Learning Chess Structure Without Rules: Discrete Diffusion on FEN Token Sequences

Ashish Behal
Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY
akbehal2@buffalo.edu

Abstract

We investigate whether a discrete diffusion model can learn the structural properties of chess board positions from raw data alone, without any explicit knowledge of chess rules. Using 500,000 puzzle positions from the Lichess database encoded as 72-token sequences, we train a Discrete Denoising Diffusion Probabilistic Model (D3PM) [Austin et al., 2021] and evaluate generated positions at two levels: syntactic validity and distributional realism. The model achieves 69.5% valid position generation compared to 16.3% for a random baseline, and closely matches the training distribution on pawn count and passed pawn metrics. Results suggest the model learns genuine chess structure implicitly from data, though the model overestimates how many pieces should be on the board, generating positions that resemble early game states more than the tactically complex puzzle positions it was trained on.

1 Introduction

Chess is a domain with crisp, verifiable rules and rich structure. Every legal board position must simultaneously satisfy dozens of constraints — piece counts, king placement, pawn ranks, castling rights — yet these rules are never explicitly taught to our model. Instead, we ask: *can a generative model learn what a valid chess position looks like purely from examples?*

This question matters beyond chess. Many real-world symbolic domains — protein structures, program syntax, formal logic — have implicit combinatorial constraints that are difficult to supervise directly. A model that learns these constraints from data alone would be broadly useful.

We approach this using D3PM [Austin et al., 2021], a discrete diffusion framework that corrupts token sequences with noise and learns to reverse the corruption. Applied to chess board positions encoded as token sequences, the model is never told the rules — it must discover them from the statistical patterns in 500,000 real puzzle positions.

Our contribution is twofold. First, we apply D3PM to chess position generation, a setting not studied in prior work. Second, we introduce a two-level evaluation framework that goes beyond simple legality checks, using distributional comparison against both the training data and a random baseline to quantify how much structure the model actually learned.

2 Related Work

Prior work on diffusion for discrete data is anchored by Austin et al. [2021], who extended the DDPM framework [Ho et al., 2020] to categorical token spaces. Rather than Gaussian noise, D3PM corrupts tokens by randomly replacing them — uniformly, toward semantically similar tokens, or irreversibly toward a mask token — and learns the reverse process.

In the chess domain, Ruoss et al. [2024] demonstrated that a large transformer trained on Lichess games could play at near-grandmaster level without any explicit search, suggesting neural networks can internalize deep chess knowledge from data. Monroe & Chalmers [2024] showed that tokenization design — how squares and metadata are encoded — matters more than model architecture for downstream performance. Wu et al. [2025] applied discrete diffusion to chess move sequences, showing that diffusion can learn structured game dynamics. El-Kishky et al. [2025] benchmarked generative architectures on the Lichess puzzle dataset and showed that binary legality checks are insufficient to distinguish good generated positions from trivial ones — directly motivating our dual-level evaluation.

Our project occupies a gap at the intersection of these threads: unconditional D3PM generation of individual chess positions, evaluated not just on whether individual positions are legal, but on whether the overall distribution of generated positions matches the statistical character of real chess puzzles — an approach inspired by distributional evaluation methods in the generative modeling literature [Heusel et al., 2017].

3 Data

We use the Lichess puzzle database [Lichess, 2024], a publicly available collection of nearly 6 million chess puzzles extracted from real games. Each puzzle is stored as a FEN (Forsyth-Edwards Notation) string — a compact text encoding of the full board state including piece placement, side to move, castling rights, en passant availability, and move counters.

We extract 500,000 FEN strings from the dataset. Puzzle positions are tactically rich by design — they are drawn from real games at moments of high tension, typically involving sacrifices, combinations, or mating attacks. This gives the dataset a distinctive statistical fingerprint: fewer pieces than average game positions (puzzles skew toward middlegame and endgame), more passed pawns, and more uneven material balance. These properties make the dataset ideal for our evaluation, since a model that genuinely learns the distribution should reproduce this fingerprint without being told about it.

Each FEN string is converted to a 72-token integer sequence. The first 64 tokens represent the 64 board squares, each taking one of 13 values (empty or one of 12 piece types). The remaining 8 tokens encode side to move, castling rights (4 binary tokens), en passant target, halfmove clock, and fullmove counter. This tokenization scheme follows the approach of Ruoss et al. [2024], who showed that clean structured token representations outperform raw PGN notation for neural chess models.

4 Methods

4.1 D3PM Framework

We train a D3PM [Austin et al., 2021] on the 72-token FEN sequences. The forward process gradually corrupts a clean token sequence by replacing tokens with random values from the vocabulary, over 1,000 timesteps. The model learns to reverse this process — starting from a fully corrupted (random noise) sequence and recovering something that looks like a real chess position.

The neural network inside the D3PM is a DDiT-Llama transformer [cloneofsimo, 2021] with 30.8 million parameters, 512-dimensional embeddings, and 6 layers. We train for 10 epochs over 500,000 FEN sequences with batch size 256 and learning rate 2×10^{-4}, using an AdamW optimizer with linear warmup. Training runs on an NVIDIA B200 GPU and takes approximately 25 minutes.

4.2 Two-Level Evaluation Framework

We evaluate generated positions at two levels:

Level 1 — Syntactic validity. We convert generated token sequences back to FEN strings and run `board.is_valid()` from the `python-chess` library. This checks hard legality constraints: exactly one king per side, no pawns on the back rank, valid piece counts, and that the non-moving side is not in check.

Level 2 — Structural realism. We compare distributional statistics of generated positions against the training data and a random baseline using KL divergence. Four metrics are measured: white pawn count, material balance, passed pawn count, and total material. The random baseline consists of positions generated by randomly placing pieces on the board subject to the hard legality constraints, representing a model that learned nothing beyond the rules.

4.3 Innovation

The primary innovation of this project is the evaluation framework itself. Prior work on generative chess models uses either binary legality checks or task-specific quality criteria such as puzzle uniqueness [El-Kishky et al., 2025]. Neither approach quantifies how much structure the model learned relative to a null model. Our two-level comparison — generated positions vs. training data vs. random baseline — adapts the distributional evaluation philosophy of metrics like FID [Heusel et al., 2017] to the discrete symbolic domain of chess, and repurposes standard chess annotation logic as measurement instruments for generative quality.

5 Experiments and Results

5.1 Training

Figure 1 shows the training loss curve across 10 epochs, plotted as an Exponential Moving Average (EMA) to smooth out batch-to-batch fluctuations and reveal the overall trend. The loss drops sharply from 4.39 at step 1 to approximately 0.33 by the end of epoch 1, then continues to decrease gradually to a final value of 0.308. The rapid early convergence indicates the model quickly learns the basic statistical patterns of FEN token sequences. The slow continued improvement in later epochs reflects learning of finer-grained structural dependencies.

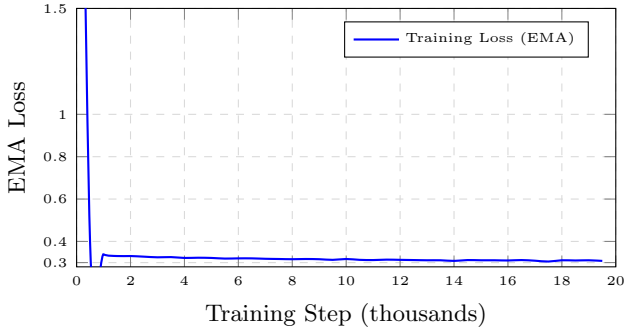


Figure 1: Training loss over 10 epochs (19,540 steps). Loss drops sharply in epoch 1 and converges gradually thereafter.

5.2 Level 1: Syntactic Validity

We generate 1,000 positions from the trained model and check each one for legality. Results are shown in Table 1.

Table 1: Level 1 validity results across 1,000 generated positions.

Source	Valid	Valid %
D3PM (ours)	695 / 1000	69.5%
Random baseline	1000 / 6148 attempts	16.3%
Training data	1000 / 1000	100.0%

The D3PM generates valid positions at 69.5%, more than four times the rate of the random baseline (16.3%). This is a strong result for a model that was never given the rules of chess — the model has implicitly learned the hard constraints of legal position construction from data alone.

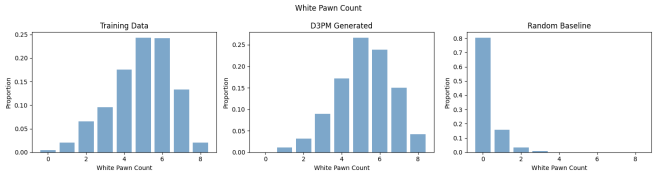
5.3 Level 2: Distributional Realism

Table 2 reports KL divergence between the training distribution and each comparison group across four structural metrics. Lower values indicate closer match to the training data.

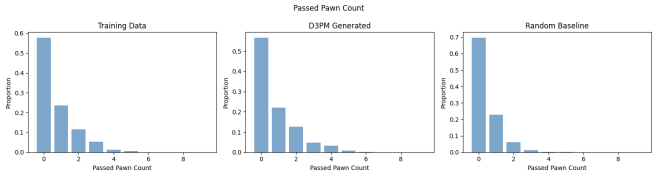
Table 2: KL divergence from training distribution. Lower is better.

Metric	D3PM	Random
White pawn count	**0.092**	17.640
Passed pawn count	**0.010**	0.088
Material balance	0.572	0.577
Total material	**3.217**	5.582

Figures 2 and 3 show the full distributions for each metric.

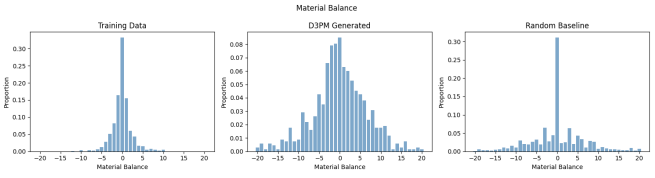


(a) White pawn count distribution.

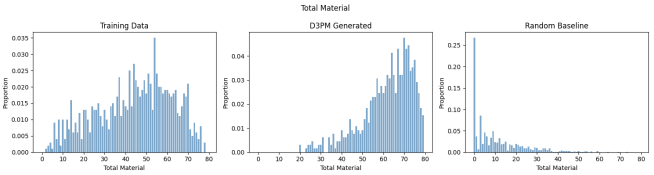


(b) Passed pawn count distribution.

Figure 2: Pawn structure distributions. The D3PM (center) closely matches the training data (left). The random baseline (right) is dramatically different.



(a) Material balance distribution.



(b) Total material distribution.

Figure 3: Material distributions. The D3PM learns material balance less well than pawn structure, and systematically overestimates total material on the board.

The model excels at pawn count (KL 0.092 vs 17.640 for random) and passed pawn rate (KL 0.010 vs 0.088), demonstrating it has learned fine-grained pawn structure from data. Material balance is learned less well — both the model and random baseline produce wider, flatter distributions than the training data, which is sharply peaked near zero. Total material is learned partially — the model scores better than random (3.217 vs 5.582) but still shows a systematic bias toward positions with more pieces on the board.

Table 3 shows the game phase distribution, which makes this bias explicit.

Table 3: Game phase distribution across position sources.

Source	Opening	Middlegame	Endgame
Training data	22.6%	54.2%	23.2%
D3PM (ours)	67.2%	30.8%	2.0%
Random baseline	0.4%	9.4%	90.2%

The random baseline heavily skews toward endgame because random piece placement tends to produce sparse boards. The D3PM overcorrects in the opposite direction, generating positions with many pieces that resemble opening positions. The model has learned that chess boards should be piece-rich — correcting the random baseline's sparseness — but has not fully learned the puzzle dataset's middlegame and endgame skew.

6 Project Limitations

The most significant limitation is the game phase bias. The model generates too many opening-like positions relative to the puzzle training distribution. This likely reflects the model's difficulty learning long-range dependencies — the total material on the board is determined

by the interaction of all 64 square tokens simultaneously, which is harder to learn than local pawn patterns. Additionally, game phase classification uses a simple material threshold (opening ≥ 60 points, middlegame ≥ 30, endgame < 30) rather than a more sophisticated engine-based approach, which may introduce some imprecision in the phase distribution analysis.

The evaluation sample size of 1,000 generated positions introduces variance in the metrics, particularly for Level 1 validity which fluctuated between 50% and 93% during training on 16-sample checkpoints. A larger generation set would produce more stable estimates.

7 Conclusion and Future Work

We trained a D3PM on 500,000 chess puzzle positions encoded as 72-token sequences and showed that the model learns genuine chess structure without being told the rules. The model generates valid positions 69.5% of the time — more than four times the random baseline rate — and closely matches the training distribution on pawn count and passed pawn metrics. These results suggest that discrete diffusion models can recover meaningful combinatorial structure from i.i.d. samples of a constrained symbolic space.

The primary limitation is a systematic bias toward piece-rich positions, suggesting the model has learned that boards should be full but not yet learned the specific material distribution of puzzle positions. Future work could address this by training longer, using a larger dataset, or designing a structured transition matrix that encodes chess-specific domain knowledge into the D3PM corruption process.

References

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34.

cloneofsimo (2021). d3pm: Discrete denoising diffusion probabilistic models. https://github.com/cloneofsimo/d3pm.

El-Kishky, A., et al. (2025). Generating creative chess puzzles. *arXiv preprint arXiv:2510.23881*.

Heusel, M., et al. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*, 30.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33.

Lichess (2024). Lichess open database. https://database.lichess.org/#puzzles.

Monroe, D. and Chalmers, P. A. (2024). Mastering chess with a transformer model. *arXiv preprint arXiv:2409.12272*.

Ruoss, A., et al. (2024). Grandmaster-level chess without search. *arXiv preprint arXiv:2402.04494*.

Wu, Y., et al. (2025). Implicit search via discrete diffusion: A study on chess. *arXiv preprint arXiv:2502.19805*.