Proposal: Optimizing Chess Engine Training Efficiency via Guided Curriculum Self-Play

Problem Statement and Motivation

Modern chess engines like Stockfish, Leela Chess Zero, and AlphaZero have surpassed human grandmasters by leveraging deep neural networks and massive self-play training (Silver et al., 2017; Linscott et al., n.d.; Stockfish Development Team, n.d.). However, this success comes with an enormous computational and energy cost. Training these engines from scratch requires playing millions of full games, where early moves are often random and uninformative. This wastes computation and electricity while the model slowly discovers basic patterns.

As AI systems scale, this inefficiency becomes increasingly unsustainable. Reducing the energy cost of training could accelerate AI innovation while lowering its environmental impact. This project proposes a curriculum-based training strategy to improve early learning efficiency, reducing both convergence time and energy consumption without sacrificing playing strength.

Background: The Game of Chess

Chess is a two-player, turned based, deterministic strategy game played on an 8×8 grid of 64 squares. Each player commands sixteen pieces: one king, one queen, two rooks, two bishops, two knights, and eight pawns. Player's pieces are arranged symmetrically at the start of the game. The objective is to checkmate the opponent's king, meaning the king is under direct attack and cannot escape capture in a single turn. Each piece moves according to fixed rules: pawns advance forward but capture diagonally; knights move in an "L" pattern; bishops move diagonally; rooks move horizontally or vertically; the queen combines rook and bishop movement; and the king moves one square in any direction. Chess is a game of perfect information, as both players always have complete visibility of the game state, with no hidden information or randomness. This property makes chess a well-suited environment for studying machine learning systems that depend on fully observable decision-making processes.

Related Work

Traditional chess engines combine two main components:

1. Search algorithms such as minimax with alpha-beta pruning, which explore game trees and prune suboptimal branches.

2. Evaluation functions that estimate board strength, historically using handcrafted heuristics (material, king safety, pawn structure) but increasingly based on neural networks trained via reinforcement learning (Silver et al., 2017; Stockfish Development Team, n.d.).

Leela Chess Zero and AlphaZero demonstrated the power of self-play reinforcement learning, but their approach requires massive compute resources to achieve top-tier performance. Prior works have shown that curriculum learning — starting with simpler tasks and gradually increasing difficulty — can speed up neural network training in other domains, though this has not been extensively tested for self-play chess engines (Bengio et al., 2009). This project aims to close that gap.

Initial Hypotheses

1. A guided curriculum self-play strategy, starting with short games and gradually increasing game length, will accelerate early learning and produce higher Elo strength per unit of energy than unguided full-game self-play.
2. The curriculum-based approach may develop a bias toward positions with higher material count even when they are not necessarily strategically better, potentially affecting its long-term decision-making compared to the baseline

Goals

1. Develop two neural network evaluation functions within the same chess engine framework:
- Baseline (Unguided): Trained via full-game self-play from the beginning.
- Guided (Curriculum): Trained on progressively longer games (e.g., 10–20 moves per side increasing to full games).
2. Measure training efficiency in terms of:
- Energy consumption (electricity usage during training)
- Convergence rate (Elo improvement over time)
3. Compare final engine strength (Elo ratings) and calculate Elo per unit of energy consumed.

Proposed Solution

The proposed approach implements two distinct self-play training regimes using identical engine code and network architectures:

- Unguided Full-Game Self-Play: Standard approach where the network plays complete games from the outset, receiving win/loss rewards.

- Guided Curriculum Self-Play: Begins with short games (e.g., 10–20 moves per side) and gradually increases the maximum allowed length as performance improves. Intermediate rewards are based on material advantage at a set move horizon, with full rewards given for checkmates.

Training logs will record both GPU/CPU energy usage and performance metrics over time. By controlling hardware and software differences, any observed performance gap can be attributed to the training curriculum itself.

Evaluation

Evaluation will involve two major components:

- Model Performance Metrics:
  - Elo rating against standard benchmark engines - Convergence curves showing Elo vs. training steps (Elo, 1978; Henderson et al., 2018)
- Energy Efficiency Metrics:
  - Total energy consumed during training (kWh)
  - Elo gained per kWh consumed

The final analysis will present a side-by-side comparison of both training regimes on these metrics, identifying which approach produces the strongest engine for the least energy.

References

- Silver, D. et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. arXiv (2017).
- Linscott, J. et al. Leela Chess Zero. https://lczero.org/
- Stockfish Development Team. Stockfish Chess Engine. https://stockfishchess.org/ Bengio, Y. et al. Curriculum Learning. ICML (2009).
- Henderson, P. et al. Deep Reinforcement Learning That Matters. AAAI (2018) — includes methods for tracking energy consumption and compute usage during training.
- Elo, A. E. The Rating of Chessplayers, Past and Present. Arco Publishing (1978) — foundational description of the Elo rating system used to evaluate engine strength.