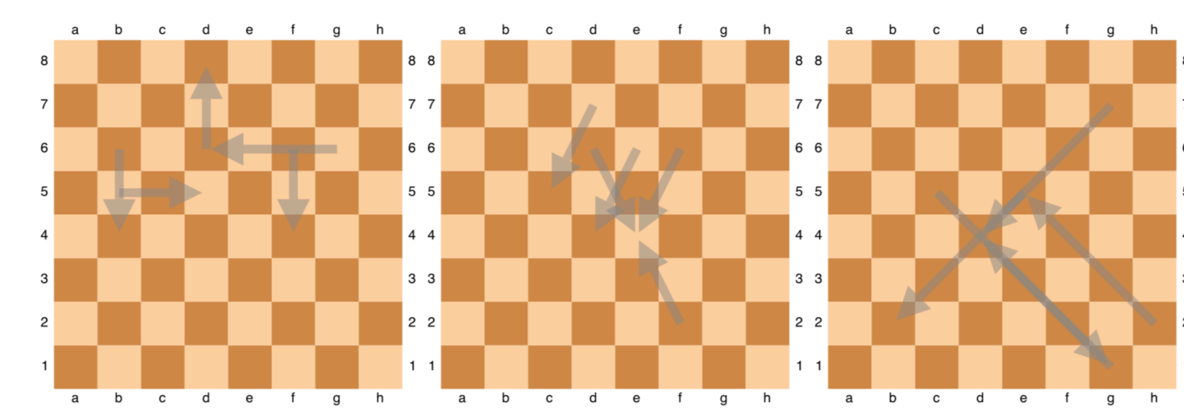


PCA (continued)

Principal components of Stockfish moves



Loading vectors of Stockfish moves



Read the paper for a discussion on the **insights** this decomposition provides about the game of chess!

Position-dependent Piece Vectors

Potential Problem: Piece vectors are constant with respect to chess positions.

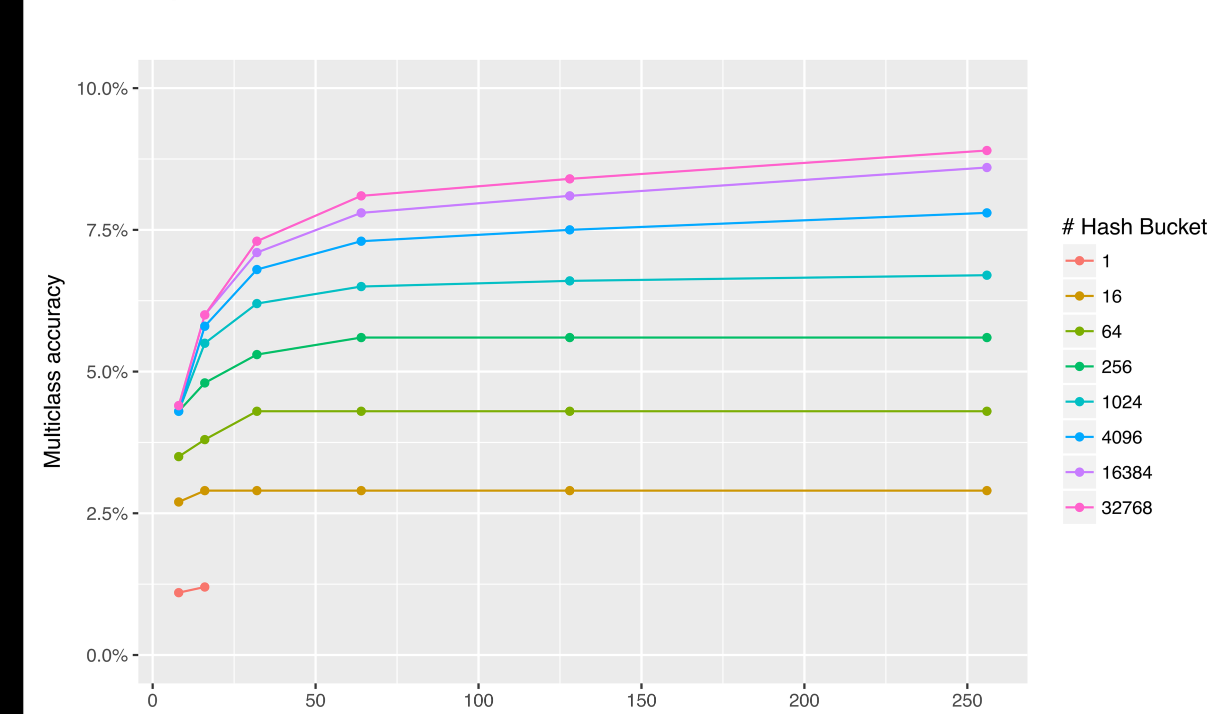
Solution: Use Zobrist hashing!

Instead of generating a unique vector for each piece, we generate one for each piece and hash bucket. The hash partitions the space of all chess positions.

Intuition: The reconstruction approximates the number of times a move has been selected for all the chess positions that are hashed into that bucket.

Experiments

Increasing the number of hash buckets improves the accuracy of predicting the chosen board move given a chess position on held-out.



Ongoing and Future Work

1) Implemented a neural network to compare how different chess representations perform in mapping chess positions to observed moves.

2) Implemented a chess environment in OpenAI gym to compare how different chess representations perform when playing against a Stockfish agent.