UMARM: UNIFIED MULTI-AGENT REWARD MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Behavior cloning, contrastive representation learning, and preference optimization are independently-motivated approaches behavioral modeling. We show that under a player-conditioned reward $r(s, a, z, c)$, these objectives share common structure: the behavior cloning optimum recovers the InfoNCE-optimal critic up to action-independent terms (Theorem 1), a principled adaptive schedule with super-linear decay guarantees convergence to the behavioral and preference objectives alone (Theorem 2), and the combined loss is equivalent to a MaxEnt goal-conditioned RL problem whose optimal reward recovers the player-conditioned data distribution (Theorem 3). UMARM (**U**nified **M**ulti-**A**gent **R**eward **M**atching) instantiates this framework for multi-agent behavioral modeling, jointly learning a continuous player latent $\boldsymbol{z}_p$ and a conditioned policy $\pi(a \mid s, \boldsymbol{z}_p, c)$ from raw trajectories alone — no skill ratings, agent identifiers, or per-agent parameters are required. Applied to chess across $\sim$4.3 million players spanning novice to grandmaster, UMARM achieves 56.9% top-1 move prediction accuracy, outperforming the parameter-matched Maia-3 model. Constructing $\boldsymbol{z}_p$ zero-shot from moves in the current game, accuracy reaches 56.1% overall and 62.9% after 40 moves of context. ELO ratings are recoverable from the latent geometry ($R^2 = 0.79$) despite never being supplied, and the preference head enables match outcome prediction between unrelated opponents. All capabilities emerge from the unified objective without task-specific supervision.

1 INTRODUCTION

Superhuman engines such as AlphaZero (Silver et al., 2018), Stockfish (The Stockfish developers, 2008), and Leela Chess Zero (The LCZero Authors, 2018) have largely solved the problem of optimal play in chess. However, optimal play is only one in a vast space of strategies: a novice's moves, a grandmaster's intuition under time pressure, and engine lines at different search depths are all qualitatively different strategiees for the same position. Modeling this full distribution, not only the best move but the move a *particular agent* would make under *particular conditions*, remains an open problem with applications in opponent modeling, cheating detection, human-AI alignment, and computational cognitive science.

Three independently-developed families bear on this problem. Behavior cloning (Pomerleau, 1991) learns to predict actions from demonstrations, contrastive representation learning (van den Oord et al., 2018; Chen et al., 2020) distinguishes entities by pushing similar representations together and dissimilar ones apart. and preference optimization (Rafailov et al., 2023; Xu et al., 2024) learns which behaviors are better by comparing outcomes. Each has been applied to behavioral modeling in isolation (Maia (McIlroy-Young et al., 2020), Maia 2 (Tang et al., 2024b) and Maia 3 (Monroe et al., 2026) use behavior cloning on ELO, contrastive methods have been used for player identification (McIlroy-Young et al., 2021), preference methods for policy alignment) but no prior work has examined their joint structure or asked whether they are in fact different views of the same underlying objective.

We show that they are. Under a player-conditioned reward $r_\theta(s, a, z, c)$, the three objectives collapse: behavior cloning via negative log-likelihood already performs implicit contrastive learning over players (Theorem 1); we adopt a reference-free Bradley-Terry likelihood directly as the preference objective, avoiding the reference model entirely; and the two-term loss is equivalent to a MaxEnt

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

goal-conditioned RL problem whose optimal reward recovers the player-conditioned data distribution for "playing like player z" (Theorem 3). We also derive a principled adaptive schedule (Theorem 2) that retains the explicit contrastive loss as an early optimization aid, with provable super-linear decay guarantees ensuring the asymptotic objective is determined by behavior cloning and preference alignment alone.

We call this framework **Unified Multi-Agent Reward Matching** (UMARM), a self-supervised offline method for learning the distribution of policies across a population of agents. Applied to chess, UMARM models agents ranging from novice players to grandmasters, encoding each player's strategic identity in a continuous latent $\boldsymbol{z}_p$ learned jointly with the policy, with no acecss to ELO or other skill information at training time.

Our contributions are:

- **A theoretical framework** revealing that behavior cloning, contrastive player identification, and preference-based quality calibration are governed by a shared reward structure. We show the NLL and InfoNCE optima are aligned (Theorem 1), derive an adaptive contrastive decay schedule with convergence guarantees (Theorem 2), and establish equivalence to goal-conditioned RL (Theorem 3), yielding the unified loss $\mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}} + \lambda_{\text{CL}}^{(t)}\mathcal{L}_{\text{CL}}$ in which all three objectives shape a shared player latent through coupled gradients while a separate preference head keeps quality calibration from directly altering move predictions.
- **Competitive human chess move prediction** conditioned on learned player latents rather than scalar ELO, with zero-shot generalization to unseen players, ELO estimation from a handful of games, and match outcome prediction across unrelated opponents with greater accuracy than raw ELO provides — all without per-player parameters or task-specific supervision, at a per-position inference cost matching the 5M Maia-3 baseline.

2 Related works

Superhuman chess engines AlphaZero (Silver et al., 2018), MuZero (Schrittwieser et al., 2020), Leela Chess Zero (The LCZero Authors, 2018), and Stockfish (The Stockfish developers, 2008) have achieved superhuman play via self-play RL, learned dynamics, and alpha-beta search, respectively; recent Transformer-based architectures (Monroe & Chalmers, 2024; Ruoss et al., 2024) further advance engine strength. All learn a single optimal or population-level policy; none model individual player heterogeneity from raw trajectories.

Human move prediction Maia (McIlroy-Young et al., 2020) first prioritized human-likeness over strength, training separate LC0 networks per ELO bracket and reaching $\sim 51\%$ move-matching accuracy. Each model, however, is fully independent and produces incoherent predictions across skill levels. Maia 2 (Tang et al., 2024b) addressed these gaps with attention conditioned on scalar ELO improving accuracy to roughly 53%, and Maia 3 (Monroe et al., 2026) scales the approach on a Chessformer backbone with a novel geometric attention bias. Allie (Zhang et al., 2025) instead models human play from move sequences and adds time-adaptive search. McIlroy-Young et al. (2022) demonstrated personalised fine-tuning of Maia to individual players via transfer learning; Maia4All (Tang et al., 2026) introduced a prototype-enriched meta-learning approach that adapts to individual players from as few as 20 games by bridging population and individual-level models; however, it still requires per-player embedding optimization. These approaches each consider behavior cloning in isolation: they condition on scalar metadata (ELO) rather than learned multidimensional latents, do not leverage contrastive signals to learn discriminative player representations, and lack any preference-based quality calibration. We show here that these three capabilities, behavioral prediction, player discrimination, and quality ordering, emerge naturally from a single unified reward structure that directly models policies per-agent.

Contrastive representation learning InfoNCE (van den Oord et al., 2018) and SimCLR (Chen et al., 2020) established contrastive learning for discriminative representations, and McIlroy-Young et al. (2021) use learned embeddings to identify chess players from their games. Eysenbach et al. (2022) proved contrastive learning on action-labeled trajectories is equivalent to goal-conditioned RL, with inner-product representations corresponding to Q-functions. We build on this to show the NLL

loss over any softmax-inducing reward already recovers the InfoNCE-optimal critic at convergence (Theorem 1), revealing that behavior cloning and contrastive learning are not independent objectives but share the same discriminative structure.

Preference optimization RLHF (Ziegler et al., 2019; Ouyang et al., 2022) and DPO (Rafailov et al., 2023) align models with human preferences via reward learning or direct optimization; CPO (Xu et al., 2024) eliminates the reference model through a uniform-prior upper bound. We adopt the same reference-free Bradley-Terry (Bradley & Terry, 1952) form but arrive at it directly as a likelihood, since the CPO bound can be vacuous in chess, where blunders produce arbitrarily sharp distributions. A separate preference head ensures that preference signals calibrate the player latent space without directly altering the policy head's move-prediction weights. In contrast, DPO and RLHF entangle preference and behavioral gradients in a single set of model parameters (Azar et al., 2024; Tang et al., 2024a); UMARM avoids this by routing preference gradient through a dedicated head sharing no parameters with the policy head.

Multi-agent imitation and behavioral modeling Few-shot imitation learning (Duan et al., 2017) indexes latents by *task* rather than *agent*; GAIL (Ho & Ermon, 2016) imitates a single demonstrator; Decision Transformers (Chen et al., 2021) condition on return-to-go for a single agent. MHR-Maia (Omi et al., 2025) models 47,000 players via per-player PEFT routing vectors, reaching $\sim 59\%$ with a median of 3,479 games per player, but requires per-player trainable parameters, precludes zero-shot inference, and lacks theoretical grounding connecting behavioral objectives or quality-calibrated latents. UMARM trades per-player accuracy for zero-shot generalization from as few as 5 reference games with no per-player optimization, unifying prediction, discrimination, and quality calibration under one objective with convergence guarantees.

3 Background

We model chess as a two-player, zero-sum MDP with deterministic transitions $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ over states $s \in \mathcal{S}$ (board configurations), legal actions $\mathcal{A}(s)$, and trajectories $\tau = (s_0, a_0, \ldots, s_T)$ with outcomes $y \in \{-1, 0, 1\}$. The standard formulation (Silver et al., 2018) learns a single policy maximizing win probability; we extend it to model heterogeneous *player-specific* behavior.

We augment this MDP with player latents and context: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{P}, \mathcal{Z}, \mathcal{C})$ where $\mathcal{P}$ is the set of players, $\mathcal{Z} \subseteq \mathbb{R}^{d_z}$ encodes player-specific information (tactical ability, positional style, opening preferences), and $\mathcal{C} \subseteq \mathbb{R}^{d_c}$ captures observable side information (time control, thinking time). Our objective is to learn $\pi_\theta(a \mid s, z, c)$ modeling the conditional action distribution of player z in context c:

$$\tau = \{s_i, a \mid s_{i-1} \sim \pi_\theta(\cdot \mid s_{i-1}, z^{p(i)}, c_{i-1})\}_{i=1}^{T}$$

where $p(i)$ alternates between white and black. We use a dataset $\mathcal{D} = \{(\tau_i, p_i^w, p_i^b, c_{i,\cdot}, y_i)\}_{i=1}^{M}$.

4 Methods

We begin from three objectives that are independently motivated for multi-agent behavioral modeling: behavior cloning ($\mathcal{L}_{\text{NLL}}$), contrastive player-matching ($\mathcal{L}_{\text{CL}}$), and comparative preference alignment ($\mathcal{L}_{\text{DPO}}$). Each objective captures a different aspect of the problem: $\mathcal{L}_{\text{NLL}}$ predicts actions, $\mathcal{L}_{\text{CL}}$ discriminates players, and $\mathcal{L}_{\text{DPO}}$ orders policies by quality. A natural starting point might be to optimize their weighted sum; we show, however, that these objectives are not independent: under a reward parameterization $r_\theta(s, a, z, c)$ that conditions on player latents, they share a common structure that collapses the three-term sum to a two-term loss with a decaying auxiliary. We adopt a reference-free Bradley-Terry likelihood directly as the preference objective, the contrastive loss becomes redundant at the NLL optimum, and a separate preference head provides quality calibration without directly altering move predictions. We present these reductions in sequence.

Let the per-player state-action-context reward be

$$r_\theta(s, a, z, c)$$

where r_θ maps a state s, action a, player latent $z \in \mathbb{R}^{d_z}$, and context c to a scalar reward. Our results (Theorems 1–3) hold for any parameterization of r_θ that induces a softmax policy $\pi \propto \exp(r)$; they do not require a specific functional form.

We define the policy to be the standard normalized reward exponential (softmax)

$$\pi_\theta(a \mid s, z, c) \triangleq \frac{\exp(r_\theta(s, a, z, c))}{\sum_{a' \in \mathcal{A}(s)} \exp(r_\theta(s, a', z, c))}$$

and our undiscounted trajectory return is

$$G(\tau, z, c) \triangleq \sum_{t=0}^{T} r_\theta(s_t, a_t, z, c)$$

Base objective The three independently-motivated objectives are

$$\mathcal{L}_{\text{NLL}} = -\mathbb{E}_{(\tau,z,c)\sim\mathcal{D}} \left[\frac{1}{|\tau|} \sum_t \log \pi_\theta(a_t \mid s_t, z, c) \right]$$

$$\mathcal{L}_{\text{CL}} = -\mathbb{E}_{i\in\mathcal{B}, j\in\mathcal{P}^+(i)} \log \frac{\exp(\frac{1}{\kappa} \tilde{z}_i^T \tilde{z}_j)}{\sum_{k\neq i} \exp(\frac{1}{\kappa} \tilde{z}_i^T \tilde{z}_k)}$$

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}[\log \sigma(\beta(\Delta R(\tau^+, z^+, c) - \Delta R(\tau^-, z^-, c)))]$$

$$\Delta R(\tau, z, c) = \sum_t (\log \pi_\theta(a_t \mid s_t, z, c) - \log \pi_{\text{ref}}(a_t \mid s_t, c))$$

where $z_p = g_\phi(\{\tau_{p,k}\}_{k=1}^K) \in \mathbb{R}^{d_z}$ is the latent the player encoder produces for player p from its K=5 reference games (Appendix C), $\mathcal{B}$ is the set of contrastive views in a batch, $\tilde{z}_i$ is an ℓ_2-normalized view — a latent for one player computed from a subset of that player's reference games — $\mathcal{P}^+(i)$ is the set of other views in $\mathcal{B}$ of view i's player, κ is the contrastive temperature, and π_{ref} is a reference policy in the style of DPO that is player-unconditioned — we eliminate this reference policy for simplification, recovering the direct Bradley-Terry likelihood over trajectory returns. In practice, each player's K=5 reference games are split into two fixed, disjoint groups, $\{1, 2\}$ and $\{3, 4, 5\}$, and the player encoder's final cross-game stage is re-run on each group separately to give two views. The two views of a player form a positive pair and the views of the other players in the batch serve as negatives; because the groups share no game, a positive pair shares no position.

$\mathcal{L}_{\text{NLL}}$ minimizes the NLL of observed player actions under the learned policy (behavior cloning). $\mathcal{L}_{\text{CL}}$ pushes views of the same player, built from disjoint sets of its reference games, together and views of different players apart, independent of context (contrastive identification). $\mathcal{L}_{\text{DPO}}$ teaches the model which game outcomes are preferred, providing quality-calibrated gradients to the player embeddings (preference alignment). Naively, one would optimize the weighted sum $\mathcal{L}_{\text{NLL}} + \lambda_{\text{CL}}\mathcal{L}_{\text{CL}} + \lambda_{\text{DPO}}\mathcal{L}_{\text{DPO}}$, treating the three as independent regularizers. We now show this is unnecessary: these objectives share sufficient structure that two of the three terms can be simplified or absorbed.

Reduction I: eliminating the reference policy While our preference objective was initially motivated by DPO, we note that the reference policy doubles memory and compute and we can simplify the objective by framing it directly as the Bradley-Terry (Bradley & Terry, 1952) likelihood. Given a preference pair (τ^+, τ^-, c), the Bradley-Terry model says the probability of τ^+ being preferred over τ^- is $\sigma(\beta(G(\tau^+, z^+, c) - G(\tau^-, z^-, c)))$. The negative log-likelihood of the observed preference ordering is then

$$\mathcal{L}_{\text{prefer}} = -\mathbb{E}_{(\tau^+,\tau^-,c)\sim\mathcal{D}_{\text{pref}}} \log \sigma(\beta(G(\tau^+, z^+, c) - G(\tau^-, z^-, c)))$$

Our preference term $\mathcal{L}_{\text{prefer}}$ coincides with the preference loss of Contrastive Preference Optimization (Xu et al., 2024), which arrives at the same form via an upper-bound argument on DPO; we adopt it directly as a likelihood. The CPO upper bound can be vacuous in chess, where blunders produce arbitrarily sharp distributions that a reference policy would never generate.

Reduction II: aligning behavior cloning and contrastive learning We now show that the softmax policy parameterization makes the explicit contrastive loss $\mathcal{L}_{\text{CL}}$ redundant at the optimum because $\mathcal{L}_{\text{NLL}}$ already performs contrastive learning over the players:

Theorem 1 (Alignment of behavior cloning and contrastive learning at the optima)**.** *At the NLL optimum, for any reward parameterization $r_\theta(s, a, z, c)$ inducing a softmax policy $\pi \propto \exp(r)$ and any players p, p' and any (s, a, c),*

$$r^*(s, a, z_p, c) - r^*(s, a, z_{p'}, c)$$
$$= \underbrace{\log \frac{p_{data}(a \mid s, z_p, c)}{p_{data}(a \mid s, z_{p'}, c)}}_{\text{InfoNCE optimal critic}} + \underbrace{\log Z^*(s, z_p, c) - \log Z^*(s, z_{p'}, c)}_{\perp\!\!\!\perp a}$$

where the difference in log-partitions is action-independent and reflects the differences in policy entropy of the players' true policies.

This alignment applies to optima and not optimization dynamics: the NLL gradient for player separation is indirect, while InfoNCE provides direct gradient on the player latents. In practice, removing $\mathcal{L}_{\text{CL}}$ entirely causes the player encoder to lose some discriminative structure. Theorem 1's practical value is that it justifies *decaying* $\mathcal{L}_{\text{CL}}$ (Theorem 2) rather than keeping it at fixed weight: because the two losses share the same discriminative signal at the optimum, $\mathcal{L}_{\text{CL}}$ accelerates convergence without introducing bias. We retain $\mathcal{L}_{\text{CL}}$ as an optimization aid with an adaptive coefficient:

Theorem 2 (Convergence guarantees for adaptive contrastive decay)**.** *Let $\varepsilon_{NLL}^{(t)} \triangleq \mathcal{L}_{NLL}^{(t)} - \mathcal{L}_{NLL}^*$ and $\varepsilon_{CL}^{(t)} \triangleq \mathcal{L}_{CL}^{(t)} - \mathcal{L}_{CL}^*$ denote the suboptimality gaps in the behavior cloning and contrastive losses at step t with the normalized gaps $\overline{\varepsilon}_{NLL}^{(t)} \triangleq \frac{\varepsilon_{NLL}^{(t)}}{\varepsilon_{NLL}^{(0)}}$, $\overline{\varepsilon}_{CL}^{(t)} \triangleq \frac{\varepsilon_{CL}^{(t)}}{\varepsilon_{CL}^{(0)}}$. Then for some initial contrastive weight $\lambda_0 > 0$ and the adaptive contrastive coefficient*

$$\lambda_{CL}^{(t)} \triangleq \lambda_0 \overline{\varepsilon}_{CL}^{(t)} \min\left(1, \frac{\overline{\varepsilon}_{CL}^{(t)}}{\overline{\varepsilon}_{NLL}^{(t)}}\right),$$

the unified adaptive loss

$$\mathcal{L}^{(t)}(\omega) \triangleq \mathcal{L}_{NLL} + \mathcal{L}_{prefer} + \lambda_{CL}^{(t)} \mathcal{L}_{CL}$$

converges under smoothness constraints to $\mathcal{L}_{NLL} + \mathcal{L}_{prefer}$.

Concretely, if both losses have finite global minima with finite initial gaps $\varepsilon^{(0)}$, $\mathcal{L}_{CL}$ is L-smooth near ω^*, and the suboptimality gaps vanish, then $\lambda_{CL}^{(t)} \to 0$, $\mathcal{L}^{(t)} \to \mathcal{L}_{NLL} + \mathcal{L}_{prefer}$, and $\omega^{(t)}$ converges to a local minimizer of $\mathcal{L}_{NLL} + \mathcal{L}_{prefer}$. Full conditions and proof are in Appendix A.

Theorem 2 provides a principled justification for the adaptive schedule rather than a finite-capacity guarantee. This L-smoothness assumption is standard for Euclidean-norm InfoNCE. The simultaneous convergence assumption is the strongest requirement; it holds empirically in our setting (both losses decrease monotonically) but could fail under insufficient model capacity or in other settings. Theorem 2's practical value is in identifying why our choice of schedule is well-motivated (superlinear decay, minimizer-set preservation) and under what conditions the contrastive auxiliary vanishes without distorting the asymptotic objective. We note that other schedules (e.g., linear warmup) might also avoid collapse; the adaptive schedule's advantage is automatic calibration of the relative magnitude between the contrastive and NLL losses. A detailed discussion is in Appendix D.

The unified loss The two reductions yield a single objective. The policy head $\pi_\theta(a \mid s, z, c)$ is trained by $\mathcal{L}_{\text{NLL}}$ for behavioral fidelity. A separate preference head $h_\xi : \mathbb{R}^{d_z} \to \mathbb{R}$ maps the player latent z to a scalar quality score and is trained by $\mathcal{L}_{\text{prefer}}$ for quality calibration. The preference head shares no parameters with the policy head, so $\mathcal{L}_{\text{prefer}}$ cannot directly alter move predictions. However, both heads share the player encoder output $\boldsymbol{z}_p$: $\mathcal{L}_{\text{prefer}}$ gradient flows through h_ξ to $\boldsymbol{z}_p$ and into the player encoder, actively shaping the latent space so that player quality is encoded in $\boldsymbol{z}_p$'s geometry. The unified loss is

$$\mathcal{L}^{(t)}(\omega) = \mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}} + \lambda_{\text{CL}}^{(t)} \mathcal{L}_{\text{CL}}$$

where $\mathcal{L}_{\text{prefer}} = -\mathbb{E}_{(\tau^+, \tau^-)}[\log \sigma(\beta(h_\xi(\boldsymbol{z}^+) - h_\xi(\boldsymbol{z}^-)))]$ and $\lambda_{\text{CL}}^{(t)} \to 0$ ensures the loss converges to $\mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}}$. All three losses send gradient to the player encoder through $\boldsymbol{z}_p$.

Remark. Theorems 1–3 hold for any reward parameterization $r(s, a, z, c)$ inducing $\pi \propto \exp(r)$; they do not require a specific functional form. Theorem 1 justifies coupling all three losses at the player encoder: the gradient signals reinforce rather than conflict.

Interpretation as goal-conditioned RL The unified loss admits a goal-conditioned RL (Kaelbling, 1993; Schaul et al., 2015) interpretation: the goal is to play like player z, and the reward measures style match.

Theorem 3 (Goal-conditioned RL equivalence)**.** *Under the assumptions of Theorem 2, $\mathcal{L}^{(t)}$ converges to $\mathcal{L}_{NLL} + \mathcal{L}_{prefer}$. The behavior cloning term decomposes via the log-partition identity as*

$$\mathcal{L}_{NLL} = \underbrace{-\mathbb{E}_{data}[G(\tau, z, c)]}_{\text{data return}} + \underbrace{\mathbb{E}_{data}\Big[\sum_t V_\theta(s_t, z, c)\Big]}_{\text{soft value}}$$

$$V_\theta(s, z, c) \triangleq \log Z_\theta(s, z, c) = \max_q \big\{\mathbb{E}_q[r_\theta(s, \cdot, z, c)] + \mathbb{H}[q]\big\}$$

where the maximum is attained at $q = \pi_\theta(\cdot \mid s, z, c)$. This is the maximum-entropy IRL objective (Ziebart et al., 2008; Ziebart, 2010) for a goal-conditioned MDP: π_θ is the entropy-regularized optimal policy for r_θ at each state, and minimizing $\mathcal{L}_{NLL}$ fits r_θ so demonstrated actions attain the soft value. At the optimum, $r^*_\theta(s, a, z, c) = \log p_{data}(a \mid s, z, c) + h_1(s, z, c)$.

The three original objectives — behavior cloning, contrastive identification, and preference alignment — are thus three views of a single goal-conditioned RL problem.

Algorithm See Algorithm 1 in Appendix B for the full pseudocode.

5 Architecture

Our architecture builds on a 5M Maia-3 (Chessformer) backbone (Monroe et al., 2026) (an 8-layer transformer (Vaswani et al., 2017) with geometric attention bias and residual dimension 256), trained from scratch with its value head, ponder head and both ELO embedding tables removed, so no rating exists in our parameter set at training or test time. Maia-3 concatenates a conditioning vector onto the 96 board channels (8 history frames $\times$ 12 piece planes) before the stem, and that vector is natively $\mathrm{concat}(\text{self ELO}[128], \text{opponent ELO}[128])$; we keep the two-player structure and substitute learned latents, $\mathrm{concat}(\boldsymbol{z}_p, \boldsymbol{z}_{\text{opp}}) \in \mathbb{R}^{512}$, so that each square token is formed as

$$\boldsymbol{t}_s \leftarrow W_{\text{tok}}[\boldsymbol{b}_s \| \boldsymbol{z}_p \| \boldsymbol{z}_{\text{opp}}], \qquad \boldsymbol{b}_s \in \mathbb{R}^{96}, \quad W_{\text{tok}}^T \in \mathbb{R}^{608\times 256}$$

where $\boldsymbol{b}_s$ are the board planes at square s and both latents are broadcast across all 64 squares. This is the *only* point at which player information enters. Figure 1 presents the full architecture.

The player encoder g_ϕ produces $\boldsymbol{z}_p \in \mathbb{R}^{256}$ from $K{=}5$ reference games by attention-pooling squares to tokens, plies to games, and games to a player (described in Appendix C). A separate preference head $h_\xi : \mathbb{R}^{d_z} \to \mathbb{R}$ maps $\boldsymbol{z}_p$ to a scalar quality score, sharing no parameters with the policy head; $\mathcal{L}_{\text{prefer}}$ gradient flows through h_ξ into the encoder, calibrating the latent space for quality ordering. All three losses ($\mathcal{L}_{\text{NLL}}$, $\mathcal{L}_{\text{CL}}$, $\mathcal{L}_{\text{prefer}}$) reach the encoder through $\boldsymbol{z}_p$. ELO is deliberately excluded from the context so that skill must be recovered from gameplay alone. Full architectural and training details are in Appendix D.

6 Experiments

We evaluate UMARM along four dimensions: player-conditioned move prediction accuracy, skill recovery from the learned latent space, zero-shot move prediction without reference games (across all plies), and match outcome prediction between unrelated opponents. All experiments use the same trained model and the same held-out players; no task-specific fine-tuning is performed.

6.1 Dataset

Training and evaluation data are drawn from the Lichess open database (Lichess, 2026). To compare with Maia 3 (Monroe et al., 2026) we use only blitz games from the same window, January 2023 through July 2025, and applied the same game-level filtering criteria as Maia (McIlroy-Young et al., 2020) and Maia 2 (Tang et al., 2024b) (rated games, first 10 plies excluded, positions with <30s remaining excluded). We used August 2025 with the same filtering for evaluation.

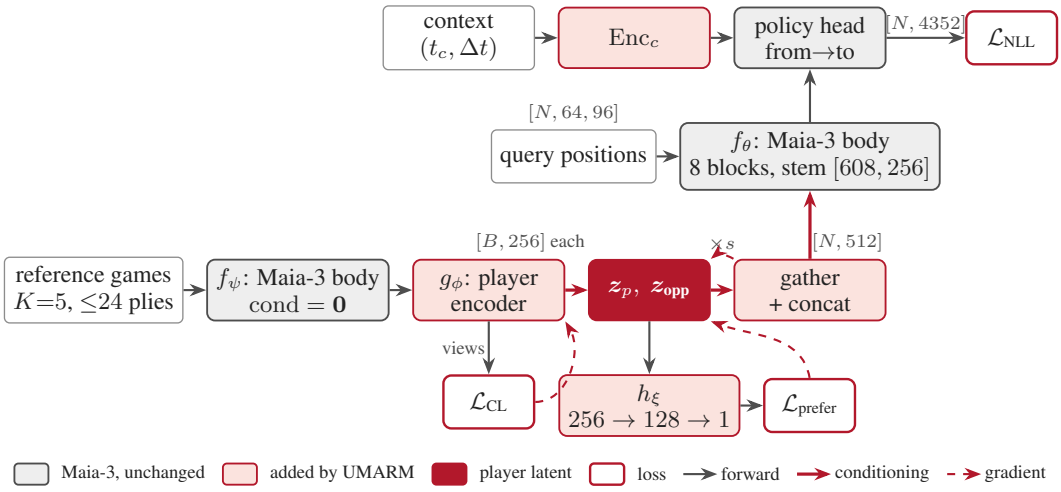


Figure 1: **UMARM architecture.** Gray components are Maia-3's, unchanged; red components are added by UMARM. g_ϕ runs twice on shared weights, on the player's reference games and on the opponent's, and both latents carry gradient. Gathered per query position and concatenated to $[N, 512]$, they replace Maia-3's $\text{concat}(\text{self ELO}, \text{opponent ELO})$ slot, widening the stem to $[608, 256]$; this is the only point at which player information enters. The reference body f_ψ runs at $\text{cond} = \mathbf{0}$ and is player-blind. N = query positions, B = players. Appendix C details g_ϕ.

The training pool contains ∼4.3M unique players and training runs for 620,000 steps at 48 players per batch, 3 games per player, at 15 subsampled positions per game, subsampled to 512 query slots per step yielding ∼254M loss-contributing positions — slightly less than 50% of Maia 3's sample budget (Appendix D). The evaluation set contains ∼4.15M positions from 5,000 held-out players spanning ELO ratings from below 1000 to above 2500. No ELO or player identity information is provided to the model during training or inference.

We evaluate UMARM, Maia 3, and Maia 2 on the same evaluation set for a fair comparison; we also include the Maia4All and Maia on their own eval sets for contextual comparison, with the Maia 3 and Maia 2 comparisons as the primary benchmarks.

6.2 Move Prediction Accuracy

We evaluate UMARM top-1 move-matching accuracy on the validation set under two situations: **Normal** (correct z_p from the player's own K=5 reference games) and **Shuffled** (z_p from a random different player). As the primary comparison, we evaluate two (5M, whose architecture we use and is our direct comparison, and 23M, which is a parameter-matched comparison) released Maia-3 checkpoints (Monroe et al., 2026) and Maia-2 (Tang et al., 2024b) on our eval. We also include the Maia (McIlroy-Young et al., 2020) and Maia4All (Tang et al., 2026) results on their own dataset for context.

UMARM (player encoder plus 5M Maia-3 as backbone) beats the base Maia-3 5M by $+1.30\%$ on average and the "parameter-matched" 23M by $+0.22\%$. While the two are technically parameter-matched, the 17M parameters held by the player encoder yield two 256-dimensional z_p, which are used as replacement for ELO in UMARM's 5M-parameter backbone; consequentially, we compare UMARM against both a 5M and 23M Maia-3 checkpoint for thoroughness but believe that the 5M model is a more relevant comparison.

6.3 Skill Recovery from Learned Latents

We test whether skill is encoded by z_p's geometry by fitting probes from z_p to ELO with 5-fold cross-validation over held-out players; no ELO is provided at training or inference. The probe results show that there is significant correlation between the learned z_p and ELO.

Table 1: **Move prediction.** Top-1 move prediction accuracy (%). The published baselines are evaluated on their own eval sets; UMARM and the other numbers are on our eval. Brackets follow Maia-2's grouping. Bold marks the best result in each row among models evaluated on our test set; UMARM is shaded.

	UMARM (5M+17M)		Maia-3		Maia-2	Published†			
Bracket	Normal	Shuffled	5M	23M	23M	Maia	Maia-2	M4A$_S$	M4A$_P$
Skilled	**54.95**	52.33	53.44	54.35	51.72	51.5	51.7	52.3	52.6
Advanced	**58.54**	55.61	57.28	58.10	52.91	52.6	54.2	53.8	54.1
Master	59.60	54.91	58.69	**60.40**	54.00	53.2	53.9	54.8	55.5
Avg	**56.94**	53.74	55.64	56.72	52.83	52.4	53.3	53.4	53.8

†Published figures on the authors' own test sets. M4A = Maia4All (Tang et al., 2026); S = Strength, P = Prototype. M4A uses 20 games/player with per-player fine-tuning; UMARM uses 5 games/player and no fine-tuning.

Table 2: **Skill recovery.** ELO prediction from the player latent $\boldsymbol{z}_p$ (5-fold CV, $n = 5{,}000$ held-out players). No ELO information is provided at training or inference.

Probe	R^2	MAE
Chance (predict mean)	0.00	414
Nonlinear (MLP)	0.77 ± 0.02	182 ± 6
Linear (Ridge)	$\mathbf{0.79 \pm 0.01}$	$\mathbf{172 \pm 5}$

6.4 Zero-shot move prediction

UMARM enables zero-shot inference: predicting moves for unseen players by constructing $\boldsymbol{z}_p$ incrementally from the moves observed so far within a single game, without any reference games. At position t, we encode positions $0, \ldots, t-1$ into a player latent and predict move t. Accuracy climbs monotonically with context, from 52.1% at 2–5 observed moves to 62.9% at 41+, surpassing the few-shot ($K = 5$ reference games) baseline of 56.9% as sufficient in-game context accumulates.

Table 3: **Zero-shot move prediction.** Top-1 accuracy (%) by number of moves observed in the current game.

	Moves seen					
	2–5	6–10	11–20	21–40	41+	Few-shot (K=5)
Zero-shot	52.1	55.6	56.6	59.1	**62.9**	56.9

6.5 Match outcome prediction

The preference head enables outcome prediction between arbitrary player pairs: given latents $\boldsymbol{z}_A, \boldsymbol{z}_B$, we predict $P(A \text{ wins}) = \sigma(h_\xi(\boldsymbol{z}_A) - h_\xi(\boldsymbol{z}_B))$, against an ELO baseline (Elo, 1978) fit by MLE.

Table 4: **Match outcome prediction.** Accuracy on validation games, averaged **per game**; draws are not counted.

Method	Accuracy (%)
ELO baseline (logistic)	56.1
UMARM ($h_\xi(\boldsymbol{z}_p)$)	**61.4**

7 Discussion & Limitations

UMARM reaches 56.94% top-1 on held-out players, beating both the 5M and 23M Maia-3 checkpoints as well as the Maia-2 checkpoint (23M) while having less than 23M effective parameters.

Critically, UMARM requires only 5 reference games per player with no fine-tuning: UMARM learns a shared encoding function that maps any player's games to a common strategy latent space.

The zero-shot results (Table 3) are particularly noteworthy. Averaged across all positions, zero-shot accuracy is 56.1%, slightly lagging behind the few-shot $K = 5$ baseline and 23M Maia-3 but without any ELO or other provided skill rating. Accuracy improves monotonically with context all the way to 62.9% after 41+ moves. The fact that zero-shot accuracy surpasses the few-shot baseline at 21+ moves suggests that in-game context can be more informative than cross-game reference trajectories played under different conditions.

The ELO recovery experiment ($R^2 = 0.79$, linear probe) confirms that the latent space organizes by skill without ELO supervision, and the fact that a linear probe outperforms a nonlinear MLP suggests that skill information is encoded in a geometrically clean, linear subspace of $\boldsymbol{z}_p$. This linear structure is consistent with the theoretical prediction that the contrastive and behavioral losses organize the latent space around the density ratio $\log(p(a \mid s, z, c)/p(a \mid s, c))$ (Theorem 1), which varies smoothly with skill.

We do not compare against MHR-Maia (Omi et al., 2025), which achieves $\sim 59\%$ per-player accuracy using per-player LoRA routing vectors trained on a median of 3,479 games per player. MHR-Maia's approach is fundamentally different from UMARM's: it requires training per-player parameters for each of 47,000 players, precluding inference on any player not seen during training. UMARM, by contrast, performs truly zero-shot inference on unseen players from as few as 2 moves of in-game context. The two approaches occupy complementary points in the accuracy–generalization tradeoff: MHR-Maia maximizes per-player fidelity at the cost of requiring extensive per-player data, while UMARM maximizes generalization at the cost of per-player accuracy.

Several limitations suggest directions for future work. We did not experiment with other potentially stronger conditional mechanisms; deeper player-conditioning mechanisms (e.g., per-layer modulation) may increase reliance on $\boldsymbol{z}_p$. Also, extending UMARM to other multi-agent domains beyond chess would test the generality of the theoretical framework.

AI USE STATEMENT

In this work, we used generative AI tools for assistance with implementing and debugging training code. We have not used generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, or interpret results. Additionally, we used generative AI tools to summarize or analyze existing literature, edit a research paper to improve readability, and format references. All AI-assisted code was manually reviewed, tested, and validated by the authors. All AI-assisted text was rewritten and verified by the authors, who take full responsibility for the correctness of all claims, proofs, and experimental results in this work.

ETHICS STATEMENT

The ability to model individual playing styles has positive applications in personalized coaching and training tools. However, improved fidelity of human-like play could lower the barrier to cheating in online chess by enabling bots that mimic a specific player's style. We note that the same latent representations enabling style imitation could also serve as features for improved cheating detection. All training and evaluation data come from the Lichess open database (Lichess, 2026), released under the Creative Commons CC0 license. The data contain no personally identifiable information beyond pseudonymous usernames, which we use only to group games by player and to form player-disjoint splits; usernames are never an input to the model.

REPRODUCIBILITY STATEMENT

All theoretical results (Theorems 1—3, Lemma 1) are stated with explicit assumptions and proved in full in Appendix A. The complete training algorithm is given in Appendix B. All architectural details, hyperparameters, optimization schedules, hardware specifications, and training times are reported in

Appendix D. Random seeds are fixed for all stochastic components. Training and evaluation data are drawn from Lichess, a publicly available platform; we specify the exact temporal window (a 31-month span of rated blitz games, January 2023 to July 2025, matching Maia 3's), the player-centric sampling and the filtering criteria in Section 6.1, and the held-out player split and evaluation slice, to enable reproduction. We train from scratch; nothing is pretrained or warm-started.

REFERENCES

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Rémi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 4447–4455. PMLR, 2024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

Amir Beck. *First-Order Methods in Optimization*, volume 25 of *MOS-SIAM Series on Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017. ISBN 978-1-61197-498-0. doi: 10.1137/1.9781611974997.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432, 2013.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs. `http://github.com/jax-ml/jax`, 2018. Version 0.3.13.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, December 1952. doi: 10.1093/biomet/39.3-4.324.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097. Curran Associates, Inc., 2021.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. arXiv preprint arXiv:1604.06174, 2016.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem. `http://github.com/google-deepmind`, 2020.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Publishing, New York, 1978.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35603–35620. Curran Associates, Inc., 2022. doi: 10.52202/068431-2580.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). arXiv preprint arXiv:1606.08415, 2016.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and João Carreira. Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 2021.

Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 1094–1099, 1993.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pp. 9–50. Springer, 1998. doi: 10.1007/3-540-49430-8_2.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 2019.

Lichess. lichess.org open database. `https://database.lichess.org`, 2026. Accessed: 2026-09-26.

Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman AI with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1677–1687. ACM, 2020. doi: 10.1145/3394486.3403219.

Reid McIlroy-Young, Yu Wang, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Detecting individual decision-making style: Exploring behavioral stylometry in chess. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24482–24497. Curran Associates, Inc., 2021.

Reid McIlroy-Young, Russell Wang, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Learning models of individual behavior in chess. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1253–1263. ACM, 2022. doi: 10.1145/3534678.3539367.

Diganta Misra. Mish: A self regularized non-monotonic activation function. In *Proceedings of the 31st British Machine Vision Conference (BMVC)*, 2020.

Daniel Monroe and Philip A. Chalmers. Mastering chess with a transformer model. arXiv preprint arXiv:2409.12272, 2024.

Daniel Monroe, George Eilender, Philip Chalmers, Zhenwei Tang, and Ashton Anderson. Chessformer: A unified architecture for chess modeling. In *International Conference on Learning Representations*, 2026. URL `https://proceedings.iclr.cc/paper_files/paper/2026/hash/3d167db04a90885ad5208fe8b273668b-Abstract-Conference.html`.

Nabil Omi, Lucas Caccia, Anurag Sarkar, Jordan T. Ash, and Siddhartha Sen. Generative modeling of individual behavior at scale. arXiv preprint arXiv:2502.14998, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022. doi: 10.52202/068431-2011.

Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991. doi: 10.1162/neco.1991.3.1.88.

Ben Poole, Sherjil Ozair, Aaron van den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5171–5180. PMLR, 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023. doi: 10.52202/075280-2338.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. arXiv preprint arXiv:1710.05941, 2017.

Marvin Ritter, Ihor Indyk, Aayush Singh, Andrew Audibert, Anoosha Seelam, Camelia Hanes, Eric Lau, Jacek Olesiak, Jiyang Kang, and Xihui Wu. Grain - feeding JAX models. `http://github.com/google/grain`, 2023. Version 0.2.12.

Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, Cannada A. Lewis, Joel Veness, and Tim Genewein. Amortized planning with large-scale transformers: A case study on chess. In *Advances in Neural Information Processing Systems*, volume 37, pp. 65765–65790, 2024. doi: 10.52202/079017-2102.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320. PMLR, 2015.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, 2020. doi: 10.1038/s41586-020-03051-4.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

Yunhao Tang, Zhaohan Daniel Guo, Zeyu Zheng, Daniele Calandriello, Rémi Munos, Mark Rowland, Pierre Harvey Richemond, Michal Valko, Bernardo Ávila Pires, and Bilal Piot. Generalized preference optimization: A unified approach to offline alignment. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47725–47742. PMLR, 2024a.

Zhenwei Tang, Difan Jiao, Reid McIlroy-Young, Jon Kleinberg, Siddhartha Sen, and Ashton Anderson. Maia-2: A unified model for human-AI alignment in chess. In *Advances in Neural Information Processing Systems*, volume 37, pp. 20919–20944, 2024b. doi: 10.52202/079017-0659.

Zhenwei Tang, Difan Jiao, Eric Xue, Reid McIlroy-Young, Jon Kleinberg, Siddhartha Sen, and Ashton Anderson. Learning to imitate with less: Efficient individual behavior modeling in chess. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL `https://openreview.net/forum?id=iw4kjcw319`.

The LCZero Authors. Leela Chess Zero. https://lczero.org, 2018. Open-source neural network chess engine, https://github.com/LeelaChessZero/lc0. Accessed: 2026-09-25.

The Stockfish developers. Stockfish. https://stockfishchess.org, 2008. Open-source UCI chess engine, https://github.com/official-stockfish/Stockfish. Accessed: 2026-09-25.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 55204–55224. PMLR, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Yiming Zhang, Athul Paul Jacob, Vivian Lai, Daniel Fried, and Daphne Ippolito. Human-aligned chess with a bit of search. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=bc2H72hGxB.

Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, December 2010.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. arXiv preprint arXiv:1909.08593, 2019.

A PROOFS

Lemma 1 (Optimal reward as a density ratio)**.** *At the NLL optimum,* $r^*(s,a,z,c) = \log p_{data}(a \mid s,z,c) + h_1(s,z,c)$ *with* $h_1 = \log Z^*(s,z,c)$ *action-independent; equivalently* $r^* = \log \frac{p_{data}(a|s,z,c)}{p_{data}(a|s,c)} + h_1(s,z,c) + h_2(s,a,c)$*,* $h_2 = \log p_{data}(a \mid s,c)$*.*

Proof of Lemma 1: optimal reward as density ratio. We prove that the reward is optimized by the ratio of densities of the player's true policy and the player-unconditioned policy. Recall that the behavior cloning loss is $\mathcal{L}_{\text{NLL}}(\theta,\phi) = -\mathbb{E}_{(s,a,z,c)\sim p_{\text{data}}}[\log \pi_\theta(a \mid s,z,c)]$ with expectation over the data-generating distribution. This is the cross-entropy $H(p_{\text{data}},\pi_\theta) = H(p_{\text{data}}) + D_{\text{KL}}(p_{\text{data}}\|\pi_\theta)$ between the true conditional distribution and the model, which is minimized when the model matches the data distribution and becomes the data entropy $\mathbb{H}(p_{\text{data}})$.

The optimal reward under the data distribution policy is

$$\pi_\theta(a \mid s,z,c) = \frac{\exp(r_\theta(s,a,z,c))}{\sum_{a'}\exp(r_\theta(s,a',z,c))} \to r_\theta(s,a,z,c) = \log \pi_\theta(a \mid s,z,c) + \log \underbrace{Z_\theta(s,z,c)}_{\sum_{a'}\exp(r_\theta(s,a',z,c))}$$

$$r^*_\theta(s,a,z,c) = \log p_{\text{data}}(a \mid s,z,c) + \underbrace{\log Z^*_\theta(s,z,c)}_{\triangleq h_1(s,z,c)}$$

where $Z_\theta(s, z, c)$ is the partially-marginalized partition function and the optimal reward r^* and partition function Z^*_θ are under the optimal policy $\pi^*_\theta = p_{\text{data}}$. By Bayes' rule, $\frac{p_{\text{data}}(a|s,z,c)}{p_{\text{data}}(a|s,c)} = \frac{p_{\text{data}}(z|s,a,c)}{p_{\text{data}}(z|s,c)}$ (both ratios equal $\frac{p_{\text{data}}(a,z|s,c)}{p_{\text{data}}(a|s,c)\,p_{\text{data}}(z|s,c)}$), so

$$\log p_{\text{data}}(a \mid s, z, c) = \log \frac{p_{\text{data}}(a \mid s, z, c)}{p_{\text{data}}(a \mid s, c)} + \log p_{\text{data}}(a \mid s, c) = \log \frac{p_{\text{data}}(z \mid s, a, c)}{p_{\text{data}}(z \mid s, c)} + \log p_{\text{data}}(a \mid s, c)$$

Substituting yields that the optimal reward depends on the ratio of the player-conditioned and unconditioned data densities:

$$r^*_\theta(s, a, z, c) = \log \frac{p_{\text{data}}(a \mid s, z, c)}{p_{\text{data}}(a \mid s, c)} + \underbrace{\log Z^*(s, z, c)}_{h_1(s,z,c)} + \underbrace{\log p_{\text{data}}(a \mid s, c)}_{h_2(s,a,c)}$$

Neither residual term affects the policy's player-discriminative structure: $h_1(s, z, c) = \log Z^*(s, z, c)$ depends on z but not a, so it contributes a per-player, per-position constant that is absorbed by the softmax normalization over actions; $h_2(s, a, c) = \log p_{\text{data}}(a \mid s, c)$ depends on a but not z, so it shifts all players' rewards identically and is cancelled by any player-contrastive comparison. The density ratio $\log(p_{\text{data}}(a \mid s, z, c)/p_{\text{data}}(a \mid s, c))$ is therefore the only term that carries both action-specific and player-specific information. □

Proof of Theorem 1: implicit contrastive learning. We show that at the NLL optimum, the per-move reward differences between players recover the same discriminative signal that the InfoNCE-optimal critic uses for player identification. This alignment means that a model achieving the NLL optimum has already learned the discriminative structure that $\mathcal{L}_{\text{CL}}$ seeks to enforce, making $\mathcal{L}_{\text{CL}}$ redundant *at convergence*: we do not claim that two losses are equivalent in general, and in fact we see $\mathcal{L}_{\text{CL}}$ as a training aid to provide extra convergence pressure and richer, diverse latents.

From Lemma 1, the optimal reward difference between two players p, p' under the same state-action-context is

$$r^*_\theta(s, a, z_p, c) - r^*_\theta(s, a, z_{p'}, c)$$
$$= \log \frac{p_{\text{data}}(a \mid s, z_p, c)}{p_{\text{data}}(a \mid s, z_{p'}, c)} + \underbrace{\log Z^*(s, z_p, c) - \log Z^*(s, z_{p'}, c)}_{\triangleq \delta_Z(s,c,z_p,z_{p'})}$$

where δ_Z is action-independent, reflecting purely the difference in policy entropy of both players.

From van den Oord et al. (2018) (see also Poole et al., 2019), the optimal InfoNCE critic satisfies

$$f^*(\tilde{z}_i, \tilde{z}_j) = \log \frac{p(\tilde{z}_j \mid p_j = p_i)}{p(\tilde{z}_j)} + c(i)$$

where f^* is the similarity function that maximizes the InfoNCE objective. Because $\tilde{z}_j = g_\phi(\tau_j)$ is deterministic, the density ratio equals the trajectory likelihood ratio

$$\log \frac{p(\tau_j \mid p_i)}{p(\tau_j)} = \sum_t \log \frac{p(a_{j,t} \mid s_{j,t}, z_{p_i}, c_j)}{p(a_{j,t} \mid s_{j,t}, c_j)}$$

At the NLL optimum, Lemma 1 gives $r^*(s, a, z, c) = \log(p_{\text{data}}(a \mid s, z, c)/p_{\text{data}}(a \mid s, c)) + h_1(s, z, c) + h_2(s, a, c)$, so the trajectory return under the optimal reward is

$$G^*(\tau_j, z_{p_i}, c) = \sum_t r^*(s_{j,t}, a_{j,t}, z_{p_i}, c)$$

$$= \sum_t \log \frac{p_{\text{data}}(a_{j,t} \mid s_{j,t}, z_{p_i}, c_j)}{p_{\text{data}}(a_{j,t} \mid s_{j,t}, c_j)} + \sum_t h_1(s_{j,t}, z_{p_i}, c_j) + \sum_t h_2(s_{j,t}, a_{j,t}, c_j)$$

The first sum is exactly the InfoNCE-optimal trajectory density ratio. The second sum $\sum_t h_1(s_{j,t}, z_{p_i}, c_j)$ depends on z_{p_i} but not on the actions $a_{j,t}$; while it varies across trajectories (through $s_{j,t}$), the InfoNCE softmax normalizes over the player index i for a fixed trajectory j, so this player-dependent, action-independent sum contributes a per-player offset that is absorbed by

the normalization. The third sum depends on the trajectory but not on the player identity p_i, so it contributes a constant across all player comparisons and is likewise absorbed.

Thus, the NLL-optimal trajectory return $G^*(\tau_j, z_{p_i}, c)$ recovers the InfoNCE-optimal trajectory-level critic up to additive terms that are absorbed by the InfoNCE normalization. The NLL and InfoNCE optima are aligned: both use the trajectory-level sum of per-move log-likelihood ratios as the discriminative signal for player identification.

We emphasize that this alignment is a statement about *optima*, not about optimization dynamics. At a suboptimal policy, the NLL gradient for player separation is indirect (flowing through the softmax and factored reward), while InfoNCE provides a direct gradient on the player embeddings. This motivates retaining $\mathcal{L}_{\text{CL}}$ as an optimization aid during training while decaying it as the NLL drives the embedding toward the aligned optimum (Theorem 2). □

Proof of convergence guarantees for adaptive contrastive decay. We prove that for $\lambda^{(t)} = \lambda_0 \bar{\varepsilon}_{\text{CL}}^{(t)} \min\left(1, \frac{\bar{\varepsilon}_{\text{CL}}^{(t)}}{\bar{\varepsilon}_{\text{NLL}}^{(t)}}\right)$ under the three assumptions:

- $\mathcal{L}_{\text{NLL}}^*, \mathcal{L}_{\text{CL}}^* = \min_{\mathcal{L}_{\text{NLL}}, \mathcal{L}_{\text{CL}}}$ and $\varepsilon_{\text{NLL}}^{(0)}, \varepsilon_{\text{CL}}^{(0)} > 0$ are all finite
- There exists a constant $L \,\text{s.t.}\, \|\nabla\mathcal{L}_{\text{CL}}(\omega) - \nabla\mathcal{L}_{\text{CL}}(\omega')\| \leq L\|\omega - \omega'\|$ for all parameters ω, ω' in a neighborhood of the optimum — the contrastive loss gradient is L-Lipschitz, where $\omega = (\theta, \phi, \xi)$ denotes the joint parameter vector
- $\varepsilon_{\text{NLL}}^{(t)}, \varepsilon_{\text{CL}}^{(t)} \to 0, \bar{\varepsilon}_{\text{NLL}}^{(t)} > 0$ — both losses converge in the limit and the NLL loss difference ratio does not converge for finite t. We note that simultaneous convergence of both gaps is a strong assumption; it may fail under insufficient model capacity or when the two losses impose conflicting requirements on the shared parameters (though we observed during our experiments that both losses dropped monotonically over the player encoder and head),

we achieve the following three results with gradient flow analyses:

- The unified loss becomes $\mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}}$ in the limit.
- The contrastive loss gradient converges superlinearly with the suboptimality gap —

$$\|\lambda^{(t)}\nabla\mathcal{L}_{\text{CL}}^{(t)} \in \mathcal{O}\left((\bar{\varepsilon}_{\text{CL}}^{(t)})^{3/2}\right)$$

- $\text{argmin}_\omega\, \mathcal{L}^{(t\to\infty)} = \text{argmin}_\omega\, \mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}}$ — the parameter minimizers for the unified loss are not just a subset of but are equal to those of the NLL and preference loss sum without the contrastive loss: our contrastive weight decay schedule loses no parameter expressivity

We first observe that the asymptotic loss is $\mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}}$:

Proof.

$$0 \leq \min(1, \bar{\varepsilon}_{\text{CL}}^{(t)}/\bar{\varepsilon}_{\text{NLL}}^{(t)}) \leq 1 \implies 0 \leq \lambda^{(t)} \leq \lambda_0\bar{\varepsilon}_{\text{CL}}^{(t)} = \lambda\frac{\varepsilon_{\text{CL}}^{(t)}}{\varepsilon_{\text{CL}}^{(0)}}$$

From the suboptimality gap assumption, $\varepsilon_{\text{CL}}^{(t)} \to 0$ so $\bar{\varepsilon}_{\text{CL}}^{(t)} \to 0$ and by squeeze theorem $\lambda^{(t)} \to 0$. Thus, for any $\delta > 0$, there exists a $T \,\text{s.t.}\, \forall t > T, \lambda^{(t)}\mathcal{L}_{\text{CL}}^{(t)} < \delta$ and the asymptotic loss is $\mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}}$ and the contrastive term does not change the loss in the limit. □

We bound the contrastive gradient using the smoothness assumption on the contrastive loss:

Proof. By the descent lemma (Beck, 2017) on $\mathcal{L}_{\text{CL}}$, we bound the suboptimality of $\omega^{(t)}$ via a second-order Taylor expansion. Recall that $\mathcal{L}_{\text{CL}}^{(t)}$ has parameter $\omega^{(t)}$ and $\mathcal{L}_{\text{CL}}^{*}$ has parameter ω^{*}. Then,

$$\mathcal{L}_{\text{CL}}^{(t)} \leq \mathcal{L}_{\text{CL}}^{*} + \nabla \mathcal{L}_{\text{CL}}^{*\,T}(\omega^{(t)} - \omega^{*}) + \frac{L}{2}\|\omega^{(t)} - \omega^{*}\|^2$$

$$\rightarrow \mathcal{L}_{\text{CL}}^{(t)} - \mathcal{L}_{\text{CL}}^{*} \leq \frac{L}{2}\|\omega^{(t)} - \omega^{*}\|^2 \rightarrow \|\omega^{(t)} - \omega^{*}\| \leq \sqrt{\frac{2\varepsilon_{\text{CL}}^{(t)}}{L}}$$

so

$$\|\nabla \mathcal{L}_{\text{CL}}^{(t)}\| = \|\nabla \mathcal{L}_{\text{CL}}^{(t)} - \nabla \mathcal{L}_{\text{CL}}^{*}\| \leq L\|\omega^{(t)} - \omega^{*}\| \leq \sqrt{2L\varepsilon_{\text{CL}}^{(t)}}$$

This bound is tight: for $\mathcal{L}_{\text{CL}}(\omega) = \frac{L}{2}\|\omega\|^2$, we achieve the bound exactly.

Combining the gradient bound with the contrastive decay yields

$$\|\lambda^{(t)} \nabla \mathcal{L}_{\text{CL}}^{(t)}\| \leq \lambda_0 \frac{\varepsilon_{\text{CL}}^{(t)}}{\varepsilon_{\text{CL}}^{(0)}} \sqrt{2L\varepsilon_{\text{CL}}^{(t)}} = \frac{\lambda_0 \sqrt{2L}}{\varepsilon_{\text{CL}}^{(0)}} \varepsilon_{\text{CL}}^{(t)\,3/2} \leq \lambda_0 \sqrt{2L\varepsilon_{\text{CL}}^{(0)}} \in \mathcal{O}(\varepsilon_{\text{CL}}^{(t)\,3/2})$$

which is the desired superlinear decay. □

We finally prove that the set of optimal parameters of $\mathcal{L}$ and $\mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}}$ are exactly the same:

Proof. Let $F(\omega) \triangleq \mathcal{L}_{\text{NLL}}(\omega) + \mathcal{L}_{\text{prefer}}(\omega), V(\omega) = F(\omega) - F(\omega^{*})$. Let also $\dot{\omega}(t)$ be a perturbed gradient flow with contrastive perturbation $p(t)$ and $\dot{\omega}$ be the associated unperturbed flow

$$\dot{\omega} = -\nabla F(\omega)$$

$$\dot{\omega}(t) = -\nabla F(\omega) - \underbrace{p(t)}_{\lambda^{(t)} \nabla \mathcal{L}_{\text{CL}}(\omega(t))}$$

We show that V is a strict Lyapunov function for the unperturbed flow by observing that along $\dot{\omega}$,

$$\dot{V}(\omega) = \nabla V(\omega) \cdot (-\nabla F(\omega)) = -\|\nabla F(\omega)\|^2 < 0 \text{ for } \omega \notin \Omega^{*}$$

Thus, ω^{*} is asymptotically stable.

Along the perturbed flow,

$$\dot{V}(\omega(t)) = \nabla V(\omega(t))^T \dot{\omega} = \nabla F(\omega(t))^T(-\nabla F(\omega(t)) - p(t)) = -\|\nabla F(\omega(t))\|^2 - \nabla F(\omega(t))^T p(t)$$

$$\overset{\text{C.S.}}{\leq} -\|\nabla F(\omega(t))\|^2 + \|\nabla F(\omega(t))\|\|p(t)\| = -\|\nabla F(\omega(t))\|(\|\nabla F(\omega(t))\| - \|p(t)\|)$$

$$= -\|\nabla F(\omega(t))\|(\|\nabla F(\omega(t))\| - \underbrace{\|\lambda^{(t)} \nabla \mathcal{L}_{\text{CL}}^{(t)}\|}_{\mathcal{O}(\varepsilon_{\text{CL}}^{(t)\,3/2})})$$

Because $\|p(t)\|$ decays superlinearly while $\|\nabla F(\omega(t))\|$ decays at most linearly near ω^{*}, there exists a T such that for all $t > T$,

$$-\|\nabla F(\omega(t))\| \underbrace{(\|\nabla F(\omega(t))\| - \|p(t)\|)}_{>0}$$

and

$$\dot{V}(\omega(t > T)) < 0$$

Because V is strictly decreasing along the perturbed flow for $\omega(t) \neq \omega^{*}$ and V is bounded below (by 0), $V(\omega(t))$ is bounded and eventually monotonically decreasing. Thus, it converges to some $V_{\infty} \geq 0$.

We show by contradiction that $V_{\infty} = 0$ and thus $\omega(t) \rightarrow \omega^{*}$. Assume $V_{\infty} > 0$ so that $\|\omega(t) - \omega^{*}\| \geq \delta > 0$ for all $t > T$. Then, in the compact sublevel set $\{V(\omega(t > T)) \leq V(\omega(T))\}$, the gradient $\|\nabla F(\omega)\|$ is bounded below by some positive constant c_{min} on $\{\omega : \|\omega - \omega^{*}\| \geq \delta\}$ because ∇F vanishes only at ω^{*}. However, since $p(t) \rightarrow 0$ we have for sufficiently large t that $\dot{V}(\omega(t)) \leq -c_{\text{min}}^2/2 < 0$ so $V(\omega(t)) \rightarrow -\infty$, contradicting $V \geq 0$. Thus $V_{\infty} = 0$, $V(\omega(t)) \rightarrow 0$, and $\omega(t) \rightarrow \omega^{*} \in \Omega^{*}$. □

□

Proof of GCRL equivalence. We show that $\mathcal{L}_{\text{NLL}}$ decomposes into the MaxEnt goal-conditioned RL objective and that the optimal reward recovers the player-conditioned data distribution.

The negative log-likelihood, dropping the per-trajectory normalization $1/|\tau|$ (which does not affect the optimum), is

$$\mathcal{L}_{\text{NLL}} = -\mathbb{E}_{(\tau,z,c)} \sum_t \log \pi_\theta(a_t \mid s_t, z, c) = -\mathbb{E}_{(\tau,z,c)} \sum_{t=0}^{T} \left(r_\theta(s_t, a_t, z, c) - \log Z_\theta(s_t, z, c)\right)$$

$$= -\underbrace{\mathbb{E}_{(\tau,z,c)}\left[\sum_{t=0}^{T} r_\theta(s_t, a_t, z, c)\right]}_{=\mathbb{E}[G(\tau,z,c)]} + \mathbb{E}_{(\tau,z,c)}\left[\sum_{t=0}^{T} \log Z_\theta(s_t, z, c)\right]$$

The log-partition decomposes for any $a \in \mathcal{A}$ as

$$\pi_\theta(a \mid s, z, c) = \exp(r_\theta(a))/Z_\theta \implies \log Z_\theta = r_\theta(a) - \log \pi_\theta(a)$$

Then,

$$\log Z_\theta(s, z, c) = \sum_{a \in \mathcal{A}} \pi_\theta(a)(r_\theta(a) - \log \pi_\theta(a)) = \mathbb{E}_{\pi_\theta}[r_\theta(\cdot \mid s, z, c)] + \mathbb{H}[\pi_\theta(\cdot \mid s, z, c)]$$

Thus,

$$\mathcal{L}_{\text{NLL}} = -\underbrace{\mathbb{E}_{\text{data}}\left[\sum_t r_\theta(s_t, a_t, z, c)\right]}_{\text{(i) data return}} + \underbrace{\mathbb{E}_{\text{data}}\left[\sum_t \mathbb{E}_{\pi_\theta}[r_\theta(s_t, \cdot, z, c)]\right]}_{\text{(ii) on-policy return}} + \underbrace{\mathbb{E}_{\text{data}}\left[\sum_t \mathbb{H}[\pi_\theta(\cdot \mid s_t, z, c)]\right]}_{\text{(iii) entropy bonus}}$$

Minimizing $\mathcal{L}_{\text{NLL}}$ increases the data's return relative to the soft value $\sum_t V_\theta(s_t, z, c)$. Since $V_\theta = \max_q \{\mathbb{E}_q[r_\theta] + \mathbb{H}[q]\}$ is attained by π_θ, this is the MaxEnt IRL objective (Ziebart et al., 2008; Ziebart, 2010), under which player z is modeled as the entropy-regularized optimal policy for r_θ in a goal-conditioned MDP whose goal is to play like player z.

Minimizing $\mathcal{L}_{\text{NLL}}$ jointly maximizes the expected return of the data-generating player z, minimizes the model's expected return (regularizing against off-distribution rewards), and maximizes the learned policy's entropy. This is precisely the MaxEnt RL decomposition (Ziebart, 2010) applied to a goal-conditioned MDP with goal to play like player z.

At the optimum, Lemma 1 gives

$$r^*_\theta(s, a, z, c) = \log p_{\text{data}}(a \mid s, z, c) + \log Z^*(s, z, c)$$

The optimal reward recovers the player-conditioned policy up to action-independent log-partition $\log Z^*(s, z, c)$, absorbed by the policy's softmax. Thus, $\pi^*_\theta(a \mid s, z, c) = p_{\text{data}}(a \mid s, z, c)$: the optimal policy exactly imitates the target player under the given context.

The preference constraint $\mathcal{L}_{\text{prefer}}$ operates through a separate preference head $h_\xi(\boldsymbol{z}_p)$ that maps the player latent to a scalar quality score. The preference head does not share any parameters with the policy head, so $\mathcal{L}_{\text{prefer}}$ cannot directly alter move-prediction weights. However, $\mathcal{L}_{\text{prefer}}$'s' gradient flows through h_ξ into $\boldsymbol{z}_p$ and the player encoder, which shapes the latent space so that player quality is encoded in $\boldsymbol{z}_p$'s geometry. The combined objective thus simultaneously learns to imitate each player's behavior (via $\mathcal{L}_{\text{NLL}}$), discriminate players (via $\mathcal{L}_{\text{CL}}$), and order players by quality (via $\mathcal{L}_{\text{prefer}}$) in a shared latent space. □

B ALGORITHM

Algorithm 1 is the training procedure as implemented: per step, one fused scalar objective $\mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}} + \lambda^{(t)}_{\text{CL}} \mathcal{L}_{\text{CL}}$, one gradient, and one Adam step per optimizer group. All three terms reach the player encoder ϕ.

Algorithm 1 UMARM training. One objective per step with all three losses reaching $\boldsymbol{z}_p$.

Require: Query data $\mathcal{D} = \{(\tau_i, p_i, o_i, \boldsymbol{c}_i, y_i)\}_{i=1}^{M}$ with result $y_i \in \{-1, 0, +1\}$ from p_i's view; per player a fixed reference set of K=5 games, disjoint from every game that player is scored on.
Require: β, λ_0, warmup W, views n_{pos}, temperature $\kappa_{\max} \to \kappa_{\min}$ over T_κ, NLL-to-$\boldsymbol{z}_p$ scale s, per-group Adam rates $\eta_\theta, \eta_\phi, \eta_\xi$, and fixed minima $\mathcal{L}^*_{\text{NLL}}$=1.0, $\mathcal{L}^*_{\text{CL}}$=1.43 with normalizers $\varepsilon_X^{(0)}$ pinned at step 0.
Ensure: $\omega = (\theta, \phi, \xi)$: body f_θ with policy head and Enc_c; encoder g_ϕ with reference body f_ψ; preference head h_ξ.

for $t = 0, 1, \ldots, T-1$ **do**
  Draw B players; query games disjoint from their own reference games. Preference pairs are the decisive games of the batch itself.
  $w^{(t)} \leftarrow \min(1, t/W)$; $\beta^{(t)} \leftarrow w^{(t)}\beta$; $s^{(t)} \leftarrow w^{(t)}s$; $\kappa^{(t)} \leftarrow \kappa_{\max} + (\kappa_{\min} - \kappa_{\max})\min(1, t/T_\kappa)$
  $\bar{\varepsilon}_X^{(t)} \leftarrow \big[(\mathcal{L}_X^{(t-1)} - \mathcal{L}_X^*)/\varepsilon_X^{(0)}\big]_0^1$; $\lambda_{\text{CL}}^{(t)} \leftarrow w^{(t)}\lambda_0\,\bar{\varepsilon}_{\text{CL}}^{(t)}\min\big(1, \bar{\varepsilon}_{\text{CL}}^{(t)}/\bar{\varepsilon}_{\text{NLL}}^{(t)}\big)$ ▷ Theorem 2
  $\boldsymbol{z}_p \leftarrow g_\phi(\{\tau_p^{(k)}\}_{k=1}^K)$ and $\boldsymbol{z}_{opp} \leftarrow g_\phi(\{\tau_o^{(k)}\}_{k=1}^K)$, from f_ψ features at cond=$\mathbf{0}$ ▷ shared weights; both carry gradient
  $\boldsymbol{u}_i \leftarrow \text{sg}(\boldsymbol{v}_i) + s^{(t)}(\boldsymbol{v}_i - \text{sg}(\boldsymbol{v}_i))$ for $\boldsymbol{v}_i = [\,\boldsymbol{z}_{p_i} \,\|\, \boldsymbol{z}_{opp_i}\,] \in \mathbb{R}^{512}$ ▷ forward exact, backward scaled
  $\boldsymbol{h}_i \leftarrow f_\theta\big([\,96 \text{ board} \,\|\, \boldsymbol{u}_i\,] \in \mathbb{R}^{608}\big)$; $\pi_\theta \leftarrow$ legal-masked softmax of the factored head on $\boldsymbol{h}_i + W_c\text{Enc}_c(\boldsymbol{c}_i)$
  $\mathcal{L}_{\text{NLL}} \leftarrow -\frac{1}{|\mathcal{B}|}\sum_i \frac{1}{|\tau_i|}\sum_t \log \pi_\theta(a_t \mid s_t, \boldsymbol{z}_{p_i}, \boldsymbol{z}_{opp_i}, \boldsymbol{c}_i)$ ▷ scored slots, ply > 10
  $\mathcal{L}_{\text{prefer}} \leftarrow -\frac{1}{|\mathcal{B}_{\text{pref}}|}\sum_j \log\sigma\big(\beta^{(t)} y_j [h_\xi(\boldsymbol{z}_{p_j}) - h_\xi(\boldsymbol{z}_{opp_j})]\big)$
  Split the K reference games into n_{pos} disjoint contiguous groups (for K=5, n_{pos}=2: games $\{1,2\}$ and $\{3,4,5\}$) and re-run only the cross-game aggregation per group to give ℓ_2-normalized views $\tilde{\boldsymbol{z}}_p^{(m)}$ ▷ no query game enters $\mathcal{L}_{\text{CL}}$
  $\mathcal{L}_{\text{CL}} \leftarrow -\frac{1}{|\mathcal{V}|}\sum_{(p,m)\in\mathcal{V}} \frac{1}{|\mathcal{P}^+(p,m)|}\sum_{(p,m')\in\mathcal{P}^+(p,m)} \log \frac{\exp(\tilde{\boldsymbol{z}}_p^{(m)\top}\tilde{\boldsymbol{z}}_p^{(m')}/\kappa^{(t)})}{\sum_{(q,n)\neq(p,m)}\exp(\tilde{\boldsymbol{z}}_p^{(m)\top}\tilde{\boldsymbol{z}}_q^{(n)}/\kappa^{(t)})}$ ▷ in-batch keys
  $\mathcal{L}^{(t)} \leftarrow \mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{prefer}} + \lambda_{\text{CL}}^{(t)}\mathcal{L}_{\text{CL}}$; $(\boldsymbol{g}_\theta, \boldsymbol{g}_\phi, \boldsymbol{g}_\xi) \leftarrow \nabla_{(\theta,\phi,\xi)}\mathcal{L}^{(t)}$
  $\theta \leftarrow \theta - \eta_\theta\text{Adam}(\boldsymbol{g}_\theta)$; $\phi \leftarrow \phi - \eta_\phi\text{Adam}(\boldsymbol{g}_\phi)$; $\xi \leftarrow \xi - \eta_\xi\text{Adam}(\boldsymbol{g}_\xi)$ ▷ no clip
end for
return (θ, ϕ, ξ).

C Player Encoder

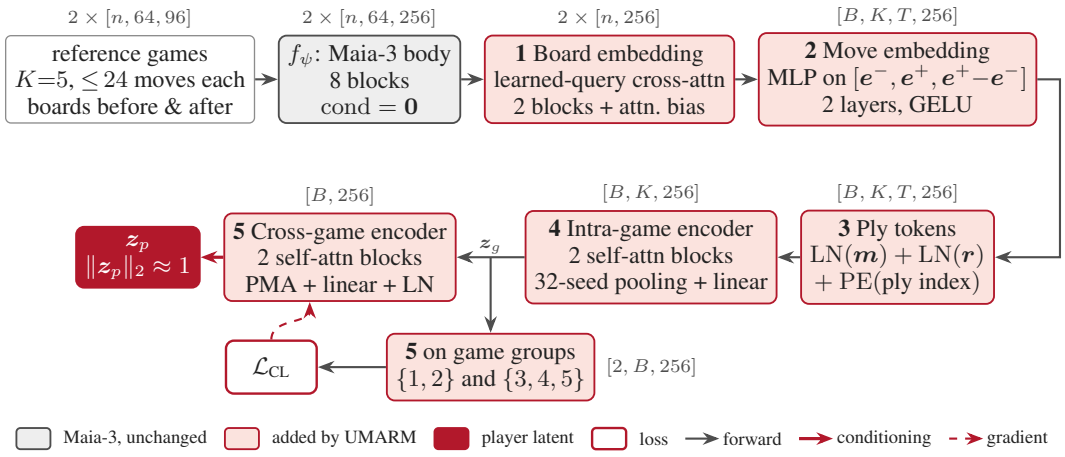


Figure 2: **Player encoder g_ϕ.** Gray components are Maia-3's, unchanged; red components are added by UMARM. Stages 1–2 embed each move from the boards before and after it, Stage 3 adds the move's ply index, Stage 4 pools a game's moves into $\boldsymbol{z}_g$, and Stage 5 pools a player's games into $\boldsymbol{z}_p$. $\mathcal{L}_{\text{CL}}$ re-runs only Stage 5, once on each of two disjoint groups of the player's reference games. Output shapes are shown above each block, with $n = B \times K \times T$. Per-stage parameter counts are in Table 5.

Input and reference body (Figure 2) Each player has a fixed set of $K{=}5$ reference games, disjoint from every game on which the player is scored so that $\boldsymbol{z}_p$ is a deterministic per player given the parameters. From each game we take up to $T{=}24$ of the player's own moves, drawn uniformly. Move t of game k contributes to the board before and after it, $\boldsymbol{b}^-_{k,t}, \boldsymbol{b}^+_{k,t} \in \mathbb{R}^{64\times 96}$, each encoded as in Maia-3 (8 history frames $\times$ 12 piece planes per square). Both boards pass through the reference body, a Maia-3 body with its own parameters run with a zero conditioning input:

$$\boldsymbol{F}^{\pm}_{k,t} = f_\psi\big(\boldsymbol{b}^{\pm}_{k,t};\ \mathrm{cond} = \mathbf{0}\big) \in \mathbb{R}^{64\times d}, \qquad d = d_z = 256$$

f_ψ is player-blind, matching the backbone f_θ except for the stem: its $[352, 256]$ kernel keeps a 256-row conditioning slot that always receives zeros whereas f_θ's $[608, 256]$ kernel receives $[\boldsymbol{z}_p \,\|\, \boldsymbol{z}_{\text{opp}}]$. It transforms the input from $[n, 64, 96] \to [n, 64, 256]$ for each of the two board stacks, with $n = B \times K \times T$ boards for B players.

Transformer blocks All attention in g_ϕ uses one post-norm transformer block (Vaswani et al., 2017) with LayerNorm (**?**) and GELU (**?**),

$$\mathrm{Blk}(\boldsymbol{U}, \boldsymbol{X}) = \mathrm{LN}\big(\boldsymbol{V} + \mathrm{FFN}(\boldsymbol{V})\big), \qquad \boldsymbol{V} = \mathrm{LN}\big(\boldsymbol{U} + \mathrm{MHA}(\boldsymbol{U}, \boldsymbol{X})\big),$$
$$\mathrm{FFN}(\boldsymbol{v}) = \boldsymbol{W}_2\,\mathrm{GELU}(\boldsymbol{W}_1\boldsymbol{v} + \boldsymbol{b}_1) + \boldsymbol{b}_2,$$

where $\mathrm{MHA}(\boldsymbol{U}, \boldsymbol{X})$ is multi-head attention with queries from sequence $\boldsymbol{U} \in \mathbb{R}^{m\times d}$ and keys and values from context $\boldsymbol{X} \in \mathbb{R}^{m'\times d}$: 4 heads of width 64, no projection biases, and an optional additive logit bias and key mask. $\mathrm{Blk}(\boldsymbol{X})$ denotes self-attention, $\mathrm{Blk}(\boldsymbol{X}, \boldsymbol{X})$. The pooling variant Blk° omits the query residual $\boldsymbol{V} = \mathrm{LN}(\mathrm{MHA}(\boldsymbol{U}, \boldsymbol{X}))$. Each block maps $[m, 256]$ to $[m, 256]$ with an FFN of width 1,024 and has 788,736 parameters; g_ϕ contains eight, numbered Blk_1 to Blk_8 below.

The five stages

- **Stage 1: board embedding.** We attention-pool each board to a single vector: a learned query $\boldsymbol{q} \in \mathbb{R}^d$ cross-attends to the board's 64 square features through two blocks with an additive attention bias computed from the board itself:

  $$\boldsymbol{e}^{\pm}_{k,t} = \mathrm{Blk}_2\big(\mathrm{Blk}_1(\boldsymbol{q}, \boldsymbol{F}^{\pm}_{k,t}; \boldsymbol{\beta}), \boldsymbol{F}^{\pm}_{k,t}\big), \qquad \boldsymbol{\beta}(\boldsymbol{F}) = \boldsymbol{W}_g\mathrm{Swish}\big(\boldsymbol{W}_h\mathrm{vec}(\boldsymbol{F}\boldsymbol{W}_c) + \boldsymbol{b}_h\big) + \boldsymbol{b}_g$$

  Pooling with a learned query rather than averaging the squares lets the embedding weight the squares that matter per position. The bias $\boldsymbol{\beta}$ follows the Smolgen design of Leela Chess Zero (The LCZero Authors, 2018; Monroe & Chalmers, 2024): each square's feature is compressed to 32 dimensions ($\boldsymbol{W}_c \in \mathbb{R}^{d\times 32}$, no bias), the board is flattened to 2,048, and a Swish (**?**) nonlinearity of width 256 produces one logit bias per head and square ($\boldsymbol{W}_g$: $256 \to 4 \times 64$). The same bias is added to the attention logits of both blocks. The pre- and post-move boards are embedded independently with the same weights. This transforms the input from $[n, 64, 256]$ to $[n, 256]$ for each stack.

- **Stage 2: move embedding.** A two-layer MLP maps each pair of board embeddings to a single move embedding:

  $$\boldsymbol{m}_{k,t} = \boldsymbol{W}_4\mathrm{GELU}\big(\boldsymbol{W}_3[\boldsymbol{e}^-_{k,t}; \boldsymbol{e}^+_{k,t}; \boldsymbol{e}^+_{k,t} - \boldsymbol{e}^-_{k,t}] + \boldsymbol{b}_3\big) + \boldsymbol{b}_4$$

  The input concatenates the embeddings before and after the move with their difference, which exposes the change the move made; the rest of the encoder therefore sees the move played rather than two positions. The two $[n, 256]$ stacks are arranged into padded per-game sequences $[B, K, T, 256]$ with a mask over missing/unsampled moves, concatenated to $[B, K, T, 768]$, and mapped $768 \to 1{,}024 \to 256$.

- **Stage 3: ply tokens.** Each move embedding is normalized and combined with a learned role embedding $\boldsymbol{r} \in \mathbb{R}^d$ and a sinusoidal positional encoding (Vaswani et al., 2017) of its ply index $\ell_{k,t}$ in the original game:

  $$\boldsymbol{x}_{k,t} = \mathrm{LN}(\boldsymbol{m}_{k,t}) + \mathrm{LN}(\boldsymbol{r}) + \mathrm{PE}(\ell_{k,t}).$$

  The encoding uses the true ply index rather than the position within the sampled sequence. Only the player's own moves are encoded so the role embedding acts as a learned constant offset; one row of the $[2, 256]$ role table is never used. This transforms $[B, K, T, 256] \to [B, K, T, 256]$, with two LayerNorms and no learned projection.

- **Stage 4: intra-game encoder.** The T tokens of game k, $\boldsymbol{X}_k \in \mathbb{R}^{T\times d}$, pass through two self-attention blocks masked to the moves present; 32 learned seeds $\boldsymbol{S}_g \in \mathbb{R}^{32\times d}$ then cross-attend them, and a linear layer maps the flattened seed outputs to the game embedding:

$$\boldsymbol{H}_k = \mathrm{Blk}_4\big(\mathrm{Blk}_3(\boldsymbol{X}_k)\big), \qquad \boldsymbol{z}_{g,k} = \boldsymbol{W}_5\,\mathrm{vec}\big(\mathrm{Blk}_5^{\circ}(\boldsymbol{S}_g, \boldsymbol{H}_k)\big) + \boldsymbol{b}_5.$$

  The PMA is Perceiver-style latent cross-attention (**?**). Using 32 seeds rather than one lets the game embedding combine several distinct summaries of the game. This yields the transformations $[B,K,T,256] \to [B,K,T,256] \to [B,K,32,256] \to [B,K,8{,}192] \to [B,K,256]$.

- **Stage 5: cross-game encoder.** The K game embeddings $\boldsymbol{Z}_g = [\boldsymbol{z}_{g,1};\ldots;\boldsymbol{z}_{g,K}]$ pass through two self-attention blocks masked to the games present, followed by pooling by multi-head attention (PMA; **?**) with 32 learned seeds $\boldsymbol{S}_p$, a linear layer, and a final LayerNorm:

$$\boldsymbol{Z}' = \mathrm{Blk}_7\big(\mathrm{Blk}_6(\boldsymbol{Z}_g)\big), \qquad \boldsymbol{z}_p = \tfrac{1}{\sqrt{d}}\,\mathrm{LN}\Big(\boldsymbol{W}_6\,\mathrm{vec}\big(\mathrm{Blk}_8^{\circ}(\boldsymbol{S}_p, \boldsymbol{Z}')\big) + \boldsymbol{b}_6\Big).$$

  This stage is a Set Transformer encoder (**?**): the games carry no positional encoding, so $\boldsymbol{z}_p$ does not depend on their order; this yields transformation $[B,K,256] \to [B,K,256] \to [B,32,256] \to [B,8{,}192] \to [B,256]$.

Table 5: **Player encoder parameters.** Parameters of g_ϕ by stage. Each of its eight transformer blocks (two in Stage 1, three in Stage 4, three in Stage 5) has 788,736 parameters: four 256×256 attention projections, a $256 \to 1024 \to 256$ FFN, and two LayerNorms.

Stage	Contents	Tensors	Parameters
1. Board embedding	query, attention-bias generator, 2 cross-attn blocks	30	2,176,256
2. Move embedding	$768 \to 1024 \to 256$ MLP	4	1,049,856
3. Ply tokens	2 LayerNorms, $[2,256]$ role embedding	5	1,536
4. Within-game encoder	2 self-attn blocks, seed pooling, $8{,}192 \to 256$	39	4,471,808
5. Across-game encoder	2 self-attn blocks, PMA, $8{,}192 \to 256$, LayerNorm	41	4,472,320
Total g_ϕ		**119**	**12,171,776**

Contrastive views $\mathcal{L}_{\mathrm{CL}}$ is computed on views: latents built from subsamples of a player's reference games. The K=5 games are split into the fixed disjoint groups $\mathcal{G}_1 = \{1,2\}$ and $\mathcal{G}_2 = \{3,4,5\}$, and Stage 5 is re-run with its game mask restricted to each group:

$$\tilde{\boldsymbol{z}}_p^{(m)} = \frac{\boldsymbol{u}_p^{(m)}}{\|\boldsymbol{u}_p^{(m)}\|_2}, \qquad \boldsymbol{u}_p^{(m)} = g_\phi^{(5)}\big(\{\boldsymbol{z}_{g,k}\}_{k\in\mathcal{G}_m}\big), \qquad m \in \{1,2\},$$

where $g_\phi^{(5)}$ denotes Stage 5. Over the B players of a step,

$$\mathcal{L}_{\mathrm{CL}} = -\frac{1}{2B}\sum_{p=1}^{B}\sum_{m=1}^{2}\log\frac{\exp\big(\tilde{\boldsymbol{z}}_p^{(m)\top}\tilde{\boldsymbol{z}}_p^{(\bar{m})}/\kappa\big)}{\sum_{(q,m')\neq(p,m)}\exp\big(\tilde{\boldsymbol{z}}_p^{(m)\top}\tilde{\boldsymbol{z}}_q^{(m')}/\kappa\big)},$$

where $\bar{m}$ is the other view and κ follows the schedule in Appendix D.

D Implementation, training, and compute details

Backbone The backbone f_θ is the Maia-3 5M architecture (Monroe et al., 2026): eight post-norm transformer blocks (RMSNorm; **?**) at width 256 with 8 attention heads, GELU feed-forward layers, Maia-3's per-block geometric attention bias, and a final LayerNorm. It is trained from scratch, with Maia-3's value head, ponder head and ELO embedding tables removed. Its input follows Maia-3: 96 planes per square (8 history frames $\times$ 12 piece planes), concatenated with $\mathrm{concat}(\boldsymbol{z}_p, \boldsymbol{z}_{\mathrm{opp}}) \in \mathbb{R}^{512}$ broadcast over the 64 squares, so the token projection is $[608, 256]$ (Section 5). Every Maia-3 block is rematerialized in the backward pass (**?**). All weights use LeCun normal initialization (**?**) with zero biases and unit normalization gains, except where stated below.

Player encoder g_ϕ (Appendix C) maps $K{=}5$ reference games per player, at most 24 plies each, to $\boldsymbol{z}_p \in \mathbb{R}^{256}$ with $\|\boldsymbol{z}_p\|_2 \approx 1$, reading features from the reference body f_ψ: a second Maia-3 body with its own parameters, run with $\mathrm{cond} = \mathbf{0}$.

Contrastive learning $\mathcal{L}_{\mathrm{CL}}$ is the InfoNCE loss of Appendix C over $n_{\mathrm{pos}}{=}2$ disjoint-game views of each of the 48 players in a step (96 rows, in-batch negatives only). The temperature anneals linearly from $\kappa = 0.5$ to 0.2 over the first 2,000 steps and is held at 0.2 thereafter.

Policy head The policy head is Maia-3's factored source→destination head: two 256×256 projections of the per-square backbone features give source and destination embeddings whose scaled dot products form 4,096 from–to logits, and a $256 \to 4$ projection of the back-rank destination embeddings adds 256 promotion logits. The 4,352 logits are masked to legal moves before the softmax. The head takes no latent argument but reads the conditioned backbone features.

Preference head h_ξ is a $256 \to 128 \to 1$ MLP (GELU) sharing no parameters with the policy head. $\mathcal{L}_{\mathrm{prefer}}$ is a per-game Bradley–Terry likelihood of player against opponent over the decisive games whose opponent is encoded (draws carry no signal), with the logit clamped to ± 50 for numerical stability.

Context encoder The context is two scalars per position: the time control (base plus $40\times$ increment, in units of 600 s) and the think time spent on the move (in units of 10 s). A $2 \to 128 \to 64$ MLP (Mish; **?**) maps it to $\mathsf{Enc}_c(\boldsymbol{c}) \in \mathbb{R}^{64}$, which a zero-initialized $64 \to 256$ projection adds to the backbone features at all 64 squares before the policy head. ELO is excluded from the context, so skill must be recovered from gameplay alone.

Opponent conditioning $\boldsymbol{z}_{\mathrm{opp}}$ is produced by the same g_ϕ from the opponent's own five reference games; up to 96 distinct opponents are encoded per step. A position whose opponent is not among them receives a learned unknown-opponent vector in place of $\boldsymbol{z}_{\mathrm{opp}}$.

Optimization The parameters form six groups (Table 6) each with an independent Adam optimizer (**?**): the backbone, reference body, player encoder and policy head at 5×10^{-5}, and the preference head and context encoder at 2×10^{-4}. Only the player encoder has a learning-rate warmup, linear from zero over 5000 steps.

Batch composition Each step draws 48 players, three query games per player (disjoint from that player's reference games) and up to 15 positions per game, subsampled without replacement to exactly 512 padded query slots so that every tensor shape is static. The loss mask keeps positions past ply 10. Each step also encodes the five reference games of its 48 players and of up to 96 opponents with at most 24 plies each.

Data pipeline Our training data are Lichess blitz games from January 2023 to July 2025 (31 months), matching Maia-3's source, window, and time controls but sampled per player rather than per game. The window holds $\sim$4,280,000 distinct players, split by player into disjoint training, validation and test pools, and the sampler draws from $\sim$1,053,000 training players. One epoch draws every sampler player once (21,937 steps), so 620,000 steps are 28.3 epochs: 317,440,000 query slots, of which 254,269,440 contribute to a loss — 49.66% of Maia-3's 512M (1,000,000 steps $\times$ 512).

Loss weights and schedule $\lambda_{\mathrm{CL}}^{(t)}$ follows Theorem 2 with $\lambda_0 = 1.0$ and clamped normalized gaps $\bar{\varepsilon}_X^{(t)} = [(\mathcal{L}_X^{(t-1)} - \mathcal{L}_X^*)/\varepsilon_X^{(0)}]_0^1$, computed from the previous step's losses; the normalizers $\varepsilon_X^{(0)}$ are measured at step 0 and then held fixed. $\lambda_0 = 1.0$ suffices because the contrastive and behavioral gradients are of the same order. The Bradley-Terry scale is $\beta = 1.0$ and the NLL-to-latent gradient scale is $s = 1.0$. λ_{CL}, β and s ramp linearly from zero over the first 5000 steps.

Loss floors We fix $\mathcal{L}_{\mathrm{NLL}}^* = 1.0$ and $\mathcal{L}_{\mathrm{CL}}^* = 1.43$ for the adaptive schedule. $\mathcal{L}_{\mathrm{NLL}}^* = 1.0$ is a deliberate lower bound, below the NLL the model reaches ($\sim$1.3), so that $\varepsilon_{\mathrm{NLL}}$ stays positive. Because $\lambda_{\mathrm{CL}}^{(t)} \le \lambda_0 \bar{\varepsilon}_{\mathrm{CL}}^{(t)}$ for any value of $\bar{\varepsilon}_{\mathrm{NLL}}$, an error in $\mathcal{L}_{\mathrm{NLL}}^*$ rescales λ_{CL} but does not change that it vanishes. $\mathcal{L}_{\mathrm{CL}}^* = 1.43$ is measured by training on $\mathcal{L}_{\mathrm{CL}}$ alone for 1.5M steps and observing its plateau.

Theorem 2 assumptions discussion L-smoothness near the optimum is the standard regularity condition for InfoNCE on ℓ_2-normalized embeddings, whose similarities are bounded; after its 2,000-step ramp κ is constant. The simultaneous-convergence assumption is the strongest. Here the encoder's features come from f_ψ, which is itself trained by all three losses, so $\mathcal{L}_{\text{CL}}$ can converge without relying on pretrained features, and its floor is measured rather than assumed, so ε_{CL} is a gap to a loss the encoder actually reaches. Empirically, $\mathcal{L}_{\text{CL}}$ decays toward its floor and $\mathcal{L}_{\text{NLL}}$ decreases to a plateau. The assumption could fail if the behavioral and discriminative objectives imposed conflicting requirements on the shared encoder.

Gradient architecture All three losses couple at the player encoder through the latents, while the policy and preference heads share no parameters:

- the backbone, policy head and context encoder receive gradient from $\mathcal{L}_{\text{NLL}}$ only;
- the player encoder g_ϕ and reference body f_ψ receive gradient from $\mathcal{L}_{\text{CL}}$ (through the views), from $\mathcal{L}_{\text{prefer}}$ (through h_ξ, on both latents), and from $\mathcal{L}_{\text{NLL}}$ at scale s through both latents at the stem, via the straight-through form (**?**) $\text{sg}(\boldsymbol{z}) + s(\boldsymbol{z} - \text{sg}(\boldsymbol{z}))$, whose forward value is exact;
- the preference head receives gradient from $\mathcal{L}_{\text{prefer}}$ only.

Hardware Training runs on a single NVIDIA A100-SXM4-40GB GPU with 216 GiB of host RAM.

Training time A step takes $\sim$2.4 s, so the 620,000-step run takes $\sim$413 GPU-hours ($\sim$17 days).

Model size Table 6 reports parameter counts by optimizer group. Only the backbone, policy head and context encoder are evaluated per position; the other groups run once per player, and $\boldsymbol{z}_p$ is cached as 256 floats at evaluation.

Table 6: **Model size.** Parameter counts by optimizer group.

Group	Parameters	Evaluated per
Backbone f_θ (incl. unknown-opponent vector)	5,030,400	position
Policy head (incl. context projection)	148,480	position
Context encoder	8,640	position
Reference body f_ψ	4,964,608	player
Player encoder g_ϕ	12,171,776	player
Preference head h_ξ	33,025	player
Per position	5,187,520	
Per player	17,169,409	
Total	**22,356,929**	

Software JAX 0.6.0 (jaxlib 0.6.0; **?**), optax 0.2.8 (**?**), grain 0.2.13 (**?**) and NumPy 1.26.4 on Python 3.10.12. Training is fp32 throughout with no mixed-precision or distributed training. The Maia-3 body is a JAX port of the official Maia-3 implementation, verified against its released checkpoints (relative error $\leq 2.1 \times 10^{-6}$, identical top-1 moves).

Reproducibility A single seed (0) drives parameter initialization and the loader's shuffle, so per-epoch player order and position sampling are deterministic. A checkpoint every 2,500 steps stores the parameters, all six Adam states, the step and the data-iterator position.

E Additional result tables

Table 7 isolates each component. Ablations marked * are evaluation-time only (zeroing or reducing inputs to the trained model); all others are full retraining runs with identical data, hyperparameters, and pipeline.

Table 7: **Component ablations.** *Evaluation-time ablation; all others are full retraining runs. N/S = Normal/Shuffled top-1 (%).

Configuration	Avg N	Avg S	Δ(N−S)
Full UMARM	**56.94**	53.74	+3.20
$\boldsymbol{z}_p = 0^*$	55.52	—	—
K=1*	56.39	53.72	+2.67
No $\mathcal{L}_{\text{CL}}$	56.01	54.70	+1.31
Fixed λ_{CL}=1	56.48	53.81	+2.67
No $\mathcal{L}_{\text{prefer}}$	56.68	54.22	+2.46