UniMaia: Steering Chess Policies with Language for Human-like Play

Anonymous ACL submission

Abstract

Recent advances in large language models have enabled natural language to serve as a flexible interface for controlling complex systems, but often at the cost of large-scale multimodal training or weakened domain-specific inductive biases. In structured decision-making domains such as chess, specialized policy networks achieve strong performance but lack semantic controllability, while prompt-conditioned language models are more flexible yet typically exhibit weaker domain grounding. We propose **UniMaia**, a framework for prompt-conditioned policy modulation that adapts a frozen Lc0-based chess policy network using a parameter-efficient text encoder and a ControlNet-style conditioning mechanism. UniMaia enables semantic control over gameplay, including opening selection and player strength, while preserving the pretrained policy representations. We further introduce **UniMaia-Aux**, which incorporates auxiliary temporal conditioning and behavioral prediction objectives. To support this work, we construct a large-scale metadata-augmented Lichess dataset, develop a semi-automated prompt-generation pipeline, and introduce benchmarks spanning both prompt-conditioned and metadata-conditioned settings. UniMaia achieves state-of-the-art expected accuracy on several prompt-conditioned benchmarks and competitive top-move accuracy on general instruction-following tasks, while remaining competitive with dedicated metadata-conditioned approaches on human move prediction benchmarks. UniMaia-Aux further improves expected accuracy and behavioral modeling across several evaluation settings, with modest trade-offs in top-move accuracy. Overall, our results demonstrate that prompt-conditioned control of domain-specific policy networks is feasible without end-to-end multimodal training, while highlighting trade-offs between controllability and predictive performance.

1 Introduction

Natural language has become an increasingly flexible interface for controlling machine learning systems. Recent advances in large language models have enabled prompt-based interaction across a wide range of tasks, including reasoning, planning, and tool use. However, in structured decision-making domains, purely language-based approaches often struggle to match the performance and inductive biases of specialized policy models. This creates a tension between **controllability** and **domain grounding**: prompt-conditioned systems are highly flexible and steerable, while domain-specific policy networks achieve stronger task performance but typically rely on rigid structured conditioning.

Chess provides a useful testbed for studying this trade-off. Prior work on human-like chess modeling has largely focused on metadata-conditioned policy networks that predict human moves from board states and structured signals such as player rating or time control. These approaches achieve strong move-prediction performance but limited semantic controllability. In contrast, language-based chess models can follow natural language instructions but often exhibit weaker domain grounding and lower move-prediction accuracy. End-to-end multimodal approaches provide a unified alternative, but typically require substantial compute and large-scale curated datasets.

In this work, we investigate whether natural language prompts can be used to steer a pretrained chess policy while preserving the inductive biases of a strong domain-specific model. We introduce **UniMaia**, a framework for **prompt-conditioned modulation of pretrained chess policies**. UniMaia combines a frozen Lc0-based policy network with a parameter-efficient text encoder and a ControlNet-style conditioning mechanism, enabling natural language control over gameplay

behaviors such as opening selection and player strength while retaining the underlying chess representations of the base model.

To support this setting, we construct a large-scale metadata-augmented Lichess dataset and a prompt-generation pipeline for diverse natural language conditioning templates. We further introduce evaluation settings spanning both **prompt-conditioned** and **metadata-conditioned** benchmarks, enabling systematic comparison between language-conditioned and structured-conditioning approaches.

UniMaia achieves strong performance across both benchmark families. The model outperforms prior prompt-conditioned baselines on instruction-following and opening-control tasks while remaining competitive with metadata-conditioned approaches on human move prediction. We further show that the learned policy varies continuously with prompted Elo, suggesting that lightweight prompt-conditioned modulation can provide semantic control without discarding domain-specific inductive biases.

Our contributions are as follows:

1. We propose UniMaia, a parameter-efficient framework for prompt-conditioned modulation of pretrained chess policy networks.

2. We introduce a large-scale metadata-augmented chess dataset and prompt-generation pipeline for language-conditioned policy learning.

3. We evaluate prompt-conditioned and metadata-conditioned policy models under a unified benchmark framework spanning instruction following, opening control, and human move prediction.

4. We demonstrate that lightweight prompt-conditioned modulation enables controllable human-like chess behavior while remaining competitive with structured-conditioning approaches.

2 Background and Related Work

Chess Policy Networks. Modern neural chess engines model gameplay as policy learning over structured board representations. AlphaZero (Silver et al., 2017) demonstrated that deep neural networks trained through self-play reinforcement learning could achieve superhuman performance when combined with Monte Carlo Tree Search (MCTS). Leela Chess Zero (Lc0), an open-source reproduction of AlphaZero for chess, later introduced stronger transformer-based ChessFormer (CF) architectures (Monroe and Chalmers, 2024; Monroe et al., 2026). These models encode the current board state together with a fixed history of previous positions using spatial feature planes and predict move probabilities over a fixed legal move space. Unlike language-only chess models, Lc0-style architectures incorporate strong domain-specific inductive biases through structured board encodings and specialized policy heads, and have proven highly effective for both strong play and human gameplay modeling (McIlroy-Young et al., 2020; Monroe et al., 2026; Feng et al., 2024).

Human Move Prediction and Metadata Conditioning. Prior work on human move prediction adapts chess policies to model human gameplay distributions rather than optimal play. Maia (McIlroy-Young et al., 2020) fine-tuned Lc0 CNN-based networks on human games across rating ranges, while Maia-2 (Tang et al., 2024) extended this approach with unified skill conditioning using rating embeddings. Allie (Zhang et al., 2024b) further incorporated interpolated skill representations, time controls, and auxiliary behavioral objectives. Although effective, these approaches rely primarily on structured metadata rather than flexible natural-language control.

Prompt-Conditioned Chess Models. Other approaches instead model chess directly as a language task. ChessGPT (Feng et al., 2024) trains language models for chess generation and reasoning using specialized chess tokenization, while ChessCLIP (Feng et al., 2024) learns joint prompt–board representations. Additional work has explored large language models for gameplay and chess generation (Karvonen, 2024; Zhang et al., 2024a; Collet, 2024; Carlini, 2023). While these models provide flexible semantic control, they generally lack the structured inductive biases of specialized chess policy networks.

Multimodal Conditioning and Control. Our approach is also related to multimodal conditioning methods that adapt pretrained models using external signals. Cross-attention conditioning is widely used in vision–language and diffusion models (Alayrac et al., 2022; Liu et al., 2023; Rombach et al., 2022). In particular, ControlNet (Zhang et al.,

2023; Chen et al., 2024) introduces lightweight conditioning through zero-initialized residual pathways while preserving a frozen base model. UniMaia applies a similar principle to chess by conditioning a pretrained policy network on natural-language prompts while preserving its underlying structured representations.

3 Dataset and Prompt Construction

We construct a large-scale prompt-conditioned chess dataset from the Lichess Open Database (The Lichess team, n.d.c), designed to support natural-language control of pretrained chess policies.

3.1 LichessGames

Training data consists of publicly available Lichess PGN archives from 2013–2023, totaling approximately 5.2 billion games. We process the raw PGNs into a structured Parquet dataset, LICHESSGAMES, containing move histories, player ratings, time controls, openings, termination details, and game metadata.

To support controllable policy learning, we augment games with normalized opening annotations and canonical opening move sequences. Because Lichess opening names and ECO classifications changed over time, we align historical annotations with the current Lichess opening database (The Lichess team, n.d.b) using rule-based normalization, fuzzy matching, and manual verification. For each game, we recover the longest opening prefix consistent with both the annotated opening and played moves.

Games are additionally partitioned by rating range to improve sampling coverage across skill levels.

3.2 Prompt Template Construction

Rather than generating a unique prompt for every game, we construct synthetic natural-language templates instantiated from game metadata. This enables large-scale prompt-conditioned training without requiring human annotation for billions of examples.

Templates are generated using GPT-4 Turbo (Achiam et al., 2023) and converted into parameterized Jinja2 templates through a post-processing pipeline. Metadata such as player ratings, titles, time controls, openings, and opening move sequences are inserted programmatically during training.

Template generation follows an iterative human-in-the-loop process: generated prompts are manually reviewed, and the conversion pipeline is refined as needed. Existing templates are updated accordingly, while later revisions prioritize backward compatibility over regeneration.

Although the templates are synthetically generated, the effective prompt space is substantially larger than the number of base templates due to combinatorial variation in metadata inclusion, prompt framing, opening descriptions, player attributes, time controls, and sampled game metadata. As a result, instantiated prompts often differ considerably in structure and lexical content despite sharing the same template family.

We construct two complementary template families: LICHESSTEMPLATES-PRETRAIN and LICHESSTEMPLATES-INSTRUCT.

Pretraining templates. LICHESSTEMPLATES-PRETRAIN consists of 160 metadata-rich templates designed to expose the model to diverse chess metadata during training. These templates include combinations of player skill levels, time controls, opening descriptions, and varying prompt verbosity, producing prompts that resemble descriptive game summaries while preserving most available metadata.

Instruction templates. LICHESSTEMPLATES-INSTRUCT consists of 2,048 instruction-oriented templates designed to encourage controllability under partial or naturalistic specifications. These templates vary whether player names, titles, ratings, skill descriptions, openings, and opening move sequences are included, and whether prompts are phrased as requests or game descriptions.

Unlike the pretraining templates, the instruction templates intentionally omit portions of the metadata, requiring the model to infer plausible gameplay behavior from incomplete natural-language specifications. For example, templates vary across 2^{10} metadata inclusion configurations in addition to lexical and game-level variation.

During training, examples are formatted using either template family with equal probability.

4 Method

We propose **UniMaia**, a prompt-conditioned chess policy model that adapts a pretrained Lc0 policy network using natural language prompts. Rather than training a language model to play chess end-to-

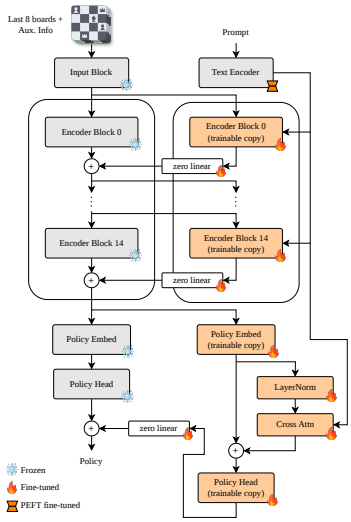

Figure 1: UniMaia architecture. A frozen Lc0-CF backbone encodes the board state, while a ControlNet-style branch incorporates prompt embeddings through cross-attention. Both the frozen backbone layer and the conditioning layer at depth n receive the residual-updated representation from layer $n-1$.

end, UniMaia conditions a frozen domain-specific chess policy through lightweight residual modulation.

4.1 Architecture

Given a board state s and natural language prompt p, the model predicts a distribution over legal chess moves. UniMaia consists of: (i) a frozen Lc0-CF backbone, (ii) a LoRA-adapted ChessGPT-base text encoder (Hu et al., 2022), and (iii) a ControlNet-style conditioning branch that injects prompt-conditioned residual updates into the policy computation.

Figure 1 illustrates the architecture.

UniMaia builds on a pretrained Lc0-CF policy network initialized from BT4 weights. The backbone remains frozen during training and maps board states s to hidden representations via a sequence of transformer layers B_n, a policy embedder PE, and a policy head PH.

The natural language prompt p is encoded by a text encoder

$$e = T_\phi(p),$$

which produces a sequence of prompt embeddings. To keep adaptation lightweight, the text encoder is fine-tuned using LoRA adapters together with rsLoRA and LoRA+ (Kalajdzievski, 2023; Hayou et al., 2024).

UniMaia conditions the frozen chess policy using a layer-coupled ControlNet-style branch. Both branches receive the residual-updated representation from the previous layer:

$$h_n = B_n(h_{n-1}+r_{n-1}),\ \tilde{h}_n = C_n(h_{n-1}+r_{n-1},\ e),$$

where C_n includes self-attention, cross-attention over the prompt embeddings e, and a feed-forward block, and $h_{-1} + r_{-1}$ denotes the output of the frozen input block (with $r_{-1} = 0$). The residual update is then

$$r_n = \mathcal{Z}_n(\tilde{h}_n),$$

projected through a zero-initialized layer $\mathcal{Z}_n$, ensuring the model is initially equivalent to the pretrained backbone.

After the final layer, UniMaia applies prompt modulation directly to the intermediate policy representation using a *controllable policy head*. The controllable policy head consists of a cloned policy embedder $\widehat{\mathrm{PE}}$, a pre-LN cross-attention layer over the prompt embeddings, and a cloned policy head $\widehat{\mathrm{PH}}$.

A cloned policy embedder produces

$$z = \widehat{\mathrm{PE}}(h_N + r_N),$$

which is passed through the controllable policy head and a final zero-initialized projection:

$$r_\pi = \mathcal{Z}_\pi\Big(\widehat{\mathrm{PH}}(\mathrm{CrossAttn}(\mathrm{LN}(z),\ e))\Big).$$

The final policy logits are $\mathrm{PH}(\mathrm{PE}(h_N + r_N)) + r_\pi$, over which we apply softmax to obtain the move distribution $\pi(a \mid s, p)$.

4.2 Training

UniMaia is trained on supervised move prediction using human games from the Lichess dataset, with only the prompt encoder adapters and conditioning branch updated; the Lc0 backbone remains frozen throughout. Training data consists of games from 2013–2022, while games from 2023 are held out for evaluation. Monthly Lichess partitions are streamed from disk or Hugging Face. Following the LichessGames partitioning scheme described in Section 3.1, we first apply Unimax downsampling (Chung et al., 2023) across the Elo partitions to improve coverage across skill levels while allocating a fixed training budget. Within each sampled Elo partition, games are then grouped by time control, and Unimax is applied again independently

Change	Δ Dev Score	$\Delta\mu_{\text{loss, Dec13}}$
ControlNet + policy head	**+0.0684**	**–0.1005**
NorMuon optimizer	+0.0110	–0.0349
LoRA vs DoRA	+0.0016	–0.0049
BT4 backbone	+0.0059	–0.0219
Unimax sampling	+0.0195	–0.0295
+ LoRA rank 16	+0.0017	+0.0011
Full configuration	**+0.1061**	**–0.1811**

Table 1: Sequential design improvements during model development. Each row modifies the previous configuration and should therefore be interpreted as a development trace rather than an independent ablation.

to balance the distribution of time controls without allowing dominant groups to overwhelm the dataset.

Prompts are drawn from the LICHESSTEMPLATES-PRETRAIN and LICHESSTEMPLATES-INSTRUCT template families with equal probability. Unless otherwise specified, prompts may include opening information, player ratings, time controls, and partial move histories. Full benchmark details, prompting strategies, auxiliary evaluations, and ablations are provided in the appendix.

UniMaia-Aux denotes an additional fine-tuning stage using temporal metadata and auxiliary prediction targets, including game outcome, termination type, remaining plies, and move delay. Further details appear in Appendix G.2.

4.3 Design Evolution

UniMaia was developed through an iterative sequence of architectural, optimization, and data-centric modifications. Rather than presenting a fully factorial ablation study, we summarize the primary design changes that led to the final model in Table 1. Each row represents a modification to the previous configuration, beginning with an early-fusion baseline containing a single cross-attention layer.

For model development, we used a small diagnostic benchmark suite consisting of opening-following, instruction-following, and human move prediction tasks. We report the *development score* (Dev score), defined as the mean of the normalized accuracies across these tasks.

Each ablation was trained on 2013 data. We report the mean training loss over the December 2013 partition, denoted $\mu_{\text{loss, Dec13}}$.

The largest improvement arises from replacing the early-fusion baseline with a layer-coupled ControlNet architecture and controllable policy head, suggesting that explicit residual modulation of the pretrained chess policy is substantially more effective than shallow cross-attention conditioning. Data-centric modifications, particularly Unimax sampling across time controls, also provide substantial gains, while later scaling decisions exhibit diminishing returns.

Final evaluation results on the complete benchmark suite are reported in Section 6. Additional ablations, parameter analyses, training curves, and negative results are provided in Appendix F.

5 Experimental Setup

We evaluate UniMaia on benchmarks spanning prompt-conditioned and metadata-conditioned settings. Prompt-conditioned benchmarks measure adaptation to natural-language instructions, while metadata-conditioned benchmarks evaluate human move prediction given structured context such as player rating and time control. Across all settings, the primary objective is agreement with human moves.

Prompt-conditioned benchmarks. We evaluate opening controllability by creating the Lichess Openings Benchmark (LOB), which pairs positions with natural-language prompts describing target openings. We report results on two variants: LOB-Partial (LOB-P; 31,169 positions), where prompts contain only partial opening information up to the current position, and LOB-Canonical (LOB-C; 32,736 positions), where prompts additionally include the canonical continuation of the opening. LOB-P provides a more realistic and challenging setting, while LOB-C measures upper-bound opening-following performance.

To evaluate general instruction following, we introduce the Lichess Instruction-Following Benchmark (LIF; 25,000 positions), constructed from Lichess games formatted with diverse prompt templates. LIF evaluates adaptation to natural-language descriptions of player strength, time controls, and gameplay context.

We additionally introduce the Lumbra's Gigabase Benchmark (LGB; 25,000 positions), constructed from curated over-the-board and online games from Lumbra's Gigabase (Jansen, n.d.). We use the OTB 2020–2024 subset to evaluate generalization beyond the Lichess training distribution. Compared to LIF, LGB contains games from a broader range of tournaments, federations,

and historical sources, providing a stronger out-of-distribution evaluation.

Finally, we define temporally augmented variants of these benchmarks, denoted LIF-Aux and LGB-Aux, which append metadata describing the time control, remaining clock time, and previous-move delay when available. These benchmarks additionally evaluate the auxiliary behavioral targets introduced during auxiliary-target fine-tuning.

Metadata-conditioned benchmarks. To compare against prior human imitation models, we evaluate on metadata-conditioned human move prediction benchmarks. We report results on the Maia-1 Subset (M1-S; 11,225 positions), the full Maia-1 benchmark (M1; 11.3M positions) (McIlroy-Young et al., 2020), the Maia-2 Rapid Benchmark (M2R; 127,852 positions) (Tang et al., 2024), and the Allie Blitz Benchmark (ABB) (Zhang et al., 2024b).

M1-S is a stratified subset of the Maia-1 benchmark that we construct by grouping positions by player rating and time control. These benchmarks evaluate how well models predict human moves when contextual information is provided explicitly rather than through natural-language prompts. Additional benchmark details and model scores are provided in the appendix.

Comparability across model classes. Prompt-conditioned benchmarks use randomly sampled templates with varying metadata, whereas metadata-conditioned benchmarks use fixed templates with consistent metadata fields. Consequently, comparisons are most meaningful within each conditioning regime. Metadata-conditioned models may receive metadata absent from prompts, while prompts may contain information unavailable to metadata-conditioned models. Appendix I.5 compares model inputs.

Evaluation metrics. We report both top-move accuracy (Acc@1) and expected accuracy ($\mathbb{E}[\mathrm{Acc}]$). Expected accuracy measures the probability assigned to the ground-truth move and is reported only for models that produce normalized move distributions. For auxiliary-target evaluations, we additionally report weighted F1 ($\mathrm{F1}_w$) for categorical targets.

Following prior work (McIlroy-Young et al., 2020; Zhang et al., 2024b), we additionally report results with the first 10 plies excluded on ABB and M1-S to focus on general gameplay. For LIF and LGB, we include all plies, since some prompts explicitly target opening play.

Baselines. We compare UniMaia against metadata-conditioned human imitation models, including Maia-1, Maia-2, and Allie, as well as prompt-conditioned approaches including ChessGPT and GPT-3.5 Turbo Instruct. We additionally compare against pretrained Lc0-CF policy networks.

For autoregressive language models, generated moves are greedily decoded, constrained to valid chess formats where possible, and illegal moves are replaced with uniformly sampled legal moves during evaluation. We focus on GPT-3.5 Turbo Instruct following prior observations that newer GPT models frequently generate illegal chess moves despite stronger general capabilities (Carlini, 2023). Additional prompting details and benchmark variants are provided in Appendices I and J.

6 Results

Tables 2 and 3 summarize the main benchmark results. Overall, the models exhibit a clear trade-off: language models perform best on opening reconstruction, metadata-conditioned models achieve the strongest conventional human move prediction performance, and UniMaia performs best on prompt-conditioned instruction-following benchmarks.

UniMaia achieves the strongest Acc@1 performance on the instruction-following benchmarks LIF and LGB while remaining competitive with metadata-conditioned approaches on conventional human move prediction tasks. In contrast, ChessGPT-base performs best on the opening-focused LOB benchmarks but degrades substantially on broader gameplay benchmarks, suggesting weaker generalization beyond opening continuation prediction.

Allie achieves the strongest overall metadata-conditioned performance on ABB and M1-S, whereas UniMaia provides substantially stronger prompt-conditioned controllability with only modest reductions on metadata-conditioned tasks. UniMaia also remains competitive with GPT-3.5 and Allie despite being optimized primarily for natural-language policy modulation.

Table 3 reports expected accuracy results. UniMaia and UniMaia-Aux achieve the strongest overall performance across most settings, with UniMaia-Aux often outperforming UniMaia on $\mathbb{E}[\mathrm{Acc}]$ despite slightly lower Acc@1. This suggests that auxiliary temporal conditioning improves

Model	LOB-P	LOB-C	LIF	LGB	ABB	M1-S	M1	M2R
Lc0-CF (BT4)	0.5025	0.5039	0.3924	0.4770	0.4133	0.4255	0.4305	0.4079
Maia-1*	0.5582	0.5602	0.4797	0.4857	0.5043	0.5072	0.5119	0.5094
Maia-2 (Blitz)	0.4058	0.4040	0.4452	0.4961	0.5230	0.5246	0.5300	0.5284
Allie-Policy	0.6150	0.6164	0.5390	0.5572	**0.5643**	**0.5733**	**0.5717**	**0.5598**
ChessGPT-base†	**0.7998**	**0.8955**	0.4464	0.4788	0.4336	0.4375	–	0.4305
GPT-3.5§	0.5748	0.6431	0.5312	0.5514	0.5498	0.5667	–	0.5489
UniMaia	0.7094	0.8162	**0.5587**	**0.6158**	0.5548	0.5667	–	0.5536
UniMaia-Aux	0.7071	0.7801	0.5481	0.6086	0.5471	0.5624	–	0.5493

Table 2: Main benchmark comparison measured by Acc@1. Best values are in bold and second-best values are underlined. *Evaluated with the model with the closest Elo to the active player. †Opening benchmarks evaluated with the prompt and PGN. Remaining benchmarks evaluated with the Elo and time control PGN header. §Opening benchmarks evaluated with the GM PGN header. Remaining benchmarks evaluated with the Elo and time control PGN header. Further details in Appendix I.4.

Model	LOB-P	LOB-C	LIF	LGB	ABB	M1-S	M1	M2R
Lc0-CF (BT4)	0.3657	0.3655	0.2772	0.3436	0.3063	0.3175	0.3198	0.3036
Maia-1*	0.4130	0.4138	0.3653	0.3651	0.3816	0.3840	0.3862	0.3827
Maia-2 (Blitz)	0.2990	0.2971	0.3423	0.3860	0.4102	0.4144	0.4166	0.4137
Allie-Policy	0.4905	0.4906	0.4104	0.4480	**0.4409**	0.4476	**0.4475**	0.4354
UniMaia	0.5899	**0.7140**	0.4297	0.4834	0.4224	0.4317	–	0.4182
UniMaia-Aux	**0.5973**	0.6823	**0.4331**	**0.4891**	0.4382	**0.4488**	–	**0.4427**

Table 3: Main benchmark comparison measured by $\mathbb{E}[\text{Acc}]$. Best values are in bold and second-best values are underlined. *Evaluated with the model with the closest Elo to the active player.

policy calibration even when temporal metadata is unavailable at evaluation time.

Unlike autoregressive language models, UniMaia inherits the legality constraints and inductive biases of the pretrained chess policy, substantially reducing illegal move generation and improving gameplay stability.

6.1 Auxiliary Temporal Metadata

Benchmark	Model	Acc@1	$\mathbb{E}[\text{Acc}]$	F1_w Resigned
LIF-Aux	Allie-Policy	0.5390	0.4104	0.6409
	UniMaia-Aux	**0.5549**	**0.4318**	**0.7279**
LGB-Aux	Allie-Policy	0.5572	0.4480	0.0933
	UniMaia-Aux	**0.5872**	**0.4637**	**0.6643**

Table 4: Auxiliary targets results. Best values are in bold and second-best values are underlined.

Table 4 reports results on the temporally augmented LIF-Aux and LGB-Aux benchmarks, which append temporal metadata such as time controls, remaining clock information, and previous move delay to the prompt.

UniMaia-Aux consistently outperforms Allie-Policy on both benchmarks in Acc@1 and expected accuracy while additionally supporting auxiliary behavioral prediction tasks. The largest gains occur for resignation prediction, suggesting improved modeling of human gameplay behavior beyond next-move prediction.

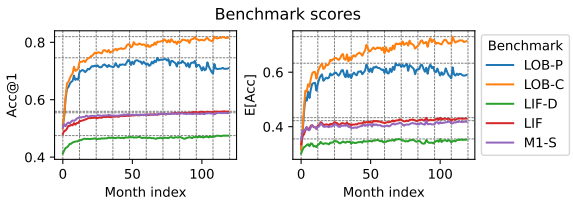

Figure 2: Benchmark performance across monthly checkpoints during training. Most benchmarks continue improving throughout training, although gains diminish over time. Vertical lines indicate year boundaries.

6.2 Training Dynamics

Figure 2 shows benchmark performance across monthly checkpoints. Most benchmarks improve steadily throughout training, although gains diminish over time.

Interestingly, LOB-C continues improving, whereas LOB-P plateaus and slightly declines. Since LOB-C includes canonical opening continuations while LOB-P provides only partial opening information, this suggests that UniMaia increas-

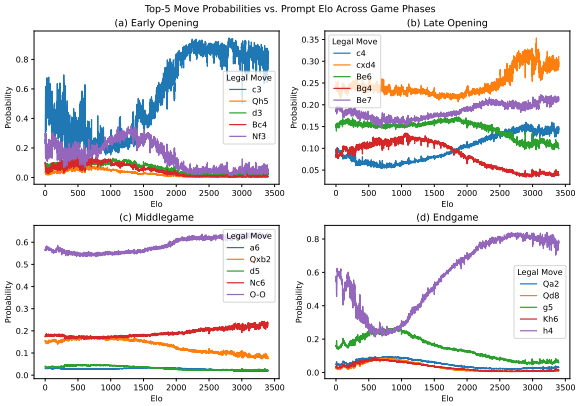

Figure 3: Top-5 move probabilities predicted by UniMaia as a function of Elo across representative game stages.

ingly benefits from stronger prompt constraints as training progresses.

These improvements come despite UniMaia being trained on a substantially smaller dataset than competing approaches: approximately 67M plies, compared to 9.15B for Maia-2 and over 262B tokens for Allie. This suggests that prompt-conditioned modulation of a pretrained backbone can be highly data-efficient, though the LLM backbone imposes higher computational costs due to memory and batch size constraints.

7 Analysis

We analyze how prompt-conditioned modulation affects UniMaia's learned policy, focusing on policy continuity across Elo. Although UniMaia uses discrete Elo tokens, the induced policies vary continuously across rating levels, producing smooth behavioral changes under varying conditioning signals.

Figure 3 shows top-5 move probabilities as a function of Elo for representative opening, middlegame, and endgame positions. Across positions, move distributions exhibit clear global trends as Elo increases, suggesting that UniMaia learns a continuous notion of player strength rather than isolated rating-specific behaviors. Local fluctuations remain, particularly in openings with many plausible continuations.

For Elo values far below the training distribution, the policy partially reverts toward stronger play, likely reflecting the influence of the frozen pretrained Lc0 backbone under limited low-Elo supervision. Ratings below 600 are absent during training due to a hard-coded minimum Elo threshold on Lichess (Duplessis, 2019; Fiekas, 2023).

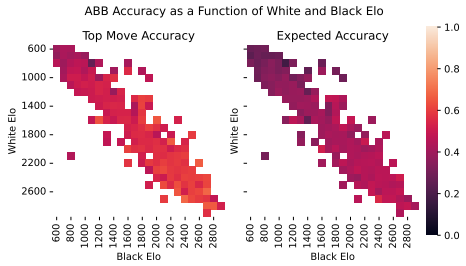

Figure 4: ABB accuracy as a function of White and Black Elo. Left: top-move accuracy. Right: expected accuracy.

We further analyze rating-dependent behavior using the metadata-conditioned ABB benchmark. Figure 4 shows that both top-move accuracy and expected accuracy generally increase with the Elo of both players, consistent with higher-rated games being more predictable and lower-rated games exhibiting greater stochasticity.

Compared to prompt-conditioned benchmarks such as LIF, the ABB heatmap is substantially smoother across rating combinations. We attribute this to ABB using a fixed prompt template, whereas LIF samples diverse templates with varying metadata fields, introducing additional variability.

Overall, UniMaia exhibits coherent rating-dependent behavior while remaining sensitive to contextual factors such as opening structure and prompt formulation.

8 Conclusion

UniMaia demonstrates that pretrained domain-specific policy networks can be steered through natural language while preserving the inductive biases of specialized models. By combining a frozen Lc0-based chess policy with lightweight prompt-conditioned modulation, UniMaia achieves strong controllability over behaviors such as opening selection and player strength while remaining competitive on human move prediction benchmarks. Our results suggest that controllable policy modulation in structured domains depends more on conditioning design, including data composition and integration with pretrained expert policies, than on raw model scale. More broadly, this work highlights lightweight language-conditioned adaptation as a practical alternative to fully end-to-end multimodal training for structured decision-making systems, providing a general framework for controllable policies built on pretrained expert models.

Limitations

Despite strong performance across prompt-conditioned and metadata-conditioned benchmarks, UniMaia has several limitations.

First, the model remains sensitive to prompt phrasing and tokenization. Small lexical or formatting changes can noticeably alter the policy, especially for uncommon or out-of-distribution prompts. Cross-attention analysis also reveals widespread attention sink behavior, where many layers disproportionately attend to the first prompt token (Appendix L.2.2), suggesting both prompt-distribution and architectural limitations in robustness.

Second, the prompt data is synthetically generated using LLM-assisted templates with semi-automated validation and manual review. While scalable, this process may introduce artifacts and does not fully capture the variability or ambiguity of real human instructions. Some metadata dimensions are also underrepresented or omitted, such as time control information in the primary instruction-template distribution. This limitation affects UniMaia but not UniMaia-Aux, whose prompts are augmented with detailed temporal metadata. Maintaining correctness under iterative template modifications is also challenging: independently generated metadata snippets can introduce grammatical inconsistencies, duplicated phrases, or formatting errors across many prompt instantiations, and exhaustive validation became increasingly difficult as the template space expanded.

Third, the opening benchmarks rely on curated opening names and canonical continuations. While this improves reproducibility, it may partially reward memorization of opening terminology rather than deeper semantic understanding. Benchmark performance may also not fully reflect perceived human likeness or stylistic coherence during gameplay.

The evaluation scope is similarly limited. Experiments rely entirely on offline automated benchmarks rather than human studies or live gameplay, leaving open questions about perceived human likeness, long-term controllability, and robustness to paraphrased or adversarial prompts.

The training data is limited to Lichess games, introducing platform-specific biases related to player demographics, openings, time controls, and online play behavior. Generalization to over-the-board or correspondence chess may therefore be imperfect.

Architecturally, UniMaia prioritizes modularity and preservation of the pretrained Lc0 policy through a layer-coupled ControlNet design. However, later ablations suggest that deeper fusion and full fine-tuning can achieve stronger raw performance, indicating a trade-off between controllability, modularity, and peak accuracy. UniMaia-Aux also increases optimization complexity while yielding only modest gains in some settings.

Many design choices, including template construction and auxiliary objectives, were tuned specifically for chess. Although the overall approach could potentially generalize to other domains by pairing pretrained expert policies with domain-specific instruction data, such adaptation would require substantial changes to the data collection and conditioning pipeline.

Finally, UniMaia's frozen 3B-parameter LLM backbone introduces substantial computational overhead. The resulting batch size of 64, compared to 8,192 for Maia-2, requires many more gradient steps per training position. Full training took 31.6 days on two RTX Pro 6000 GPUs, exceeding the compute budgets of Maia-2 (13 days on two 80-GB A100s) and Allie (14 days on eight A6000s), even after accounting for per-GPU throughput differences.

However, smaller-scale experiments were considerably more accessible: the 2013-only ablation runs, which used the same architecture and parameter count, completed in roughly 7 hours on a single consumer-grade RTX 5090 with a smaller batch size. This suggests that meaningful experimentation and ablation studies remain feasible on modern consumer hardware, despite the high cost of full-scale training.

Nevertheless, the overall preprocessing pipeline and multi-GPU training over millions of plies may still limit reproducibility and accessibility. Future work could reduce this overhead through smaller language model backbones, more efficient conditioning mechanisms, or more aggressive quantization.

Acknowledgments

Generative AI was used to proofread, reorganize, and improve the conciseness of the text. For code, autocomplete suggestions were used, and some brief example snippets were modified for our own use.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, and 1 others. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Noah Amsel, David Persson, Christopher Musco, and Robert M Gower. 2025. The polar express: Optimal matrix sign methods and their application to the muon algorithm. *arXiv preprint arXiv:2505.16932*.

Max Bachmann. n.d. GitHub - rapidfuzz/RapidFuzz: Rapid fuzzy string matching in Python using various string metrics — github.com. https://github.com/rapidfuzz/RapidFuzz. [Accessed 22-06-2024].

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, and 1 others. 2022. Gpt-neox-20b: An open-source autoregressive language model. In *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136.

Nicholas Carlini. 2023. Playing chess with large language models — nicholas.carlini.com. https://nicholas.carlini.com/writing/2023/chess-llm.html. [Accessed 10-07-2024].

Junsong Chen, Yue Wu, Simian Luo, Enze Xie, Sayak Paul, Ping Luo, Hang Zhao, and Zhenguo Li. 2024. Pixart-{\delta}: Fast and controllable image generation with latent consistency models. *arXiv preprint arXiv:2401.05252*.

Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. 2023. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. *arXiv preprint arXiv:2304.09151*.

Hailey Collet. 2024. Exploring mamba's capabilities with chess: Ablations, probing inner states, and transformer comparisons. https://haileystorm.github.io/mamba-chess/exploring-capabilities/. [Accessed 10-07-2024].

Timothy Dozat. 2016. Incorporating nesterov momentum into adam. *ICLR Workshop Track*.

Thibault Duplessis. 2014. auto threefold draw setting + improve prefs page · lichess-org/lila@de1ed4f — github.com. https://github.com/lichess-org/lila/commit/de1ed4f0e42bea222c9bc424129b2cad2fbec584. [Accessed 07-04-2026].

Thibault Duplessis. 2019. lower minimum rating to 600 · lichess-org/lila@b248195 — github.com. https://github.com/lichess-org/lila/commit/b248195f9bd4b1539d4078e903cff28d661569e3. [Accessed 24-05-2026].

Thibault Duplessis. 2022. change default threefold setting to always - closes #11812 · lichess-org/lila@c113413 — github.com. https://github.com/lichess-org/lila/commit/c1134134d398be97701672b5d916d5ed91bfbd43. [Accessed 07-04-2026].

Arpad E Elo. 1961. The uscf rating system - a scientific achievement. *Chess Life*, 16(6):160–161.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. 2024. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems*, 36.

Niklas Fiekas. 2023. lower minRating: 600 -> 400 by niklasf · Pull Request #12488 · lichess-org/lila — github.com. https://github.com/lichess-org/lila/pull/12488/changes. [Accessed 03-02-2026].

Niklas Fiekas. n.d. GitHub - niklasf/python-chess: A chess library for Python, with move generation and validation, PGN parsing and writing, Polyglot opening book reading, Gaviota tablebase probing, Syzygy tablebase probing, and UCI/XBoard engine communication — github.com. https://github.com/niklasf/python-chess. [Accessed 18-06-2024].

Henrik Forstén. 2020. Moves left head by Ttl · Pull Request #961 · LeelaChessZero/lc0 — github.com. https://github.com/LeelaChessZero/lc0/pull/961#issuecomment-587112109. [Accessed 03-03-2026].

Mark E Glickman. 2012. Example of the glicko-2 system. *Boston University*, 28:2012.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Michael Jansen. n.d. Free chess game database for Scid and PGN — lumbrasgigabase.com. https://lumbrasgigabase.com/en/. [Accessed 14-04-2026].

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. 2024. Muon: An optimizer for hidden layers in neural networks.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*.

Adam Karvonen. 2024. Emergent world models and latent variable estimation in chess-playing language models. *arXiv preprint arXiv:2403.15498*.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and 1 others. 2022. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. 2025. Normuon: Making muon more efficient and scalable. *arXiv preprint arXiv:2510.05491*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, and 1 others. 2025. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. 2020. 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*, pages 1677–1687.

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

Daniel Monroe, George Eilender, Philip Chalmers, Zhenwei Tang, and Ashton Anderson. 2026. Chessformer: A unified architecture for chess modeling. In *The Fourteenth International Conference on Learning Representations*.

OpenAI. n.d. GPT-3.5 Turbo Model | OpenAI API — developers.openai.com. https://developers.openai.com/api/docs/models/gpt-3.5-turbo. [Accessed 22-04-2026].

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, and 1 others. 2025. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

SeatGeek. n.d. GitHub - seatgeek/thefuzz: Fuzzy String Matching in Python — github.com. https://github.com/seatgeek/thefuzz. [Accessed 22-06-2024].

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, and 1 others. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Zhenwei Tang, Difan Jiao, Reid McIlroy-Young, Jon Kleinberg, Siddhartha Sen, and Ashton Anderson. 2024. Maia-2: A unified model for human-ai alignment in chess. *Advances in Neural Information Processing Systems*, 37:20919–20944.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, and 1 others. 2022. Ul2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025a. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, and 1 others. 2025b. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

The Lichess team. 2024. Frequently Asked Questions — lichess.org. https://lichess.org/faq#threefold. [Accessed 10-07-2024].

The Lichess team. n.d.a. Chess players and leaderboards — lichess.org. https://lichess.org/player. [Accessed 03-02-2026].

The Lichess team. n.d.b. lc0/src/neural/encoder.cc at master – LeelaChessZero/lc0 — github.com. https://github.com/LeelaChessZero/lc0/blob/master/src/neural/encoder.cc. [Accessed 03-03-2026].

The Lichess team. n.d.c. lczero-training/tf/lc0_az_policy_map.py at master – LeelaChessZero/lczero-training — github.com. https://github.com/LeelaChessZero/lczero-training/blob/master/tf/lc0_az_policy_map.py. [Accessed 03-03-2026].

The Lichess team. n.d.d. Weekly Rapid rating distribution — lichess.org. https://lichess.org/stat/rating/distribution/rapid. [Accessed 03-02-2026].

The Lichess team. n.d.a. Chess rating systems — lichess.org. https://lichess.org/page/rating-systems. [Accessed 03-02-2026].

The Lichess team. n.d.b. GitHub - lichess-org/chess-openings: An aggregated data set of chess opening names — github.com. https://github.com/lichess-org/chess-openings. [Accessed 22-06-2024].

The Lichess team. n.d.c. lichess.org open database — database.lichess.org. https://database.lichess.org/. [Accessed 18-06-2024].

The Lichess team. n.d.d. LM Title (Lichess Master Title) — lichess.org. https://lichess.org/forum/general-chess-discussion/lm-title--lichess-master-title. [Accessed 29-03-2026].

Unknown. 2026. Historical peak elo and identity data for notable lichess users.

Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, and 1 others. 2025. Embeddinggemma: Powerful and lightweight text representations. *arXiv preprint arXiv:2509.20354*.

Maurice Weber, Daniel Y Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, and 1 others. 2024. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37:116462–116492.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. 2024. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*.

Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Biao Zhang, Paul Suganthan, Gaël Liu, Ilya Philippov, Sahil Dua, Ben Hora, Kat Black, Gus Martins, Omar Sanseviero, Shreya Pathak, and 1 others. 2025a. T5gemma 2: Seeing, reading, and understanding longer. *arXiv preprint arXiv:2512.14856*.

Edwin Zhang, Vincent Zhu, Naomi Saphra, Anat Kleiman, Benjamin L Edelman, Milind Tambe, Sham M Kakade, and Eran Malach. 2024a. Transcendence: Generative models can outperform the experts that train them. *arXiv preprint arXiv:2406.11741*.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Yiming Zhang, Athul Paul Jacob, Vivian Lai, Daniel Fried, and Daphne Ippolito. 2024b. Human-aligned chess with a bit of search. *arXiv preprint arXiv:2410.03893*.

A Leela Chess Zero Encoding

UniMaia builds upon pretrained Leela Chess Zero (Lc0) ChessFormer policy networks (Monroe and Chalmers, 2024; Monroe et al., 2026). Because UniMaia conditions a frozen Lc0 policy through a ControlNet-style conditioning branch, the underlying Lc0 input and output representations remain unchanged. We briefly summarize the encoding scheme used by Lc0 below.

A.1 Input Encoding

Lc0 represents each position using 112 spatial feature planes defined over the 8×8 chess board (team, n.d.b). The representation encodes the current board state together with a fixed history of the previous seven positions, following the AlphaZero-style formulation (Silver et al., 2017).

The first $13 \times 8 = 104$ planes encode the 8 most recent board positions, ordered from most recent to oldest. For each position, the first six binary planes encode the locations of the white player's pawns, knights, bishops, rooks, queens, and king, respectively, followed by six corresponding planes for the black player's pieces in the same order. A thirteenth plane indicates whether the position has previously occurred in the game, enabling detection of repetition states. Missing history positions are padded with zeros.

The remaining 8 planes encode auxiliary game-state information using spatially constant values across all board squares. In order, these planes encode white queenside castling rights, white kingside castling rights, black queenside castling rights, black kingside castling rights, whether it is black's turn to move, the halfmove clock normalized by 99, a deprecated all-zero plane, and an all-ones plane used to help identify board boundaries.

As in AlphaZero, board states are always encoded from the perspective of the current player, so the side to move is consistently represented as playing from the bottom of the board (Silver et al., 2017). Lc0 additionally uses a simplified repetition encoding compared to AlphaZero, employing a single repetition plane per historical position rather than two separate repetition-count planes. Although this encoding does not distinguish higher-order repetitions such as fivefold repetition, such events are extremely rare in practice (see Appendix D).

A.2 Output Encoding

Lc0 predicts move probabilities over a fixed output space of 1,858 legal UCI move tokens (team, n.d.c). The encoding covers all legal moves from the perspective of the current player, including pawn promotions represented as distinct move types. Knight promotions are handled implicitly by treating ordinary pawn moves from the seventh to eighth rank as knight promotions when the moved piece is a pawn.

In addition to the policy head, modern Lc0 models predict a three-dimensional value distribution corresponding to win, draw, and loss probabilities, rather than the scalar value prediction used in AlphaZero (Silver et al., 2017). During Monte Carlo Tree Search, the scalar evaluation is computed as $\mathbb{P}(\text{win}) - \mathbb{P}(\text{loss})$.

Lc0 models also include a moves-left head that predicts the expected number of remaining plies in the game (Forstén, 2020). This auxiliary prediction helps reduce degenerate MCTS behavior in decisively winning or losing positions, where many continuations receive similar value estimates, by encouraging shorter wins and longer losses and reducing move shuffling.

B Lichess Rating System and Rating Ranges

Lichess uses the Glicko-2 rating system (Glickman, 2012; The Lichess team, n.d.a), an extension of the Elo rating system (Elo, 1961) that additionally models rating uncertainty and volatility. In Glicko-2, each player is associated with a rating, a rating deviation measuring uncertainty, and a volatility parameter controlling expected rating fluctuation over time. Rating deviations decrease as more games are played and increase during inactivity.

Although Lichess internally uses Glicko-2, player ratings are still recorded in the public PGN archives using the conventional Elo terminology (The Lichess team, n.d.c). We therefore use the term "Elo" throughout the paper for consistency with prior chess literature and existing datasets.

Lichess maintains separate ratings for each time control category, including Ultrabullet, Bullet, Blitz, Rapid, Classical, and Correspondence chess (team, 2024). New accounts begin at a rating of 1500 (The Lichess team, n.d.a). Historically, the minimum possible rating was 600, but this floor was lowered to 400 in March 2023 (Fiekas, 2023). The theoretical maximum rating is 4000, though observed ratings are substantially lower in practice.

As of early 2026, Lichess ratings span approximately 400–3400 depending on the time control and player population. The rating distribution is highly concentrated in the intermediate range, with median rapid ratings near 1400 and most active players falling roughly between 900 and 1900 (team, n.d.a,n). Extremely high ratings above 2500 are comparatively rare and correspond primarily to titled or professional-level players.

These rating ranges are important for interpreting both the training distribution and the controllability behavior of UniMaia. Because the underlying game data is heavily concentrated in intermediate rating ranges, the model receives substantially more supervision signal for common amateur and club-level play than for extreme low- or high-rating regimes.

C Dataset Construction Details

This appendix provides implementation details for constructing LICHESSGAMES, prompt templates, and opening normalization, complementing the high-level overview in Section 3.

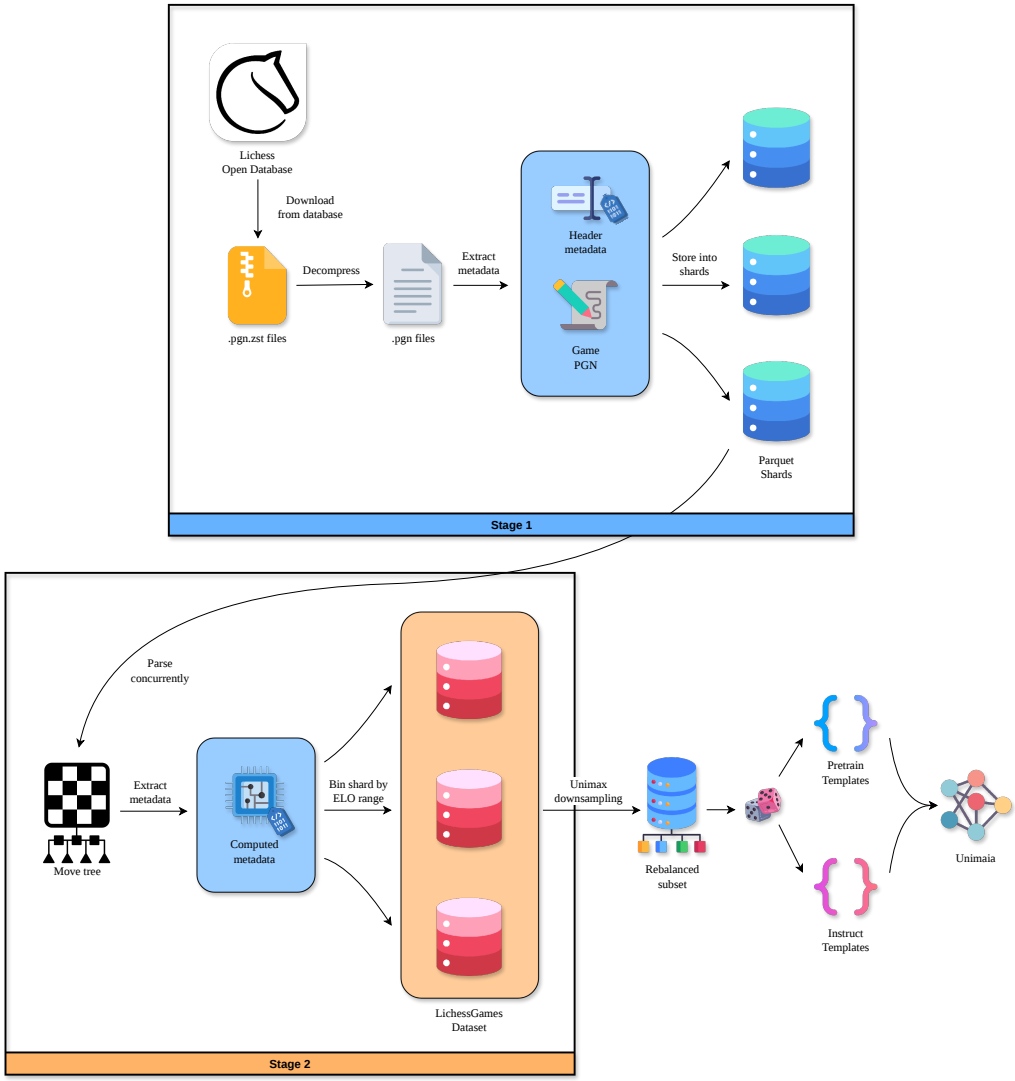

Figure 5: Overview of the data processing pipeline. Stage 1 performs lightweight string-based processing of PGNs into Parquet metadata shards, while Stage 2 parses games into move trees and extracts additional game-level features. During training, games are formatted using randomly sampled prompt templates from the pretraining and instruction-template families.

C.1 Data Sources

Training data consists of publicly available Lichess PGN archives from 2013–2023. Additional auxiliary metadata sources are stored separately and joined dynamically when constructing training examples or prompt-conditioned benchmarks.[1] After preprocessing, the resulting LICHESSGAMES dataset contains approximately 5.2B games and 265.2B positions.

[1]The version of the openings database used in this work contains 3,464 openings, although newer releases contain additional entries.

C.2 Creating LICHESSGAMES

As illustrated in Figure 5, dataset construction proceeds in two stages.

Stage 1: Metadata extraction. The first stage performs lightweight string-only processing of PGNs without parsing move trees. Metadata headers and raw PGN strings are extracted and stored in Parquet shards using `python-chess` (Fiekas, n.d.), specifically with the cheaper `chess.pgn.read_header` routine.

Games from the same month are stored sequentially in a single streaming pipeline with batched Parquet writes. Each shard stores at most 4096 ×

Statistic	Value
Total games	5.185×10^9
Total positions	2.65197×10^{11}
Years covered	2013–2023
Source	Lichess Open Database
License	CC0
ELO range	600–$\approx$ 3400
Templates (pretrain)	160
Templates (instruct)	2048

Table 5: Summary statistics of the LICHESSGAMES dataset.

Dataset	Purpose	Entries	License
Historical Peak Elo and Identity Data for Notable Lichess Users	Player name augmentation	5,793	CC0
Lichess Chess Openings Database	Opening normalization and canonical move sequences	3,464	CC0

Table 6: Auxiliary metadata sources used during training-example and benchmark construction. These sources are stored separately from LICHESSGAMES and joined dynamically as needed.

1024 games to avoid memory issues during later processing.

To support concurrent parsing in Stage 2, byte offsets are tracked for each PGN and the file pointer is rewound after reading headers. Because UTF-8 characters may occupy multiple bytes, offsets must be adjusted accordingly:

```
diff = len(pgn.encode("utf-8")) - len(pgn)
f.seek(prev_offset)
pgn = f.read(offset - prev_offset - diff)
```

Listing 1: Handling UTF-8 byte offsets when recovering PGN strings.

Table 7 summarizes the metadata fields extracted during Stage 1.

Stage 2: Move-tree parsing and partitioning. The second stage fully parses PGNs into move trees using `chess.pgn.read_game`, which is substantially more CPU-intensive. Because PGNs are already separated into arrays during Stage 1, this stage can be parallelized efficiently across workers on a SLURM cluster.

Additional features extracted during this stage are summarized in Table 8.

Field	Description
Event	The event name
TimeControlName	The time control, if provided
UTCDate	The date that the game was played
Year	The year that the game was played
Month	The month that the game was played
Day	The day that the game was played
TcDelay	Base time in seconds, if provided
TcIncrement	Increment in seconds, if provided
WhiteElo	White player Elo
BlackElo	Black player Elo
WhiteRatingDiff	White player rating change
BlackRatingDiff	Black player rating change
Round	Tournament round, if applicable
Result	Game result
Variant	Chess variant
ECO	ECO opening code
Opening	Opening name
Termination	Coarse game termination info
Annotator	PGN annotator
IsFischerRandom	Whether the game is Fischer random chess
EvalDepth	Stockfish evaluation depth, if available
OriginalPGN	Original PGN string

Table 7: Metadata fields extracted during Stage 1.

Field	Description
Length	Game length in halfmoves
TerminationOutcome	Outcome under standard chess rules
TerminationWinner	Winning player
LastComment	Final PGN comment
TerminationReason	Combined termination information
MeanElo	Mean player Elo
DiffElo	Absolute Elo difference

Table 8: Additional features extracted during Stage 2.

Games are partitioned into 100-point Elo bins based on mean player Elo and Elo difference. These partitions are later combined with Unimax sampling (Chung et al., 2023) during training.

C.3 Prompt Template Construction

This section describes implementation details of the prompt-template generation pipeline summarized in Section 3.2.

Prompt templates are initially generated using `gpt-4-turbo-2024-04-09` (Achiam et al., 2023), then converted into parameterized Jinja2 templates by replacing metadata spans with programmatically generated snippets.

Template construction follows an iterative human-in-the-loop workflow:

1. Generate natural-language prompts conditioned on game metadata.

2. Convert metadata spans into parameterized Jinja2 fields.

3. Instantiate templates across many games.

4. Manually review outputs and refine formatting rules.

A major challenge is maintaining correctness under iterative modifications. Because templates combine independently generated metadata snippets, small changes can introduce grammatical or formatting errors across many instantiations.[2]

Consequently, some prompts contain minor artifacts such as duplicated words or awkward phrasing. These artifacts are retained in both training and evaluation for reproducibility and do not appear to materially affect model behavior.

C.3.1 LICHESSTEMPLATES-PRETRAIN

LICHESSTEMPLATES-PRETRAIN contains 160 templates generated from combinations of:

- 5 seed games,
- 4 time controls (bullet, blitz, rapid, classical),
- common versus uncommon openings,
- concise versus regular verbosity.

Two prompts are generated per configuration, yielding:

$$2 \times 5 \times 4 \times 2 \times 2 = 160$$

templates.

C.3.2 LICHESSTEMPLATES-INSTRUCT

LICHESSTEMPLATES-INSTRUCT contains 2,048 templates varying:

- player titles,
- Elo ratings,
- textual skill descriptions,
- usernames,
- real names,
- opening names,
- opening move sequences,
- white versus black perspective,
- request versus descriptive framing.

Combined with common versus uncommon openings and prompt paraphrases, this yields:

$$2^9 \times 2 \times 2 = 2048$$

templates.

C.4 Enriching Player Metadata with Real Names

The player identity mapping data was obtained from a publicly distributed, anonymized file (`famous_lichess.tsv`) originally shared on an online chess community forum. Because the original hosting platform and author are no longer active or identifiable, the dataset has been independently preserved and re-hosted for reproducibility purposes (Unknown, 2026).

C.5 Opening Normalization and Canonical Move Sequences

Opening normalization uses ECO codes and opening names to align historical Lichess annotations with the current Lichess Chess Openings database (The Lichess team, n.d.b).

Unresolved openings are matched using `thefuzz` (SeatGeek, n.d.) and `rapidfuzz` (Bachmann, n.d.), where token-sort ratio performed best empirically.

When automated matching failed, openings were resolved manually by:

- inspecting repository history using `git log -S`,
- searching for the opening sequence externally,
- verifying the resulting sequence using the Lichess analysis board.

At prompt-construction time, the historical opening name is preserved in the text prompt, while the normalized opening entry is used to recover the canonical move sequence.

Because multiple move sequences may correspond to the same opening name, we identify the longest common prefix consistent with the played PGN and use this prefix as the canonical opening continuation.

[2] Substantial effort was invested in designing metadata formatting rules that produce well-formed text across edge cases, including correct pluralization, handling zero versus nonzero increments, and decomposing time controls into hours, minutes, and seconds. However, as the template space expanded, the combinatorial growth in template instantiations made exhaustive validation increasingly difficult.

D Termination Statistics

This appendix reports the frequencies of termination reasons for representative dataset shards used in the analysis of repetition-related outcomes discussed in Appendix A. We include shards from March 2014, October 2022, and January 2023 to demonstrate that five-fold repetition remains consistently rare across time, including both early data and data collected after changes to Lichess draw-claim behavior. Specifically, Lichess introduced automatic draw claims for threefold repetition under low time conditions in April 2014 and later made automatic claiming the default in November 2022 (Duplessis, 2014, 2022).

Termination Reason	Counts	Percentage (%)
Resigned	266,449	37.425
Time forfeit	231,410	32.503
Checkmate	193,718	27.209
Threefold repetition	5,615	0.789
Insufficient material	5,098	0.716
Stalemate	4,868	0.684
Draw by agreement	3,966	0.557
Fivefold repetition	640	0.090
Fifty moves	196	0.028
Total	711,960	100.000

Table 9: Termination reason frequencies for the only shard of March 2014 games in LICHESSGAMES, after applying the filtering criteria described in Appendix F.4.

Termination Reason	Counts	Percentage (%)
Resigned	1,326,763	34.911
Time forfeit	1,288,942	33.915
Checkmate	1,061,309	27.926
Threefold repetition	50,813	1.337
Stalemate	31,683	0.834
Insufficient material	26,885	0.707
Draw by agreement	9,675	0.255
Fivefold repetition	3,576	0.094
Fifty moves	735	0.019
Abandoned	50	0.001
Rules infraction	27	0.001
Total	3,800,458	100.000

Table 10: Termination reason frequencies for the first shard of October 2022 games in LICHESSGAMES, after applying the filtering criteria described in Appendix F.4.

E Additional Design Evolution Details

Table 12 summarizes the parameter changes introduced by each sequential design modification from Section 4.3. Architectural scaling contributes the majority of the parameter increase.

Termination Reason	Counts	Percentage (%)
Resigned	1,321,464	34.705
Time forfeit	1,294,684	34.002
Checkmate	1,064,219	27.949
Threefold repetition	59,097	1.552
Stalemate	30,763	0.808
Insufficient material	26,698	0.701
Draw by agreement	9,385	0.246
Fivefold repetition	731	0.019
Fifty moves	626	0.016
Rules infraction	34	0.001
Abandoned	2	0.000
Total	3,807,703	100.000

Table 11: Termination reason frequencies for the first shard of January 2023 games in LICHESSGAMES, after applying the filtering criteria described in Appendix F.4.

Change	**Trainable LLM Parameters**	**Lc0 Parameters**
ControlNet + policy head	–	+124.486M
NorMuon optimizer	–	–
LoRA vs DoRA	–0.328M	–
BT4 backbone	–	+279.038M
Unimax sampling	–	–
+ LoRA rank 16	+3.932M	–
Full configuration	**+3.604M**	**+403.524M**

Table 12: Parameter changes associated with each design modification. The BT4 backbone accounts for most of the parameter increase (+279M of +404M total), despite comparatively modest gains in development score.

Figure 6 compares the training losses of the initial and final configurations. Although most modifications add little or no parameter overhead, their combined effect substantially improves optimization and final loss.

Overall, most improvements are parameter-efficient, while later scaling decisions contribute the majority of the parameter growth for comparatively smaller gains.

F Ablation Study

This appendix provides additional details on the ablation studies used during UniMaia development. The main paper summarizes the sequential design changes leading to the final model; here, we describe the shared experimental setup and detailed ablation results. Full-scale training dynamics are discussed separately in Appendix H.

All ablations are trained on Lichess games from January–December 2013. This subset preserves realistic game distributions while enabling rapid iteration over architecture, optimization, parameter-

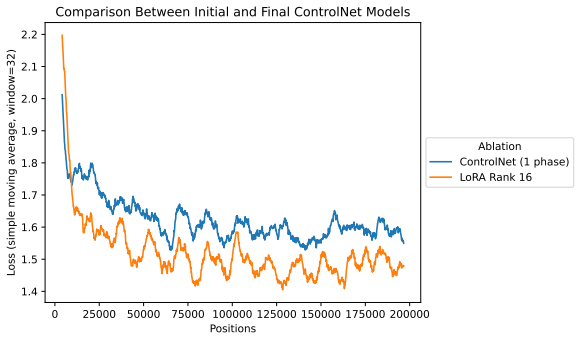

Figure 6: Comparison of training losses between the initial single-phase, layer-coupled ControlNet and the final model.

efficient fine-tuning, backbone scaling, and data-sampling strategies. Unless otherwise specified, all ablations use the same preprocessing pipeline, prompt construction procedure, optimizer settings, and evaluation protocol.

F.1 Training Data and Procedure

Ablations follow the same monthly partitioning scheme as the full training run, restricted to the 2013 subset. Since globally shuffling the dataset is impractical, each monthly partition is divided into independently shuffled shards processed sequentially. Although this may introduce mild temporal drift, it enables efficient checkpointing and incremental training.

Within each shard, games are grouped by mean player Elo and absolute Elo difference. Unimax downsampling (Chung et al., 2023) is applied across Elo buckets to maintain coverage across skill levels while enforcing a fixed training budget. Additional data-sampling ablations (Appendix F.9) are applied within each Elo bucket to isolate the effects of balancing by opening, time control, or game length.

Each ablation trains on 2^{14} plies per shard, totaling 196,608 training plies across 2013. Since the subset contains one shard per month, training proceeds sequentially from January through December. To reduce prompt-template overfitting, prompts are lowercased with probability 0.1 and stripped of diacritics with probability 0.5.

To ensure hardware stability during long training runs, we insert a cooldown period after processing each month to prevent GPU thermal throttling and system crashes.

F.2 Optimization

Unless otherwise specified, ablations use the same optimization hyperparameters as the corresponding compared configuration. We use weight decay 0.01 and $\epsilon = 1 \times 10^{-8}$. Learning rates follow the development schedule: linear warmup to 5×10^{-5}, followed by staged monthly decay. Optimizer-specific changes, including the transition from AdamW (Loshchilov and Hutter, 2017) to NorMuon, are evaluated separately in Appendix F.6.

The ablations are designed for rapid iteration rather than full-scale training. In practice, the 2013-only runs complete in several hours on modern consumer GPUs such as the RTX 3090 or RTX 5090, depending on batch size and memory constraints.

F.3 Evaluation Protocol

Ablations are compared using both training loss and downstream benchmark performance. Since per-step loss curves are noisy, we summarize the training loss using the mean loss over the December 2013 partition, denoted $\mu_{\text{loss, Dec13}}$.

For downstream evaluation, we use a compact diagnostic benchmark suite:

- **LOB-P**: opening-following under partial opening information;
- **LIF-D**: diagnostic instruction-following over a restricted Elo range;
- **M1-S**: human move prediction on a stratified subset of the Maia-1 benchmark.

LIF-D is used only during development because it partially overlaps with the training distribution. Final evaluation instead uses the full Lichess Instruction-Following Benchmark described in Section 5. As described in the main paper, we summarize ablation performance using the development score: the mean normalized accuracy across these diagnostic tasks.

F.4 Initial Baseline Configuration

The initial ablation baseline is an early-fusion model with a single cross-attention conditioning layer. It combines prompt embeddings from ChessGPT-base (Feng et al., 2024) with a frozen Lc0-CF policy backbone (Monroe and Chalmers, 2024; Monroe et al., 2026) initialized from the `t3` checkpoint.

Unless otherwise stated, ablations exclude bot games, Lichess Master games[3], correspondence games, games missing Elo or time-control metadata, and games shorter than 32 plies unless they end in checkmate or forced draw. We retain the standard Lc0 input representation, including its repetition handling, since five-fold repetition is extremely rare in the dataset.

The following sections evaluate the main design choices explored during development. Each ablation isolates one factor where possible, while the sequential design table in the main paper should be interpreted as a development trace rather than a fully factorial study.

F.5 Architecture Design

We study how conditioning effectiveness depends on both the frequency of cross-modal interaction and the integration of prompt signals into the frozen chess policy. We compare early fusion, deep fusion, and ControlNet-style conditioning.

F.5.1 Fusion-Based Conditioning

We first evaluate two standard fusion approaches that inject prompt information directly into the Lc0 encoder.

Single-step cross-attention (early fusion). Inspired by early-fusion multimodal models such as LLaVA (Liu et al., 2023), this variant performs a single cross-attention operation between the prompt embeddings and the shallow board representation before the Transformer stack. The fused representation is then processed by all encoder layers and output heads without further prompt interaction.

For training stability, we adopt a three-stage procedure:

(i) both the text encoder and Lc0 encoder are frozen,

(ii) the text encoder is adapted using DoRA (Liu et al., 2024),

(iii) all non-policy components are frozen while the policy outputs are fine-tuned.

Each stage is trained on disjoint month ranges (Jan–Mar, Apr–Jun, Jul–Dec).

[3]Lichess Master was a rare discretionary title formerly awarded by Lichess (The Lichess team, n.d.d); all other titles are officially awarded by FIDE.

Per-layer cross-attention (deep fusion). This variant inserts cross-attention blocks into every Transformer layer, enabling iterative prompt-conditioned refinement of the board representation. All cross-attention output projections are zero-initialized for stability, while the pretrained Lc0 parameters remain frozen.

F.5.2 ControlNet-Based Conditioning

We next evaluate ControlNet-style conditioning (Zhang et al., 2023), which introduces a parallel trainable pathway alongside the frozen backbone.

Layer-coupled ControlNet. A trainable copy of the encoder is added in parallel to the frozen Lc0 encoder. After each layer, the ControlNet activations are projected and injected residually into the corresponding backbone layer. Cross-attention modules inserted between the self-attention and feed-forward blocks enable prompt-conditioned modulation of the hidden board representation.

Following the PixArt-δ ControlNet design (Chen et al., 2024), both cross-attention outputs and residual injections are zero-initialized for stability.

Unlike the original sequential ControlNet formulation, both the base layer and the ControlNet layer at depth n receive the base hidden state from depth $n-1$. We refer to this as the *layer-coupled* design.

We evaluate:

- a two-stage variant, where the text encoder is initially frozen and later adapted with DoRA, and
- a single-stage variant, where the text encoder is adapted throughout training.

Controllable policy head. To enable direct modulation of the output distribution, we additionally evaluate a controllable policy head (CPH). A pre-normalized cross-attention block is inserted between the cloned policy embedding and the cloned policy head. Given a policy embedding x, cross-attention over the prompt embeddings produces an update y, which is added residually before the cloned policy head:

$$x \leftarrow x + y.$$

This allows prompt information to directly influence the policy logits.

Gated attention. We further evaluate gated attention (Qiu et al., 2025) within the controllable policy head to mitigate attention sinks observed during preliminary experiments.

Sequential ControlNet. Finally, we evaluate a sequential ControlNet variant closer to the original ControlNet connectivity pattern, where each ControlNet layer consumes the previous ControlNet hidden state rather than the base hidden state.

F.5.3 Training Regime

Unless otherwise specified, all architecture ablations freeze the pretrained Lc0 backbone and train only the newly introduced conditioning components. This setting was initially chosen for stability, as early experiments with full joint optimization produced large loss spikes and unstable convergence.

Although later experiments partially mitigated these issues, we retain the frozen-backbone regime throughout the primary architecture study to isolate the effect of the conditioning mechanisms themselves.

F.5.4 Results

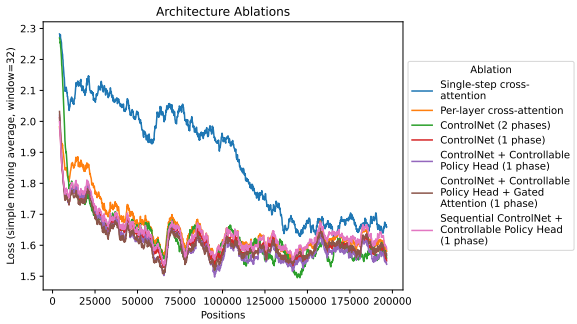

Figure 7: Simple moving averages of the training losses for the architecture ablations. Due to the small batch size (8), the raw losses are highly volatile.

Figure 7 shows the training losses for the architecture ablations. Fusion-based approaches consistently exhibit higher losses than ControlNet variants, particularly during early training.

Table 13 summarizes downstream benchmark performance. Overall, ControlNet variants substantially outperform both early and deep fusion in mean benchmark score and training loss.

While deep fusion achieves the strongest opening-following performance on LOB-P, it performs substantially worse on M1-S, indicating weaker alignment with human move distributions. In contrast, ControlNet variants perform consistently well across benchmarks, suggesting that structured residual conditioning is more effective than repeated cross-attention alone in the frozen-backbone regime.

Adding a controllable policy head further improves both benchmark performance and loss, supporting the hypothesis that direct policy modulation is beneficial. Gated attention slightly improves M1-S performance, but reduces instruction-following accuracy overall. The sequential ControlNet variant also underperforms the layer-coupled formulation and exhibits higher training loss.

Overall, the layer-coupled ControlNet with a controllable policy head achieves the best trade-off between convergence and downstream performance, and is therefore selected as the baseline architecture for subsequent experiments.

F.5.5 Full Fine-Tuning (Retrospective Study)

We additionally conduct a retrospective study in which selected fusion architectures are trained with full joint optimization rather than a frozen backbone.

Under full fine-tuning, both early and deep fusion improve substantially, particularly on M1-S. Deep fusion with a controllable policy head achieves the strongest overall benchmark performance.

These results are not directly comparable to the primary architecture study, which assumes a frozen backbone. Instead, they suggest that the relative effectiveness of conditioning strategies depends strongly on whether the pretrained chess policy is fixed or jointly optimized.

Finally, fusion-based methods remain substantially more parameter-efficient than ControlNet, which approximately doubles the number of trainable Lc0 parameters, highlighting a trade-off between parameter efficiency and conditioning modularity.

F.6 Optimization Strategy

We next evaluate whether alternative optimizers improve convergence and downstream performance. Unless otherwise specified, all optimizer ablations use the same architecture, training data, and hyperparameters.

We compare three optimizers:

1. **AdamW:** AdamW (Loshchilov and Hutter, 2017), used in our initial experiments.

2. **Muon:** Muon (Jordan et al., 2024) orthogonalizes updates for 2D non-embedding and non-output parameters. Following prior work (Jordan et al., 2024; Liu et al., 2025), we use AdamW for all remaining parameters.

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
Single-step cross-attention (early fusion)	0.4696	0.4176	0.4315	0.4396	1.6685
Per-layer cross-attention (deep fusion)	**0.5745**	0.4325	0.3379	0.4483	1.6092
ControlNet (2 phases)	0.5384	0.4332	0.5118	0.4944	1.5753
ControlNet	0.5469	0.4351	0.5122	0.4981	1.5865
ControlNet + controllable policy head	0.5704	**0.4358**	0.5179	**0.5080**	**1.5680**
ControlNet + CPH + gated attention	0.5059	0.4253	**0.5228**	0.4847	1.5942
Sequential ControlNet + CPH	0.5653	0.4311	0.5116	0.5027	1.6223

Table 13: Architecture ablation results. Best values are shown in bold and second-best values are underlined.

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
Early fusion (frozen)	0.4696	0.4176	0.4315	0.4396	1.6685
Early fusion (full FT)	0.5393	0.4294	0.5094	0.4927	1.6329
Deep fusion (frozen)	0.5745	0.4325	0.3379	0.4483	**1.6092**
Deep fusion (full FT)	0.5742	**0.4400**	0.5115	0.5086	1.6205
Deep fusion + CPH (full FT)	**0.5822**	0.4389	**0.5122**	**0.5111**	1.6189

Table 14: Full fine-tuning results for selected architectures. Best values are shown in bold and second-best values are underlined.

3. **NorMuon:** NorMuon (Li et al., 2025) extends Muon with neuron-wise normalization using second-order momentum statistics.

Figure 8 shows the training losses. Although Muon and NorMuon initially converge more slowly, both quickly outperform AdamW and maintain lower losses throughout training.

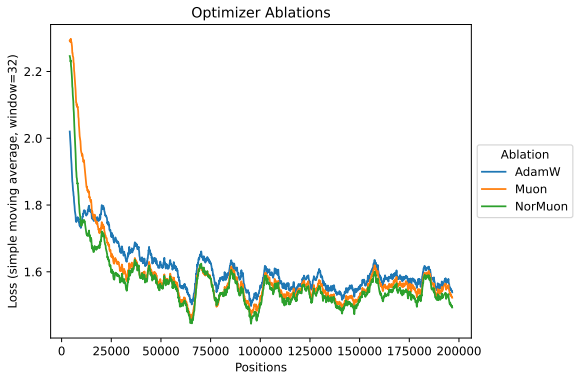

Figure 8: Simple moving averages of the training losses for each optimizer.

These improvements also transfer to downstream performance (Table 15). Muon improves opening-following performance and training loss over AdamW, while NorMuon achieves the best overall results. Based on these results, we use NorMuon in all subsequent experiments.

F.7 Parameter-Efficient Fine-Tuning

We next evaluate parameter-efficient fine-tuning (PEFT) strategies for adapting the text encoder. The baseline uses DoRA with rsLoRA and LoRA+, motivated by prior work showing improved adaptation with minimal additional overhead.

We compare:

1. **DoRA:** Decomposes low-rank updates into magnitude and direction components (Liu et al., 2024). Unless otherwise specified, DoRA additionally uses rsLoRA and LoRA+.

2. **LoRA:** Standard LoRA adapters (Hu et al., 2022) without DoRA magnitude decomposition.

3. **No rsLoRA:** Removes rsLoRA scaling (Kalajdzievski, 2023).

4. **No LoRA+:** Removes LoRA+ (Hayou et al., 2024), using identical learning rates for the LoRA matrices A and B instead of the default ratio $\frac{\eta_B}{\eta_A} = 2$.

5. **Frozen text encoder:** Fully freezes the text encoder while retaining the remaining ControlNet conditioning architecture.

Figure 9 shows that all LoRA-based variants converge similarly. Although the frozen text encoder reaches a comparable loss, it consistently underperforms on downstream benchmarks, indicating that adapting the language representation remains beneficial even with a frozen chess backbone.

Among trainable variants, standard LoRA slightly outperforms DoRA in both mean benchmark score and final loss (Table 16) while using fewer trainable parameters (Table 17), suggesting

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
AdamW*	0.5704	0.4358	0.5179	0.5080	1.5680
Muon	0.5906	0.4346	0.5171	0.5141	1.5525
NorMuon	**0.5919**	**0.4432**	**0.5219**	**0.5190**	**1.5331**

Table 15: Optimizer ablation results. Best values are in bold and second-best values are underlined.

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
DoRA*	0.5919	0.4432	0.5219	0.5190	1.5331
LoRA	**0.6015**	0.4408	0.5196	**0.5206**	**1.5282**
No rsLoRA	0.5799	0.4441	0.5200	0.5147	1.5333
No LoRA+	0.5781	**0.4471**	**0.5246**	0.5166	1.5401
Frozen text encoder	0.5545	0.4350	0.5145	0.5013	1.5501

Table 16: PEFT ablation results. Best values are in bold and second-best values are underlined.

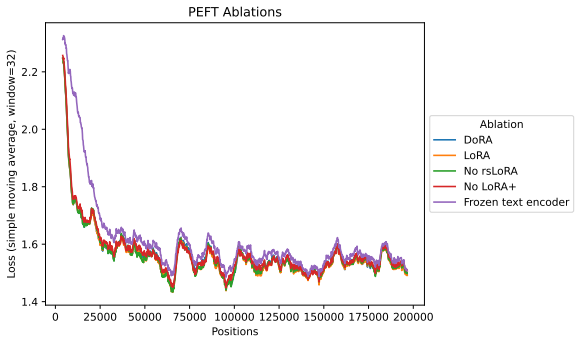

Figure 9: Simple moving averages of the training losses for the PEFT ablations.

Experiment	Trainable LLM Parameters
DoRA*	4.260M
LoRA	3.932M
Frozen text encoder	0

Table 17: Trainable text-encoder parameter counts for the PEFT variants.

limited benefit from DoRA's additional magnitude parameters.

Removing rsLoRA or LoRA+ slightly reduces overall performance, though the effect is small. Since both introduce negligible overhead while modestly improving stability and accuracy, we retain them in the final configuration. Overall, these results suggest that lightweight low-rank adaptation is sufficient for conditioning the frozen chess policy, with more complex PEFT variants providing diminishing returns.

F.8 Lc0 Scaling

We next investigate whether scaling the frozen Lc0-ChessFormer backbone improves downstream controllability and human move prediction performance. Since the ControlNet branch mirrors the backbone architecture, increasing the size of the underlying Lc0 model approximately doubles the additional parameters introduced by the conditioning network. Consequently, backbone scaling substantially increases total model size and memory requirements.

We compare three pretrained Lc0-CF checkpoints from Monroe and Chalmers (2024); Monroe et al. (2026). All models use 15 Transformer layers with 32-dimensional attention heads, while larger variants increase the number of attention heads and embedding dimensions. All checkpoints additionally use stochastic weight averaging (Izmailov et al., 2018).

1. **t3:** A 0.10B-parameter model initialized from `t3-512x15x16h-swa-2815000`.
2. **BT3:** A 0.12B-parameter model initialized from `BT3-768x15x24h-swa-2790000`.
3. **BT4:** A 0.21B-parameter model initialized from `BT4-1024x15x32h-swa-5000000`.

Increasing the backbone width generally improves performance. While BT3 achieves the highest mean benchmark score in this ablation, BT4 attains the lowest loss and remains competitive across all benchmarks.

We note that BT3 also outperforms BT4 in the standalone Lc0 evaluation (see Appendix K), particularly on human move matching. This likely contributes to its stronger benchmark performance in the ControlNet setting.

Nevertheless, we select BT4 as the baseline for subsequent experiments, prioritizing lower loss and higher capacity despite a small drop in benchmark

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
Lc0-CF-t3 (0.10B)*	0.6015	0.4408	0.5196	0.5206	1.5282
Lc0-CF-BT3 (0.12B)	**0.6154**	0.4432	**0.5249**	**0.5278**	1.5136
Lc0-CF-BT4 (0.21B)	0.6071	**0.4449**	0.5244	0.5255	**1.5063**

Table 18: Lc0 backbone scaling ablation results. Best values are shown in bold and second-best values are underlined.

performance. We do not further disentangle the effect of backbone quality versus ControlNet adaptation and leave this to future work.

F.9 Data Sampling

We use Unimax downsampling (Chung et al., 2023) to allocate a fixed training budget of plies across the dataset while preventing large groups from dominating the sample distribution. Unimax combines approximately uniform allocation with a cap on the number of samples drawn from each source; unless otherwise specified, we use a maximum of one epoch per group, allowing smaller groups to be fully utilized while truncating larger groups to satisfy the global budget. In all experiments, Unimax is first applied across Elo buckets. This ablation studies how to perform additional downsampling *within* each Elo bucket.

We consider two independent design choices:

1. **Group-level balancing**: whether to partition games into groups (e.g., by opening or time control) and allocate separate budgets to each group using Unimax.

2. **Per-game budget allocation**: whether to allocate plies proportionally to game length or using Unimax, which caps the contribution of long games.

In all settings, plies are sampled uniformly within each game after budgets are assigned. Unless otherwise specified, the baseline uses Unimax grouped by opening.

Table 19 shows that balancing across time controls consistently outperforms balancing across openings. This suggests that improving coverage across time controls improves generalization, whereas enforcing a uniform opening distribution may distort the naturally skewed opening frequencies observed in Lichess. Although proportional allocation occasionally improves individual metrics, Unimax-based per-game allocation yields better overall benchmark performance. Consequently, we adopt Unimax (time control) for subsequent experiments.

F.10 LoRA Rank Ablations

We evaluate LoRA ranks 2^k for $k \in [0, 7]$ while keeping the LoRA scaling factor (α) fixed.

Table 20 shows that benchmark performance peaks at rank 16, with rank 8 performing similarly. In contrast, training loss does not vary monotonically with rank and is only weakly correlated with downstream performance.

Higher LoRA ranks increase the number of trainable parameters, but yield diminishing returns beyond moderate ranks. Combined with the PEFT ablations in Table 16, these results suggest that most gains arise from introducing low-rank adaptation itself, while increasing the adaptation capacity beyond a moderate rank provides comparatively small improvements.

F.11 Learning Rate Schedule Ablations

We compare our custom learning rate schedule against simplified Warmup–Stable–Decay (WSD-S) schedules proposed by Wen et al. (Wen et al., 2024). Since the ablations are restricted to 2013 data, the custom schedule behaves similarly to a cosine-like decay.

We evaluate two WSD-S variants:

1. **WSD-S (monthly):** warmup and decay are applied independently each month;

2. **WSD-S (annual):** a single warmup–stable–decay cycle is applied across all of 2013.

Because the original schedule spends nearly all of January 2013 in warmup, we shorten the warmup phase to 512 steps and allocate an additional 512 decay steps, leaving 1,024 stable-training steps.

As shown in Figure 11, the WSD-S schedules reduce loss more rapidly early in training due to their shorter warmup phases. However, the custom schedule achieves lower final loss and better benchmark performance overall (Table 21), particularly after the WSD-S variants enter their decay phase.

Although the custom schedule performs best overall, the ablation suggests that the original

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
Proportional (global)	0.6357	0.4469	0.5312	0.5380	1.5500
Proportional (opening)	0.6154	0.4492	0.5319	0.5321	1.5651
Proportional (opening + time control)	0.6384	0.4503	0.5260	0.5382	1.6006
Proportional (time control)	0.6503	**0.4503**	0.5312	0.5440	1.5192
Unimax (global)	0.6228	0.4472	**0.5322**	0.5341	**1.4716**
Unimax (opening)*	0.6071	0.4449	0.5244	0.5255	1.5063
Unimax (opening + time control)	0.6182	0.4494	0.5272	0.5316	1.5562
Unimax (time control)	**0.6580**	0.4450	0.5320	**0.5450**	1.4768

Table 19: Downsampling ablation results. Best values are in bold and second-best values are underlined.

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
LoRA rank = 1	0.6494	0.4433	0.5264	0.5397	1.4863
LoRA rank = 2	0.6340	0.4425	0.5299	0.5355	1.4874
LoRA rank = 4	0.6504	0.4442	0.5300	0.5415	1.4866
LoRA rank = 8*	0.6559	**0.4457**	**0.5304**	0.5440	1.4863
LoRA rank = 16	**0.6625**	0.4448	0.5297	**0.5457**	1.4874
LoRA rank = 32	0.6540	0.4453	0.5300	0.5431	1.4892
LoRA rank = 64	0.6545	0.4428	0.5299	0.5424	**1.4832**
LoRA rank = 128	0.6467	0.4429	0.5247	0.5381	1.4843

Table 20: LoRA rank ablation results. Best values are in bold and second-best values are underlined.

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
Custom LR schedule*	**0.6580**	**0.4450**	**0.5320**	**0.5450**	**1.4768**
WSD-S (monthly)	0.6019	0.4407	0.5222	0.5216	1.5014
WSD-S (annual)	0.6389	0.4434	0.5226	0.5350	1.4845

Table 21: Learning rate schedule ablation results. Best values are in bold and second-best values are underlined.

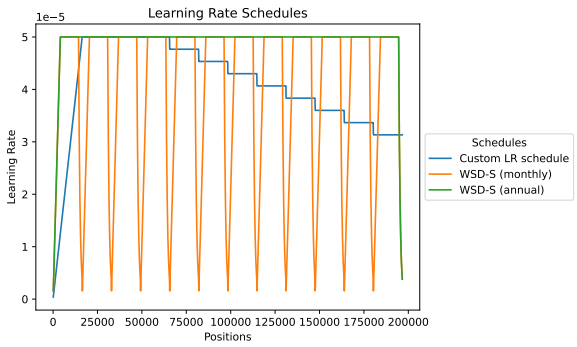

Figure 10: Learning rate schedules evaluated in the ablation study.

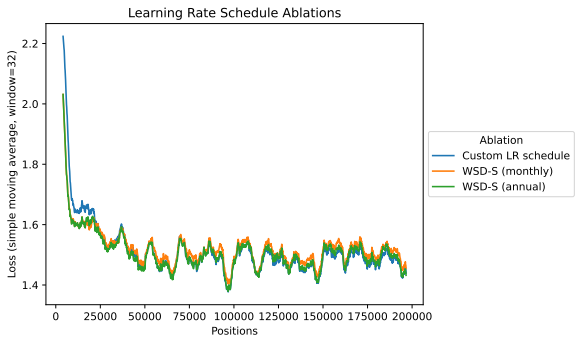

Figure 11: Simple moving averages of the training losses for the learning rate schedule ablations. The WSD-S schedules converge faster initially due to their shorter warmup, but the custom schedule catches up during the decay phase.

warmup duration is unnecessarily long. Consequently, we reduce the warmup length from 2,048 to 1,024 steps in the full training run.

F.12 Negative Results

This section summarizes exploratory ablations that did not affect the final UniMaia configuration, but nevertheless provide insight into training behavior and data composition effects.

F.12.1 Template Mixture Ablations

We evaluate different sampling mixtures between LICHESSTEMPLATES-PRETRAIN and LICHESSTEMPLATES-INSTRUCT:

1. **50/50 mixture**: sample from both template families with equal probability;
2. **Pretrain only**: sample exclusively from

LICHESSTEMPLATES-PRETRAIN;

3. **Instruct only**: sample exclusively from LICHESSTEMPLATES-INSTRUCT.

Using only instruct templates yields the strongest average benchmark performance, while the 50/50 mixture achieves comparable downstream accuracy with lower training loss. We therefore adopt the mixed-template configuration in the final model.

One possible explanation is that the instruct template pool (2,048 templates) is substantially larger than the pretrain pool (160 templates), making the pretrain-only setting more susceptible to overfitting. Nevertheless, we retain the mixed setting because the pretrain templates contain richer structured metadata, which may provide a stronger semantic grounding signal during training.

We also experiment with restricting training to rapid games and omitting the first 10 plies, following Maia-2 Rapid (Tang et al., 2024). Although this substantially lowers training loss, it produces only marginal gains on rapid-only human move prediction benchmarks. Since UniMaia is intended as a unified prompt-conditioned model across time controls, we do not pursue this direction further. This result is also consistent with prior work such as Allie (Zhang et al., 2024b), which shows that unified multi-time-control models can match or outperform specialized rapid-only systems.

F.12.2 Text Model

We evaluate several text encoders differing in architecture family, parameter scale, and training data composition. Specifically, we compare pretrained and chess-specialized models from the GPT-NeoX (Black et al., 2022), Gemma (Team et al., 2025a), and Qwen (Zhang et al., 2025b) families, including autoregressive, encoder-decoder, and embedding-focused variants.

Within the GPT-NeoX family, we compare RedPajama INCITE Base (3B) (Weber et al., 2024), ChessGPT-base, and ChessGPT-chat (Feng et al., 2024). We additionally evaluate Gemma 3 (1B), T5 Gemma 2 encoder variants (Tay et al., 2022; Zhang et al., 2025a), Embedding Gemma (Kusupati et al., 2022; Vera et al., 2025), and Qwen 3 Embedding (0.6B).

Model scale alone does not consistently improve performance. For example, the larger T5 Gemma 2 encoder underperforms its smaller counterpart. Instead, training data composition appears more important: models trained on both chess and natural language data consistently outperform general-purpose pretrained models. Embedding-oriented models also perform competitively despite their smaller size.

Overall, ChessGPT-base achieves the best average performance and lowest training loss, suggesting that domain-specialized pretraining is more beneficial than increasing model scale alone. We therefore retain ChessGPT-base for all subsequent experiments.

F.12.3 Fine-tuning ChessGPT-base

We investigate whether fine-tuning the language model prior to ControlNet training improves downstream instruction-following performance.

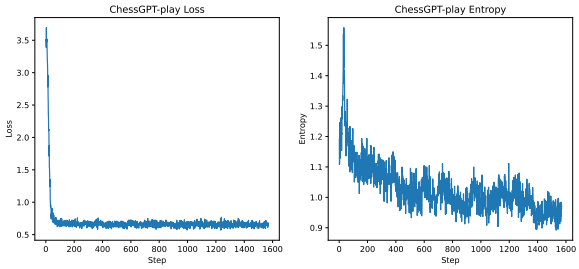

Figure 12: Training loss (left) and entropy (right) during ChessGPT-play training. While loss plateaus early, entropy continues to decrease, indicating increasingly confident predictions.

We evaluate four variants:

1. **ChessGPT-play:** ChessGPT-base fine-tuned directly for next-move prediction on the 2013 split using the template `{prompt}<|endoftext|>{pgn}`.

2. **ControlNet (ChessGPT-base):** The standard ControlNet configuration using the pretrained ChessGPT-base encoder.

3. **ControlNet (ChessGPT-play):** The same ControlNet configuration, but replacing the text encoder with ChessGPT-play.

4. **ControlNet (ChessGPT-play), end-of-text PGNs:** A variant using the explicit end-of-text separator `{prompt}<|endoftext|>{pgn}` during ControlNet training to reduce distribution shift relative to ChessGPT-play pretraining.

ChessGPT-play converges rapidly within the first $\sim$ 50 steps (Figure 12). Interestingly, prediction

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
50/50 pretrain, instruct templates*	0.5704	0.4358	0.5179	0.5118	1.5680
Pretrain templates	0.4401	**0.4422**	0.5104	0.4794	**1.5507**
Instruct templates	**0.5819**	0.4402	**0.5221**	**0.5173**	1.5842

Table 22: Template mixture ablation results. Best values are in bold and second-best values are underlined.

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
Gemma 3 (1B)	0.4817	0.4275	0.5049	0.4714	1.6049
T5Gemma 2 Encoder (1B)	0.4555	0.4287	0.5056	0.4632	1.6111
T5Gemma 2 Encoder (0.27B)	0.5168	0.4329	0.5161	0.4886	1.5873
Embedding Gemma (0.30B)	0.5273	0.4381	**0.5212**	0.4955	1.5512
Qwen 3 Embedding (0.60B)	0.5772†	**0.4441**†	0.5207†	0.5140†	1.5733†
Redpajama Incite Base (3B)	0.5108	0.4341	0.5100	0.4850	1.5916
ChessGPT-base v1 (3B)*	**0.6015**	0.4408	0.5196	**0.5206**	**1.5282**
ChessGPT-chat v1 (3B)	0.5778	0.4433	0.5147	0.5119	1.5617

Table 23: LLM ablation results. Best values are in bold and second-best values are underlined. †The corresponding LoRA model produced NaN losses, so benchmark scores and loss are instead reported using DoRA.

entropy continues decreasing after the loss plateaus, suggesting increasing confidence without corresponding improvements in accuracy.

Although ChessGPT-play performs strongly on the opening-following benchmark, it substantially underperforms on LIF-D and M1-S. Replacing the pretrained ChessGPT-base encoder with ChessGPT-play similarly degrades overall ControlNet performance, suggesting that specialization toward direct move prediction harms general instruction-following behavior.

To further evaluate opening-following performance, we additionally test the models on canonical opening reconstruction benchmarks.

As shown in Table 25, the strong opening-following performance of ChessGPT-play does not transfer when the model is used as the ControlNet text encoder. Adding an explicit end-of-text separator slightly improves performance, but remains below the original ChessGPT-base configuration. Interestingly, inference performance for the end-of-text variant remains nearly unchanged even when the separator and PGN are omitted, although we do not report these results.

F.12.4 LoRA Target Modules Ablations

We vary the set of modules targeted by LoRA in the text encoder:

1. **QKV projections:** Apply LoRA only to the query, key, and value projections.

2. **QKV + O projections:** Additionally adapt the output projection.

3. **QKV + O + FFN:** Further extend LoRA to the feedforward layers.

Table 26 shows that all configurations perform similarly. Targeting both QKV and O projections achieves the highest mean score, while adding FFN adapters slightly reduces downstream performance despite a marginally lower loss. The differences are small and likely within the variance induced by random initialization. Overall, expanding LoRA beyond attention projections provides limited benefit in this setting and increases parameter count with diminishing returns.

F.12.5 LoRA Rank with Scaled Alpha Ablations

We additionally scale the LoRA scaling factor with rank while keeping the same rsLoRA multiplier used in the rank-16 setting from Appendix F.10. Specifically, we evaluate ranks 2^k with $\alpha = 4 \cdot 2^{k/2}$ for $k \in [2, 6]$.

Table 27 shows that performance again peaks at moderate ranks, with rank 16 achieving the highest mean score. Increasing the rank beyond this point provides only marginal or inconsistent improvements despite the additional parameters.

Compared to the fixed-α setting (Table 20), scaling α with rank produces a similar trend, suggesting that the effective capacity of the adapters saturates in this regime. Although scaling α stabilizes training across ranks, it does not substantially change the optimal rank, reinforcing that moderate-rank adapters provide the best trade-off between performance and parameter efficiency.

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
ChessGPT-play	**0.8080**	0.3975	0.4343	**0.5466**	**0.6530**
ControlNet (ChessGPT-base)*	0.6625	0.4448	0.5297	0.5457	1.4874
ControlNet (ChessGPT-play)	0.6355	0.4445	**0.5314**	0.5371	1.4975
ControlNet (ChessGPT-play), end-of-text PGN	0.6494	**0.4453**	0.5281	0.5409	1.4872

Table 24: Fine-tuned base model ablation results. Best values are in bold and second-best values are underlined.

Experiment	LOB-P	LOB-C	LOB-P†	LOB-C†	Mean
ChessGPT-play	**0.8080**	**0.8799**	**0.8305**	**0.8327**	**0.8378**
ControlNet (ChessGPT-base)*	0.6625	0.6844	0.6569	0.6608	0.6661
ControlNet (ChessGPT-play)	0.6355	0.6523	0.6119	0.6151	0.6287
ControlNet (ChessGPT-play), end-of-text PGN	0.6494	0.6715	0.6388	0.6420	0.6504

Table 25: Fine-tuned base model ablations evaluated on opening instruction-following benchmarks. Best values are in bold and second-best values are underlined. †Prompted with the PGN header.

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
LoRA target = W_{QKV}	**0.6455**	**0.4445**	0.5260	0.5387	1.4867
LoRA target = W_{QKV}, W_O*	0.6443	0.4426	**0.5315**	**0.5394**	1.4871
LoRA target = W_{QKV}, W_O, FFN	0.6407	0.4419	0.5288	0.5371	**1.4855**

Table 26: LoRA target modules ablation results. Best values are in bold and second-best values are underlined.

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
LoRA rank = 4, $\alpha = 8$	0.6511	0.4438	0.5272	0.5407	**1.4866**
LoRA rank = 8, $\alpha = 8\sqrt{2}$	0.6441	0.4427	0.5288	0.5385	1.4908
LoRA rank = 16, $\alpha = 16$*	**0.6625**	0.4448	0.5297	**0.5457**	1.4874
LoRA rank = 32, $\alpha = 16\sqrt{2}$	0.6546	**0.4456**	**0.5338**	0.5447	1.4870
LoRA rank = 64, $\alpha = 32$	0.6468	0.4406	0.5327	0.5400	1.4868

Table 27: LoRA rank with scaled α ablation results. Best values are in bold and second-best values are underlined.

F.12.6 NorMuon Hyperparameter Ablations

We additionally evaluate Nesterov momentum (Dozat, 2016), but observe slightly worse performance. Replacing the Newton–Schulz iteration with the Polar Express algorithm (Amsel et al., 2025) for approximating polar$(\cdot)$ occasionally introduces gradient spikes in the LoRA adapters, slightly increasing loss and making NaNs more likely in some execution environments. Consistent with observations from the Kimi Team (Team et al., 2025b), Muon also produces exploding attention logits more frequently than AdamW. DoRA adapters are generally more stable than LoRA under Polar Express.

F.12.7 Key Padding Mask Ablations

We evaluate whether masking padding tokens in cross-attention improves performance. Table 28 shows that applying a key padding mask slightly degrades overall performance despite a marginal improvement on LIF-D. Adding gated attention (Qiu et al., 2025) further reduces performance and increases loss. We therefore omit both modifications in the final configuration.

F.13 Reproducibility and Sources of Non-Determinism

Despite fixing random seeds for Python, NumPy, and PyTorch (CPU and CUDA), we observe small differences in training trajectories across execution environments. Runs performed locally and on university servers exhibit consistent high-level behavior, with major loss peaks and dips aligning with dataset shard boundaries. However, within-shard dynamics differ systematically between environments, likely due to differences in data loading pipelines (e.g., streaming versus local parquet access with an `IterableDataset`) that alter sample ordering within shards.

Within a fixed environment, repeated runs produce nearly identical trajectories, with only minor numerical variation attributable to known GPU and multi-worker data-loading non-determinism. Over-

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
Without key padding mask*	**0.6625**	0.4448	**0.5297**	**0.5457**	**1.4874**
With key padding mask	0.6496	**0.4459**	0.5287	0.5414	1.4900
With key padding mask and gated attention	0.5338	0.4298	0.5256	0.4964	1.5354

Table 28: Key padding mask ablation results. Best values are in bold and second-best values are underlined.

all, reproducibility is preserved at a coarse level, although exact optimization dynamics remain sensitive to data ordering and system-level factors.

Figure 13 illustrates these effects, while Table 29 shows that benchmark performance remains highly consistent across repeated runs despite slight differences in final losses.

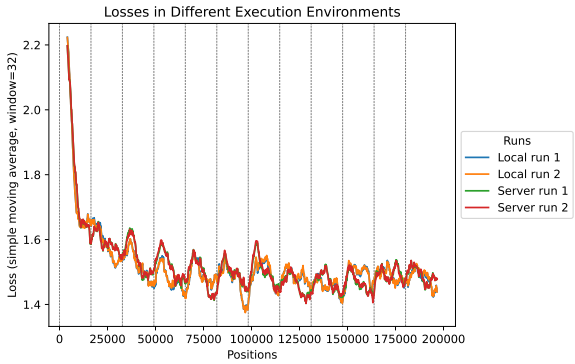

Figure 13: Training losses for two local and two server runs with identical training configurations. Runs within the same environment exhibit highly similar trajectories, while cross-environment runs differ slightly in their within-shard dynamics. Each shard contains 2^{14} plies.

G Final Training Configuration

This section summarizes the configurations used for the final **UniMaia** and **UniMaia-Aux** training runs.

G.1 UniMaia

UniMaia is trained on Lichess games from January 2013 to December 2022, corresponding to approximately 66.978B plies. Games from 2023 are reserved for evaluation. Following Appendix F.9, we apply two-stage Unimax downsampling (Chung et al., 2023): first across Elo partitions, and then independently within each time-control group using a fixed turn budget. Prompts are sampled uniformly from LichessTemplates-Pretrain and LichessTemplates-Instruct.

The model uses the BT4 Lc0-CF backbone (Monroe and Chalmers, 2024; Monroe et al., 2026) within the layer-coupled ControlNet architecture described in Section 4. The frozen Lc0 policy network is conditioned through a text encoder adapted using LoRA with rsLoRA and LoRA+, using rank 16 adapters.

Optimization uses NorMuon for all 2D non-embedding, non-output parameters and AdamW for the remaining parameters. We use weight decay 0.01, $\epsilon = 1 \times 10^{-8}$, gradient clipping with norm 1.0, and a LoRA+ learning-rate ratio of 2.0. Training uses a batch size of 64.

The learning rate is linearly warmed up over the first 1,024 steps to a peak learning rate of 5×10^{-5}. From April 2013 to June 2014, the learning rate decays linearly through monthly updates to 1.5×10^{-5}. It is then held constant until 2022, after which it decays monthly toward 5×10^{-7} by the end of training.

Training proceeds sequentially over monthly dataset partitions, each divided into independently shuffled shards to support efficient streaming and checkpoint recovery. Since training consists of a single epoch over the sampled data, the training loss is equivalent to the validation loss; we therefore report average monthly loss values throughout the paper. Monthly checkpoints are evaluated on the benchmark suite described in Appendix I.4. All experiments are conducted on a multi-GPU SLURM setup using two RTX Pro 6000 GPUs with fixed Python, NumPy, and PyTorch random seeds.

G.2 UniMaia-Aux

UniMaia-Aux performs an additional fine-tuning stage starting from the final UniMaia checkpoint trained through December 2022. Auxiliary training uses games from 2018 only, selected to ensure reliable clock metadata and termination annotations.

The auxiliary targets consist of:

(i) the final game result, represented as a categorical terminal-outcome distribution Z over $\{-1, 0, 1\}$, where a white win corresponds to 1, a draw corresponds to 0, and a black win corresponds to -1;

(ii) the number of plies remaining until the end of

Experiment	LOB-P	LIF-D	M1-S	Mean	$\mu_{\text{loss, Dec13}}$
Local run 1	**0.6580**	0.4450	**0.5320**	**0.5450**	**1.4768**
Local run 2	0.6429	**0.4470**	0.5254	0.5385	1.4813
Server run 1	0.6559	0.4457	0.5304	0.5440	1.4863
Server run 2	0.6443	0.4426	0.5315	0.5394	1.4871

Table 29: Results across repeated local and server runs. Best values are in bold and second-best values are underlined.

Setting	Value
Training data	Lichess 2013–2022
Evaluation data	Lichess 2023
Backbone	BT4 Lc0-CF
Architecture	ControlNet + controllable policy head
Frozen backbone	Yes
Text adaptation	LoRA & rsLoRA & LoRA+
LoRA rank	16
Batch size	64
Peak learning rate	5×10^{-5}
Warmup steps	1,024
Optimizer	NorMuon + AdamW
Weight decay	0.01
Optimizer ϵ	1×10^{-8}
Gradient clipping	1.0
Downsampling	Unimax by time control
Hardware	2× RTX Pro 6000

Table 30: UniMaia training configuration.

Setting	Value
Initialization	UniMaia 2022 checkpoint
Auxiliary training data	Lichess 2018
Backbone	BT4 Lc0-CF
Architecture	ControlNet + controllable policy head
Frozen backbone	Yes
LoRA rank	16
Batch size	64
Peak learning rate	5×10^{-5}
Warmup steps	1,024
Optimizer	NorMuon + AdamW
Auxiliary losses	Outcome, moves-left, termination, move delay
Move-delay source	PGN `%clk` annotations

Table 31: UniMaia-Aux training configuration.

the game, represented as $N - i - 1$, where N is the total number of plies in the game and i is the current ply index;

(iii) the termination reason (e.g., checkmate, time-out, resignation, or draw by threefold repetition); and

(iv) the move delay of the active player, denoted by Δt_a.

The scalar value estimate V can then be recovered from the predicted terminal-outcome distribution by taking its expectation:

$$V = \mathbb{E}[Z].$$

The value and moves-left heads are implemented as trainable residual updates to the frozen Lc0 heads, while additional heads predict termination type and move delay.

Auxiliary training uses the same BT4-based ControlNet architecture, LoRA rank, optimizer setup, batch size, and downsampling strategy as UniMaia. We filter out bot and correspondence games, games missing Elo or time-control metadata, and games without clock annotations; the move-delay objective additionally requires `%clk` PGN annotations.

All auxiliary objectives are optimized jointly. Outcome and termination predictions use cross-entropy losses, while moves-left and move-delay predictions use Huber losses. The learning rate schedule follows the same warmup as UniMaia, followed by a lower constant learning rate during fine-tuning.

H Training Dynamics

This appendix provides additional training dynamics analyses for UniMaia and UniMaia-Aux. The UniMaia benchmark trajectories are shown in Figure 2 in the main paper and are not repeated here.

H.1 UniMaia

Figure 14 shows the UniMaia training loss over the full run. The loss decreases rapidly early in training before gradually plateauing, while retaining noticeable shard-dependent variance. The smoothed loss also exhibits clear year-dependent structure, consistent with the reproducibility effects discussed in Appendix F.13.

To further analyze the divergent LOB-P and LOB-C behavior observed in the main paper, we partition LOB-P templates by final top-move accuracy using a threshold of 0.74. Figure 15 shows a clear bifurcation: high-performing templates continue improving throughout training, whereas

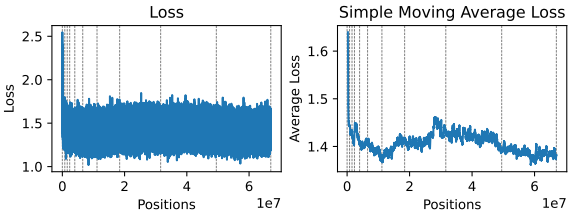

Figure 14: Left: raw training loss. Right: simple moving average (window size 1,024). Vertical lines indicate year boundaries.

lower-performing templates gradually degrade.

This split largely aligns with whether the prompt contains an explicit opening PGN. High-performing templates primarily use semantic opening descriptions without move sequences, whereas lower-performing templates typically include partial PGN conditioning. The weakest-performing templates effectively reduce to unconstrained move prediction conditioned only on partial move history.

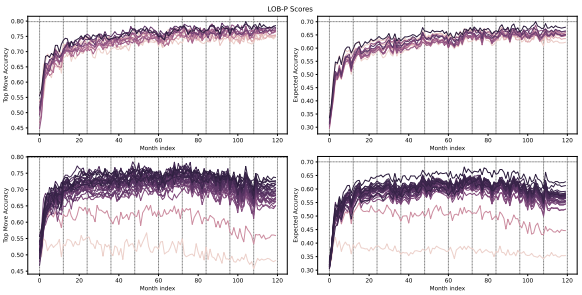

Figure 15: LOB-P templates partitioned by final top-move accuracy using a threshold of 0.74. Top: high-performing templates. Bottom: low-performing templates. Left: top-move accuracy. Right: expected accuracy.

Table 32 reports benchmark scores at December checkpoints throughout training. Most benchmarks improve steadily over time, with diminishing gains in later stages. Despite year- and month-dependent fluctuations in the training loss, benchmark performance remains relatively stable overall, with all benchmarks except LOB-P exhibiting clear upward trends. Transient loss increases tend to affect expected accuracy more than top-move accuracy, although the effect is modest.

H.2 UniMaia-Aux

Figure 16 shows optimization dynamics during auxiliary-target fine-tuning. The policy loss initially decreases before briefly increasing alongside a sharp rise in policy entropy, suggesting escape from a local policy minimum. The loss then decreases again, while the moves-left loss improves steadily throughout training.

Figure 16: Left: policy entropy. Middle: policy loss. Right: moves-left loss. All curves show simple moving averages.

Figures 17 and 18 show benchmark trajectories on the original benchmarks during fine-tuning, while Figures 19 and 20 report results on temporally augmented benchmarks.

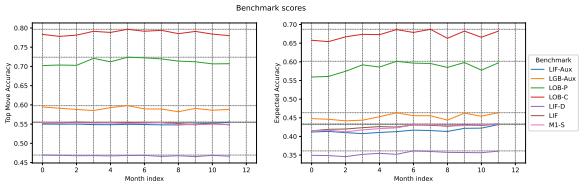

Figure 17: Top-move accuracy and expected accuracy across monthly checkpoints. Vertical lines indicate year boundaries.

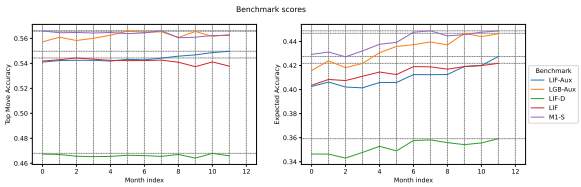

Figure 18: Top-move accuracy and expected accuracy across monthly checkpoints with opening plies excluded from evaluation.

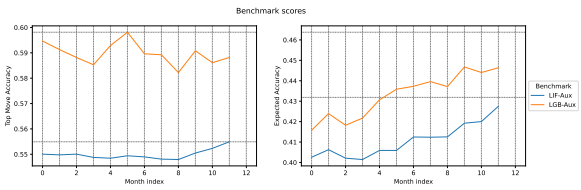

Figure 19: Top-move accuracy and expected accuracy for LIF-Aux and LGB-Aux during fine-tuning.

Unlike UniMaia, UniMaia-Aux often continues improving expected accuracy even when top-move accuracy plateaus, suggesting that auxiliary supervision primarily improves policy calibration rather than only top predictions.

Month	LOB-P	LOB-C	LIF-D	LIF	M1-S	Cumulative Positions
December 2013	0.6695	0.7079	0.4557	0.5187	0.5363	786,432
December 2014	0.7187	0.7641	0.4654	0.5374	0.5435	1,572,864
December 2015	0.7157	0.7672	0.4680	0.5394	0.5448	2,359,296
December 2016	0.7386	0.7934	0.4680	0.5439	0.5496	3,997,696
December 2017	0.7274	0.7912	0.4674	0.5477	0.5477	6,553,600
December 2018	0.7390	0.8042	0.4658	0.5508	0.5522	11,141,120
December 2019	0.7313	0.8036	0.4683	0.5532	0.5528	18,415,616
December 2020	0.7351	0.8185	0.4692	0.5567	0.5526	31,588,352
December 2021	0.6966	0.8046	0.4697	0.5575	0.5520	49,479,680
December 2022	0.7094	0.8162	0.4748	0.5587	0.5534	66,977,792

Table 32: Top move accuracy at December checkpoints during UniMaia training, along with the cumulative number of positions processed.

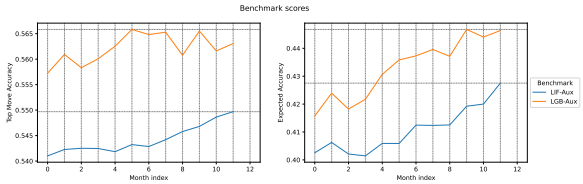

Figure 20: Top-move accuracy and expected accuracy for LIF-Aux and LGB-Aux with opening plies excluded.

Performance on temporally augmented benchmarks generally improves throughout fine-tuning, especially when the first 10 plies are omitted. In contrast, performance on the original benchmarks slightly declines later in training, suggesting partial overfitting to the auxiliary supervision setup or prompt format.

The auxiliary suffix may also weaken fixed prompt–opening associations during the opening phase. Because prompts now include temporal metadata and move-delay information, identical openings no longer necessarily share identical prompts, encouraging reliance on broader contextual signals rather than static prompt patterns.

I Additional Experimental Details

This appendix provides additional details on benchmark construction, evaluation protocols, baselines, and prompting strategies omitted from the main paper for brevity.

I.1 Benchmark Details

Lichess Openings Benchmark (LOB). LOB evaluates opening controllability using prompts derived from curated opening databases containing ECO codes, opening names, and canonical move sequences. Positions are paired with prompts sampled from LICHESSTEMPLATES-INSTRUCT.

We evaluate two variants:

- **LOB–Partial (LOB–P):** Prompts contain only the opening name and/or moves up to the current position.
- **LOB–Canonical (LOB–C):** Prompts additionally include the canonical continuation of the opening.

LOB–P avoids explicit answer leakage and therefore provides a more challenging and realistic evaluation, whereas LOB–C measures upper-bound opening-following performance.

Lichess Instruction-Following Benchmark (LIF). LIF is constructed from 2023 LICHESS-GAMES positions formatted using templates from LICHESSTEMPLATES-PRETRAIN and LICHESSTEMPLATES-INSTRUCT. The benchmark spans diverse player ratings, openings, and gameplay contexts, and serves as the primary prompt-conditioned evaluation benchmark.

Lichess Instruction-Following Diagnostic (LIF-D). LIF-D (19,606 positions) is a controlled diagnostic benchmark constructed from January 2013 positions restricted to the 900–1100 Elo range. Because it is drawn from the same temporal distribution as the training data, it is used only for ablations and training diagnostics.

Lichess Instruction-Following Top-10 Benchmark (LIF-T10). LIF-T10 (25,000 positions) is constructed from January 2023 games in which at least one player belongs to the most active Lichess accounts. The benchmark evaluates whether prompt-conditioned models can exploit player-specific information when identities are included in prompts.

Lumbra's Gigabase Benchmark (LGB). LGB is constructed from Lumbra's Gigabase (Jansen,

n.d.), a curated collection of deduplicated PGN files aggregated from multiple online and over-the-board (OTB) sources. We evaluate on the OTB 2020–2024 subset to measure generalization beyond the Lichess training distribution.

Games are formatted using LICHESSTEMPLATES-INSTRUCT. To partially account for systematic rating differences between FIDE and Lichess ratings, all player ratings are shifted by +500 Elo during evaluation.

Temporally augmented benchmarks. LIF-Aux and LGB-Aux append temporal metadata including time control, remaining clock time, and previous-move delay when available. These benchmarks additionally evaluate the auxiliary targets introduced during auxiliary-target fine-tuning.

I.2 Metadata-Conditioned Benchmarks

Maia-1 (M1). We evaluate using the publicly released "Maia KDD Testing Set" from McIlroy-Young et al. (McIlroy-Young et al., 2020). The released dataset differs slightly from the filtering protocol described in the original paper, including a broader Elo range and a small number of low-time positions. We use the released version for reproducibility.

Maia-1 Subset (M1-S). M1-S is a stratified subset of M1 grouped by player Elo (rounded to the nearest hundred) and time control. It is used for rapid evaluation during ablations and for evaluating prompt-based baselines when full M1 evaluation is computationally expensive.

Maia-2 Rapid Benchmark (M2R). M2R follows the Maia-2 evaluation protocol introduced by Tang et al. (Tang et al., 2024) using Lichess Rapid games from December 2019.

Allie Blitz Benchmark (ABB). ABB consists of Lichess Blitz games from 2022 (Zhang et al., 2024b). Following prior work, low-time positions are filtered to reduce noise.

I.3 Evaluation Protocol

Prior human-move-prediction benchmarks commonly exclude the first 10 plies and moves made under severe time pressure to reduce noise from opening preparation and time scrambles (McIlroy-Young et al., 2020; Zhang et al., 2024b).

Where applicable, we report both filtered and unfiltered results. For prompt-conditioned benchmarks such as LIF and LGB, we retain opening moves because some prompts explicitly target opening behavior.

We report:

- **Top-move accuracy (Acc@1):** Whether the highest-probability move matches the human move.
- **Expected accuracy ($\mathbb{E}[\mathrm{Acc}]$):** The probability assigned to the human move.
- **Weighted F1 ($\mathrm{F1}_w$):** Used for auxiliary categorical targets.
- **MAE:** Used for scalar auxiliary targets.

As in the main paper, expected accuracy is reported only for models that produce normalized move distributions.

I.4 Baseline Model Details

Lc0-CF baselines. We evaluate multiple pretrained Lc0-CF policy networks, including the t3, t82, BT3, BT4, and BT4-spsa checkpoints (Monroe and Chalmers, 2024; Monroe et al., 2026). These models differ in parameter count and architectural details but all use structured board encodings based on 112 spatial feature planes.

Maia family. Maia-1 consists of separate Lc0-based models trained for different Elo ranges (McIlroy-Young et al., 2020). Maia-2 introduces discrete Elo tokens and separate rapid/blitz models (Tang et al., 2024), while Maia-3 replaces discrete tokens with interpolated Elo embeddings using a ChessFormer backbone (Monroe et al., 2026).

Allie. Allie is a GPT-2-based human imitation model conditioned on Elo and time-control metadata encoded as tokens (Zhang et al., 2024b). We evaluate the ALLIE-POLICY configuration without search.

ChessCLIP. ChessCLIP scores the compatibility between prompts, candidate moves, and board states using CLIP-style contrastive representations (Feng et al., 2024). During inference, all legal moves are enumerated and ranked according to prompt similarity.

ChessGPT v1. ChessGPT is a large language model trained for chess reasoning and move prediction (Feng et al., 2024). It is based on the RedPajama INCITE 3B model (Weber et al., 2024).

ChessGPT-base is obtained by continual pretraining on a mixture of text, PGN, and FEN data, while ChessGPT-chat is further fine-tuned on conversational chess data.

ChessGPT-base. We evaluate ChessGPT-base using two prompting strategies unless otherwise specified:

1. **Prompt**: The prompt followed by the PGN move list without headers: `{prompt}\n\n{pgn}`.

2. **Elo and time control PGN header**[4]: A PGN header describing both players' Elos and the time control (Appendix J.4), followed by two newlines and the PGN move list.

To ensure reliable move extraction, we use `outlines` (Willard and Louf, 2023) to constrain the generated output with a predefined regular expression. For ChessGPT-base, the move number may optionally be preceded by a space, and both the move number and trailing punctuation are optional.

ChessGPT-chat. We evaluate ChessGPT-chat using three prompting strategies unless otherwise specified:

1. **Simple, duplicate**: We prompt ChessGPT-chat with `{prompt}\n\nPGN: {pgn}\n\nJust write your move.` Applying the chat template twice—once without the generation prompt and once with it—improves performance on the Lichess openings benchmark.

2. **Simple, original**: The same prompt as above, except the chat template is applied only once with the generation prompt.

3. **General policy**: A prompt adapted from Feng et al., consisting of a prompt to play chess in a given position, a PGN header with certain fields masked by "??", the PGN move list, and a final instruction to produce the next move.

As with ChessGPT-base, we use `outlines` (Willard and Louf, 2023) to constrain generation. For ChessGPT-chat, the output must match a move number, one or three periods, a space, and a valid SAN move.

[4]We use *PGN header* to refer to PGN-style headers that may not fully conform to the PGN specification.

GPT-3.5 Turbo Instruct. GPT-3.5 Turbo Instruct (OpenAI, n.d.) is a proprietary language model that demonstrates strong chess performance when prompted appropriately.

Prompting strategies. We evaluate the following configurations:

1. **GM PGN header**: A PGN header indicating a game between strong players (*i.e.*, Magnus Carlsen and Garry Kasparov), followed by the move list. We use the prompt from the open-sourced implementations accompanying Carlini (Carlini, 2023) and Allie (Zhang et al., 2024b), which differs slightly from the version described in the Carlini blog and reproduced in the Allie appendix (Appendix J.3).

2. **GM PGN header + prompt**: A natural-language prompt inserted between the header and the move list.

3. **Original PGN header**: The original PGN header, followed by the move list.

4. **Elo and time control PGN header**: A PGN header specifying player ratings and time control (Appendix J.4), followed by the move list.

For autoregressive language models, generated moves are constrained to valid chess formats where possible. Illegal moves are replaced with uniformly sampled legal moves during evaluation.

I.5 Model Comparison

The evaluated models differ substantially in their representations and conditioning mechanisms. Engine-based models such as Lc0 rely exclusively on structured board encodings, while metadata-conditioned models incorporate explicit Elo and time-control signals through tokens or embeddings. Prompt-conditioned language models instead encode all contextual information implicitly through text.

UniMaia combines structured Lc0 board representations with natural-language conditioning, enabling semantically flexible control while preserving domain-specific inductive biases. UniMaia-Aux further augments the model with auxiliary prediction heads for behavioral and temporal targets.

Tables 33–36 summarize the input representations, predicted outputs, and training configurations of the evaluated models.

Model	Game state	Elo	Time control	Prompt
Lc0	112 Lc0 planes	–	–	–
Maia-1	112 Lc0 planes	Separate models	–	–
Maia-2	Current board	Discrete tokens	Separate models	–
Maia-3	112 Lc0 planes	Interp. embeddings	–	–
Allie	Move history	Interp. embeddings	Discrete tokens	–
ChessCLIP	112 Lc0 planes	Via prompt	Via prompt	✓
ChessGPT	Via prompt	Via prompt	Via prompt	✓
GPT-3.5	Via prompt	Via prompt	Via prompt	✓
UniMaia	112 Lc0 planes	Via prompt	Via prompt	✓
UniMaia-Aux	112 Lc0 planes	Via prompt	Via prompt	✓

Table 33: Comparison of model inputs and conditioning mechanisms.

Model	Policy	Value	Moves left	Move delay	Resignation	Termination	Text
Lc0	✓	✓	✓	–	–	–	–
Maia-1	✓	✓	–	–	–	–	–
Maia-2	✓	✓	–	–	–	–	–
Maia-3	✓	✓	–	–	–	–	–
Allie	✓	✓	–	✓	✓	–	–
ChessCLIP	✓	–	–	–	–	–	–
ChessGPT	✓	–	–	–	–	–	✓
GPT-3.5	✓	–	–	–	–	–	✓
UniMaia	✓	–	–	–	–	–	–
UniMaia-Aux	✓	✓	✓	✓	✓	✓	–

Table 34: Comparison of predicted outputs and auxiliary targets.

Model	Batch size	Grad. acc.	# GPUs	# Positions	# Parameters
Maia-1	1,024	–	1	409.6M	92M
Maia-2	8,192	–	2	9.15B	23M
Maia-3	128	4	8	1.024B	5M, 23M, 79M
ChessCLIP	512	–	8	1.3M	67M
UniMaia	64	–	2	66.978M	3B+404M
UniMaia-Aux	64	–	2	71.565M	3B+515M

Table 35: Training configurations for structured-input models.

Model	# GPUs	Context	# Tokens	# Parameters
Allie	8	512	262.144B	355M
ChessGPT-base	8	1024	?	3B
ChessGPT-chat	8	1024	?	3B

Table 36: Training configurations for language-based models. ChessGPT is trained for one epoch. The exact dataset size is unavailable due to dataset release restrictions.

J Prompt Templates

J.1 Example LICHESSTEMPLATES Prompt Templates

Table 37 shows representative prompt templates from LICHESSTEMPLATES-PRETRAIN and LICHESSTEMPLATES-INSTRUCT, illustrating the linguistic variability and metadata coverage used during training and evaluation.

J.2 Metadata-Conditioned Benchmark Prompts

All prompt-conditioned models were provided with the following prompt template (formatted with game metadata) when evaluated on the metadata-conditioned benchmarks:

"`An anonymous white player with an ELO of {white_elo} plays chess against another anonymous black player, with an ELO of {black_elo} in a {type.lower()} game on Lichess played with {time_info}.`"

Here, "`time_control`" is rendered as "`a time`

Template Set	Example Template
PRETRAIN	This {time_control_lower} chess game was played online at Lichess.org on {date}, with each player given {td_all_seconds} and {text_time_increment} per move. The game featured {white_title_full_quot_name} {white_alias_alias_paren} as {W_lit}, with an Elo rating of {white_elo}, against {black_title_full_quot_name} {black_alias_alias_paren} as {B_lit}, who had a rating of {black_elo}. The opening used was the {opening_commas}, delineated by the moves: {opening_moves}.
	In a {ratedness_lower} {time_control_lower} chess game hosted on Lichess.org on {date}, {white_title_full_paren_title} {white_alias_known_as_quot_name}, rated {white_elo}, played as {W_lit} against {black_title_full_paren_title} {black_alias_known_as_quot_name} and rated {black_elo}. The game opened with the {opening}, starting with moves {opening_moves}. The time control for the match was {td_all_seconds} {with_text_time_increment} per move.
	On {date}, at Lichess.org, {white_title_full} {white_alias} (white) with an Elo rating of {white_elo} played against {black_title_full} {black_alias} (black) who had an Elo rating of {black_elo}. This rated {time_control_lower} game had a time control of {text_td} per player {time_increment_words_without_any_increment}. The game featured the {opening}, which began with the moves {opening_moves}.
INSTRUCT	Join {black_title} {black_alias}, ELO {black_elo}, as he plays {B_lit} utilizing the {opening} with the opening moves {opening_moves}. Challenge this {black_rank_player_a} to a game on Lichess and test your skills against his opening strategy.
	Play as {white_title_full} {white_alias}, taking the {w_lit_lower_pieces} and opening the game with {opening_moves}, known as the {opening_variant_name_of_the_main}.
	In this chess game on Lichess, the player, controlling the {b_lit_lower_pieces}, opens with the {opening} by playing {opening_moves}.

Table 37: Representative prompt templates from LICHESSTEMPLATES-PRETRAIN and LICHESSTEMPLATES-INSTRUCT.

control of <duration as human-friendly text> with (no increment|an increment of <increment as human-friendly text>)"

J.3 GM PGN Header Prompt

The GM PGN header prompt used for evaluation is based on the open-sourced implementations accompanying Carlini (Carlini, 2023) and Allie (Zhang et al., 2024b). This version differs slightly from the prompt described in the Carlini blog and reproduced in the Allie appendix.

In particular, the blog version includes a fixed game result and lists Kasparov as White and Carlsen as Black, whereas the implementation version omits the result field and lists Carlsen as White and Kasparov as Black. We use the implementation version for all experiments.

```
[White "Magnus Carlsen"]
[Black "Garry Kasparov"]
[WhiteElo "2900"]
[BlackElo "2800"]
```

J.4 Elo and Time Control PGN Header

Aside from the opening-specific benchmarks, ChessGPT-base and GPT-3.5 Turbo Instruct were prompted using a PGN header prepended to the movelist:

```
"""[White "???"]
[Black "???"]
[WhiteElo "\{white\_elo\}"]
[BlackElo "\{black\_elo\}"]
[TimeControl "\{time\_control\}"]

"""
```

This format mirrors the structured metadata representation used during training of these models, in contrast to the natural-language prompts used for UniMaia.

J.5 Auxiliary Targets Prompt Suffix

When predicting the auxiliary targets, we append the following suffix to the prompt if the time control is available:

`The time control for this {time_control.lower()} chess game was {text_td} {time_increment_text}.`

where `text_td` is the time delay in hours, minutes, and seconds, and `time_increment_text` is either `with no additional increment` or `with {time_increment} <second or seconds> of increment`, with the correct pluralization.

If the game has `%clk` annotations for move delay prediction, we append the following:

`Time taken last turn by active player: {active_player_last_move_time}. Time taken last turn by opponent: {opponent_last_move_time}. Time remaining for the active player: {active_player_time_remaining}. Time remaining for the opponent: {opponent_time_remaining}.`

K Detailed Evaluation Results

This appendix provides the complete benchmark results corresponding to the summarized results in Section 6. We include additional prompt-conditioned and metadata-conditioned evaluations, results with the first 10 plies omitted, auxiliary-target evaluations, and comparisons across model classes under a unified evaluation harness.

Because prior work often uses different benchmark suites, prompting strategies, and legality handling procedures, direct comparison across model classes is difficult. To improve comparability, all reproduced results are evaluated using a unified benchmark harness that standardizes preprocessing, prompting, legality handling, and evaluation metrics whenever possible.

K.1 Prompt-Conditioned Benchmarks

Tables 38–41 report the full prompt-conditioned benchmark results for both **Acc@1** and expected accuracy.

Overall, UniMaia achieves the strongest performance across most prompt-conditioned tasks, attaining the highest Acc@1 on LIF-D, LIF, LIF-T10, and LGB. The main exception is opening reconstruction (LOB-P and LOB-C), where ChessGPT-base performs best and UniMaia consistently ranks second.

When excluding the first 10 plies to reduce the influence of opening memorization, UniMaia retains the strongest overall performance on over-the-board prediction tasks while remaining competitive across the remaining benchmarks.

For expected accuracy, UniMaia and UniMaia-Aux achieve the strongest overall performance across nearly all prompt-conditioned settings. Although UniMaia-Aux slightly underperforms UniMaia in Acc@1 on the original prompt-conditioned benchmarks, it frequently achieves stronger expected accuracy, suggesting improved policy calibration from auxiliary temporal conditioning.

Both ChessGPT variants frequently generate illegal moves. During evaluation, illegal moves are replaced with uniformly sampled legal moves, reducing both Acc@1 and expected accuracy.

Finally, ChessCLIP rarely ranks the correct move as its top prediction despite occasionally assigning moderate probability mass to it. Its output distributions are typically diffuse after the softmax transformation, resulting in comparatively weak expected accuracy.

K.1.1 Auxiliary Temporal Metadata

Tables 42 and 43 report results for UniMaia-Aux on temporally augmented benchmarks.

Unlike the standard prompt-conditioned benchmarks, LIF-Aux and LGB-Aux include temporal metadata such as time controls, remaining clock time, and previous move delay in the prompt, more closely matching the auxiliary-target training setup.

In addition to move prediction accuracy, we evaluate the auxiliary targets introduced in Appendix G.2. We report weighted F1 ($F1_w$) for categorical targets and MAE for scalar targets.

UniMaia-Aux consistently outperforms Allie-Policy on both LIF-Aux and LGB-Aux, with and without the first 10 plies omitted. The gains are generally larger after excluding opening plies, suggesting that temporal metadata improves behavioral modeling beyond opening memorization.

UniMaia-Aux also substantially improves resignation prediction on over-the-board games, indicating stronger modeling of human gameplay behavior. Overall, it outperforms Allie-Policy on nearly all auxiliary targets except value prediction. We note that UniMaia-Aux is trained on substantially less data and uses a cross-entropy objective over discrete outcomes, whereas Allie uses MSE for value prediction, which may partially explain the MAE differences.

K.2 Metadata-Conditioned Benchmarks

Tables 44 through 47 report metadata-conditioned benchmark results, including evaluations with the first 10 plies omitted to reduce opening memorization effects.

Model	LOB-P	LOB-C	LIF-D	LIF	LIF-T10	LGB
Lc0-CF (t3)	0.5010	0.5027	0.3244	0.3942	0.3180	0.4772
Lc0-CF (t82)	0.5011	0.5024	0.3214	0.3908	0.3148	0.4749
Lc0-CF (BT3)	0.5025	0.5041	0.3252	0.3924	0.3162	0.4771
Lc0-CF (BT4)	0.5025	0.5039	0.3246	0.3924	0.3132	0.4770
Lc0-CF (BT4-spsa)	0.4744	0.4757	0.3217	0.3902	0.3122	0.4643
Maia-1*	0.5582	0.5602	0.4411	0.4797	0.4076	0.4857
Maia-1**	0.5709	0.5729	0.4411	0.4830	0.4085	0.5034
Maia-2 (Blitz)	0.4058	0.4040	0.4144	0.4763	0.4051	0.4452
Maia-2 (Rapid)	0.4129	0.4123	0.4207	0.4758	0.3943	0.4484
Allie-Policy	0.6150	0.6164	0.4658	0.5390	0.4662	0.5572
ChessCLIP	0.2730	0.2696	0.0895	0.0918	0.0747	0.1040
ChessGPT-base†	**0.7998**	**0.8955**	0.3808	0.4464	0.4153	0.4788
ChessGPT-chat‡	0.3417	0.2272	0.1494	0.1637	0.1480	0.1501
GPT-3.5§	0.5748	0.6431	0.4457	0.5312	0.4778	0.5514
UniMaia	0.7094	0.8162	**0.4748**	**0.5587**	**0.4954**	**0.6158**
UniMaia-Aux	0.7071	0.7801	0.4664	0.5481	0.4915	0.6086

Table 38: **Prompt-conditioned** benchmark results, measured by **Acc@1**. Best values are in bold and second-best values are underlined. *Evaluated only with the official models. **Evaluated with both the official and unofficial models. †Opening benchmarks evaluated with the regular prompt. Remaining benchmarks evaluated with the Elo and time control PGN header. ‡Opening benchmarks evaluated with the duplicate simple prompt. Remaining benchmarks evaluated with the general policy prompt. §Opening benchmarks evaluated with the GM PGN header. Remaining benchmarks evaluated with the Elo and time control PGN header.

Model	LIF-D	LIF	LIF-T10	LGB
Lc0-CF (t3)	0.3360	0.4052	0.3290	0.5064
Lc0-CF (t82)	0.3329	0.4013	0.3265	0.5012
Lc0-CF (BT3)	0.3364	0.4036	0.3272	0.5054
Lc0-CF (BT4)	0.3357	0.4037	0.3232	0.5067
Lc0-CF (BT4-spsa)	0.3366	0.4046	0.3249	0.5021
Maia-1*	0.4532	0.4845	0.4106	0.4515
Maia-1**	0.4532	0.4871	0.4114	0.4694
Maia-2 (Blitz)	0.4412	0.5021	0.4278	0.4850
Maia-2 (Rapid)	0.4467	0.5019	0.4187	0.4825
Allie-Policy	**0.4812**	**0.5492**	0.4735	0.5452
ChessCLIP	0.0862	0.0813	0.0633	0.0567
ChessGPT-base†	0.3813	0.4440	0.4158	0.4339
ChessGPT-chat‡	0.1490	0.1557	0.1491	0.1564
GPT-3.5†	0.4608	0.5444	**0.4914**	0.5569
UniMaia	0.4712	0.5457	0.4792	0.5638
UniMaia-Aux	0.4657	0.5378	0.4787	**0.5692**

Table 39: Prompt-conditioned benchmark results, measured by **Acc@1**. We **omit** the first 10 plies from each game in the benchmark. Best values are in bold and second-best values are underlined. *Evaluated only with the official models. **Evaluated with both the official and unofficial models. †Evaluated with the Elo and time control PGN header. ‡Evaluated with the general policy prompt.

For ABB, we additionally report published results from prior work where available. Because the Allie and Maia-3 evaluations use broadly compatible protocols, these values are approximately comparable, although all rankings in this work are based on our reproduced results under the unified benchmark harness.

Across metadata-conditioned benchmarks, UniMaia remains competitive with specialized human move prediction models despite being optimized primarily for prompt-conditioned policy modulation. Allie-Policy achieves the strongest overall Acc@1 performance, while UniMaia-Aux attains the best expected accuracy on several benchmarks, including M1-S and M2R.

Notably, UniMaia-Aux consistently improves expected accuracy relative to UniMaia despite slightly lower Acc@1 scores, suggesting that auxiliary temporal objectives improve policy calibration even when temporal metadata is unavailable at evaluation time.

Tables 46 and 47 contain nearly identical trends, although Maia-2 scores are lower because the first 10 plies were excluded during training.

K.3 Inference Strategy Ablations

This section provides detailed results for different prompting and inference strategies used with large language models, including ChessGPT and GPT-3.5 Turbo Instruct (Appendix I.4), illustrating the sensitivity of prompt-conditioned models to input formatting and prompt design.

L Extended Analysis

This appendix provides additional analysis of UniMaia, focusing on policy continuity, internal resid-

Model	LOB-P	LOB-C	LIF-D	LIF	LIF-T10	LGB
Lc0-CF (t3)	0.3628	0.3627	0.2310	0.2742	0.2298	0.3390
Lc0-CF (t82)	0.3633	0.3630	0.2311	0.2744	0.2293	0.3398
Lc0-CF (BT3)	0.3644	0.3642	0.2322	0.2756	0.2308	0.3405
Lc0-CF (BT4)	0.3657	0.3655	0.2328	0.2772	0.2311	0.3436
Lc0-CF (BT4-spsa)	0.3815	0.3813	0.2462	0.2948	0.2448	0.3616
Maia-1*	0.4130	0.4138	0.3321	0.3653	0.3195	0.3651
Maia-1**	0.4299	0.4307	0.3321	0.3678	0.3204	0.3827
Maia-2 (Blitz)	0.2990	0.2971	0.3227	0.3670	0.3216	0.3423
Maia-2 (Rapid)	0.3056	0.3040	0.3224	0.3650	0.3149	0.3401
Allie-Policy	0.4905	0.4906	0.3510	0.4104	0.3525	0.4480
ChessCLIP	0.0402	0.0403	0.0658	0.0616	0.0496	0.0405
UniMaia	0.5899	**0.7140**	0.3522	0.4297	**0.3678**	0.4834
UniMaia-Aux	**0.5973**	0.6823	**0.3606**	**0.4331**	0.3600	**0.4891**

Table 40: **Prompt-conditioned** results, measured by $\mathbb{E}[\mathbf{Acc}]$. Best values are in bold and second-best values are underlined. *Evaluated only with the official models. **Evaluated with both the official and unofficial models.

Model	LIF-D	LIF	LIF-T10	LGB
Lc0-CF (t3)	0.2360	0.2763	0.2336	0.3624
Lc0-CF (t82)	0.2364	0.2766	0.2336	0.3651
Lc0-CF (BT3)	0.2372	0.2781	0.2349	0.3657
Lc0-CF (BT4)	0.2379	0.2800	0.2352	0.3712
Lc0-CF (BT4-spsa)	0.2530	0.2994	0.2508	0.3937
Maia-1*	0.3438	0.3713	0.3248	0.3434
Maia-1**	0.3438	0.3733	0.3254	0.3570
Maia-2 (Blitz)	0.3463	0.3913	0.3446	0.3842
Maia-2 (Rapid)	0.3463	0.3886	0.3375	0.3792
Allie-Policy	**0.3652**	0.4215	**0.3607**	0.4393
ChessCLIP	0.0705	0.0662	0.0518	0.0422
UniMaia	0.3444	0.4129	0.3507	0.4252
UniMaia-Aux	0.3592	**0.4218**	0.3507	**0.4521**

Table 41: **Prompt-conditioned** results, measured by $\mathbb{E}[\mathbf{Acc}]$. We **omit** the first 10 plies from each game in the benchmark. Best values are in bold and second-best values are underlined. *Evaluated only with the official models. **Evaluated with both the official and unofficial models.

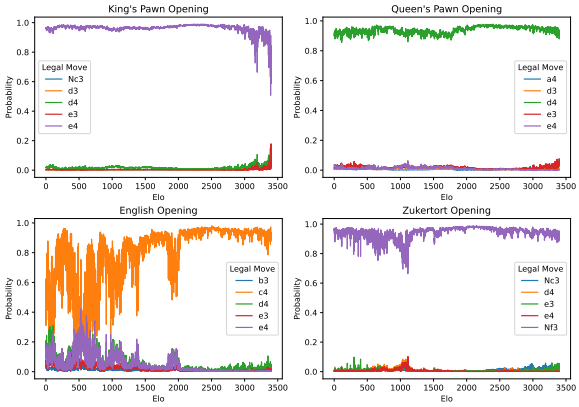

Figure 21: Top-5 move probabilities predicted by UniMaia as a function of Elo for opening-conditioned prompts.

ual dynamics, cross-attention behavior, and prompt sensitivity.

L.1 Policy Continuity Across Elo

We provide additional policy-continuity analyses beyond those shown in Section 7. All evaluations use Elo values from 0–3400, including regions below the training distribution (Appendix B).

Figure 3 in Section 7 evaluates four representative positions spanning the early opening, late opening, middlegame, and endgame. The corresponding PGNs and prompts are provided in Appendix M.1.1.

We additionally analyze continuity under opening conditioning by fixing the initial position and varying the opening prompt. We consider prompts corresponding to the King's Pawn, Queen's Pawn, English, and Zukertort openings. Full prompts are provided in Appendix M.1.2.

Most openings exhibit relatively stable top-move probabilities across Elo, while the English Opening shows larger variation in the range 400–1300 Elo, likely reflecting common transpositions and differing opening-selection behavior across rating levels.

L.1.1 Quantitative Diagnostics of Policy Continuity

Figures 22–25 provide quantitative diagnostics of policy continuity across Elo values, complementing the qualitative analysis in Section 7 and Appendix L.1. For each position, we measure cosine similarity and L2 distance between policies at adjacent Elo values, together with their corresponding distributions.

Overall, adjacent policies remain highly similar across Elo ranges, while policy differences vary

Benchmark	Model	Acc@1↑	$\mathbb{E}[\text{Acc}]$ ↑	F1_w ↑ Result	Termination	Resigned	$N-i-1$	MAE↓ V	Δt_a
LIF-Aux	Allie-Policy	0.5390	0.4104	–	–	0.6409	–	**0.2176**	4.8651
	UniMaia-Aux	**0.5549**	**0.4318**	**0.6718**	**0.5448**	**0.7279**	**21.3624**	0.3229	**4.0252**
LGB-Aux	Allie-Policy	0.5572	0.4480	–	–	0.0933	–	**0.3711**	127.7715
	UniMaia-Aux	**0.5872**	**0.4637**	**0.3814**	**0.6494**	**0.6643**	**35.6365**	0.4092	**124.9658**

Table 42: Auxiliary targets results. Best values are in bold and second-best values are underlined.

Benchmark	Model	Acc@1↑	$\mathbb{E}[\text{Acc}]$ ↑	F1_w ↑ Result	Termination	Resigned	$N-i-1$	MAE↓ V	Δt_a
LIF-Aux	Allie-Policy	0.5492	0.4215	–	–	0.6388	–	**0.2139**	5.0358
	UniMaia-Aux	**0.5495**	**0.4275**	**0.6618**	**0.5431**	**0.7270**	**20.4985**	0.3258	**4.3000**
LGB-Aux	Allie-Policy	0.5452	0.4393	–	–	0.1231	–	**0.3337**	122.3217
	UniMaia-Aux	**0.5622**	**0.4464**	**0.3824**	**0.6135**	**0.6223**	**26.3534**	0.3981	**118.3532**

Table 43: Auxiliary targets results. We **omit** the first 10 plies from each game in the benchmark. Best values are in bold and second-best values are underlined.

Model	ABB (ours)	ABB (reported)	M1-S	M1	M2R
Lc0-CF (t3)	0.4179	–	0.4318	0.4351	0.4136
Lc0-CF (t82)	0.4113	–	0.4285	0.4289	0.4062
Lc0-CF (BT3)	0.4157	–	0.4310	0.4340	0.4127
Lc0-CF (BT4)	0.4133	–	0.4255	0.4305	0.4079
Lc0-CF (BT4-spsa)	0.4145	–	0.4223	0.4303	0.4091
Maia-1*	0.5078	0.516 ± 0.001	0.5094	0.5135	0.5135
Maia-1**	0.5089	–	0.5149	0.5173	0.5138
Maia-2 (Blitz)	0.5230	0.520 ± 0.001	0.5246	0.5300	0.5284
Maia-2 (Rapid)	0.5204	–	0.5253	0.5294	0.5311
Maia-3-79M (Blitz)	–	$\mathbf{0.571 \pm 0.001}$	–	–	–
Allie-Policy	**0.5643**	0.557 ± 0.001	**0.5733**	**0.5717**	**0.5598**
ChessCLIP	0.0813	–	0.0813	0.0803	0.0838
ChessGPT-base†	0.4336	–	0.4375	–	0.4305
ChessGPT-chat‡	0.1697	–	0.1719	–	0.1647
GPT-3.5§	0.5410	0.537 ± 0.001	0.5565	–	0.5390
GPT-3.5†	0.5498	–	0.5667	–	0.5489
UniMaia	0.5548	–	0.5667	–	0.5536
UniMaia-Aux	0.5471	–	0.5624	–	0.5493

Table 44: **Metadata-conditioned** benchmark prediction results, measured by **Acc@1**. We **omit** the first 10 plies from each game in the benchmark. Best values are in bold and second-best values are underlined. *Evaluated only with the official models. **Evaluated with both the official and unofficial models. †Evaluated with the Elo and time control PGN header. ‡Evaluated with the general policy prompt. §Evaluated with the GM PGN header.

smoothly rather than exhibiting abrupt transitions. This behavior is consistent across opening, middlegame, and endgame positions, supporting the interpretation that Elo conditioning induces a largely continuous modulation of the policy distribution.

L.2 Mechanistic Interpretability

To better understand how prompt conditioning modifies the pretrained chess policy, we analyze the internal behavior of UniMaia on the first 256 examples of LIF.

L.2.1 Layer-wise Residual Contributions

We extract the outputs of the zero-initialized projection layers in the ControlNet branch, which we refer to as *ControlNet residual updates*. These residuals quantify the magnitude of the modifications introduced relative to the frozen Lc0 backbone.

Figure 26 shows that encoder layers 0–13 produce relatively small residual updates, whereas encoder layer 14 and the controllable policy head produce substantially larger updates. This suggests that prompt-conditioned modulation is concentrated near the output of the network rather than uniformly distributed across all layers.

Model	ABB	M1-S	M1	M2R
Lc0-CF (t3)	0.3035	0.3145	0.3165	0.3009
Lc0-CF (t82)	0.3028	0.3137	0.3159	0.3002
Lc0-CF (BT3)	0.3049	0.3170	0.3183	0.3027
Lc0-CF (BT4)	0.3063	0.3175	0.3198	0.3036
Lc0-CF (BT4-spsa)	0.3244	0.3360	0.3386	0.3212
Maia-1*	0.3868	0.3883	0.3902	0.3898
Maia-1**	0.3888	0.3916	0.3932	0.3900
Maia-2 (Blitz)	0.4102	0.4144	0.4166	0.4137
Maia-2 (Rapid)	0.4056	0.4107	0.4131	0.4130
Allie-Policy	**0.4409**	0.4476	**0.4475**	0.4354
ChessCLIP	0.0700	0.0670	0.0686	0.0742
UniMaia	0.4224	0.4317	–	0.4182
UniMaia-Aux	0.4382	**0.4488**	–	**0.4427**

Table 45: **Metadata-conditioned** results, measured by $\mathbb{E}[\mathbf{Acc}]$. We **omit** the first 10 plies from each game in the benchmark. Best values are in bold and second-best values are underlined. *Evaluated only with the official models. **Evaluated with both the official and unofficial models.

Model	ABB	M1-S	M1	M2R
Lc0-CF (t3)	0.4091	0.4234	0.4249	0.4040
Lc0-CF (t82)	0.4036	0.4195	0.4199	0.3981
Lc0-CF (BT3)	0.4073	0.4231	0.4240	0.4032
Lc0-CF (BT4)	0.4053	0.4181	0.4210	0.3991
Lc0-CF (BT4-spsa)	0.4032	0.4116	0.4177	0.3972
Maia-1*	0.5043	0.5072	0.5119	0.5094
Maia-1**	0.5057	0.5122	0.5158	0.5096
Maia-2 (Blitz)	0.4961	0.4998	0.5048	0.4964
Maia-2 (Rapid)	0.4956	0.5038	0.5065	0.5031
Allie-Policy	**0.5568**	**0.5654**	**0.5650**	**0.5497**
ChessCLIP	0.0799	0.0803	0.0797	0.0821
ChessGPT-base†	0.4408	0.4452	–	0.4380
ChessGPT-chat‡	0.1671	0.1661	–	0.1631
GPT-3.5†	0.5398	0.5554	–	0.5370
UniMaia	0.5447	0.5534	–	0.5413
UniMaia-Aux	0.5361	0.5495	–	0.5356

Table 46: **Metadata-conditioned** results, measured by **Acc@1**. Best values are in bold and second-best values are underlined. *Evaluated only with the official models. **Evaluated with both the official and unofficial models. †Evaluated with the Elo and time control PGN header. ‡Evaluated with the general policy prompt.

This behavior is consistent with the zero-initialization design of ControlNet-style conditioning, which encourages earlier layers to remain close to the pretrained Lc0 representations while allowing later layers to introduce controllable modifications.

Figure 27 further shows that the controllable policy head contributes most strongly in the early opening, while encoder layer 14 becomes increasingly influential later in the game.

One possible explanation is that earlier layers are more tightly constrained by architectural biases

Model	ABB	M1-S	M1	M2R
Lc0-CF (t3)	0.2985	0.3093	0.3104	0.2955
Lc0-CF (t82)	0.2977	0.3088	0.3098	0.2948
Lc0-CF (BT3)	0.2997	0.3118	0.3120	0.2970
Lc0-CF (BT4)	0.3009	0.3123	0.3133	0.2979
Lc0-CF (BT4-spsa)	0.3176	0.3295	0.3306	0.3140
Maia-1*	0.3816	0.3840	0.3862	0.3827
Maia-1**	0.3840	0.3877	0.3897	0.3829
Maia-2 (Blitz)	0.3860	0.3921	0.3938	0.3861
Maia-2 (Rapid)	0.3819	0.3890	0.3907	0.3858
Allie-Policy	**0.4310**	**0.4388**	**0.4386**	0.4225
ChessCLIP	0.0652	0.0627	0.0640	0.0685
UniMaia	0.4084	0.4181	–	0.4025
UniMaia-Aux	0.4222	0.4330	–	**0.4234**

Table 47: **Metadata-conditioned** results, measured by $\mathbb{E}[\mathbf{Acc}]$. Best values are in bold and second-best values are underlined. *Evaluated only with the official models. **Evaluated with both the official and unofficial models.

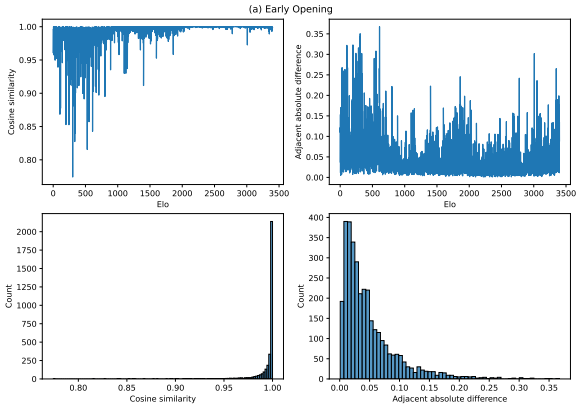

Figure 22: Quantitative policy continuity diagnostics for an early-opening position.

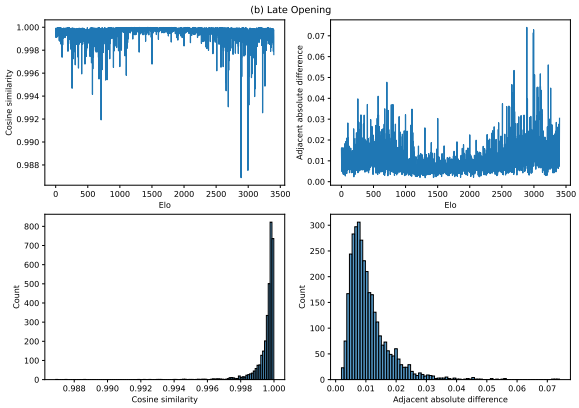

Figure 23: Quantitative policy continuity diagnostics for a late-opening position.

inherited from Lc0-CF, such as the geometric attention bias (GAB) (Monroe and Chalmers, 2024; Monroe et al., 2026). Later layers and the controllable policy head interface more directly with the policy output and may therefore admit greater flex-

Model	LOB-P	LOB-C	LIF-D	LIF	LIF-T10	LGB
ChessGPT-base (prompt)	**0.7998**	**0.8955**	0.3801	**0.4503**	0.4060	**0.5436**
ChessGPT-base (Elo + time control PGN)	0.6100	0.6186	**0.3808**	0.4464	**0.4153**	0.4788
ChessGPT-chat (simple, duplicate)	0.3417	0.2272	0.0981	0.1097	0.0964	0.1532
ChessGPT-chat (simple, original)	0.2601	0.1730	0.0892	0.0992	0.0831	0.1243
ChessGPT-chat (general policy)	0.1630	0.1503	0.1494	0.1637	0.1480	0.1501

Table 48: ChessGPT **prompt-conditioned** results, measured by **Acc@1**. Best values are in bold and second-best values are underlined.

Model	LIF-D	LIF	LIF-T10	LGB
ChessGPT-base (prompt)	0.3672	0.4303	0.3888	0.4336
ChessGPT-base (Elo + time control PGN)	**0.3813**	**0.4440**	**0.4158**	**0.4339**
ChessGPT-chat (simple, duplicate)	0.0933	0.1020	0.0874	0.1129
ChessGPT-chat (simple, original)	0.0873	0.0939	0.0727	0.0918
ChessGPT-chat (general policy)	0.1490	0.1557	0.1491	0.1564

Table 49: ChessGPT **prompt-conditioned** results, measured by **Acc@1**. We **omit** the first 10 plies from each game in the benchmark. Best values are in bold and second-best values are underlined.

Model	ABB	M1-S	M2R
ChessGPT-base (prompt)	0.4330	0.4330	0.4310
ChessGPT-base (Elo + time control PGN)	**0.4408**	**0.4452**	**0.4380**
ChessGPT-chat (simple, duplicate)	0.1128	0.1006	0.1146
ChessGPT-chat (simple, original)	0.0933	0.0889	0.0998
ChessGPT-chat (general policy)	0.1671	0.1661	0.1631

Table 50: ChessGPT **metadata-conditioned** results, measured by **Acc@1**. Best values are in bold and second-best values are underlined.

Model	ABB	M1-S	M2R
ChessGPT-base (prompt)	0.4260	0.4257	0.4248
ChessGPT-base (Elo + time control PGN)	**0.4336**	**0.4375**	**0.4305**
ChessGPT-chat (simple, duplicate)	0.0980	0.0898	0.1071
ChessGPT-chat (simple, original)	0.0874	0.0821	0.0932
ChessGPT-chat (general policy)	0.1697	0.1719	0.1647

Table 51: ChessGPT **metadata-conditioned** results, measured by **Acc@1**. We **omit** the first 10 plies from each game in the benchmark. Best values are in bold and second-best values are underlined.

Model	LOB-P	LOB-C	LIF-D	LIF	LIF-T10	LGB
GPT-3.5 (GM PGN header)	**0.5748**	**0.6431**	0.4344	0.5218	0.4684	**0.5525**
GPT-3.5 (GM PGN header + Prompt)	0.5118	0.5186	0.4341	0.5166	0.4607	0.5138
GPT-3.5 (Prompt + Original PGN header)	0.5196	0.5183	0.4338	0.5166	0.4593	0.5147
GPT-3.5 (Elo + Time Control PGN header)	–	–	**0.4457**	**0.5312**	**0.4778**	0.5514

Table 52: GPT-3.5 **prompt-conditioned results**, measured by **Acc@1**. Best values are in bold and second-best values are underlined.

ibility for prompt-conditioned modulation. We emphasize, however, that this interpretation remains speculative and is not directly verified in this work.

To analyze how concentrated the residual updates are, we define the *residual concentration ratio* as the ratio of the maximum absolute residual update to the mean absolute residual update.

As shown in Figure 28, the ratio is approximately 6 for encoder layers 0–13, approximately 3 for encoder layer 14, and approximately 5 for the controllable policy head. This suggests that residual updates in the early encoder layers and policy

Model	LIF-D	LIF	LIF-T10	LGB
GPT-3.5 (GM PGN header)	0.4531	0.5352	0.4841	0.5562
GPT-3.5 (GM PGN header + Prompt)	0.4584	0.5411	0.4849	0.5555
GPT-3.5 (Prompt + Original PGN header)	0.4579	0.5410	0.4849	0.5543
GPT-3.5 (Elo + Time Control PGN header)	**0.4608**	**0.5444**	**0.4914**	**0.5569**

Table 53: GPT-3.5 **prompt-conditioned** results, measured by **Acc@1**. We **omit** the first 10 plies from each game in the benchmark. Best values are in bold and second-best values are underlined.

Model	ABB	M1-S	M2R
GPT-3.5 (GM PGN header)	0.5298	0.5451	0.5248
GPT-3.5 (GM PGN header + Prompt)	0.5271	0.5409	0.5212
GPT-3.5 (Elo + Time Control PGN header)	**0.5398**	**0.5554**	**0.5370**

Table 54: GPT-3.5 **metadata-conditioned** results, measured by **Acc@1**. Best values are in bold and second-best values are underlined.

Model	ABB	M1-S	M2R
GPT-3.5 (GM PGN header)	0.5410	0.5565	0.5390
GPT-3.5 (GM PGN header + Prompt)	0.5493	0.5640	0.5461
GPT-3.5 (Elo + Time Control PGN header)	**0.5498**	**0.5667**	**0.5489**

Table 55: GPT-3.5 **metadata-conditioned** results, measured by **Acc@1**. We **omit** the first 10 plies from each game in the benchmark. Best values are in bold and second-best values are underlined.

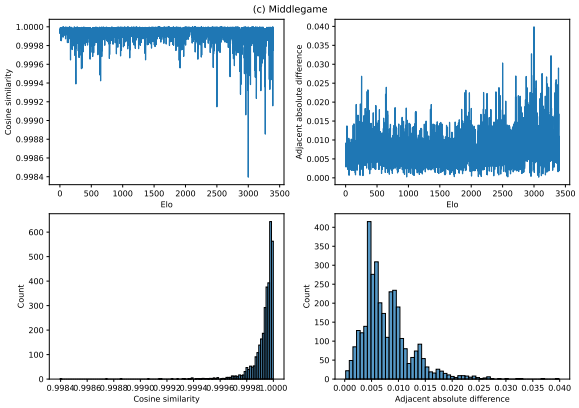

Figure 24: Quantitative policy continuity diagnostics for a middlegame position.

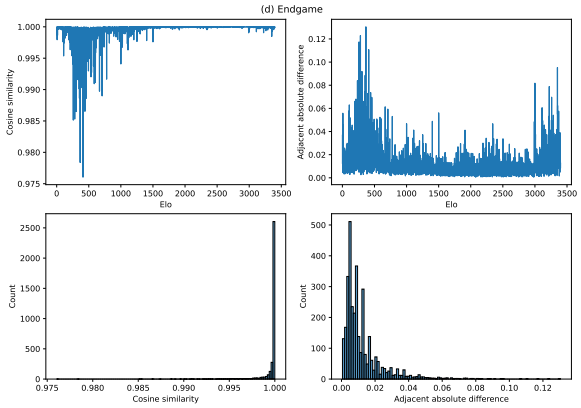

Figure 25: Quantitative policy continuity diagnostics for an endgame position.

head are concentrated in relatively few dimensions, whereas encoder layer 14 exhibits more distributed modifications.

Because residual updates in layers 0–13 are small in magnitude, we further compare the encoder representations of UniMaia and the frozen Lc0 backbone using cosine similarity.

The cosine similarity has mean 0.551 and standard deviation 0.075, indicating moderate similarity between the induced representations.

We observe a moderate positive correlation between cosine similarity and mean player Elo ($r = 0.396$, $p = 4.70 \times 10^{-11}$), suggesting that higher-rated games induce representations closer to the pretrained Lc0 policy. This is consistent with the interpretation that stronger play aligns more closely with the pretrained backbone, whereas lower-rated or stylistically distinct play requires larger prompt-conditioned deviations. We additionally observe a weak negative correlation with ply index ($r = -0.157$, $p = 0.0119$), indicating that representations diverge slightly from the base model as games progress.

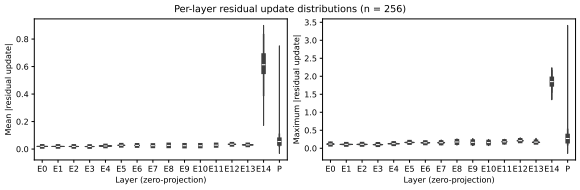

Figure 26: Distribution of mean and maximum absolute residual updates for each encoder layer and the controllable policy head.

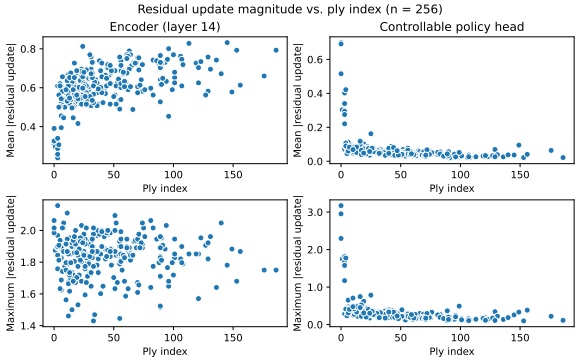

Figure 27: Mean and maximum absolute residual updates as a function of ply index for encoder layer 14 and the controllable policy head.

L.2.2 Cross-attention Sinks

To analyze the interaction between prompts and board representations, we examine cross-attention behavior in the ControlNet branch.

Following prior work on attention sinks (Xiao et al., 2023), we compute the proportion of examples for which the largest cross-attention weight is assigned to the first token. Attention weights are averaged across heads and board squares before computing this statistic. Note that the ChessGPT tokenizer does not include a beginning-of-sequence token, so the first token is part of the prompt itself.

Most layers exhibit strong attention-sink behavior, whereas a small subset rarely or never does so. In particular, encoder layers 2 and 14 almost never exhibit attention sinks ($\approx 1\%$ and 0%, respectively), while most remaining layers and the controllable policy head do so in more than 90% of evaluated prompts. The controllable policy head, together with encoder layers 8 and 13, exhibits attention sinks for all evaluated prompts.

Cross-attention activations are also highly similar across board squares. For a given text token, activations from different spatial locations remain nearly identical, as illustrated in Figure 30.

Attempts to mitigate this behavior did not improve performance. Masking padding tokens in the cross-attention keys (Appendix F.12.7) slightly degraded performance, and adding gated attention (Qiu et al., 2025) (Appendix F.5) did not recover this loss. We leave further investigation of these effects to future work.

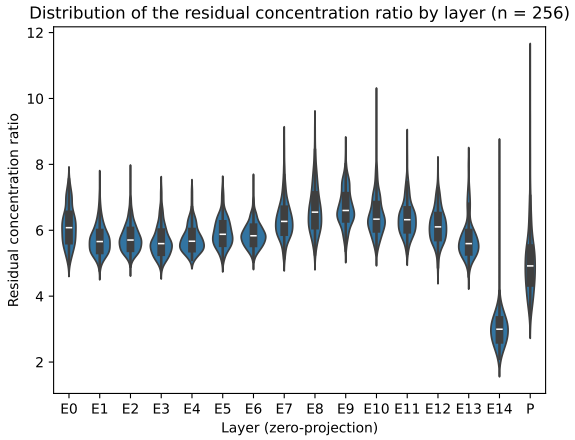

Figure 28: Residual concentration ratio for each encoder layer and the controllable policy head.

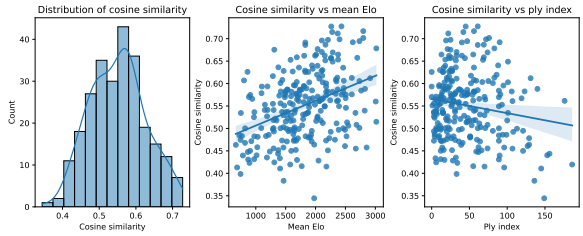

Figure 29: Cosine similarity between flattened encoder layer 13 representations of UniMaia and the frozen Lc0 backbone. Left: distribution over examples. Middle: similarity as a function of mean Elo. Right: similarity as a function of ply index.

L.3 Failure Modes and Stress Cases

UniMaia remains sensitive to relatively small prompt variations, even when the underlying instruction is semantically unchanged.

For example, the prompts "`Open with the English Opening`" and "`Open with the English Opening.`" can produce noticeably different move distributions despite differing only by punctuation.

Similarly, lexical variations can substantially affect opening-following behavior. After the position "`1. e4 c5`", the prompts "`Play the Alapin Sicilian`" and "`Play the Alapin Sicilian Defense`" both assign highest probability to `c3`, the defining move of the Alapin Sicilian. In contrast, the prompt "`Open with the Alapin Sicilian`" instead assigns highest probability to `Nf3`, with `c3` becoming only the second-ranked move.

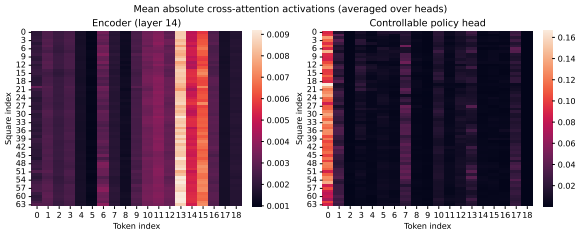

Figure 30: Mean absolute cross-attention activations averaged across heads. Left: encoder layer 14. Right: controllable policy head.

One contributing factor is the distribution of prompt templates used during training. None of the pretraining templates, and only approximately 33.4% ($\frac{684}{2048}$) of instruction templates, begin with the verb "Play", while none begin with "Open". Many templates instead begin with contextual prefixes such as player names or titles. Consequently, prompts that more closely match the training distribution tend to produce more stable behavior than semantically equivalent but distributionally different phrasing.

This sensitivity is also qualitatively consistent with the prevalence of attention sinks (Appendix L.2.2). If a small subset of tokens disproportionately captures attention, minor tokenization or phrasing differences may propagate through the network and induce large downstream policy changes. While this does not establish causality, it provides a plausible explanation for the observed instability.

These examples are intended to illustrate stress cases rather than typical behavior. Nevertheless, they highlight an important limitation of prompt-conditioned chess policies: relatively small prompt perturbations can induce disproportionately large policy changes.

M Policy Continuity Evaluation Details

This appendix summarizes the evaluation prompts used in the policy continuity analyses of Section 7. All evaluations sweep Elo values from 0–3400 to probe both in-distribution and out-of-distribution behavior.

M.1 Policy Continuity Evaluation Prompts

We evaluate policy continuity in two settings: (1) across representative positions spanning different game stages and (2) under opening-conditioned prompts from the initial position.

M.1.1 Policy Continuity Across Positions

For the analysis in Figure 3, we evaluate four representative positions spanning the early opening, late opening, middlegame, and endgame. The corresponding prompts and PGNs are listed in Table 56.

M.1.2 Policy Continuity Under Opening Conditioning

For the analysis in Figure 21 in Appendix L.1, we evaluate policy continuity from the initial position under different opening-conditioned prompts:

1. **King's Pawn Opening**: `You are an anonymous white player rated {elo}, playing against an anonymous black player rated {elo}, using the King's Pawn Opening`
2. **Queen's Pawn Opening**: `You are an anonymous white player rated {elo}, playing against an anonymous black player rated {elo}, using the Queen's Pawn Opening`
3. **English Opening**: `You are an anonymous white player rated {elo}, playing against an anonymous black player rated {elo}, using the English Opening`
4. **Zukertort Opening**: `You are an anonymous white player rated {elo}, playing against an anonymous black player rated {elo}, using the Zukertort Opening`

N Performance by Elo Range

Figures 31 and 33 show benchmark performance as a function of White and Black Elo.

Both top-move and expected accuracy generally increase with player Elo, reflecting the greater consistency of higher-rated play. Lower-rated games exhibit more stochasticity and stylistic variation, making moves harder to predict.

The benchmark families also exhibit different heatmap patterns. LIF shows greater variability across Elo combinations because it uses randomly sampled prompt templates with varying metadata, whereas ABB uses a fixed template with consistent metadata, producing more stable patterns.

We present the accuracy and size heatmaps for all benchmarks. Together, these visualizations illustrate how performance varies across player skill levels and dataset distributions.

Stage	PGN	Prompt Template
Early Opening	1. e4 c5	You are an anonymous white player rated {elo}, playing against an anonymous black player rated {elo}, using the Sicilian Alapin
Late Opening	1. d4 d5 2. c4 e6 3. Nc3 c5 4. cxd5 exd5 5. Nf3 Nf6 6. g3 Nc6 7. Bg2	You are an anonymous black player rated {elo}, playing against an anonymous white player rated {elo}, opening with the main line of the Tarrasch Defense's Prague Variation in the QGD.
Middlegame	1. e4 c5 2. Nf3 Nc6 3. Bc4 g6 4. c3 Bg7 5. Qb3 e6 6. d4 cxd4 7. cxd4 Nxd4 8. Nxd4 Bxd4 9. Qd3 Qb6 10. Be3 Bxe3 11. fxe3 Ne7 12. Nc3	Username prashu_20091, a beginner with an ELO of {elo}, confidently plays as White, utilizing the Sicilian Defense: Old Sicilian with the opening moves 1. e4 c5 2. Nf3 Nc6.
Endgame	1. e4 d5 2. exd5 Qxd5 3. Nc3 Qa5 4. d4 c6 5. Nf3 Bg4 6. Be2 Bxf3 7. Bxf3 Nf6 8. O-O e6 9. Bf4 Bb4 10. Ne4 Nxe4 11. Bxe4 O-O 12. c3 Be7 13. b4 Qd8 14. Qh5 g6 15. Qh3 Nd7 16. Bh6 Re8 17. a3 Nf6 18. Bf3 Bf8 19. Bxf8 Rxf8 20. a4 Rc8 21. Rad1 Nd7 22. c4 Nb6 23. d5 cxd5 24. cxd5 Nxd5 25. Bxd5 exd5 26. Qb3 Qc7 27. Rxd5 Rfd8 28. Rxd8+ Rxd8 29. h3 Rd2 30. Qe3 Qd8 31. Qxa7 Rd1 32. Rxd1 Qxd1+ 33. Kh2 Qd6+ 34. g3 Qf6 35. Qe3 Qd6 36. Qc5 Qd2 37. a5 Kg7 38. Kg2 Kf6 39. Qb6+ Kg7 40. Qxb7 h5 41. a6	The game in question was a rated blitz chess match hosted on the online chess platform Lichess.org on January 1, 2023. The match featured a rapid time control of five minutes with no additional time per move, denoted as "300+0." Playing with the white pieces was a player using the username "Talca" who held a rating of {elo}. The black pieces were commanded by player "pierobaldo" who held an identical rating of {elo} at the time of the game. The opening played in this game was the Scandinavian Defense, specifically the Main Line variation, which is introduced with the moves 1. e4 d5 2. exd5 Qxd5 3. Nc3 Qa5. The game featured an average of {elo} Elo with a 0-point difference

Table 56: PGN and prompt templates used for policy continuity evaluation across representative positions (early opening, late opening, middlegame, and endgame).

O Future Work

Several directions may further improve controllable chess policy modeling.

Future work should improve the realism and robustness of the prompt distribution. Although the current pipeline uses LLMs for prompt generation and paraphrasing, it still relies on a largely fixed template set and manual validation. Incorporating natural user prompts, broader linguistic variation, and richer stylistic descriptions may improve generalization and controllability. Automated validation methods, including heuristic filters or LLM-based judges, could also help scale template verification while reducing prompt artifacts.

Another important direction is hybrid conditioning. Rather than relying solely on natural language or structured metadata, future systems could combine both. Natural language provides flexible high-level control, while structured inputs such as Elo, time control, clock usage, and player identity may improve consistency and behavioral realism. The results of UniMaia-Aux suggest that temporal and behavioral metadata are particularly promising conditioning signals.

Architecturally, the strong performance of deep fusion under full fine-tuning motivates further comparisons between ControlNet-style conditioning, deep fusion, early fusion, and hybrid approaches. Smaller embedding-focused text encoders may also provide more efficient conditioning than large language models, particularly when adapted to chess-specific corpora (Appendix F.12.2). Other conditioning techniques, including classifier-free guidance, distilled guidance, and retrieval-augmented personalization, may further improve controllability and human move prediction.

Future work should also expand the training objective beyond next-move prediction. Bidirectional, multi-task, or self-supervised objectives could strengthen the relationship between prompts, metadata, and gameplay behavior.

Evaluation and training data may likewise be extended beyond Lichess. Incorporating over-the-

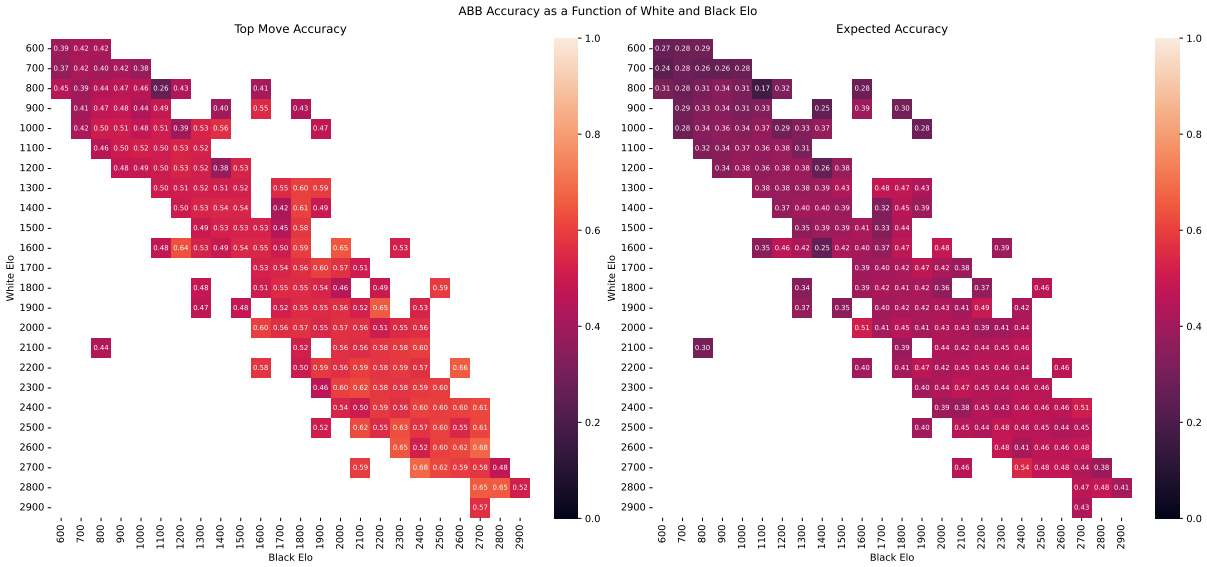

Figure 31: ABB accuracy as a function of White and Black Elo. Left: top-move accuracy. Right: expected accuracy.

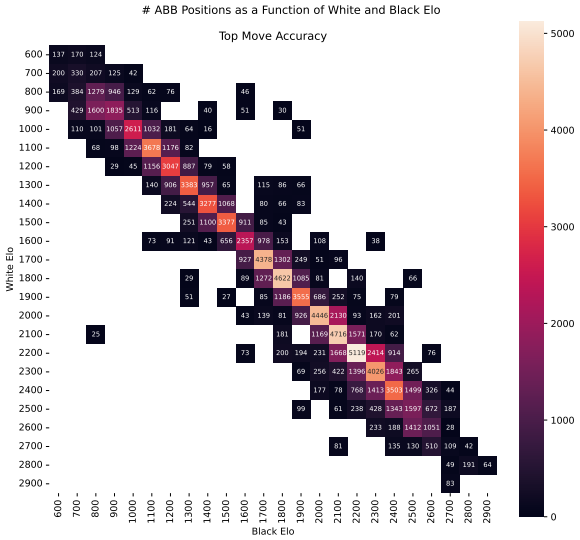

Figure 32: Number of ABB entries in each rounded White and Black Elo bucket.

board games, additional online platforms, curated engine-analysis datasets, and human evaluations would enable more robust assessment of behavioral realism and prompt controllability in real-world settings.

Finally, future interpretability work should move beyond descriptive analysis toward causal intervention. Techniques such as activation patching, residual ablations, prompt counterfactuals, and sparse autoencoders may help identify which components mediate opening control, Elo conditioning, and human move prediction, as well as how language conditioning modifies pretrained Lc0 representations. In particular, sparse autoencoders could be used to disentangle interpretable latent features within the cross-attention and policy-modulation layers, enabling analysis of how specific semantic concepts influence the pretrained chess representations.

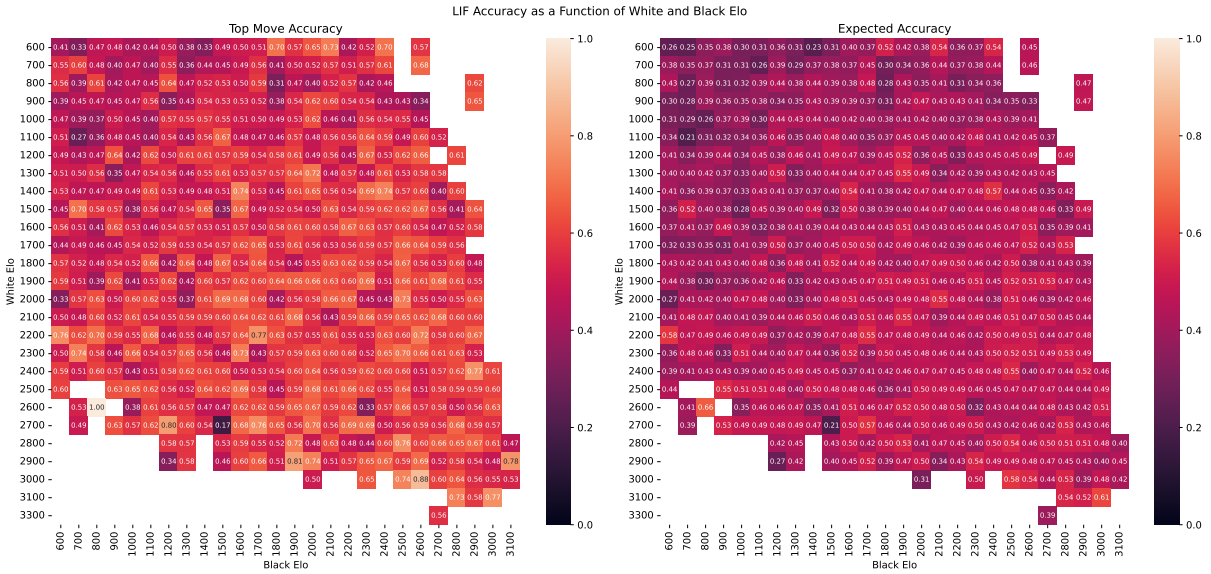

Figure 33: LIF accuracy as a function of White and Black Elo. Left: top-move accuracy. Right: expected accuracy.

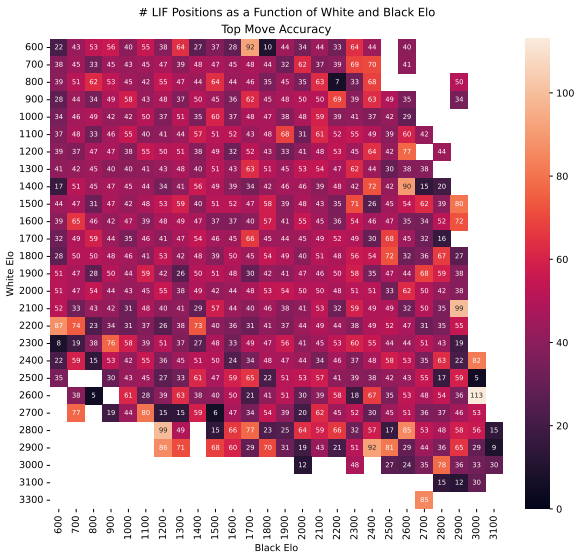

Figure 34: Number of LIF entries in each rounded White and Black Elo bucket.

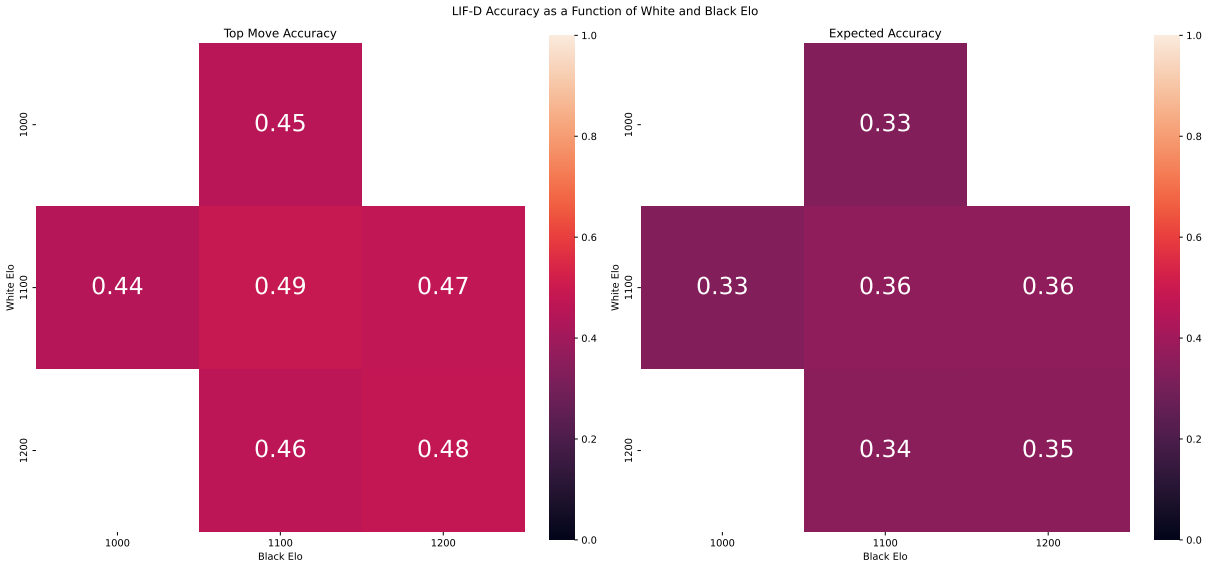

Figure 35: LIF-D accuracy as a function of White and Black Elo. Left: top-move accuracy. Right: expected accuracy.

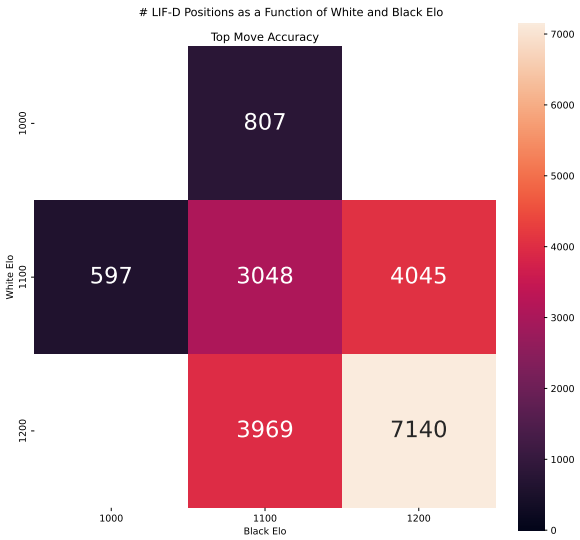

Figure 36: Number of LIF-D entries in each rounded White and Black Elo bucket.

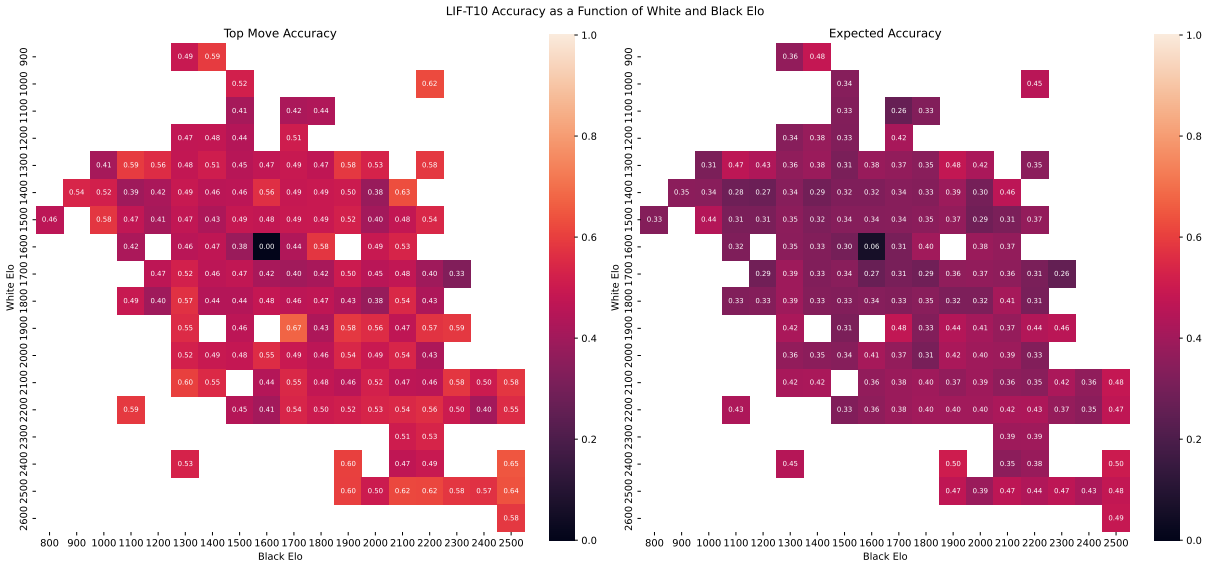

Figure 37: LIF-T10 accuracy as a function of White and Black Elo. Left: top move accuracy. Right: expected accuracy

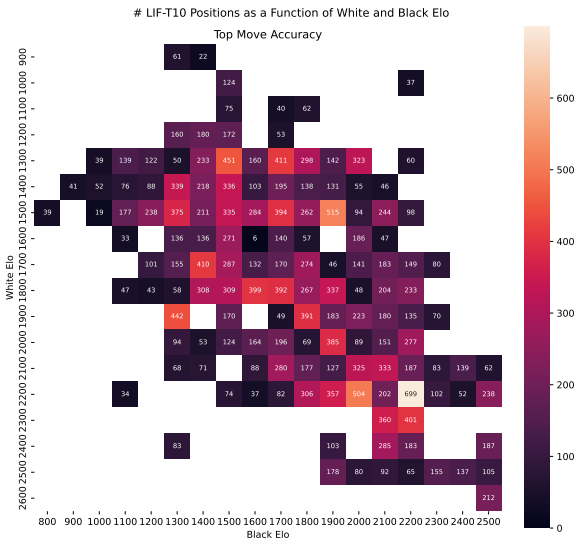

Figure 38: Number of LIF-T10 entries in each rounded White and Black Elo bucket.

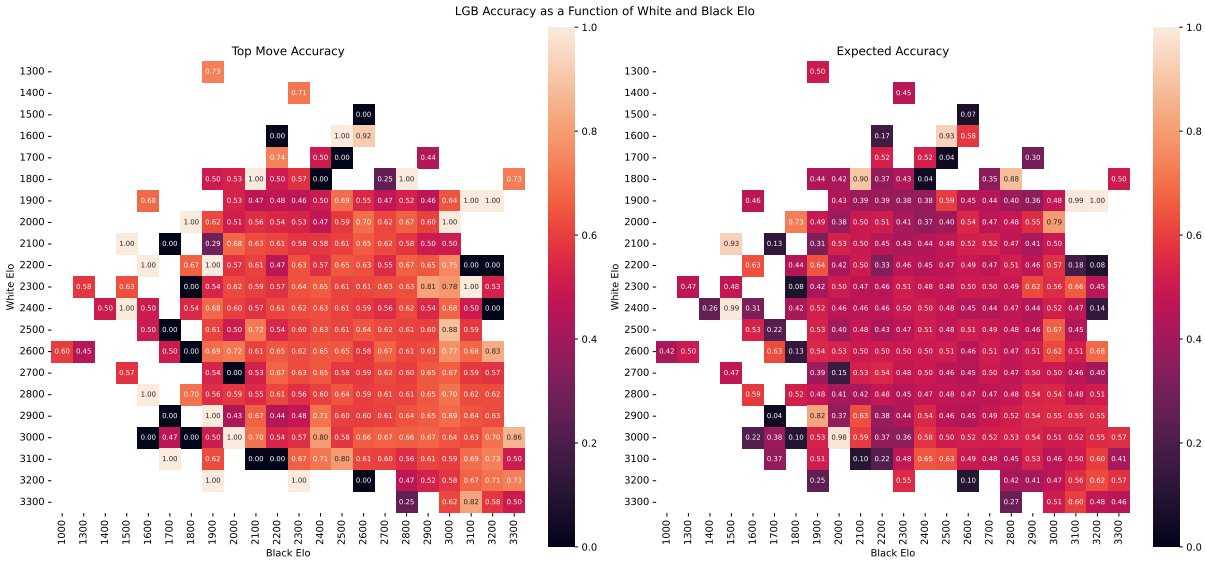

Figure 39: LGB accuracy as a function of White and Black Elo. Left: top move accuracy. Right: expected accuracy

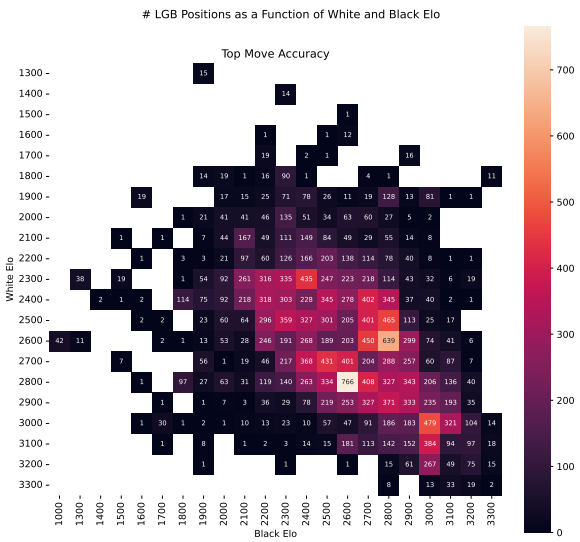

Figure 40: Number of LGB entries in each rounded White and Black Elo bucket.

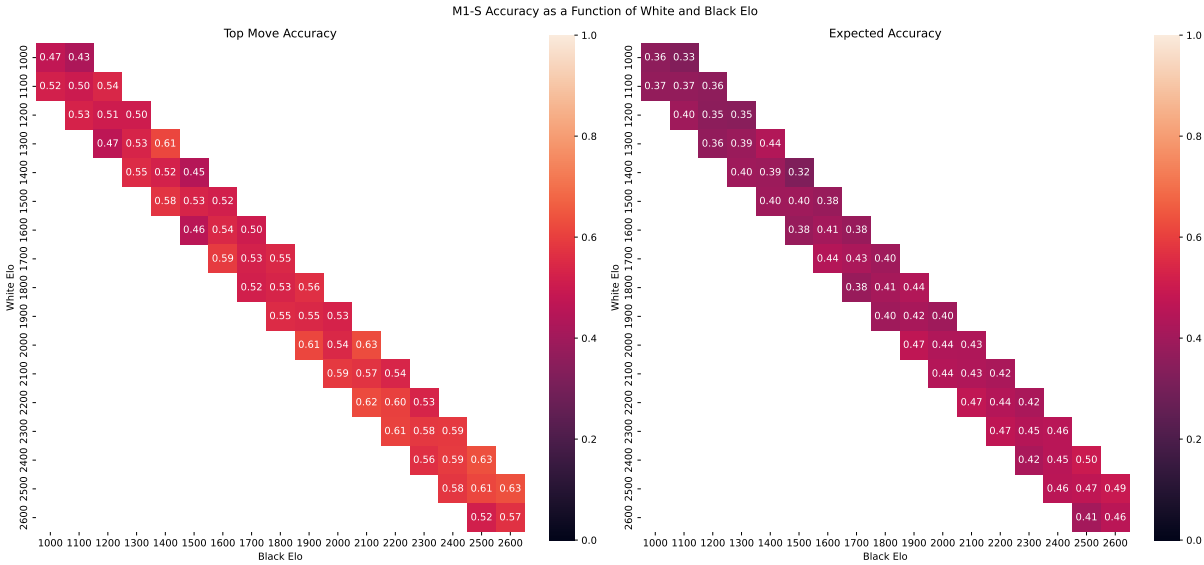

Figure 41: M1-S accuracy as a function of White and Black Elo. Left: top move accuracy. Right: expected accuracy

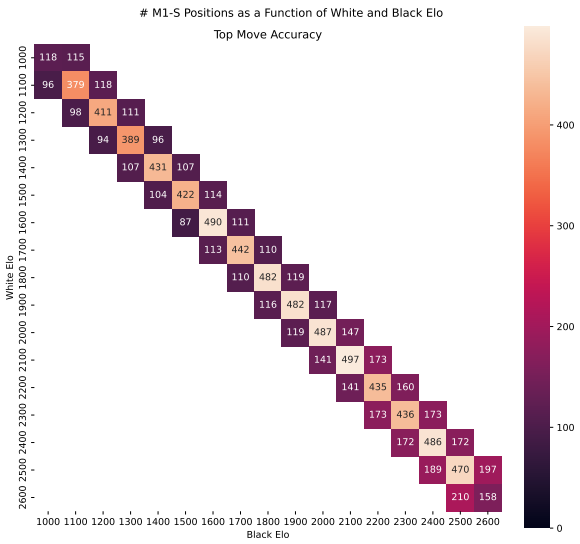

Figure 42: Number of M1-S entries in each rounded White and Black Elo bucket.

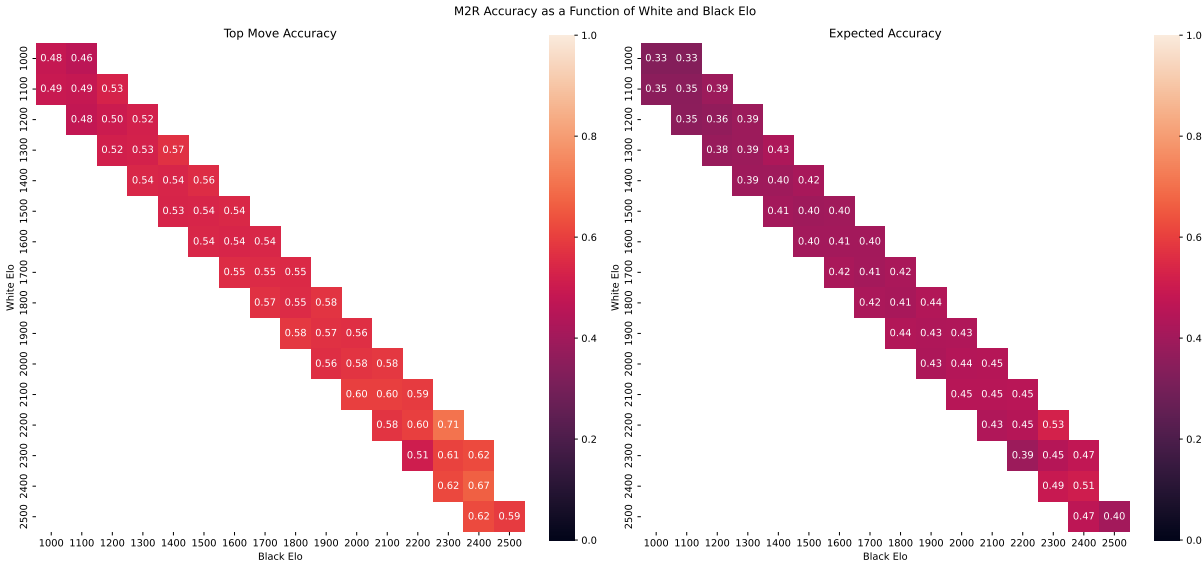

Figure 43: M2R accuracy as a function of White and Black Elo. Left: top move accuracy. Right: expected accuracy

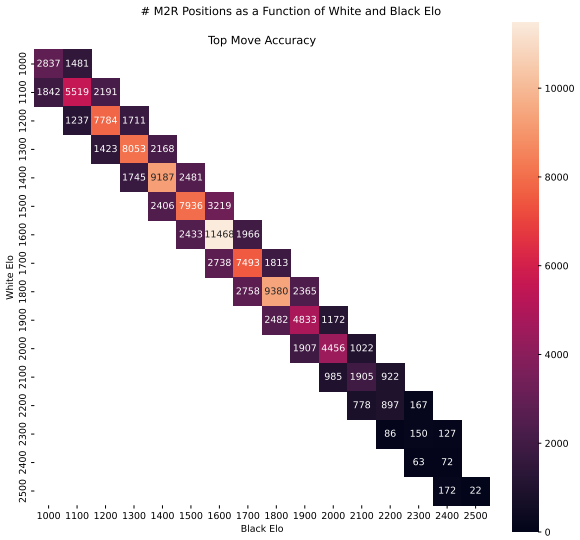

Figure 44: Number of M2R entries in each rounded White and Black Elo bucket.