Hexaïssa: Standing on Giants' Shoulders – Routing the Best Chess Engines with Mixture-of-Experts and Latent Reward Learning

Bach Ngo[1,*] **and Nguyen Hoang Khoi Do**[2]

[1]The Frazer School for Competitive Academics, FL, USA
[2]University of Florida, FL, USA
[*]Correspondence to: bachhyngo@gmail.com

Abstract

We present Hexaïssa, a novel framework for adaptive chess engine routing that formulates expert selection as a Mixture-of-Experts (MoE) problem. Hexaïssa learns a gating policy that dynamically selects among heterogeneous state-of-the-art engines such as Stockfish, LCZero, and Obsidian, depending on the tactical and strategic complexity of each board state. This adaptive mechanism enables stronger performance and more efficient computation than any fixed engine or static configuration. However, training such a gating policy is fundamentally challenging due to sparse optimization signals and long-horizon credit assignment in chess games. To address these, we introduce a score-based inverse reinforcement learning (IRL) method that models expert engine trajectories as samples from a latent distribution over optimal behaviors. By recovering the Stein score function of this distribution via stochastic differential equations (SDEs), we infer dense, per-move reward signals consistent with potential-based IRL. These latent rewards allow efficient training of the gating network without requiring additional environment interaction or human supervision. Empirical results on standard chess benchmarks demonstrate that Hexaïssa significantly outperforms individual engines, conventional MoE, and IRL baselines.

The project page — https://hexaissa.org

Introduction

Chess has long been a cornerstone for research in sequential decision-making under perfect information (McIlroy-Young et al. 2021; Silver et al. 2018). Its well-defined rules, large combinatorial search space, and long-term planning requirements make it an ideal environment for developing and evaluating intelligent agents. Over the years, a variety of algorithmic approaches have been applied to chess, ranging from symbolic search to deep learning and hybrid methods (David, Netanyahu, and Wolf 2016; Maharaj, Polson, and Turk 2022). In recent years, many state-of-the-art (SOTA) chess engines such as Stockfish (Isenberg 2021), LCZero (LCZero Team 2021), and Obsidian (Gabriel 2024) have emerged. Each reflects different inductive biases and exhibits complementary strengths in distinct types of positions.

Importantly, the strength of a chess engine depends not only on its core architecture but also on how it is configured. Parameters such as search depth, evaluation time, and available hardware resources all influence performance, and their impact varies across different game positions. A challenging position may require deeper search from Stockfish, while a well-understood position may be efficiently handled by Obsidian at lower depth, for example. This observation motivates adaptive chess engine selection, where each engine can be chosen dynamically based on each board state per move. Such adaptivity enables better allocation of engines under low time-control settings, and significantly improves overall performance.

In this paper, we propose Hexaïssa, a novel framework for adaptive chess engine routing that learns to select among heterogeneous engine configurations at every board position. Specifically, Hexaïssa frames this routing problem as a Mixture-of-Experts (MoE) setup, where a gating policy selects the top-performing engine for each move, conditioned on the current board state of the game. Unlike static engine selection or fixed-depth evaluation, our approach learns to dynamically allocate computational effort, leading to stronger play and improved efficiency. However, training such a gating policy is non-trivial due to two fundamental challenges. First, optimization signals in chess are sparse, as feedback is typically provided only at the end of a match in the form of a win, draw, or loss, which offers a very limited signal for optimizing its routing decisions throughout the game. Second, even when the final outcome is known, it is difficult to determine which expert choices along the trajectory contributed positively or negatively, posing a credit assignment problem (Gupta et al. 2021).

To address these challenges, Hexaïssa introduces a novel score-based IRL framework grounded in stochastic differential equations (SDEs). The key idea is to treat trajectories of strong chess engines as samples drawn from a latent distribution of high-quality decisions. In this latent space, where the structure of trajectories is smoother and easier to model, we estimate the score function, defined as the gradient of the log-probability density, using denoising score matching (Chao et al. 2022). To connect this to the original trajectory domain, Hexaïssa theoretically pulls back the latent score to the trajectory space using the Jacobian of the decoder. This produces a trajectory-level score field that accurately reflects

the underlying expert distribution. From this field, Hexaïssa defines a reward potential function whose gradient is trained to align with the recovered score, yielding a smooth and differentiable reward landscape that guides policy learning. As a result, Hexaïssa consistently outperforms all individual engines, including Stockfish, on standard chess benchmarks, establishing a new state of the art in adaptive engine control.

Related Work

Chess Engines as Experts. Modern chess engines have progressed from handcrafted systems, such as early versions of Stockfish (Isenberg 2021) to neural-based approaches, including LCZero (LCZero Team 2021), Obsidian (Gabriel 2024), and AlphaZero (Silver et al. 2017). More recent models such as Unplugged MuZero (Schrittwieser et al. 2021), Gumbel AlphaZero (Danihelka et al. 2022), and transformer-based systems (Ruoss et al. 2024), pursue fully end-to-end learning across entire games. Each of these engines can be viewed as a specialized expert, reflecting distinct decision-making paradigms. In practice, the notion of an expert is defined not only by the type of engine but also by its configuration, such as maximum search depth, evaluation time, or inference speed. In Hexaïssa, we select a representative set of strong open-source engines, Stockfish, LCZero, and Obsidian, and instantiate multiple expert configurations from each. These experts are treated as fixed, black-box policies. The selected engines are diverse in inductive bias (e.g., symbolic vs. neural), robust across different types of positions, and widely adopted in competitive play. Rather than training a single unified model to generalize across all scenarios, we learn to adaptively select the most appropriate expert and configuration per chess board state.

Mixture-of-Experts in Decision Making. MoE architectures have shown promise in scalable decision-making systems by enabling dynamic routing among specialized modules (Willi et al. 2024; Riquelme et al. 2021; Chen et al. 2022; Do et al. 2025b; Nguyen et al. 2025; Do et al. 2025a). Recent efforts explore MoE under offline or multitask settings (Chen et al. 2023), often assuming access to supervised task labels or decomposable expert transitions. A closely related one is an unpublished work M2CTS (Helfenstein et al. 2024), which trains separate neural networks for the opening, middle, and end-game phases in chess. Although these setups improve performance, they do not learn reward signals for each move. Further, the MoE is trained with standard supervised learning, without any reward learning. In contrast, we formulate MoE at the trajectory level over black-box chess engines, where each expert is a full decision-making policy. Our method learns a gating network that selects the top expert per move, driven by dense reward signals inferred through score-based modeling. This design enables fully offline and simulator-free MoE training in a domain with highly complex and long-horizon decision dynamics.

Inverse Reinforcement Learning. Modern IRL approaches, including IQLearn (Garg et al. 2021), LS-IQ (Al-Hafez et al. 2023), T-REX (Jiang et al. 2023), RankGame IRL (Sikchi et al. 2022), and Energy Guidence MaxEnt (Chao et al. 2024) variants, have made offline reward inference more tractable by avoiding adversarial losses or environment interaction. They leverage ranking constraints, Bellman consistency, or successor features to extract scalar reward functions. Yet, they still suffer from reward ambiguity, particularly with near-optimal demonstrations. Our method departs from scalar paradigms entirely. Using score-based generative modeling with stochastic differential equations (SDEs), we fit the gradient field of expert trajectory distributions and recover dense per-step rewards via potential matching, hence, no simulator or rollout required.

Preliminaries

Mixture of Experts. MoE is a conditional computation framework where a set of K expert models $\mathcal{E} = \{f_1, f_2, \ldots, f_K\}, \quad f_k : \mathbb{R}^d \to \mathbb{R}^l$, $k = 1, \ldots, K$, is selectively activated via an input-dependent gating mechanism. Given input $\mathbf{x} \in \mathbb{R}^d$, a gating function $\mathcal{G}_\eta : \mathbb{R}^d \to \Delta^{K-1}$ produces a categorical distribution $\mathcal{G}_\eta(\mathbf{x}) = [\mathcal{G}_\eta(1 \mid \mathbf{x}), \ldots, \mathcal{G}_\eta(K \mid \mathbf{x})]$ over experts, where Δ^{K-1} denotes the probability simplex. To reduce computation, only the top-k experts are activated per input. Let $H(\mathbf{x}) = \mathbf{x}W_g + \mathbf{b}_g + \boldsymbol{\varepsilon}$ be the pre-activation logits with noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \Sigma)$, and define $\mathcal{I}_{\text{top}} = \text{TopK}(H(\mathbf{x}), k)$ as indices of the k largest entries with $\widetilde{H}i(\mathbf{x}) = H_i(\mathbf{x})$ if $i \in \mathcal{I}_{\text{top}}$, and $-\infty$ otherwise. The gated distribution is computed via masked softmax:

$$\mathcal{G}_\eta(i \mid \mathbf{x}) = \exp(\widetilde{H}_i(\mathbf{x}))/ \sum_{j \in \mathcal{I}_{\text{top}}} \exp(\widetilde{H}_j(\mathbf{x})). \quad (1)$$

The final output of MoE is a weighted sum of the top-k expert outputs $y(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\text{top}}} \mathcal{G}_\eta(i \mid \mathbf{x}) \cdot f_i(\mathbf{x})$.

In standard settings, $\mathcal{G}_\eta$ is often trained using supervised signals, such as classification loss or routing labels. However, in chess, there is no such supervision for which expert or configuration is optimal at each position. The only available signal is the game outcome, encoded as a scalar reward (+1, 0, -1), which is observed only at the end of the game. This creates a long-horizon credit assignment problem that standard supervised or RL struggles to address. To overcome this, we formulate expert selection as an IRL problem, learning dense, per-move rewards from expert trajectories. This enables the gating policy to imitate high-quality routing behavior without environment interaction or ground-truth labels.

Inverse Reinforcement Learning. In many real-world scenarios, including chess, the reward function that drives expert behavior is not explicitly known, but we may observe expert trajectories. IRL aims to infer this unknown reward function by assuming that the expert acts (approximately) optimally under some latent reward. Formally, we consider a finite-horizon MDP $\mathrm{M} = (\mathcal{S}, \mathcal{A}, P, \gamma)$, where a trajectory $\tau = (s_0, a_0, \ldots, s_T, a_T)$ is assigned a cumulative return $R_\theta(\tau) = \sum_{t=0}^{T} \gamma^t r_\theta(s_t, a_t)$. As in standard IRL, the expert can be modeled as sampling trajectories from the distribution:

$$p_\theta(\tau) = \frac{\exp(R_\theta(\tau))}{Z_\theta}, \quad Z_\theta = \int_\tau \exp(R_\theta(\tau))\, d\tau. \quad (2)$$

The objective is to learn a reward function r_θ such that the induced distribution $p_\theta(\tau)$ approximates the unknown

expert distribution $p^*(\tau)$. As we cannot access $p^*(\tau)$ but we have expert demonstrations $\mathcal{D}_{\text{exp}} = \{\tau^{(i)}\}_{i=1}^N \sim p^\star(\tau)$, the learning objective becomes learning θ that maximizes the log-likelihood of the data:

$$\begin{aligned}\theta^\star &= \arg\max_\theta \frac{1}{N}\sum_{i=1}^N \log p_\theta(\tau^{(i)}) \\ &= \arg\max_\theta \left[\frac{1}{N}\sum_{i=1}^N R_\theta(\tau^{(i)}) - \log Z_\theta\right]\end{aligned} \tag{3}$$

While elegant, this formulation requires computing the model distribution $p_\theta(\tau)$, which involves evaluating the partition function $Z_\theta = \int_\tau \exp(R_\theta(\tau))d\tau$. This integral sums over all possible trajectories $\tau = (s_0, a_0, \ldots, s_T, a_T)$, whose number grows exponentially with time horizon T. Since the state space $\mathcal{S}$ and action space $\mathcal{A}$ are often very large, the trajectories space becomes combinatorially intractable. Thus evaluating Z_θ exactly, or even approximating it via sampling, requires extensive simulator rollouts. In offline, black-box settings like ours, such interaction is not feasible. This motivates our alternative approach, which avoids estimating the partition function and instead learns dense reward signals directly from expert demonstrations.

Hexaïssa

This section introduces our Hexaïssa, a framework for expert engine routing in chess. It addresses two core challenges: reward sparsity and long-horizon credit assignment to beat the current best chess engine Stockfish. We begin by collecting a diverse set of game trajectories, ranging from expert-like to sub-optimal trajectories. These trajectories are embedded into a latent space where their structure reveals a natural separation: expert-like behaviors cluster into a high-quality region, while weaker behaviors map to lower-quality areas. To infer dense per-move reward without environment interaction, we apply stochastic perturbations to latent trajectories and learn a Stein score function, which estimates the gradient of the underlying expert trajectory distribution. This score field consistently points toward the high-quality region in latent space, allowing us to indirectly extract a potential-based reward function that attributes higher reward to moves aligned with expert behavior. In the second phase, we train a gating network to select the best engine per board position, using policy gradients guided by the latent reward. The overview of our Hexaïssa is depicted in Fig. 1.

Reward Learning

In this phase, we target two key goals: (1) finding a compact representation of trajectories, and (2) avoiding explicit computation of Z_θ. For (1), we project trajectories into a lower-dimensional latent space via a trajectory autoencoder. If the decoder is well-trained, each expert trajectory $\tau^{(i)} \sim p^*(\tau)$ can be accurately encoded as some latent point z, and the expert distribution $p^*(\tau)$ is implicitly represented by a transformed density $p^*(z)$. The learning problem then reduces to modeling this latent distribution. For (2), instead of modeling $p^*(z)$ directly, we estimate its score function $\nabla_z \log p^*(z)$, which enables efficient gradient-based learning without evaluating partition function Z_θ (see Appendix **B**).

To transform $p^*(\tau)$ to $p^*(z)$, we first partitioned the offline dataset into two types of trajectories: $\mathcal{D}^+ = \{\tau^{+(i)}\}_{i=1}^{N_+}$ and $\mathcal{D}^- = \{\tau^{-(j)}\}_{j=1}^{N_-}$, where each $\tau^{+(i)} = (s_1, a_1, \ldots, s_T, a_T)$ corresponds to a high-quality trajectory (e.g., from Stockfish or Obsidian in deep configurations), and each $\tau^{-(j)}$ represents a lower-quality counterpart (e.g., from shallow or time-constrained settings). Our goal is a low-dimensional latent embedding is learned in which desirable and undesirable behaviors are separated into distinct regions. This is achieved via a Variational Auto-Encoder (VAE), denoted as $\mathfrak{Z}(\tau, \phi, \omega) = d_\omega \circ e_\phi$, with two components: (i) Encoder $e_\phi : \mathbb{R}^n \to \mathbb{R}^m$, which maps each trajectory τ to a latent vector $z = e_\phi(\tau)$ on a Riemannian manifold $\mathcal{M}^d$; (ii) Decoder $d_\omega : \mathbb{R}^m \to \mathbb{R}^n$, which reconstructs the trajectory from its latent representation. The encoder–decoder pair is trained by maximizing the standard Evidence Lower Bound (ELBO) on the marginal log-likelihood $\log p(\tau)$:

$$\begin{aligned}\mathcal{L}_{\text{ELBO}}(\phi, \omega) = \mathbb{E}_{q_\phi(z|\tau)}\left[\log p_\omega(\tau \mid z)\right] \\ - \text{KL}\left(q_\phi(z \mid \tau) \,\|\, p_0(z)\right),\end{aligned} \tag{4}$$

where the prior $p_0(z)$ is typically chosen as the standard Gaussian $\mathcal{N}(0, I)$ to encourage smoothness and full support over the latent space. Importantly, because the score function is learned in the latent space, we must map it back to the original trajectory space in order to obtain a reward signal that is meaningful for actual gameplay decisions. The following lemma formalizes the relationship between $p^*(z)$ and $p^*(\tau)$ via volume correction, allowing us to pull back score information to the trajectory domain. (All proofs are in Appendix **A**).

(Push-forward density under immersion) Let $d_\omega : \mathbb{R}^m \to \mathbb{R}^n$ be a C^1 immersion of constant rank $m \leq n$, with manifold $\mathcal{M} = d_\omega(\mathbb{R}^m) \subset \mathbb{R}^n$. Given the expert distribution over trajectories $\tau \in \mathcal{M}$ satisfies $p^*(\tau) \propto \exp(R^*(\tau))$, the induced density over latent variables $z \in \mathbb{R}^m$ results in $p^*(z) = p^*(d_\omega(z)) \cdot \sqrt{\det\left(J_{d_\omega}(z)^\top J_{d_\omega}(z)\right)}$ where $J_{d_\omega}(z) = \frac{\partial d_\omega(z)}{\partial z} \in \mathbb{R}^{n\times m}$.

By Lemma , the latent score can theoretically be linked back to the trajectory space via the decoder map $d_\omega : z \mapsto \tau$. Specifically, the gradient of the log-density in latent space admits the following pullback formulation:

$$\begin{aligned}\nabla_z \log p^*(z) = J_{d_\omega}(z)^\top \nabla_\tau R_\theta(\tau) \\ + \tfrac{1}{2}\nabla_z \log\det(J_{d_\omega}(z)^\top J_{d_\omega}(z))\end{aligned} \tag{5}$$

where $\tau = d_\omega(z)$ and $J_{d_\omega}(z) \in \mathbb{R}^{n\times m}$ is the Jacobian of the decoder. Since we assume $p^*(\tau) \propto \exp(R^*(\tau))$, the term $\nabla_\tau \log p^*(\tau)$ simplifies to $\nabla_\tau R^*(\tau)$, i.e., the reward gradient. This construction allows us to learn meaningful reward gradients aligned with expert behavior entirely within the latent space, while preserving interpretability in the trajectory space through the decoder.

In Eq. (5), the first term represents the pullback of the trajectory-space reward gradient via the Jacobian decoder, while the second term corresponds to the gradient of the

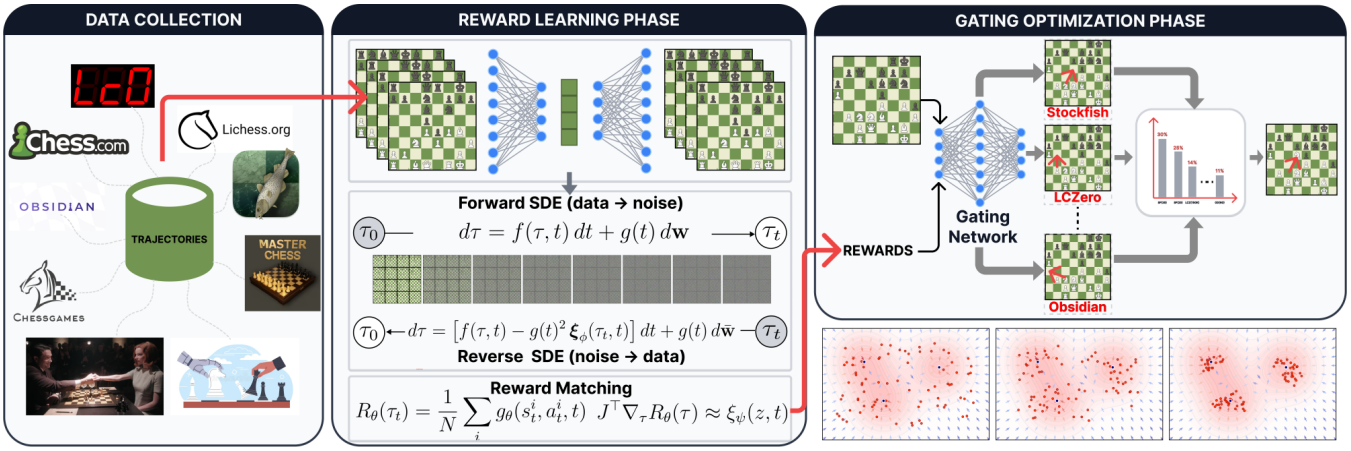


Figure 1: Overview of our offline expert-routing framework. We begin by collecting expert trajectories from multiple engines (e.g., Stockfish, LCZero, Obsidian) and human game databases. In the IRL phase (center), trajectories are encoded into a latent space and perturbed by a forward SDE; we then learn the score function and recover a reward potential via gradient matching, yielding per-step rewards. In the Gating Optimization phase (right), these learnt rewards supervise a gating policy that selects the best expert for each state, training entirely offline to maximize cumulative inferred reward. Below, contour plots visualize how score matching shapes the latent trajectory distribution.

log-determinant of the pullback metric. This second term captures the volume distortion introduced when pushing the expert distribution from trajectory space back to latent space. Intuitively, a smaller volume term implies the transformation is closer to volume-preserving. In a special case $J^\top J \approx I$, the distortion vanishes and the transformation is locally isometric.

From a practical point of view, it is often desirable to preserve volume when learning over latent spaces. A natural regularization strategy is to penalize deviations from unit determinant via $\lambda \cdot \left|\nabla_z \log\det\left(J^\top J\right)\right|$, which encourages the decoder to act locally like an isometry. However, computing the gradient $\frac{1}{2}\nabla_z \log\det\left(J^\top J\right)$ is nontrivial, especially when the latent dimension m is large or the decoder d_ω is deep and highly nonlinear. To facilitate this computation, let $G(z) = J_{d_\omega}(z)^\top J_{d_\omega}(z) \in \mathbb{R}^{m\times m}$, we first introduce a classical trace identity for the gradient of a log determinant:

(The Gradient of the Log- Determinant via Trace). Let $G(z) \in \mathbb{R}^{m\times m}$ be a C^1 family of invertible symmetric matrices parameterized by $z \in \mathbb{R}^m$, then, $\nabla_z \frac{1}{2}\log\det G(z) = \frac{1}{2}\operatorname{tr}\left(G(z)^{-1}\nabla_z G(z)\right)$.

(Hutchinson Approximation for Latent Volume Gradient). Let $d_\omega : \mathbb{R}^m \to \mathbb{R}^n$ be a C^2 map with full-rank Jacobian $J_{d_\omega}(z) \in \mathbb{R}^{n\times m}$. The gradient of the log-determinant can be approximated by $\nabla_z \frac{1}{2}\log\det G(z) \approx \frac{1}{2k}\sum_{i=1}^k v_i^\top G^{-1}\nabla_z G G^{-1} v_i$.

From Lemma and Theorem , the gradient of the log-determinant term in the latent reward can be efficiently approximated. Specifically, for the pullback metric $G(z) = J^\top J$, the gradient $\nabla_z\left(\frac{1}{2}\log\det G(z)\right)$ is expressed as a trace, and approximated using random probe vectors $\{v_i\}$ by $\frac{1}{2k}\sum_{i=1}^k v_i^\top G^{-1}\nabla_z G\, G^{-1} v_i$. This stochastic estimator makes the volume distortion term $\frac{1}{2}\nabla_z \log\det\left(J^\top J\right)$ tractable even in high dimensional settings, facilitating volume-preserving regularization in Eq. (4) through a penalty term such as $\lambda \cdot \left|\nabla_z \log\det\left(J^\top J\right)\right|$ applied to decoder d_ω.

Latent Score Learning. To estimate $\nabla_z \log p^*(z)$, we perturb z with Itô SDE (Ito et al. 1951), $dz = f(z,t)\,dt + \sqrt{2\varepsilon}dW_t$ whose marginal $p_t(z)$ evolves under the Fokker–Planck equation (Solin, Tamir, and Verma 2021). The reverse-time SDE, $dz = \left[f(z,t) - 2\varepsilon\,\nabla_z \log p_t(z)\right]dt + \sqrt{2\varepsilon}dW$ shows $\nabla_z \log p_t(z)$ drives trajectories toward high density regions (i.e expert-like behavior). We train a score network $\xi_\psi(z)$ via denoising score matching on samples from $\mathcal{D}^+$:

$$\begin{aligned}\mathcal{L}_{\text{score}}(\psi) &= \mathbb{E}_{p(z)}\left\|\xi_\psi(z) - \nabla_z \log p_t(z)\right\|^2 \\ &= \frac{1}{2}\mathbb{E}_{p(z)}\left[\xi_\psi(z)^2\right] + \mathbb{E}_{p(z)}\left[\nabla_z \xi_\psi(z)\right]\end{aligned} \tag{6}$$

At $t = 0$, the learned score network satisfies $\xi_\psi(z,0) \approx \nabla_z \log p^*(z)$, which, through the decomposition above, yields a dense gradient field reflecting expert-preferred directions in latent space. This score not only serves as a proxy for credit assignment, but also offers a principled direction along which latent scores can be adjusted to improve the quality of their corresponding trajectories. Next, we demonstrate how the reward of a trajectory τ can be indirectly recovered through $\nabla_z \log p^*(z)$.

(Reward Improvement via Latent Score Ascent). Given d_ω is volume preserving and $\xi_\phi(z) \approx \nabla_z \log p^*(z) = J^\top \nabla_\tau R^*(\tau) + \frac{1}{2}\nabla_z \log\det\left(J^\top J\right)$, where $\tau = d_\omega(z)$. Let the latent update be $z \leftarrow z + \epsilon \cdot \xi_\phi(z)$ with small $\epsilon > 0$, then the decoded trajectory $\tau' = d_\omega(z)$ satisfies $R^*(\tau') > R^*(\tau)$ whenever $\nabla_\tau R^*(\tau) \neq 0$.

Theorem guarantees that latent score ascent-i.e., taking a small step in the direction of the learned latent score function-provably improves the decoded trajectory reward, as long as the current trajectory is not at a stationary point. This result

justifies the use of $\nabla_z \log p^*(z)$ not only as a diagnostic for expert-likeness, but also as a constructive reward gradient in the latent space.

Recovering Per-Step Rewards. As we regularize the decoder with $\lambda \cdot \left|\nabla_z \log \det(J^\top J)\right|$, the volume distortion term in Eq. (5) becomes negligible. Hence, we treat the learned latent score $\xi_\psi(z)$ as a pullback of the trajectory-space reward gradient via the decoder Jacobian $\xi_\psi(z) \approx J_{d_\omega}^\top(z) \nabla_\tau R^*(\tau)$ where $\tau = d_\omega(z)$. To recover an interpretable per-step reward function $r_\theta(s, a, t)$ with $R_\theta(\tau) = \sum_{t=1}^{T} r_\theta(s_t, a_t, t)$, we train r_θ to minimize the discrepancy between its induced reward gradient and the pullback of the learned score:

$$\mathcal{L}_{\text{align}}(\theta) = \mathbb{E}_{\tau \sim \mathcal{D}^+} \left\| J_{d_\omega}^\top(z) \nabla_\tau R_\theta(\tau) - \xi_\psi(z) \right\|^2 \quad (7)$$

This alignment ensures that the recovered reward faithfully explains the score signal and is consistent with expert behavior in trajectory space.

Gating Optimization

Given the latent reward function $r_\theta(s, a, t)$ obtained via score-based IRL, we reformulate gating network $\mathcal{G}_\eta$ as policy $\pi_\eta(e \mid s)$ and train it through direct interaction with the chess environment. Unlike conventional RL setups that rely on sparse terminal rewards, our approach leverages dense, per-move latent rewards to guide expert routing via RL. Specifically, at each state s_t, the gating network selects an expert engine $e_t \sim \pi_\eta(\cdot \mid s_t)$, which proposes an action $a_t \sim e_t(\cdot \mid s_t)$. The resulting transition (s_t, a_t, s_{t+1}) is evaluated by the fixed reward function r_θ, providing a fine-grained training signal without further human supervision or engine relabeling.

To optimize the routing policy, we adopt a clipped policy gradient framework such as Proximal Policy Optimization (PPO) (Schulman et al. 2017) with advantage estimates computed from latent rewards. Letting $\hat{R}_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_\theta(s_{t'}, a_{t'}, t')$ and $\hat{A}_t = \hat{R}_t - V_\eta(s_t)$, the policy is updated using clipped importance-weighted objectives $\mathcal{L}_{\text{PPO}}(\eta) = \mathbb{E}_t[\min(\rho_t(\eta) \cdot \hat{A}_t, \text{clip}(\rho_t(\eta), 1-\epsilon, 1+\epsilon) \cdot \hat{A}_t)]$ where $\rho_t(\eta) = \pi_\eta(e_t \mid s_t) / \pi_{\text{old}}(e_t \mid s_t)$ and the value loss is given by $\mathcal{L}_{\text{value}}(\eta) = \mathbb{E}_t[(V_\eta(s_t) - \hat{R}_t)^2]$. Besides standard PPO components, we use an entropy regularization, promoting exploration over diverse engine strategies:

$$\mathcal{R}_{\text{ent}}(\eta) = -\mathbb{E}_{s \sim \mathcal{D}} \left[\sum_{e \in \mathcal{E}} \pi_\eta(e \mid s) \log \pi_\eta(e \mid s) \right] \quad (8)$$

The final training objective combines latent reward maximization with diversity-inducing penalties:

$$\begin{aligned} \mathcal{L}_{\text{total}}(\eta) = &-\mathcal{L}_{\text{PPO}}(\eta) + \lambda_V \cdot \mathcal{L}_{\text{value}} \\ &+ \lambda_{\text{ent}} \cdot \mathcal{R}_{\text{ent}}(\eta) \end{aligned} \quad (9)$$

where $\lambda_V, \lambda_{\text{ent}} \in \mathbb{R}_{\geq 0}$ are tunable coefficients. This formulation enables the gating network to refine itself through interaction, guided by dense reward signals rooted in expert imitation, while maintaining diversity and stability.

Experimental Evaluation

Settings. Our offline dataset for training the model comprises 2 million chess trajectories, covering a wide range of strategic and tactical patterns. For evaluation, we conduct experiments on two standard opening books, Opening Book Noomen S25 DivP book (Noomen 2025a), Noomen S26 DivP book (Noomen 2025b), containing 150 opening lines in total, under four types of time controls: Bullet (1m+1s), Blitz (3m+2s), Blitz (5m+2s), and the TCEC standard (30m+3s). Details of the dataset, and implementation specifics , such as training algorithms, model architecture, hyperparameter, model size, and training costs are provided in Appendix **C1**.

Baselines. We conduct a comprehensive comparison of our proposed method against three groups of baselines, each corresponding to a distinct evaluation aspect. The first aspect concerns playing strength, where Hexaïssa is evaluated through direct head-to-head competition with three widely-used commercial standalone engines, namely Stockfish v.17 (Isenberg 2021), LCZero (LCZero Team 2021), and Obsidian (Gabriel 2024), to assess its effectiveness relative to individual expert systems. The second aspect focuses on comparisons with existing MoE-based methods, including two baselines, Uniform Mixture (Zaharescu and Horaud 2009) and M2CTS (Helfenstein et al. 2024), to highlight the importance of our fine-grained reward learning mechanism compared to simple routing heuristics. The third aspect evaluates the contribution of our score-based reward learning, by replacing Hexaïssa's reward module with several representative IRL backbones, including T-REX, (Jiang et al. 2023), IQLearn (Garg et al. 2021), LS-IQ (Al-Hafez et al. 2023), and Energy Guidance MaxEnt (EGME) (Chao et al. 2024), to analyze the impact of our proposed score-based SDE reward learning approach on overall performance. Win rate serves as the primary evaluation metric across all comparisons.

Playing Strength Evaluation. Fig. 2 presents the head-to-head competition of Hexaïssa against three leading commercial engines: Stockfish (35 nodes), Obsidian (60 nodes), and LCZero (150,000 nodes). The results are averaged across four time controls, with detailed breakdowns provided in Appendix **C2**. When playing as White, Hexaïssa achieves an average win rate of 80%: 76% against Stockfish, 83% against Obsidian, and 80% against LCZero. When playing as Black, the system maintains consistently strong performance with a high draw rate and very few losses, notably achieving an 82% draw rate against Stockfish. Hexaïssa outperforms all individual engines in all time controls. Owing to the dense per-move reward signals provided by the score-based IRL component, the gating network learns to dynamically route queries to the most suitable expert for each evolving position. For simple positions that can be resolved equally well across engines, the gating network preferentially selects shallower depths to reduce computational overhead, which is particularly beneficial in fast time controls such as blitz or bullet. This, in turn, allows for more complex or critical and challenging positions later in the game to be processed using deeper configurations, with longer thinking time, leading to a very high winning rate. Detailed analysis of expert selection can be found in Appendix **C3**.

Against other MoE Chess Engines. As shown in Table 1,

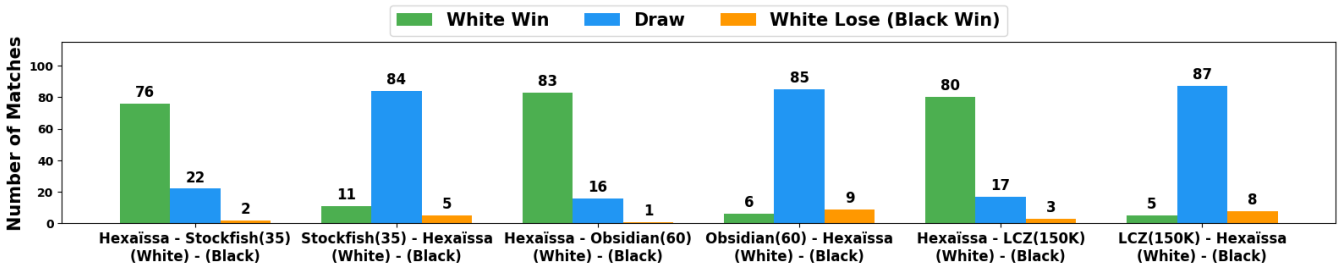


Figure 2: Head-to-head match performance of Hexaïssa against baseline engines: Stockfish (depth 35), LCZero (150K nodes), and Obsidian (depth 60). Results are averaged across four time controls.

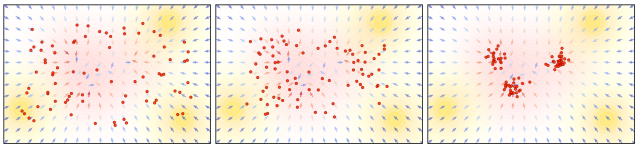


Figure 3: Visualization of the reverse SDE process in the latent space. From left to right, the learned score function (vector field) progressively refines noisy samples (red dots), guiding them to converge on the high-probability modes of the expert distribution, which correspond to high-reward trajectories.

Method	Win (%)	Draw (%)	Lose (%)
Uniform Mixture	38.8	39.1	22.1
M2CTS	47.2	33.5	19.3
T-REX (IRL Variants)	52.1	31.4	16.5
IQLearn (IRL Variants)	53.5	30.8	15.7
LS-IQ (IRL Variants)	54.8	30.2	15.0
EGME (IRL Variants)	55.5	30.1	14.4
Hexaïssa (Ours)	**58.5**	**29.0**	**12.5**

Table 1: Tournament results comparing Hexaïssa against baseline methods (Win/Draw/Loss %). Averages over matches with Stockfish, LCZero, and Obsidian.

Hexaïssa achieves a higher average win rate than M2CTS by 11.3 percentage points (47.2% vs. 58.5%) and surpasses Uniform Mixture by 19.7 percentage points (38.8% vs. 58.5%). The poor performance of Uniform Mixture is unsurprising, as randomly selecting experts lacks any form of contextual awareness, leading to inefficient decisions. Meanwhile, the performance gap with M2CTS highlights the limitations of phase-based expert routing. Instead of learning a flexible routing policy, M2CTS relies on a rigid set of rules (e.g., counting the number of remaining pieces) to categorize games into predefined phases such as 'opening', 'middlegame', or 'endgame'. This approach often leads to suboptimal expert selection, as a position with high tactical complexity may be misclassified simply due to a reduced number of pieces. In contrast, Hexaïssa addresses these limitations through a per-move expert routing mechanism, where each board state is independently evaluated to determine the most suitable expert configuration.

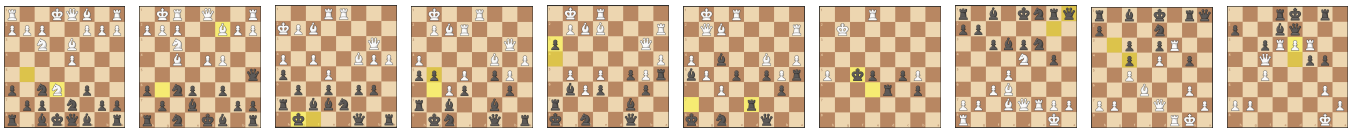
Compared to other IRL methods, Hexaïssa achieves higher average win rates compared to T-REX (+6.4 points, 52.1%), IQLearn (+5.0 points, 53.5%), LS-IQ (+3.7 points, 54.8%), and EGME (+3.0 points, 55.5%). These results demonstrate that score-based trajectory modeling provides better reward learning compared to traditional IRL methods, which struggle when relying solely on sparse terminal outcomes. This is an inherent challenge in chess, where intermediate move quality is difficult to assess without full game completion.

Reverse SDE Convergence Analysis. Fig. 3 illustrates the reverse SDE process in latent space, where the background colors indicate trajectory quality: yellow areas correspond to poor chess moves, while red areas represent expert-level play with high rewards. The red dots denote randomly initialized trajectories, initially scattered across both good and bad regions. As the reverse SDE unfolds over four stages, the learned score function gradually guides all samples away from the yellow low-reward areas toward the red expert zones. By the final stage, all trajectories have converged into distinct clusters located within high-quality regions. This convergence process highlights the effectiveness of our method in transforming any input trajectory into expert-like behavioral patterns (see Appendix **C4** for details).

Reward Function Visualization. Fig. 4 illustrates Hexaïssa's reward function learning process across 800 randomly sampled chess positions. The reward values fluctuate between [-0.1, 0.3], with high points (orange circles) corresponding to White winning positions and low points (blue circles) corresponding to Black wins or disadvantageous White positions. The results demonstrate that the system successfully learned to distinguish between good and bad board states: positions with clear tactical advantages (such as center control, king attacks) receive high rewards, while passive positions or material losses receive low rewards. This continuous variation shows that the score-based IRL approach successfully generates dense reward signals instead of relying solely on terminal game outcomes, enabling the gating network to learn effective routing policies.

Game Analysis. A strong indicator of engine strength is the presence of a win-win pair, where one engine defeats the other as both White and Black in the same opening or position. Achieving this in a balanced position shows strong performance, but doing so in an imbalanced one is far more impressive, as it reflects the ability to overcome a disadvantage and still prevail, revealing a deeper strategic understand-

Figure 4: Visualization of Hexaïssa's learned reward function over 800 chess positions. The model assigns high rewards to positions advantageous for White (orange circles) and low rewards to positions favoring Black (blue circles).



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 5: Comparative game analysis of Hexaïssa versus Stockfish D35 in Bullet where Hexaïssa won both games as Black or White on the same opening. Frames (a–g) show critical positions from a game where Hexaïssa plays as Black, while frames (h–j) depict key moments where it plays as White.

ing. We selected to analyze one such pair of Hexaïssa and Stockfish, shown in Fig. 5. More game analysis is in Appendix **C5**.

Fig. 5a shows the TCEC starting position with Stockfish D35 as White and Hexaïssa as Black, in a bullet game. White starts with a one-pawn advantage and a dominant position, as Black's development soon becomes shattered. In Fig. 5b, though Black is up a knight, White retains pressure due to Black's disorganized back rank. At this critical point, Hexaïssa calls on Obsidian to find a development plan, enabling Black to gradually regroup their pieces to more favorable squares.

At the position shown in Fig. 5c, Stockfish begins to lose its advantage. Black has managed to hold on to their extra knight and repositioned the king to safety, allowing Hexaïssa to shift to offense as Stockfish lashes out in desperation. Fig. 5d-5f illustrate how Black paves its way to wining. With Stockfish over extended on the kingside, Hexaïssa sacrifices a pawn to open space and activate its pieces. A second pawn is given for the same purpose (Fig. 5e), similar to an AlphaZero and LCZero style idea of using flank pawns to clear entry lines. Fig. 5f shows a striking transformation: from a cramped setup, Hexaïssa now controls the board, while White is left with scattered pawns and will soon fall.

Using the sudden shift in momentum to our advantage (Fig. 5g), we opt to trade off the majority of the pieces to enter an easily winning endgame. Notice how White's pawns are completely disconnected whereas ours are well protected. This game is a testament to how our Hexaïssa is capable of defending and maneuvering patiently, before going to the counterattack decisively, and sacrifice material for lasting positional gains.

In contrast, when Stockfish played Black in the same opening, it quickly fell behind. By move 21 (Fig. 5h), six of its pieces remained undeveloped, while Hexaïssa had already mobilized fully. As shown in Fig. 5i, Stockfish was forced into poor trades while struggling to untangle, ending up with shattered, isolated pawns. By move 34 (Fig. 5j), it finally activated its queenside rook, just as Hexaïssa delivered a decisive tactic that won the queen.

The key difference is that Hexaïssa prioritizes development and piece activity, even at the cost of material, while Stockfish plays more greedily and gets punished for it. These win-wins highlight Hexaïssa's strength in sacrificing material for long-term advantage, unlike many engines that cling to pieces and lose ground. By valuing activity over material, Hexaïssa consistently gains the upper hand against the top engines.

Conclusion

We present Hexaïssa, a theoretically grounded yet effective framework for adaptive chess engine routing via Mixture-of-Experts and latent score learning-based IRL. By estimating per-step rewards in latent space, it enables efficient training of gating policies in sparse, long-horizon settings. Empirical results show that Hexaïssa outperforms individual engines and MoE baselines, establishing a new direction for expert composition in chess.

References

Al-Hafez, F.; Tateo, D.; Arenz, O.; Zhao, G.; and Peters, J. 2023. Ls-iq: Implicit reward regularization for inverse reinforcement learning. *arXiv preprint arXiv:2303.00599*.

Chao, C.-H.; Feng, C.; Sun, W.-F.; Lee, C.-K.; See, S.; and Lee, C.-Y. 2024. Maximum entropy reinforcement learning via energy-based normalizing flow. *Advances in Neural Information Processing Systems*, 37: 56136–56165.

Chao, C.-H.; Sun, W.-F.; Cheng, B.-W.; Lo, Y.-C.; Chang, C.-C.; Liu, Y.-L.; Chang, Y.-L.; Chen, C.-P.; and Lee, C.-Y. 2022. Denoising Likelihood Score Matching for Conditional Score-based Data Generation. In *International Conference on Learning Representations*.

Chen, T.; Huang, S.; Xie, Y.; Jiao, B.; Jiang, D.; Zhou, H.; Li, J.; and Wei, F. 2022. Task-specific expert pruning for sparse mixture-of-experts. *arXiv preprint arXiv:2206.00277*.

Chen, Z.; Shen, Y.; Ding, M.; Chen, Z.; Zhao, H.; Learned-Miller, E. G.; and Gan, C. 2023. Mod-squad: Designing mixtures of experts as modular multi-task learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11828–11837.

Danihelka, I.; Guez, A.; Schrittwieser, J.; and Silver, D. 2022. Policy improvement by planning with Gumbel. In *International Conference on Learning Representations*.

David, O. E.; Netanyahu, N. S.; and Wolf, L. 2016. Deepchess: End-to-end deep neural network for automatic learning in chess. In *International Conference on Artificial Neural Networks*, 88–96. Springer.

Do, N. H. K.; Ngo, B.; Kashuv, Y.; Pham, C. V.; Tong, H.; and Thai, M. T. 2025a. Hephaestus: Mixture Generative Modeling with Energy Guidance for Large-scale QoS Degradation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Do, N. H. K.; Nguyen, T.; Hassanaly, M.; raed alharbi; Seo, J. T.; and Thai, M. T. 2025b. Swift Hydra: Self-Reinforcing Generative Framework for Anomaly Detection with Multiple Mamba Models. In *The Thirteenth International Conference on Learning Representations*.

Gabriel, G. 2024. Obsidian Chess Engine. GitHub Repository.

Garg, D.; Chakraborty, S.; Cundy, C.; Song, J.; and Ermon, S. 2021. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34: 4028–4039.

Gupta, D.; Mihucz, G.; Schlegel, M.; Kostas, J.; Thomas, P. S.; and White, M. 2021. Structural credit assignment in neural networks using reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 30257–30270.

Helfenstein, F.; Blüml, J.; Czech, J.; and Kersting, K. 2024. Checkmating one, by using many: Combining mixture of experts with mcts to improve in chess. *arXiv preprint arXiv:2401.16852*.

Isenberg, G. 2021. Stockfish. Chess Programming Wiki.

Ito, K.; Itô, K.; Itô, K.; Mathématicien, J.; Itô, K.; and Mathematician, J. 1951. *On stochastic differential equations*, volume 4. American Mathematical Society New York.

Jiang, Q.; Li, F.; Ren, T.; Liu, S.; Zeng, Z.; Yu, K.; and Zhang, L. 2023. T-rex: Counting by visual prompting. *arXiv preprint arXiv:2311.13596*.

LCZero Team. 2021. Announcing Ceres. LCZero Blog.

Maharaj, S.; Polson, N.; and Turk, A. 2022. Chess AI: competing paradigms for machine intelligence. *Entropy*, 24(4): 550.

McIlroy-Young, R.; Wang, Y.; Sen, S.; Kleinberg, J.; and Anderson, A. 2021. Detecting individual decision-making style: Exploring behavioral stylometry in chess. *Advances in Neural Information Processing Systems*, 34: 24482–24497.

Nguyen, H.; Dam, H.; Do, N. H. K.; Tran, C.; and Pham, C. 2025. REM: A Scalable Reinforced Multi-Expert Framework for Multiplex Influence Maximization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25): 27099–27107.

Noomen, J. 2025a. Open Book: Division P - Season 25 (Noomen). https://www.mediafire.com/file/rachfj7z85mw0ti/Noomen_DivP_S25.pgn/file.

Noomen, J. 2025b. Open Book: Division P – Season26 (Noomen). https://www.mediafire.com/file/d01rgovtittkr3c/Noomen_DivP_S26.pgn/file.

Riquelme, C.; Puigcerver, J.; Mustafa, B.; Neumann, M.; Jenatton, R.; Susano Pinto, A.; Keysers, D.; and Houlsby, N. 2021. Scaling Vision with Sparse Mixture of Experts. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 8583–8595. Curran Associates, Inc.

Ruoss, A.; Delétang, G.; Medapati, S.; Grau-Moya, J.; Wenliang, L. K.; Catt, E.; Reid, J.; and Genewein, T. 2024. Grandmaster-level chess without search. *CoRR*.

Schrittwieser, J.; Hubert, T.; Mandhane, A.; Barekatain, M.; Antonoglou, I.; and Silver, D. 2021. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34: 27580–27591.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sikchi, H.; Saran, A.; Goo, W.; and Niekum, S. 2022. A ranking game for imitation learning. *arXiv preprint arXiv:2202.03481*.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Solin, A.; Tamir, E.; and Verma, P. 2021. Scalable inference in SDEs by direct matching of the Fokker–Planck–Kolmogorov equation. *Advances in Neural Information Processing Systems*, 34: 417–429.

Willi, T.; Obando-Ceron, J.; Foerster, J.; Dziugaite, K.; and Castro, P. S. 2024. Mixture of Experts in a Mixture of RL settings. *arXiv preprint arXiv:2406.18420*.

Zaharescu, A.; and Horaud, R. 2009. Robust factorization methods using a gaussian/uniform mixture model. *International Journal of Computer Vision*, 81(3): 240–258.