Human Chess: A Novel Searchless RL-based Chess Agent Capable of Multi-ELO Human-Like Play

Prerit Choudhary, Rikhil Vagadia, and Ankush Dhawan

I. Motivation

Traditional chess engines face a compelling dual challenge that significantly limits their practical utility for human chess education and training. First, engines like Stockfish, AlphaZero, and LeelaChess require computationally intensive tree-search algorithms, evaluating millions of positions per second to determine optimal moves. Second, and more critically, these systems fail to provide realistic opponents for human players across different skill levels. Current approaches create weaker opponents by simply limiting search depth, resulting in highly inconsistent play patterns that alternate unpredictably between brilliant strategic moves and inexplicable blunders—fundamentally unlike the systematic, consistent errors characteristic of human players at specific ELO ratings. This creates an unrealistic training environment where players practice against opponents that exhibit superhuman tactical vision followed by irrational mistakes, poorly preparing them for actual human competition.

II. Method

HumanChess introduces a novel searchless architecture that fundamentally reimagines chess AI through three core innovations. First, the base model employs a spatially-aware transformer with RelativeMultiHeadAttention and 2D coordinate embeddings that explicitly model the 8×8 chess board geometry, enabling natural understanding of chess-specific spatial relationships without explicit rule programming. Second, the system implements two innovative fine-tuning approaches: (a) Low-Rank Adaptation (LoRA) with soft-label supervision that assigns probability mass to multiple viable moves based on win percentage similarity, avoiding erratic training from one-hot encoded ground truths, and (b) a discrete diffusion framework with ELO-conditioned move generation that models uncertainty in human decision-making through bounded self-play reinforcement learning with multi-temporal rewards spanning immediate tactical feedback and 4-move strategic lookahead. Third, the approach incorporates entropy-conditioned denoising during inference, progressively refining predictions by updating only the least confident tokens at each timestep while preserving high-confidence decisions.

III. Implementation and Results

The system was implemented using a 6.89 million parameter decoder-only transformer trained on the ChessBench dataset containing 15 billion Stockfish-annotated data points. Key engineering achievements include the development of Algorithm 1 for generating soft-encoded vectors that identify the 6 closest win percentage moves to enable robust training signals. Quantitative results demonstrate significant progress: the base model achieved 1.2721 loss corresponding to win percentage estimation within 2.3% accuracy without tree search. The diffusion model showed stable convergence with final training loss of 1.271, improving from initial per-token loss of 0.42 to 0.004. Most importantly, self-play evaluation against the Maia-1100 engine revealed consistent improvement in decision-making quality, with mean average move reward improving from -0.3286 to -0.0645 over 7000 episodes—a 400% reduction in material losses. Attention visualization revealed hierarchical development of chess understanding, with early layers capturing cross-positional relationships and deeper layers exhibiting file and rank-based movement patterns characteristic of sliding pieces.

IV. Discussion and Conclusion

These results matter because they demonstrate the first successful searchless chess engine capable of human-like play across multiple skill levels while requiring orders of magnitude less computation than traditional approaches. The system provides realistic training opponents that make consistent, skill-appropriate errors rather than erratic blunders, better preparing players for human competition. Limitations include performance degradation in endgame scenarios due to exponentially larger state spaces, with entropy analysis showing the model approaches near-randomness in late-game positions (entropy of 6.166 vs. 2.17 in early game). The LoRA approach revealed capacity limitations in smaller models, suggesting larger architectures are necessary for maintaining chess competency during human-style fine-tuning. Future work should prioritize expanding model size for deeper strategic understanding and exploring domain transfer to other strategic games. The outcome-focused impact extends beyond chess: this searchless, human-skill modeling framework establishes a foundation for creating more realistic AI training partners across various strategic domains while maintaining computational efficiency.

Authors are from the Stanford Electrical Engineering and Computer Science Departments, Stanford University, Stanford, CA 94305, USA. {preritc, rvagadia, ankushd}@stanford.edu.

Human Chess: A Novel Searchless RL-based Chess Agent Capable of Multi-ELO Human-Like Play

Prerit Choudhary, Rikhil Vagadia, and Ankush Dhawan

Abstract— **Traditional chess engines like Stockfish and AlphaZero achieve superhuman performance through computationally intensive tree-search algorithms, but struggle to provide realistic opponents for human players at lower skill levels. These systems typically create weaker opponents by limiting search depth, resulting in erratic play patterns that alternate between strong moves and irrational blunders—unlike the consistent, systematic errors characteristic of human players. HumanChess, presented here, is a novel searchless chess engine that addresses both computational efficiency and human-like play across multiple skill levels. Our approach combines a spatially-aware transformer architecture with innovative fine-tuning methodologies to create agents that play realistically at different ELO ratings (1200-1800). The base model incorporates 2D coordinate embeddings and RelativeMultiHeadAttention to naturally understand chess board geometry, achieving competitive performance with only two forward passes at runtime. The base model essentially serves as a maximal knowledge base for the fine-tuned ELO conditioned models to build off of. We explore two fine-tuning approaches: (1) Low-Rank Adaptation (LoRA) with soft-label supervision that assigns probability mass to multiple viable moves based on win percentage similarity, and (2) a discrete diffusion framework with ELO-conditioned move generation that models the uncertainty in human decision-making. The first approach aims to leverage our base model as a bootstrap oracle in order to avoid erratic training arising from training on one hot encoded moves as ground truths from human data. By identifying moves that are within a similar win percentage of the human move, we allow the truth label to enforce the model's current chess understanding in addition to the human rather than take large steps towards single human actions. The diffusion approach employs bounded self-play reinforcement learning with multi-temporal rewards spanning immediate tactical feedback and 4-move strategic lookahead. Our system demonstrates the ability to generate human-like chess play without tree search, requiring orders of magnitude less computation than traditional engines while providing more realistic training opponents across skill levels. Experimental results show stable convergence during training with the model achieving strong performance in early game positions, though challenges remain in endgame scenarios due to the exponentially larger state space.**

I. Introduction

Current chess engines suffer from two primary challenges. First, traditional chess engines, such as Stockfish, AlphaZero, and LeelaChess, primarily rely on tree-based search algorithms, meaning that they require large computations to evaluate positions and determine optimal moves [1]. In addition, while these engines are able to accomplish a very high, super-human level of play, existing techniques for creating a lower-rated opponents (that are useful for a chess player who is learning) depend on adjusting the depth parameters of these powerful chess engines, resulting in opponents that have highly inconsistent play patterns [2]. For instance, such systems often play well up to a certain point, then make an irrational, erratic decision, like sacrificing a queen, to offset the previous good moves. Hence, these systems fail to capture the systematic and largely consistent nature of human errors when playing chess, where these agents play in an often unrealistic manner. Moreover, Stockfish and other chess policy approximation techniques like LeelaChess and AlphaZero still rely on tree-based search techniques, which result in intensive computation.

Authors are from the Stanford Electrical Engineering and Computer Science Departments, Stanford University, Stanford, CA 94305, USA. {preritc, rvagadia, ankushd}@stanford.edu.

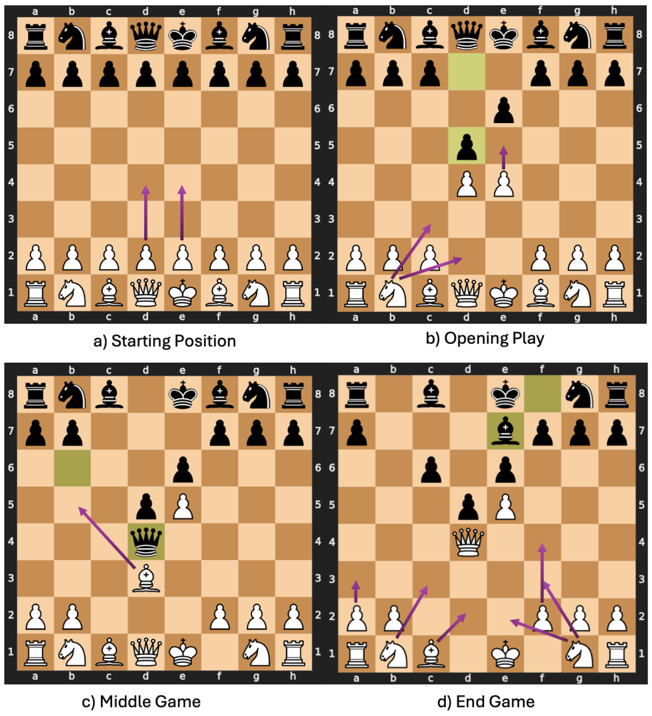

Fig. 1: **HumanChess Sytem Overview** HumanChess uses Diffusion to consider many possible trajectories for a given player skill level. (a) Decides between playing a Queen's Pawn and King's Pawn Opening, (b) Considers three variations after Black responds with the French Defense (c) After a blunder by Black, the model only considers one option since it has superior win percentage (Nc6 still leads to Black losing a Queen), (d) Many moves in the end game attain a similar game result.

In this paper, we present the development of HumanChess, a searchless chess engine that 1) plays in a humanistic manner and can be used across all skill levels to better practice and improve a player's chess abilities, as it would be more reflective of practice against humans compared to practice against the computer, and 2), is substantially more efficient by requiring only the state to compute the next action based on just two forward passes through the network at runtime.

II. Related Works

A. Dataset and Pretrained Model

Ruoss et al. (2022) introduces ChessBench, a dataset of 10 million online games annotated by the chess engine Stockfish, resulting in 15 billion data points) [3]. It is worth noting that the creation of this dataset itself is quite a substantial task, as it represents over 8800 days of unparalleled Stockfish evaluation time on one TPUv5.

The authors train a series of transformer models up to 270 M parameters in size, each on one of three different supervised objectives: action-value prediction, state-value, and behavioral cloning. The best result is seen from the action-value model, which takes a board state plus a candidate move as input and predicts the Stockfish evaluation, ultimately selecting the move with the highest estimated evaluation. The model performs at a 2895 Lichess rating, which shows great generalization, as the large state space of chess makes memorization futile. Despite this achievement, the paper fails to address an important need—the ability to get chess engines to play in the style of humans, especially at lower ratings.

While our ultimate goal is to model human-like behavior, we begin by pretraining on Stockfish-annotated data to provide the model with strong foundational knowledge of chess tactics and strategy. Relying solely on human games—particularly at lower ELOs—can introduce inconsistent or noisy decision patterns that make it difficult for the model to learn effective representations from scratch. Pretraining on expert-level evaluations helps the model internalize high-quality positional understanding and tactical motifs. We then fine-tune on real human games to specialize this prior toward realistic, rating-specific play. This two-stage training process is essential for producing agents that not only play well but also make consistent, human-like mistakes appropriate for their target skill level.

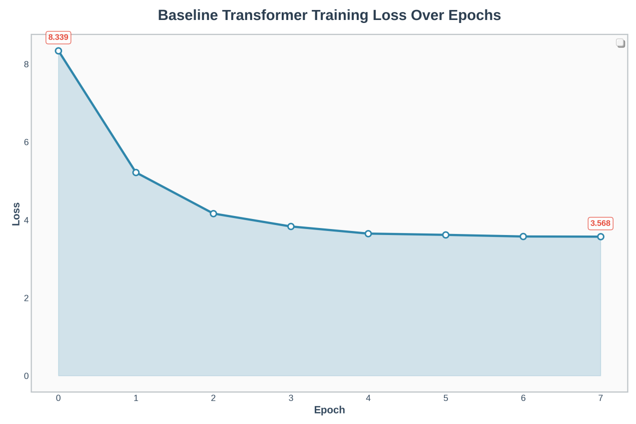

Fig. 2: **Training Loss Plot for Baseline Transformer Model:** Training loss shows convergence after 8 epochs.

B. Fine-Tuning

In the human skill level fine-tuning, we explore two different potential approaches.

We first explore Low-Rank Adaptation (LoRA). LoRA is a parameter-efficient method that inserts small trainable matrices into the attention projections of the model while keeping the majority of the pretrained weights frozen [4]. Training LoRA on top of the base model with real human-played games will create a humanistic chess agent that can play at multiple ELO ratings. Add more here probably

We also explore diffusion models for instilling implicit search capabilities in the model by modeling future world states. In [5], the authors highlight the usefulness of diffusion models for instilling search-like abilities within models to enhance their planning abilities without relying on explicit search. The diffusion process inherently models the uncertainty and variation in human decision-making, generating distributions over moves that reflect how humans consider multiple candidate moves before settling on one. The diffusion model enables learning different trajectory vector spaces for specific ELO embeddings, allowing the system to model various human skill levels with targeted rating-specific parameters. Extending this method to chess will create agents that balance strategic planning with human-like play patterns when fine-tuned on real human-played games.

C. RL Based Optimization

Silver et al. introduced AlphaZero [6], a breakthrough general reinforcement learning algorithm that achieved superhuman performance across chess, shogi, and Go through pure self-play learning. Starting from random play with no domain knowledge beyond game rules, AlphaZero combines deep neural networks with Monte Carlo Tree Search (MCTS) and learns entirely through self-play reinforcement learning. Despite evaluating significantly fewer positions per second than traditional engines (80k vs. 70M for Stockfish), AlphaZero's neural network enables more selective search and achieved superhuman chess performance within 4 hours of training. However, AlphaZero still relies on computationally intensive tree search methods and focuses exclusively on optimal play rather than modeling human-like behavior across different skill levels but their self-play RL based approach could be useful for our task.

III. Methodology

A. Base Architecture

The foundation of the HumanChess system is a decoder-only transformer that is trained on the ChessBench state-value dataset, containing state – win percentage pairs annotated by StockFish rollouts. A key innovation is that this transformer contains a spatial-aware architecture. The model incorporates a RelativeMultiHeadAttention mechanism with 2D coordinate embeddings - in addition to standard positional encodings, we also explicitly model the 8x8 chess board structure by adding a relative bias for each square's coordinates and propagating the combined positional encoding

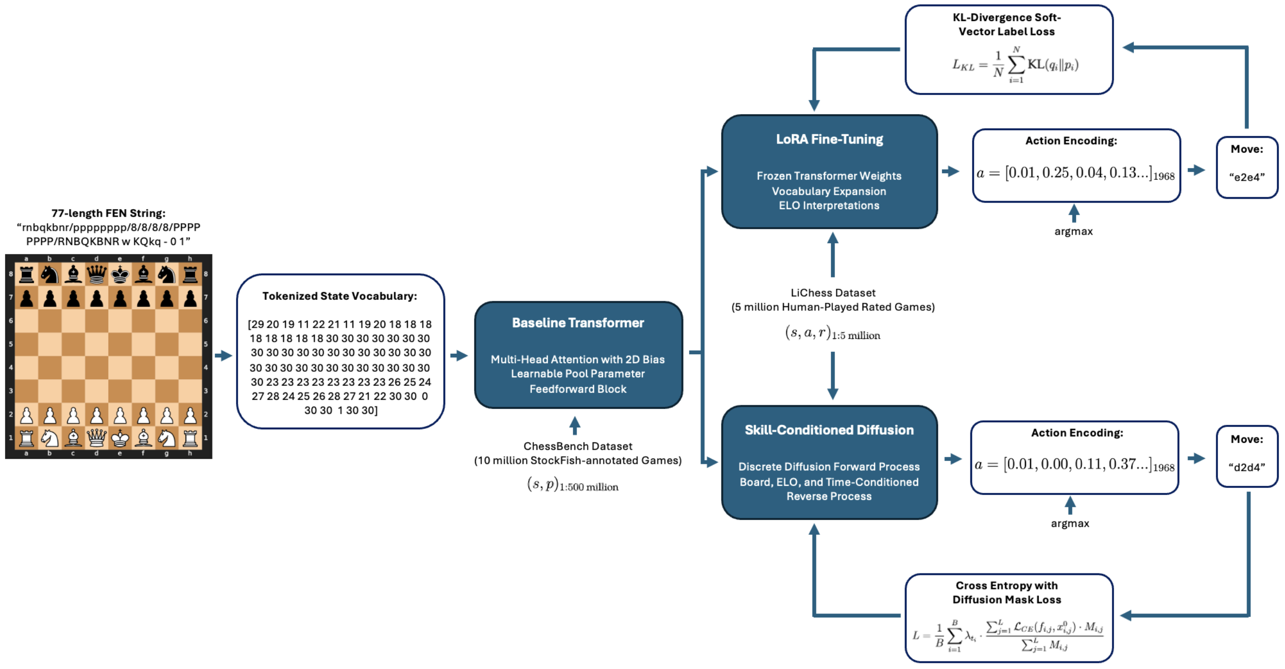
77-length FEN String:
"rnbqkbnr/pppppppp/8/8/8/8/PPPP PPPP/RNBQKBNR w KQkq - 0 1"
Tokenized State Vocabulary:
[29 20 19 11 22 21 11 19 20 18 18 18 18 18 18 18 18 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 23 23 23 23 23 23 23 23 26 25 24 27 28 24 25 26 28 27 21 22 30 30 0 30 30 1 30 30]
Baseline Transformer
Multi-Head Attention with 2D Bias
Learnable Pool Parameter
Feedforward Block
ChessBench Dataset
(10 million StockFish-annotated Games)
$(s,p)_{1:500\text{ million}}$
LoRA Fine-Tuning
Frozen Transformer Weights
Vocabulary Expansion
ELO Interpretations
KL-Divergence Soft-Vector Label Loss
$L_{KL}=\frac{1}{N}\sum_{i=1}^{N}\mathrm{KL}(q_i\|p_i)$
Action Encoding:
$a=[0.01,0.25,0.04,0.13...]_{1968}$
argmax
Move:
"e2e4"
LiChess Dataset
(5 million Human-Played Rated Games)
$(s,a,r)_{1:5\text{ million}}$
Skill-Conditioned Diffusion
Discrete Diffusion Forward Process
Board, ELO, and Time-Conditioned
Reverse Process
Action Encoding:
$a=[0.01,0.00,0.11,0.37...]_{1968}$
argmax
Move:
"d2d4"
Cross Entropy with Diffusion Mask Loss
$L=\frac{1}{B}\sum_{i=1}^{B}\lambda_{t_i}\cdot\frac{\sum_{j=1}^{L}\mathcal{L}_{CE}(f_{i,j},x^0_{i,j})\cdot M_{i,j}}{\sum_{j=1}^{L}M_{i,j}}$

Fig. 3: **Methodology:** The pipeline for this project follows a two stage training process, as explained above, and then further optimized, as explained below.

through our attention calculation. This spatial-aware architecture allows the model to naturally understand chess-specific geometric patterns like diagonals, knight moves, and piece relationships, resulting in a more humanistic understanding of chess through 2D grid structure representation.

Using Q-learning, the pretrained model achieves a maximally performing policy within the transformer parameters. The pretrained model leverages the annotated win percentages for the state-value pairs to learn high-level chess strategies without explicit search. It should be noted that this approach may not necessarily produce human-style play, as the training data consists of Stockfish games rather than actual GM or IM games.

B. Human-Style Fine-tuning

To have the model play at specific rating levels, we fine-tune the pretrained model on rating-specific human datasets using soft-label supervision derived from real game data. To this end, we process a dataset of positions (as FEN strings) and corresponding moves from a database of Lichess games, filtered by player ELO [**?**]. For this step, we explore two methods: a LoRA based action model and a discrete diffusion-based action model, which are described in detail in the following sections.

1) LoRA Action Model: Rather than treating each move as the sole correct action, we constructed soft label vectors that assign partial probability mass to alternative legal moves with similar predicted win probabilities. This approach recognizes that strong players often face multiple viable options in a given position, and creates richer, more human-aligned training signals. These relabeled actions are then used to fine-tune the base model using Low-Rank Adaptation (LoRA). This approach allows for adapting the model to different player skill levels and move preferences without the full cost of full fine-tuning as gradients will not be backpropagated through the full transformer. Using the soft label distributions and KL divergence loss, we optimize the LoRA parameters and a newly initialized output layer, enabling efficient specialization of the model to human-like play that captures high-quality moves and stylistic and strategic diversity across skill levels.

2) Discrete Diffusion for ELO-Conditioned Move Generation: In this approach, we introduce a novel framework for learning human-like chess policies through diffusion models enhanced with bounded self-play, based on the DIFFUSEARCH work. We implement a discrete diffusion model that learns skill-specific move distributions from human gameplay data, enabling our end-to-end system to generate move recommendations that authentically reflect how players at different ELO ratings approach a given chess position. Modeling this as a discrete diffusion problem allows us to directly model the categorical distribution over the chess move vocabulary.

Our diffusion model leverages an 8 layer pretrained decoder only transformer architecture trained on the ChessBench dataset described previously. The model consists of 8 transformer layers with 256-dimensional embeddings, 8 attention heads, and 1024-dimensional feed-forward networks. The pre-trained model provides crucial chess domain knowledge, including piece interaction patterns, tactical motifs, and

Algorithm 1 Generating Soft Encoded Vectors

```
Require: $s$: current state of the board (FEN String)
Require: $a_{\text{true}}$: true action ID taken by the player (1–1968)
Require: $w$: win percentage associated with the action taken
    by the player
Require: scale: float hyperparameter corresponding to probability mass associated with true taken action
Ensure: $a_{\text{out}}$: soft encoded vector of length 1968 with associated probability masses
 1: Initialize $a_{\text{out}} \leftarrow \mathbf{0}_{1968}$
 2: $\mathscr{L} \leftarrow$ legal moves in state $s$
 3: for each move in $\mathscr{L}$ do
 4:     Find next 6 closest win percentage moves to $w$
 5:     $b \leftarrow$ next 6 closest win percentage move IDs
 6: end for
 7: Create weighted probability mass for IDs in $b$ according to their similarity in win percentage to $w$, scaled to $(1-$ scale$)$
 8: $a_{\text{out}}[a_{\text{true}}] \leftarrow$ scale
 9: for $i = 1$ to $|b|$ do
10:     $a_{\text{out}}[b[i]] \leftarrow$ corresponding probability mass from weighted distribution
11: end for
12: for all remaining entries in $a_{\text{out}}$ do
13:     $a_{\text{out}}[j] \leftarrow \varepsilon$ for entries not assigned above
14: end for
15: Normalize $a_{\text{out}}$ such that $\sum_{i=1}^{1968} a_{\text{out}}[i] = 1$
16:
17: return $a_{\text{out}}$
```

positional evaluation heuristics. Rather than training from scratch, we initialize our diffusion model with these learned representations and adapt them through our novel ELO-conditioning framework.

We employ Low-Rank Adaptation (LoRA) to efficiently adapt the pre-trained architecture for our diffusion task. LoRA decomposes weight updates into low-rank matrices, dramatically reducing the number of trainable parameters while preserving model expressiveness through the expression $h = W_\theta x + \Delta W x = W_\theta x + BAx$ where $B \in R^{d\times r}$ and $A \in R^{r\times k}$ and $rank(r) << min(d,k)$. We apply LoRA with rank r=128 and scaling factor alpha=256 to the query, key, value, and output projection matrices within each attention head. This configuration reduces trainable parameters by approximately 90% while maintaining the model's capacity to learn complex skill-dependent pattern, targeting the attention mechanism where we hypothesize skill-level differences primarily manifest.

Player skill conditioning represents a core innovation in our approach. We model ELO ratings through learnable bucket embeddings that span our target range of 1200-1800 with 100-point granularity, resulting in 7 discrete buckets. To enable smooth skill interpolation rather than discrete jumps, we implement continuous ELO conditioning through linear interpolation between adjacent bucket embeddings. This process is explained further in 7.1. The continuous ELO embedding is broadcast across all sequence positions and added to the position embeddings, ensuring that skill-level information influences every aspect of the model's reasoning process. We break up this model into a few different components:

a. Discrete Diffusion Process

Our diffusion formulation operates directly on the discrete move vocabulary. The forward diffusion process gradually corrupts clean move sequences by mixing them with uniform noise according to a predefined schedule. For a clean move sequence x_0 and a timestep t, the forward process samples from:

$$q(x_t|x_0) = \alpha_t \cdot \text{OneHot}(x_0) + (1-\alpha_t) \cdot \frac{1}{|V|}\mathbf{1}_{|V|}$$

where α_t represents the noise schedule, OneHot(x_0) is the one-hot encoding of the true move token, and $|V| = 2000$ is our move vocabulary size. The term $\frac{1}{|V|}\mathbf{1}_{|V|}$ represents a uniform distribution over all possible moves, where each move has probability $\frac{1}{|V|}$.
We create a categorical mixture distribution that interpolates between the true move (represented as a one-hot vector) and uniform noise over the vocabulary. At $t = 0$, we have $\alpha_0 = 1$, recovering the clean data distribution. As t increases, α_t decreases, gradually shifting probability mass from the true move to uniformly random moves in the vocabulary. The noise schedule follows $\alpha_t = \prod_{i=1}^{t}(1-\beta_i)$ with linearly increasing β_i values from 0.025 to 0.1 over $T = 40$ diffusion steps. During training, we randomly sample timesteps t Uniform(0, T-1) and apply the corresponding noise level to ground truth move sequences. The model learns to predict the original clean moves given the noisy observations, position context, and target ELO rating.

b. Model Architecture

The input to the model is represented as follows:

$$x = [\textbf{CLS}]||\text{state\_tokens}||\text{future\_tokens}$$

where state_tokens represent the current chess position encoded as piece-square tokens, and future_tokens represent the (potentially noised) move sequence to be denoised. The exact representations of these tokens is explained in Section **??**) .
Position embeddings are applied through the inherited positional encoding from the base model, while ELO embeddings are broadcast and added across all positions. Temporal embeddings for the diffusion timestep t are learned through a separate embedding layer initialized to zero, allowing the model to gradually learn timestep-specific behaviors during training. The temporal embedding is critical for the denoising process, as it allows the model to adapt its predictions based on the current noise level. At early timesteps (high noise), the model should focus on broad strategic patterns, while at later timesteps (low noise), it should attend to precise tactical considerations.

c. Training And Loss

Our training objective focuses specifically on positions where noise was applied, reflecting the insight that the model should learn to correct corrupted tokens rather than simply copying already-clean inputs. The loss function incorporates several masking strategies:

$$L = \frac{1}{B}\sum_{b=1}^{B}\lambda_t \sum_{l=1}^{L}\Big[\mathrm{CE}\big(f_\theta(x_t^{(b)}, s^{(b)}, \mathrm{elo}^{(b)})_l, x_{0,l}^{(b)}\big) \times \text{mask\_changed}_{b,l} \times \text{mask\_valid}_{b,l}\Big]$$

$$\lambda_t = \frac{\beta_t}{1-\bar{\alpha}_t}.$$

In this formulation, λ_t is given by $\beta_t/(1-\bar{\alpha}_t)$, which provides timestep-dependent weighting to emphasize harder denoising steps. The term $\text{mask\_changed}_{b,l}$ is an indicator that equals 1 if token l in example b was corrupted by the forward noising process (and 0 otherwise), ensuring we only compute loss on positions that were actually altered. The term $\text{mask\_valid}_{b,l}$ equals 1 if token l in example b is not a padding token (and 0 if it is), which excludes padding from the loss. Finally, $\mathrm{CE}(\cdot,\cdot)$ denotes the cross-entropy loss over the move vocabulary. Weighting by the timestep is a crucial stability component to prevent the model from being dominated by easy denoising tasks and ensures that the model still learns from high-noise scenarios.

d. Diffusion Inference

During inference, we employ a reverse diffusion process, which we call entropy conditioned denoising, that iteratively denoises a randomly initialized move sequence over $T = 40$ timesteps. Starting from uniform random tokens, the model progressively refines predictions by selecting the highest-confidence tokens (the one's with minimal logit entropy) at each step. We implement an adaptive masking strategy where only the least confident tokens (determined by log-probability percentiles) are updated at each timestep, allowing high-confidence predictions to remain stable. This selective updating mechanism prevents the model from unnecessarily revising correct predictions while focusing computational effort on uncertain regions of the sequence. Additionally, the frozen tokens add to the inherent conditional information the model can use as other tokens that are still to be predicted are now conditioned on the frozen tokens. One key design choice is that we allow tokens to be "unfrozen" so that highly confident incorrect predictions do not derail the entire diffusion process. The process terminates early if no tokens require updating, typically converging within 10-15 iterations for most chess positions.

C. Self-play Evaluation

Building upon AlphaZero's self-play paradigm, we develop a novel online reinforcement learning approach adapted for our diffusion-based chess architecture. Our system trains against two fixed opponents of varying strength—our own pret-rained transformer model, then a weaker system based off of the Maia Chess bot that encompasses 3 different ELO levels —providing stable gameplay across skill levels while preventing overfitting to specific playing styles. This dual-opponent training ensures consistent learning across different strategic contexts without the instability issues common in pure self-play scenarios. To evaluate performance and maintain realistic boundaries, we implement policy gradient-based self-play constrained by boundary embeddings that encode rating-specific move patterns. Agents of multiple ELO ratings play against each other, with results used to determine if each agent plays at its intended skill level while preventing model drift into superhuman play.

Our approach introduces move-by-move policy gradient updates, contrasting with AlphaZero's end-of-game learning signals. We implement a hybrid reward system combining immediate rewards (positional changes, tactical achievements), critically important delayed rewards calculated over a 4-move lookahead window that captures positional improvements and strategic planning, and terminal rewards (game outcomes). This 4-move windowed evaluation enables the model to learn medium-term strategic concepts beyond immediate tactical gains, bridging the gap between short-term tactics and long-term strategy. The boundary embeddings ensure agents improve within realistic decision-making boundaries—for instance, if a 1200-level agent defeats a 1400-level agent, policy gradients adjust model weights to maintain skill-level consistency. After each move, parameters are updated using policy gradients where the log probability of the selected move is weighted by these multi-temporal rewards and backpropagated, enabling rapid adaptation while preserving human-like characteristics across ELO levels.

IV. Experimental Results

A. Base Model Performance

After training the base model on an Nvidia 4090 GPU (approximately 10 days of training, the model achieves a loss of 1.2721, corresponding to a correct win percentage categorization of within 2.3% when rescaled. This signifies that the model demonstrates the ability to implicitly estimate win percentages with great accuracy without performing any depth-based tree search.

The below figure visualizes the 2D spatial attention patterns across our transformer's eight layers, revealing hierarchical development of chess-specific spatial understanding. Early layers (0-1) show off-diagonal intensity patterns indicating cross-positional relationships between pieces on different squares. Deeper layers (6-7) exhibit pronounced vertical and horizontal bands, demonstrating learned file and rank-based movement patterns characteristic of sliding pieces like rooks and queens. The emergence of spatial blocks throughout the layers indicates the model processes coherent local neighborhoods, learning positional concepts such as pawn formations and king safety zones. This hierarchical attention development validates that our 2D spatial bias mechanism enables the transformer to automatically discover chess-specific spatial relationships without explicit rule programming.

B. *LoRA Model Performance*

The first method for mid-level humanistic play investigated was fine-tuning the baseline transformer using LoRA adaptation. The LoRA adapters were trained on the baseline transformer for 5 million samples extracted from the Lichess April 2025 dataset of ELO-action annotated games.

This method, while computationally efficient, revealed significant challenges when applied to our relatively small 6.89 million parameter base trasnformer. Our experiments demonstrated a critical limitation: the model experience substantial capability degradation during the human-style fine-tuning process, essentially "forgetting" how to play high-level chess in the deeper stages of the game as it adapted to human gameplay patterns.

With a larger model, LoRA typically preserves pre-trained capabilities while adding new skills [7], but in this smaller architecture limited by computation access, the model was unable to maintain the delicate balance between chess competancy and human-like play patterns. For example, the model would initially play a reasonable move to begin the game, such as `e2e4` as White or `d7d5` as Black, but then soon after play less reasonable moves in the early game such as `h2h3`, since it did not have a positional understanding with the few LoRA parameters.

1) Soft-Label Learning Limitations: Our soft-label supervision approach, which assigned probability mass to multiple viable moves based on win percentage similarity (Algorithm 1), aimed to capture the natural uncertainty in human decision-making. However, the interaction between this probabilistic training signal and the model's limited capacity created unexpected complications. The model struggled to distinguish between "multiple good moves" (which should be preserved) and "human-like suboptimal moves" (which should be learned), often converging to play patterns that were neither strategically sound nor authentically human-like. During training, the loss (KL-Divergence Loss with Soft Labeled Vectors) begins above, and converges to 1.8, indicated moderate to large differences. This may be attributed to a lack of depth in the baseline transformer and a lack of data when fine-tuning with LoRA. This suggests that the model's understanding of chess became too degraded to make coherent human-like decisins in the large state-action space of chess.

C. *Diffusion Results*

Due to substantial compute constraints, we were only able to train LoRA adapters on the baseline transformer for 5 million samples extracted from the Lichess dataset. Although we plan to test the model through tournament style self play to assess capabilities, this process could not be completed within the constraints of the class period. Throughout training, the model exhibited stable and consistent convergence patterns. The weighted cross-entropy loss incorporating our timestep-dependent weighting scheme (λ_t) demonstrated effective learning across all diffusion timesteps. Unlike traditional supervised learning where loss simply decreases monotonically, diffusion training showed characteristic oscillations reflecting the varying difficulty of denoising at different noise levels. We therefore evaluate performance using logit entropy as our primary metric. Entropy provides insight into the model's prediction confidence across the 1968-token vocabulary. This metric allows us to assess both training quality and inference prediction, as these differ substantially in diffusion models.

Our final loss during training was 1.271, decaying exponentially from a first epoch training loss of 120. Intuitively, our loss function sums the cross entropy loss of each token across the 312 predicted tokens. Because the vocabulary is of size 2000 (1968 actions + 31 states + 1 noise), a fully random prediction would result in a loss of $312 \times (ln(1/2000) \approx 7.6) = 2371.2$. With this understanding, we see that after one epoch, our model had a per token loss of 0.42, with our final epoch per token loss at 0.004. This indicates that the model can denoise artificially noised inputs extremely well, given a state. Since our noising schedule that caps at 0.1, meaning that we also need to assess the model's ability to denoise given a completely random sample during inference. Here we make use of the entropy confidence values.

We observe a notable trend: the model's average entropy for a prediction in earlier parts of the game (ie. its confidence) is substantially lower (2.17) compared to when the model approaches the middle (4.83) and end games (6.166), where the model struggles to play reasonable moves. We can see that the model is slightly better than random in the middle, but in the endgame, approaches almost complete randomness, indicating decaying performance. We note that this is likely due to a combination of a distribution mismatch in training data accompanied by the exponentially larger state space in the middle and end game compared to the beginning. Additionally, the contrast between the extremely low training loss and the inference results, suggest that the model may need to have substantially more parameters to capture deeper relationships within the middle and endgame. Moreover, the noise schedule may need to also be refined to enforce substantially more difficult noising tasks for model training. Due to time constraints and the lack of general efficiency, we have still yet to test the capability of the ELO conditional embeddings to alter chess understanding with diffusion, but we plan to test this with simulations against online bots once we have drastically augmented the diffusion model's parameter schema and training time.

D. *Self Play RL Optimization Results*

We tracked several key performance metrics during our self-play training to monitor model improvement and stability. Update steps measured the frequency of parameter updates, incrementing each time the model received a reward signal above a threshold ($|reward| > 0.001$) for both immediate tactical feedback and terminal game outcomes. In addition, we recorded game length—the total number of

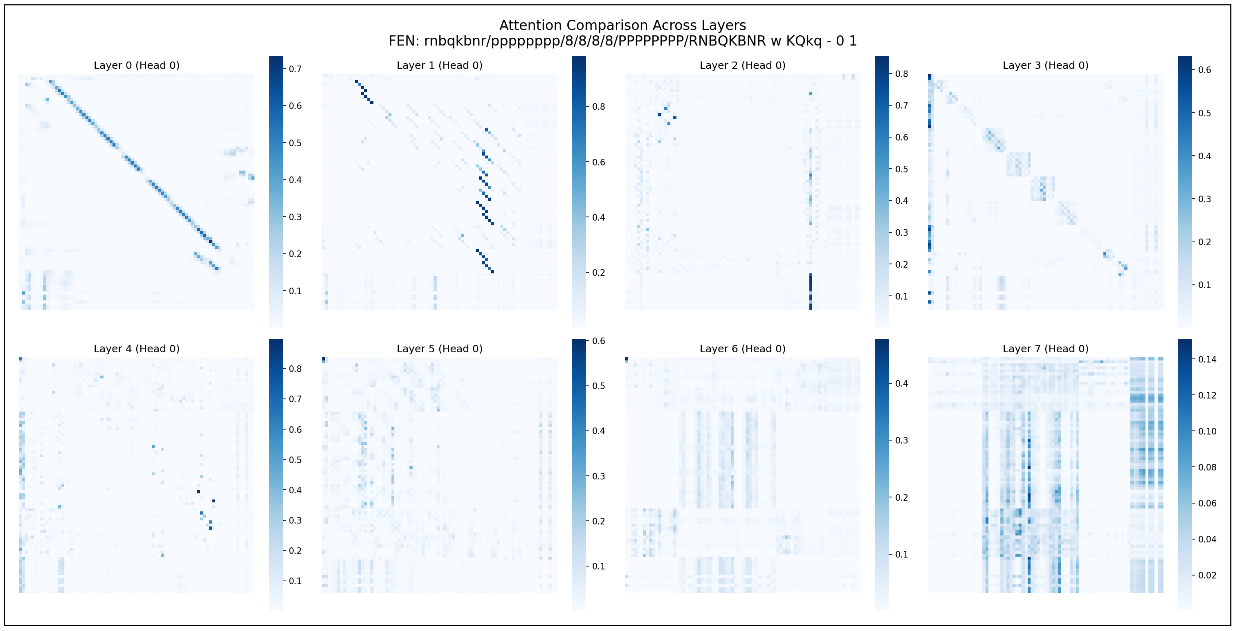

Fig. 4: **2D Spatial Attention Visualization**: x and y axes represent the tokenized state (FEN string) as 77 total tokens. Token 0 represents which player is to move, Tokens 1-64 represent the board state, and the remaining tokens characterize en-passant, castling, and move indexes.

moves per game across both players—to track the evolving complexity and duration of matches over training. We also tracked average move reward, defined as the mean pawn-equivalent gain per action, incorporating material, positional, and tactical bonuses. These metrics were automatically logged and reset every 500 episodes to provide rolling performance assessments throughout the training process.

Training against the baseline transformer unfortunately did not yield strong performance improvements for our diffusion model. Against a much stronger opponent, total game length between players is a useful proxy for strategic competence — longer matches often reflect an ability to avoid critical errors and maintain positional resilience. Over 25,000 episodes, the average self-play game length hovered between 35 and 55 moves with no clear upward trend, showing that the agent never learned to outlast its far stronger opponent. This outcome is expected, given that the pretrained baseline bot is significantly stronger and already plays at a much higher level. Still, this is a disappointing result, as we hoped the diffusion model would at least be able to extend games slightly, an indication that it is becoming more of a competent player through our self-play approach. The lack of clear progress may stem from sparse terminal rewards, compounding errors in long-term planning, or insufficient credit assignment during online updates.

By contrast, when training our diffusion model online against the Maia engine (1100 ELO), we observed clear, if modest, improvements in per-move quality. As summarized in the below table, mean average move reward improved steadily over successive 500-episode windows, confirming that the model learned to make progressively stronger moves against a mid-level opponent. A minor regression in episodes 5001–5500 (–0.1450) suggests occasional instability, but the overall upward trajectory confirms that the model steadily refines its policy against this mid-level opponent.

These results compellingly demonstrate that, even against a nontrivial opponent, our diffusion model is able to incrementally refine its decision-making, steadily narrowing its material losses and improving positional choices on each move. More broadly, this clear upward trajectory highlights the promise of our hybrid self-play framework as a powerful approach for training sophisticated, human-like decision-making agents.

Episode Window	Mean Avg Move Reward
1–500	-0.3286
501–1000	-0.3013
1001–1500	-0.2389
1501–2000	-0.2354
2001–2500	-0.2220
2501–3000	-0.2223
3001–3500	-0.1642
3501–4000	-0.1237
4001–4500	-0.1230
4501–5000	-0.1202
5001–5500	-0.1450
5501–6000	-0.1040
6001–6500	-0.0967
6501–7000	-0.0645

TABLE I: **Self-play of Diffusion Model against Maia-1100 Bot:** Mean average move reward per 500-episode window is reported, and shows an increase across episodes, indicating that the agent continually learns to make better moves.

V. Conclusion and Future Work

This research presents a significant step forward towards creating more realistic and computationally efficient chess AI systems. We successfully achieved a strong baseline transformer model capable of high-skilled chess play through our novel 2D spatial attention mechanism, demonstrating that searchless architectures can compete with traditional approaches while requiring orders of magnitude less computation. Our dual fine-tuning approaches—LoRA-based direct action learning and diffusion-based trajectory planning—showcase innovative methods for incorporating human-like decision-making across multiple skill levels.

Our experiments also identified several areas for improvement. The humanistic fine-tuning exposed constraints of our 6.89 million parameter model when faced with chess's vast state and action space, with compute limitations preventing extensive experimentation. Our models often exhibit late-game blunders, suggesting deeper networks are necessary for strategic coherence over extended sequences. Future work should prioritize expanding model size to achieve deeper chess understanding and more robust endgame play.

Beyond technical improvements, this research opens exciting possibilities for broader applications. While we use chess as our exploration domain, these searchless and human-skill modeling techniques can be adapted to various strategic domains. Most immediately, this model offers a more human-realistic training partner for chess players, providing opponents that make consistent, skill-appropriate decisions rather than the erratic play patterns of traditional engines, better preparing players for tournament competition against human opponents.

VI. Acknowledgements

We thank the course staff of CS224R for their support and guidance in the project, Tim Genewein and Anian Ruoss from DeepMind for initial guidance on fine-tuning the baseline transformer, and the ARMLab for providing compute resources for the project. Moreover, the ChessBench repo was used as a reference for dataset setup and extraction, especially with the bagz file format. Additionally, the diffusion component of this project was also a part of a CS231N project, where Prerit Choudhary was in both classes, Ankush Dhawan and Rikhil Vagadia were part of only CS224N, and then Akshay Gupta was part of only CS231N. The diffusion model was included with vision components to complete an end to end image to recommendation pipeline, but the majority of the CS231N paper was focused on vision techniques to define FEN states for chess boards.

VII. Team Contributions

For this project, most work was conducted together, from ideation to implementation to evaluation. Prerit led on training the baseline transformer on Chessbench, and implementing the diffusion model pipeline, Ankush led on the LoRA fine-tuning implementation and evaluation against the Maia models, and Rikhil led on the self play evaluation. Ankush took lead on making figures, and Rikhil and Prerit helped with paper structure and methods communication.

References

[1] Q. A. Sadmine, A. Husna, and M. Müller, "Stockfish or leela chess zero? a comparison against endgame tablebases," in Advances in Computer Games, M. Hartisch, C.-H. Hsueh, and J. Schaeffer, Eds. Cham: Springer Nature Switzerland, 2024, pp. 26–35.

[2] K. W. Regan, T. Biswas, and J. Zhou, "Human and computer preferences at chess," in Multidisciplinary Workshop on Advances in Preference Handling: Papers from the AAAI-14 Workshop. Quebec City, Canada: Association for the Advancement of Artificial Intelligence (AAAI), 2014, pp. 79–84.

[3] A. Ruoss, G. Delétang, S. Medapati, J. Grau-Moya, L. K. Wenliang, E. Catt, J. Reid, C. A. Lewis, J. Veness, and T. Genewein, "Amortized planning with large-scale transformers: A case study on chess," 2024. [Online]. Available: https://arxiv.org/abs/2402.04494

[4] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[5] J. Ye, Z. Wu, J. Gao, Z. Wu, X. Jiang, Z. Li, and L. Kong, "Implicit search via discrete diffusion: A study on chess," 2025. [Online]. Available: https://arxiv.org/abs/2502.19805

[6] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017. [Online]. Available: https://arxiv.org/abs/1712.01815

[7] D. Biderman, J. Portes, J. J. G. Ortiz, M. Paul, P. Greengard, C. Jennings, D. King, S. Havens, V. Chiley, J. Frankle, C. Blakeney, and J. P. Cunningham, "Lora learns less and forgets less," 2024. [Online]. Available: https://arxiv.org/abs/2405.09673