# Optimizing Chess Engine Training Efficiency via Guided Curriculum Self-Play

CSCE 585: Machine Learning Systems

Vito Spatafora

Milestone P2 — Final Report

# Abstract

This project investigates whether a modified self-play regime can improve the efficiency of training NNUE chess evaluation networks from scratch. A complete reinforcement loop was constructed in which Stockfish, guided by iteratively updated NNUE weights, generates training data through self-play and is retrained on its own experience. Two learning conditions were contrasted. The control condition applied a long, fixed move limit and declared truncated games as draws. The experimental condition introduced short, iteratively increased move limits and material-based adjudication to force decisive outcomes, effectively creating a curriculum that exposes the network to longer horizons only after it learns to resolve shorter games.

Each iteration produced a candidate NNUE that was benchmarked against both its predecessor and a fixed baseline to estimate Elo improvement. Replay-buffer sampling, decisive-result oversampling, persistent state handling, versioned model retention, and reproducible pipeline execution were integrated to ensure stable evaluation. The aim of the experiment is to determine whether this adjudication-and-curriculum approach accelerates convergence and stabilizes learning compared to conventional long-horizon self-play. The work thus provides evidence toward whether shaped supervision and staged move-limit escalation constitute a more efficient pathway for bootstrapping NNUE performance from scratch.

# Introduction

Efficiently Updatable Neural Networks (NNUE) have become integral to state-of-the-art chess engines, enabling strong evaluation performance with low computational overhead. Although NNUE is now widely used in competitive engines, far less is known about how to reliably train these networks from scratch. Existing systems typically rely on inherited evaluation functions, extensive curated datasets, or long self-play schedules where convergence is slow and unstable. This creates a key research problem: can NNUE models be initialized and guided toward strong play more efficiently through alternative supervision strategies?

This work examines that question through a complete self-play reinforcement pipeline in which a Stockfish-based engine generates data, trains an NNUE evaluator, and is benchmarked iteratively. The experiment isolates a central design decision: how to treat unfinished games during early learning stages. The control condition declares truncated games as draws, whereas the experimental condition applies synthetic material-based adjudication and increases allowable game length over iterations. This is essentially saying “whoever has more piece value wins” when the self-play games reach their move limit and are prematurely ended. The resulting setup

pursues two objectives: establishing a functional framework for training NNUE models from scratch and testing whether adjudication-driven supervision with staged horizon growth yields faster or more stable convergence than conventional timeout-based draws.

# Related Works

NNUE gained prominence through Stockfish, yet its published training procedures offer limited insight into how such networks learn effectively from scratch (Stockfish Developers, 2020–2024). Most community efforts reuse pretrained models or rely on large, curated datasets rather than examining first-principles convergence.

Self-play reinforcement learning frameworks such as AlphaZero demonstrated that engines can bootstrap performance without external data, but these approaches use deep policy value networks rather than efficiently updatable evaluators (Silver et al., 2018). Reward shaping and truncated game adjudication have historically appeared in classical chess programming, where material balance often substitutes for full outcomes in search termination, but these ideas have not been systematically applied to NNUE training.

This project sits at the intersection of these strands: NNUE-based evaluation, self-play bootstrapping, and curriculum-style reward shaping. By testing material adjudication and progressive horizon scaling, it addresses an open question in whether alternative supervision can accelerate NNUE convergence.

# Experimental Design

The study evaluates whether curriculum-style material adjudication yields more efficient NNUE convergence than conventional draw-on-timeout supervision. To support this comparison, an end-to-end self-play reinforcement framework was constructed using Stockfish configured with efficiently updatable neural network evaluators. The system iterates through cycles of data generation, network training, benchmarking, and model selection, enabling controlled variation of adjudication rules and move-length scheduling.

## Self-Play Generation

Training data is produced via engine self-play. For each game, Stockfish performs fixed-depth searches (depth=8) with the current NNUE model providing evaluation guidance. Opening positions are sampled randomly from an external opening set to introduce variety. During each move, positions, engine scores, and moves are logged to form training examples. A replay buffer stores large volumes of accumulated data in memory and periodically rebalances decisive outcomes via oversampling to avoid draw-heavy distributions.

### Adjudication Conditions

Two regimes define how truncated games are treated. In the control condition, a long (Move limit = 250), fixed move limit is imposed and all games exceeding this limit are labeled as draws. In the experimental conditions, the maximum move length starts small (Move limit = 20) and increases incrementally across training iterations (Additional 10 moves per iteration, e.g. second iteration has a move limit of 30). Games exceeding the limit are synthetically adjudicated using material balance: a material advantage is assigned as a win or loss, and equality is marked as a draw. This curriculum forces early decisive signals and gradually exposes the network to longer decision horizons.

### Training Pipeline

At the end of each self-play phase, a filtered subset of replay-buffer data is written to disk and converted to a binpack representation suitable for NNUE training (via the nodchip stockfish binary). A PyTorch-based trainer resumes from the previous checkpoint and performs one epoch of optimization, producing a candidate network. Training duration, replay size, and game statistics are logged for later analysis.

### Benchmarking and Model Selection

Once trained, each candidate network is evaluated through engine matches against its immediate predecessor and against a fixed baseline NNUE network (version0). Matches alternate colors to control bias, and outcomes are summarized as wins, losses, draws, and Elo differences using the standard rating approximation. If the candidate network performs above a threshold (Elo difference > -20 for first 4 Iterations, for Iterations 5 and 6 Elo difference > -5), it is promoted to the next version; otherwise, the prior version is carried forward. Versioned checkpoints and NNUE files provide persistent traceability across iterations.

### Runtime Management

The pipeline incorporates state saving with serialized replay buffers and iteration counters to support restarts. Logging infrastructure collects training statistics, enabling comparison of convergence behavior between adjudication regimes.

Implementation Procedure:

The framework was executed for six training iterations under both adjudication regimes. Each iteration used time-limited game generation (10 minutes per cycle), depth-8 search, and deterministic replay extraction. Updated networks were benchmarked through 200-game matches against the previous version and 200-game matches against a fixed baseline (version 0), with Elo estimates derived from head-to-head scoring. All models were trained by resuming from the prior checkpoint, ensuring strict continuity rather than independent restarts. Replay buffers, game outcomes, training durations, seeds, and network paths were logged to CSV and checkpoint files, enabling reproducibility and restart ability.

This design isolates adjudication strategy as the key independent variable, while keeping architecture, search depth, training settings, and benchmarking constant. The dependent measures of interest include convergence pace, Elo progression, decisiveness of play, and stability across iterations.

# Results

Model performance was tracked over six self-play iterations for both adjudication regimes. Each iteration generated training data, produced a candidate NNUE network, and benchmarked that network against both its predecessor and a fixed baseline. Three categories of outcomes emerged: differences in gameplay statistics, iterative benchmarking results, and a final head-to-head comparison between the experimental and control models (each from iteration 6).

## 4.1 Self-Play Dynamics

The adjudication strategy produced markedly different self-play behavior.
In the experimental regime, enforcing short horizons with synthetic adjudication led to high volumes of games per iteration (ranging from 901 to 9,531) and short average game lengths (approximately 20–70 plies). This design exposed the model to frequent decisive outcomes, reflecting the reward shaping intent of the curriculum.

In contrast, the control regime, which treated truncated games as draws under a long move limit, produced far fewer games (1,283–1,455 per iteration) and much longer game lengths (147–153 plies on average). Although this resulted in a comparable number of positions added to the replay buffer, the underlying game resolution distribution was more draw-heavy and temporally extended.

These behavioral differences confirm that curriculum adjudication meaningfully altered the nature of the bootstrapped experience, especially early in training where decisive outcomes were otherwise sparse (Bengio et al., 2009).

| Experimental iteration | games_played | avg_game_length_plies | positions_added | bench_prev_w | bench_prev_l | bench_prev_d | bench_prev_elo | bench_base_w | bench_base_l | bench_base_d | bench_base_elo |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 9531 | 19.99842618822789 | 190605 | 85 | 73 | 42 | 20.87120466602866 | 91 | 67 | 42 | 41.89414020800522 |
| 2 | 3017 | 19.99933708982433 | 60338 | 79 | 78 | 43 | 1.7371924043128577 | 92 | 63 | 45 | 50.73574877909367 |
| 3 | 961 | 39.9729448491155 | 38414 | 71 | 90 | 39 | -33.10621599 | 72 | 72 | 56 | 0 |
| 4 | 901 | 49.90122086570477 | 44961 | 81 | 72 | 47 | 15.64516754533055 | 78 | 72 | 50 | 10.426196175570942 |
| 5 | 1423 | 59.58889669711876 | 84795 | 67 | 84 | 49 | -29.60345765 | 75 | 87 | 38 | -20.87120467 |
| 6 | 1127 | 69.16503992901508 | 77949 | 85 | 65 | 50 | 34.860070287560106 | 77 | 76 | 47 | 1.7371924043128577 |
| | | | | | | | | | | | |
| Control iteration | games_played | avg_game_length_plies | positions_added | bench_prev_w | bench_prev_l | bench_prev_d | bench_prev_elo | bench_base_w | bench_base_l | bench_base_d | bench_base_elo |
| 1 | 1448 | 150.39433701657458 | 217771 | 68 | 66 | 66 | 3.4744716740370483 | 69 | 75 | 56 | -10.42619618 |
| 2 | 1455 | 150.74158075601375 | 219329 | 78 | 76 | 46 | 3.4744716740370483 | 69 | 72 | 59 | -5.211924701 |
| 3 | 1405 | 151.54163701067617 | 212916 | 74 | 80 | 46 | -10.42619618 | 61 | 87 | 52 | -45.42367635 |
| 4 | 1283 | 153.04832424006236 | 196361 | 66 | 75 | 59 | -15.64516755 | 58 | 100 | 42 | -74.06331161 |
| 5 | 1333 | 150.96249062265565 | 201233 | 81 | 77 | 42 | 6.949638428 | 73 | 88 | 39 | -26.10669261 |
| 6 | 1283 | 147.62042088854247 | 189397 | 80 | 73 | 47 | 12.165214579657565 | 56 | 101 | 43 | -79.53375448 |

Key:

| Metric | Definition | Context for this Study |
|---|---|---|
| Games Played | Total self-play games completed per iteration. | Experimental iterations yield higher counts due to capped game lengths. |
| Avg Game Length | Average duration of games in plies (half-moves). | Tracks the curriculum progression (20→70 plies) vs. standard play (~150 plies). |
| Positions Added | Total training examples generated. | Indicates the volume of data available for the epoch. |
| Bench Prev (Elo) | Relative strength vs. the previous iteration. | Positive values indicate incremental learning; negative values indicate regression. |
| Bench Base (Elo) | Absolute strength vs. the untrained (Genesis) network. | The primary indicator of total learning progress over time. |

## 4.2 Iterative Benchmarking Trends

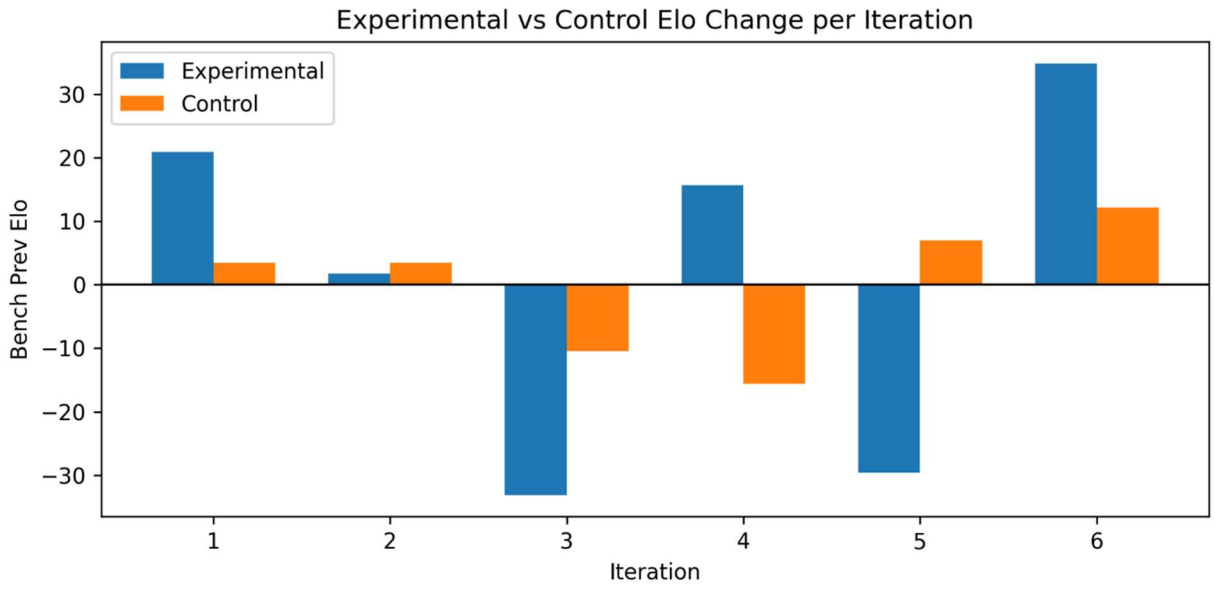
Across both regimes, benchmarking demonstrated noisy but directional improvement over time. However, qualitative differences emerged:

The experimental candidate frequently outperformed its immediate predecessor
Iteration-level Elo shifts relative to the previous model were often positive, with four of six updates registering net improvement (e.g., +20.9, +15.6, +34.9).

The control model exhibited weaker or negative iteration-to-iteration improvement
Its benchmarking swings were more volatile and frequently negative (e.g., –15.6, –45.4, –74.1). Although positive updates occurred, the net trend suggested less stable or slower convergence.

Against the fixed baseline, the experimental regime consistently exceeded parity, reaching iteration-level advantages up to +50 Elo, whereas the control regime repeatedly underperformed baseline reference strength (–10 to –79 Elo).

These results support the hypothesis that adjudication accelerates early learning by providing more consistent directional improvement signals and can be visually seen in the diagram below. Remember, negative Elo change (if substantial) will result in new model not being accepted. This mean the massive regressions we see with the Experimental model don't matter other than to discount that iteration's training (iterations 3 and 5).

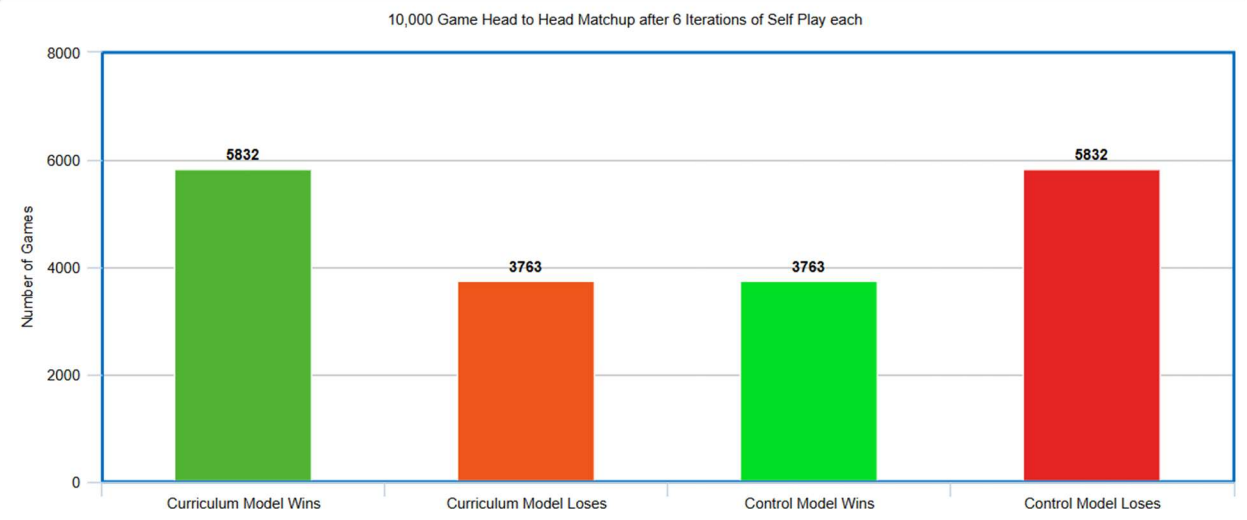


## 4.3 Final Competitive Evaluation

After six training cycles for each the control and experimental setup, the final networks produced by each regime were put against each other in a direct head-to-head matchup via 10,000 games of full chess.
The experimental model won 58.32 percent of encounters, compared to 37.63 percent losses and 4.05 percent draws. A logistic Elo estimator placed its advantage at approximately +72.9 Elo with >99.99 percent likelihood of superiority.

Notably, this advantage emerged despite both models sharing identical architecture, search depth, and training durations, with adjudication strategy being the principal differentiating factor. The match results therefore indicate that shaping supervision with staged horizon expansion produced a model that generalizes more effectively than one trained under standard long-horizon draw treatment.

| Head to Head Matchup after 6 Iterations of Self Play each | |
|---|---|
| Curriculum Perspective: Experimental Model Versus Control over 10,000 games | Value |
| **Outcome** | **10000** |
| **Wins** | **5832** |
| **Draws** | **405** |
| **Losses** | **3763** |
| **Elo Difference** | **72.9** |
| **Likihood of Superiority** | **100%** |




## 4.4 Summary of Findings

Collectively, the results provide evidence that material-based adjudication with curriculum horizon scaling yields faster and more stable NNUE improvement than conventional self-play termination rules. The experimental condition generated:

- More decisive games
- More consistent iteration-to-iteration strength gains
- Substantially stronger final-model performance.

While the absolute performance gap remains modest relative to established engines, this experiment demonstrates that shaping reinforcement signals materially affects convergence when training NNUE architectures from scratch.

# Discussion

The results indicate that reward shaping through material-based adjudication can improve early NNUE convergence. By forcing decisive outcomes in short games, the experimental regime exposed the network to clearer learning signals and reduced reward sparsity, producing more consistent iteration-to-iteration improvement and stronger final play (Schraudolph, 1999).

In contrast, the control condition experienced instability and slower progression, likely due to the prevalence of draws and weaker feedback signals. The observed Elo gap, produced under identical computational budgets and architectures, suggests that curriculum-style supervision meaningfully influences NNUE learning dynamics. While modest in absolute strength, these findings highlight how unfinished games are labeled during self-play matters for evaluator development and warrants further exploration.

# Limitations

Several constraints limit generalization of these results:

- Shallow search depth: Training and benchmarking occurred at depth 8, which may exaggerate the influence of evaluation quality relative to deeper tactical play.
- Limited iteration count: Only six training cycles were evaluated; longer runs may change convergence trends or reduce differences between conditions.
- Architecture rigidity: A fixed NNUE topology was used. It remains unknown whether larger or alternative feature sets interact differently with adjudication schemes.
- Simple adjudication heuristic: Material balance is a coarse approximation of position quality and may amplify evaluation biases rather than ground them in strategic play.

- Data generation was capped out at 10 minutes per iteration. Longer runs could further diversify training data which could chain the rate of converge

These limitations indicate that although results are promising, broader claims about optimal NNUE training require larger-scale experimentation and deeper evaluation.

# References


- Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum Learning. Proceedings of ICML.
- Silver, D., Hubert, T., Schrittwieser, J., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science.
- Stockfish Developers. (2020–2024). NNUE evaluation documentation and network integration guides. https://stockfishchess.org/
- Schraudolph, N. (1999). Local Gain Adaptation in Stochastic Gradient Descent. Proceedings of ICML (early discussion of gradient signal shaping).