Generating open source chess puzzles - notes

Aatu Selkee

February 5, 2026

Contents

1 Tokenization **2**

2 Model architecture **2**

3 Metrics **3**
3.1 Uniqueness 3
3.2 Counter intuitiveness 3

4 RL **3**

5 23.1. **3**

6 26.1 **4**

7 27.1 **5**

8 28.1 **5**

9 29.1 **5**

10 2.2 **5**

11 4.2 **6**

12 5.2 **7**

1 Tokenization

Tokenization v1

- Board
    - PNBRQKpnbrqk. = 13 tokens
- Side to move
    - wb = 1 tokens (b already counted)
- Castling
    - KQkq. = 0 tokens (already counted)
- En passant
    - abcdefgh = 7 tokens (b already counted)
    - 12345678 = 8 tokens
    - -. = 1 tokens (. already counted)
    - = 16 tokens
- Half move counter
    - 0123456789 = 2 (0 and 9 new tokens)
- Full move counter
    - 0123456789 = 0 (already counted)

= 32 tokens

Total tokens do not match the number of tokens in the paper 31 (the most obvious is that "-" might be replaced with a "."). The length of the produced string is also 76 instead of 77 for some reason. This tokenization feels bad, as e.g. side to move b is completely different to board b (black to move vs black bishop).

Tokenization v2 (my current choice): Length 76, number of tokens 48 (own tokens e.g. for black bishop and black to move)

2 Model architecture

What should be used, pre- or post-normalization ([1] says post (it says that the llama papers use post, but I think they use pre), but [2] and [3] use pre-normalization to improve stability.)

If we want some new experiments, we can find out what masking schedule produces the best results after supervised learning and then apply RL to that model.

3 Metrics

I would love to access the code they used to compute the metrics to see what the differences are.

Computing if the fen position is legal and unique is surprisingly costly (due to repeated Stockfish searches, which can be controlled by setting limits to the depth, nodes and time)

Overlap zero over the 1000 generated positions with the training set (probably due to generating the positions with only one active theme, which basically never happens in the training set).

3.1 Uniqueness

We remove one-movers, I think they might not? Similarly to the paper, we do not check if the best move wins enough (or draws when down in material) as Lichess puzzler does (actually, we do check this!). They mention minor differences in the implementation between Lichess puzzler and their metric computation.

3.2 Counter intuitiveness

I think, in the end they compute $0.8 \cdot depth_{move_{depth} == move_{highdepth}} + 0.1 \cdot value_{captured} \geq \tau_{cnt}$.

Perhaps new thing we could do is that we could modify the counter intuitiveness threshold τ_{cnt} by considering the difficulty rating of the puzzle (more difficult puzzle is more counter-intuitive)

4 RL

[1] did not use the masked diffusion for the RL, and it will probably be harder than with the autoregressive model.

Compute the log-probability of the models in the same way as with autoregressive models (sum the log probabilities of the chosen tokens). The model must be called K times, where K is the amount of tokens (the latter tokens depend on the previous tokens and teacher forcing is not possible I think?). Hence, the computational complexity is a lot higher with masked diffusion than with an autoregressive model. I may be wrong based on algorithm 2 of [4], as we call the model as many times as we have discretized the range [0, 1].

Computation of the log-probability is intractable for these diffusion models. Therefore, as it is needed in RL, we will use the ELBO as a replacement as was done in [5]. We should compute the ELBO in exactly the same way as in SFT (for a fen, sample t, compute α_t and mask with probability, then compute the ELBO as in the paper, finally apply RL with the log probabilities replaced with the negative ELBOs). This paper may also be useful [6].

5 23.1.

	Lichess puzzles	Masked Diffusion	Paper Masked Diffusion
Legal	100%	96.8%	99.72%
Unique	81.7% (95.25%)	9.67%	30.89%
Counter-intuitive	4.3% (2.25%)	23.7%	1.11%
Puzzle	4.1% (2.14%)	0.1%	0.34%

- Previously generated puzzles might not be amazing, as they were generated with a single uniformly random theme. The ones in the dataset have 3 or more themes almost every time. Most positions with a unique solution are mate-in-one.
- Uniqueness computation working pretty well (is probably slightly stricter than in the paper, as we remove one-movers (apart from mate-in-one), although we allow mates-in-one at the end of a puzzle to have multiple solutions)
- Uniqueness computation takes a long time for good puzzles that do not end in a checkmate, as we need to compute the entire solution line for the theme check. (computing the metrics (uniqueness and counter intuitiveness) for 1000 positions from the Lichess puzzle dataset took over 30 minutes, over 1.8 seconds per puzzle)
- FEN $\rightarrow$ themes working well. It was surprisingly well hidden in the Lichess Puzzler repository, but I implemented it to our code.
- Counter-intuitiveness behaves weird, as 1000 Lichess puzzles from the dataset have a realistic value, but our generations should probably not have a counter-intuitiveness of 23.7%...
- It would be amazing, if we could get the implementations of the uniqueness and the counter-intuitiveness metrics
- Talk with Arno Solin.

6 26.1

If the solution to a puzzle is unique, counter-intuitivity metric makes sense for the puzzles we have generated. If the solution is not unique, the engine has potential to find two good moves early and switch the correct move making the puzzle counter-intuitive.

Realized that the correct metrics should look the following (counter-intuitiveness was computed from all positions and not the ones with a unique solution):

Table 1: The proportion of positions satisfying a criterion. An average is taken over 1000 generated positions from the model or 1000 randomly sampled positions from the Lichess Puzzle dataset. The values in the parenthesis are the corresponding values in [1].

	Lichess puzzles	Masked Diffusion
Legal	100%	96.8% (99.72%)
Unique	81.4% (95.25%)	9.71% (30.89%)
Counter-intuitive	5.0% (2.25%)	1.1% (1.11%)
Puzzle	4.1% (2.14%)	0.1% (0.34%)

Could experiment with guidance mechanisms as in [7].

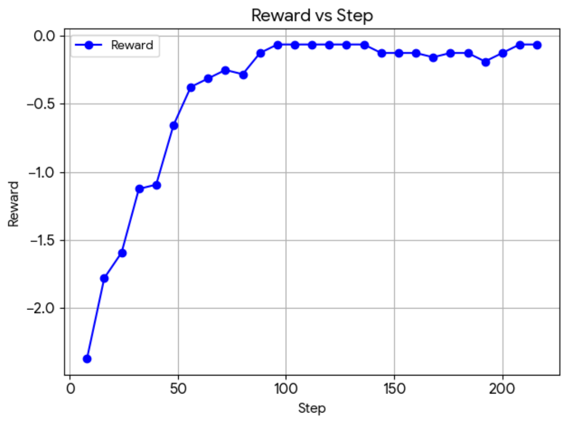
We could use a Gaussian quadrature ELBO estimate that has smaller variance from [6].

7 27.1

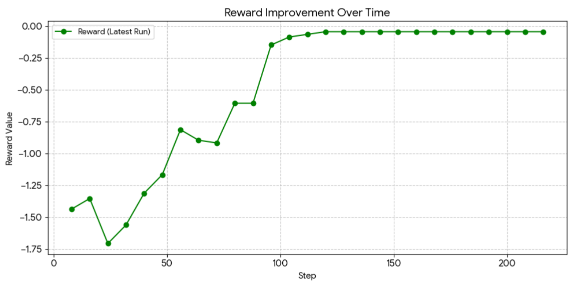
What group size should I use in GRPO? Currently, I set group size to batch size and use just one group.

8 28.1

ESPO now overfits to one position (if the reward is the negative number of nonzero tokens in the generated tokens, the model ends up generating just zeros)



(a) High Variance Estimate



(b) Low Variance Estimate

Currently, I use the Gauss-Legendre quadrature to estimate the integral. I also use antithetic sampling (share masks with current model and reference model) and coupled sampling (use a mask and the inverse mask and average the losses) to reduce variance.

9 29.1

The RL is quite sensitive to the group size G even for the simple reward.

$\mathcal{L}_{old}$ might be different to $\mathcal{L}_{ref}$ (currently I use them as the same). I probably need to change the training loop to store the elbo of the model when the positions are generated.

10 2.2

The low variance elbo estimate is currently working well. Previously, I had a sign error, as the model.elbo_loss returned an upper bound for the negative log likelihood, which we minimized during supervised training. Now I take that into account in the ESPO loss computation.

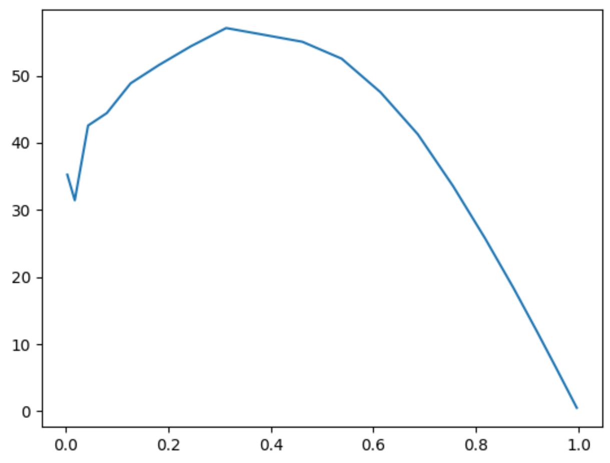
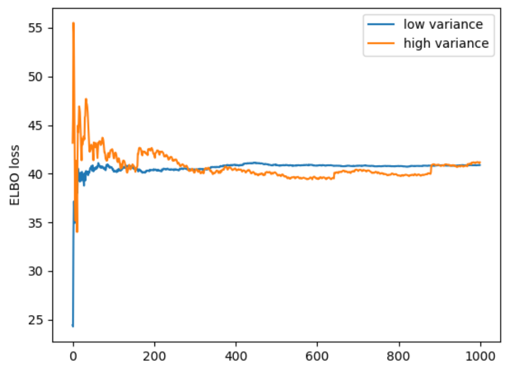
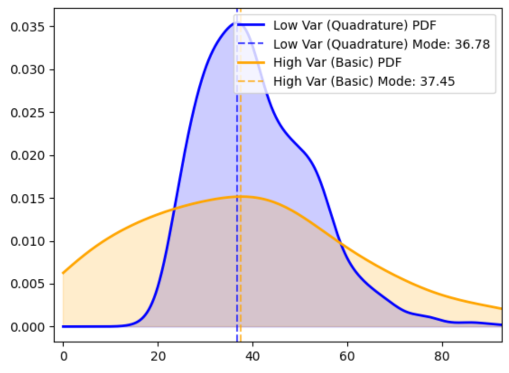


Figure 2: y-axis negative ELBO integrand, x-axis t, negative ELBO is the area under this curve divided by a constant (20, the number of grid points). (Severi told me that this should have a peak in the middle and be low on the edges.)



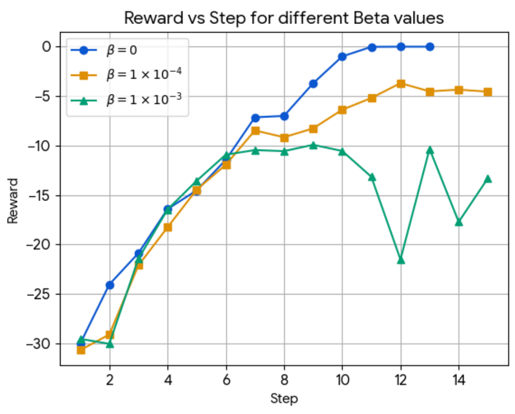
(a) Cumulative average of the negative ELBO estimates.



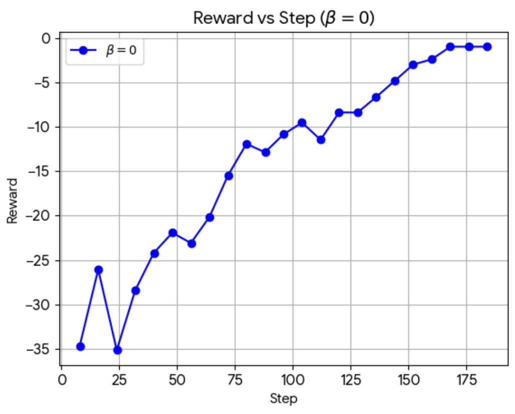
(b) Histograms of the negative ELBO estimates.

11 4.2

If n_gradient_updates_per_generation is set to 1, the performance is great. If n_gradient _updates_per_generation is set higher and importance sampling is hence used, the performance is effected greatly by the ϵ parameter (needs to be larger to not clip as many gradients, e.g. $\epsilon = 0.5$ or even higher in the test case). If epsilon is low (e.g. $\epsilon = 0.1$), the model will not converge to a good solution. There is hence no bug in the code I think, although, the RL seems to converge a lot slower if n_gradient_updates_per_generation is eight compared to when it is one (way more gradient updates AND more new generations as well).



(a) Rewards over time for different betas. If beta is zero, the model overfits completely. For larger betas, the rewards stay lower. No importance sampling is used (only one gradient update is made for each generation).



(b) Rewards over time if eight gradient updates are done for new generations. The convergence is way slower than in the image on the left (even $\frac{184}{8} = 23$ is slower than around 12 for the image on the left).

12 5.2

In the reward function, the authors check if the entropy of the generated board is high enough, which I believe requires the autoregressive decomposition to be computed efficiently (some approximation is needed with the diffusion). Hence, currently, we do not employ the entropy condition for the reward and trust that the other diversity measures are enough.

References

[1] X. Feng, V. Veeriah, M. Chiam, M. Dennis, R. Pachauri, T. Tumiel, F. Barbero, J. Obando-Ceron, J. Shi, S. Singh, S. Hou, N. Tomašev, and T. Zahavy, "Generating creative chess puzzles," 2025.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[4] A. Ruoss, G. Delétang, S. Medapati, J. Grau-Moya, L. K. Wenliang, E. Catt, J. Reid, C. A. Lewis, J. Veness, and T. Genewein, "Amortized planning with large-scale transformers: A case study on chess," 2024.

[5] J. Ou, J. Han, M. Xu, S. Xu, J. Xie, S. Ermon, Y. Wu, and C. Li, "Principled rl for diffusion llms emerges from a sequence-level perspective," 2025.

[6] K. Rojas, J. Lin, K. Rasul, A. Schneider, Y. Nevmyvaka, M. Tao, and W. Deng, "Improving reasoning for diffusion language models via group diffusion policy optimization," 2025.

[7] Y. Schiff, S. S. Sahoo, H. Phung, G. Wang, S. Boshar, H. Dalla-torre, B. P. de Almeida, A. Rush, T. Pierrot, and V. Kuleshov, "Simple guidance mechanisms for discrete diffusion models," 2025.