Watermaking Chess

Eugene Francisco

May 2025

Background

Watermarking refers to the process of embedding information in generated data so as to label that data as having been generated by a model. This project presents a way for agents to watermark their moves in a game of chess. After the game is played, a verifier can review the sequences of moves made by a player and output a prediction for whether the moves were actually made by the watermarking model.

We adapt Kirchenbauer et al.'s watermarking technique for LLMs to this domain. We use LCZero's policy network to query a list of actions and then apply Kirchenbauer's watermarking technique to a strong subset of viable actions. Our watermarking model achieves > 1900 Elo and, run on 7 matches, our watermarking detection achieves a ≤ 0.05 false positive rate and > 0.9 statistical power for detecting watermarking.

Related Work

Just a few days after beginning this project, I came across Kim et al.[1] which also applies Kirchenbauer et al.'s work to Chess, albeit in a more direct approach than I have implemented. They watermarked models with much higher Elos than the model I worked with and crossed an impressive p-value threshold of 10^{-4} for watermark detection in under 10-matches. The approaches that I present here I have applied to weaker models.

Watermarking Methods

First, here is some notation setup.

S	The set of all chess board states.
A	The set of all chess moves.
$\mathcal{P}$	The space of all probability measures over A.

Let $s_t \in S$ be the state of the board at step t. LCZero gives us a policy $\pi_{\text{lcz}} : s_t \to \mathcal{P}$ such that

$\delta = 0$	$\delta = 1$
g2g3	d2d4
d2d4	c2c4
f1g2	b1c3
e1h1	c1g5
c2c4	e2e3
f3d4	d1c2
d4b3	f1d3
c4d5	g5h4
b1c3	d3c4
c1e3	c4e2
b3d4	e1h1
e3d4	f1d1
d1a4	f3e5
c3a4	d4c5
f1d1	h4f6
a4c3	d1d7
d1d3	d7b7
d4f6	e5d3
g2d5	c2b3
c3d5	a2a4
a1d1	c3a2
d5e7	a2c1
d3d6	a1b1
e7f5	b3d1
f5d6	c1b3
d1d6	e2f3
f2f3	b3a5
d6d8	a5b3
b2b4	b3d4
g3g4	e3d4

Figure 1: Example opening moves from non-watermarked model (left) against watermarked model (right). Note that we used $\delta = 0.6$ for our trials and the choice for $\delta = 1$ here is only meant to highlight the preference towards green.

[1] https://arxiv.org/abs/2605.14283

$\pi_{\text{lcz}}(s_t)$ is a probability distribution over the feasible actions at s_t. To choose a watermarked move, we perform a (1) *nucleus sampling procedure* to make a subset of allowable actions and (2) a *coloring procedure*, and (3) a *selection procedure* to actually choose the move to take. Steps (2) and (3) are adapted from Kirchenbauer et al.

Nucleus sampling procedure: To remove clutter, let $\pi_{\text{lcz}}^{(t)}$ be the policy of actions at state s_t. Fix a constant $q \in (0, 1]$ and $\ell \in \mathbb{Z}_+$. From A, pick the actions with largest probability so that the chosen actions collectively have measure at most q. Call this set $Q \subseteq A$.[2] Finally, ensure that Q has at least ℓ elements; if elements with top ℓ probability have cumulative probability greater than q, still include it in Q. The inclusion of ℓ here is meant to allow a balance between statistical signal and model performance. If Q only has one element, there is no room for statistical signal. At the same time, pushing q too high to force larger Qs can encourage fumbles. ℓ allows us to keep q small while still giving diverse move choices.

Coloring procedure: The procedure here is taken from Kirchenbauer et al's watermarking procedure for LLMs. Fix a $\gamma \in (0, 1)$ in advance. Using s_t and a universal hash, assign to each action $a \in A$ a color $c^{(s_t)}(a)$ that is either green or red. Pick the hashing technique so that

$$\mathbb{P}(c^{(s_t)}(a) = \text{green}) = \gamma$$

where the probability here is for any fixed board state picking a uniformly and randomly.[3] Let $G, R \subseteq Q$ be the actions colored green and red that have also been selected in step (1).

Selection Procedure: Fix $\delta \in \mathbb{R}_+$ in advance. Let $\ell_a \in \mathbb{R}$ be the pre-softmax logit corresponding to action $a \in A$. For each action $a \in G$, update ℓ_a with

$$\ell_a \leftarrow \ell_a + \delta.$$

For each action $a \in A - Q$, update ℓ_a with

$$\ell_a \leftarrow -\infty.$$

Pass the updated logits through a $\mathrm{softmax}$ to yield an updated policy $\sigma_{\text{lcz}}^{(t)}$. Sample from $\sigma_{\text{lcz}}^{(t)}$ and return the sampled action.

[2] In other words, the actions in Q should all have (1) probability greater than anything in $A - Q$, (2) $\sum_{a \in Q} \pi_{\text{lcz}}^{(t)}(a) \leq q$, and (3) Q should be the largest set with properties (1) and (2).

[3] In other implementations, we select a partition of the action space into a set of size exactly $\lceil \gamma \rceil$ and $|A| - \lceil \gamma \rceil$ uniformly but here each action is assigned colors independently of the rest.

Watermark Detection

Let τ be the trajectory of actions made by both players; i.e., $\tau = [a_1, b_1, a_2, b_2, \ldots,]$ where each $a_i, b_i \in A$ and a_i correspond to actions played by white while b_i correspond to actions played by black. The goal is to determine whether actions played by white or black were chosen by a model. Suppose we want to know whether white played. To do so, perform the following:

(i) Keep a counter of green actions $k \leftarrow 0$. Keep a counter of total actions n.

(ii) For each action a_t made by white:

  (a) Let s_t be the state before a_t was taken.

  (b) Use steps (1) and (2) of the watermarking procedure to attain sets $G, R \subseteq Q$ of green and red actions.

  (c) If $a_t \in G$, increment $k \leftarrow k + 1$.

  (d) Increment $n \leftarrow n + 1$

Let K be the random variable for how many green actions are observed in n actions under the policy that picked $\{a_i\}$. We will now perform a statistical significance test to find the probability of observing k or more green actions under the null hypothesis that watermarking was not used (or a different model/human played these actions). By the way we have chosen to color actions, we have

$$\mathbb{P}(c(a_t) = \text{green} \mid \text{ no watermarking }) = \gamma.$$

Since watermarking boosts green actions' logits, we must have

$$\mathbb{P}(c(a_t) \mid \text{watermarking}) > \gamma.$$

Let ρ be the probability of coloring a color green under the policy that picked $\{a_i\}$. This give us our hypotheses:

$$\begin{aligned} H_0 &\quad \rho = \gamma \\ H_1 &\quad \rho > \mu_0. \end{aligned}$$

Observe that the colors $\{c(a_t)\}$ are i.i.d. under H_0. We care about $\mathbb{P}(K \geq k \mid H_0)$. Since $K|H_0 \sim \mathrm{Binom}(n, \gamma)$, this is the tail probability of a binomial. Let p be the resulting probability. We reject the null if $p \leq 0.05$. This fixes our false-positive rate at 0.05.

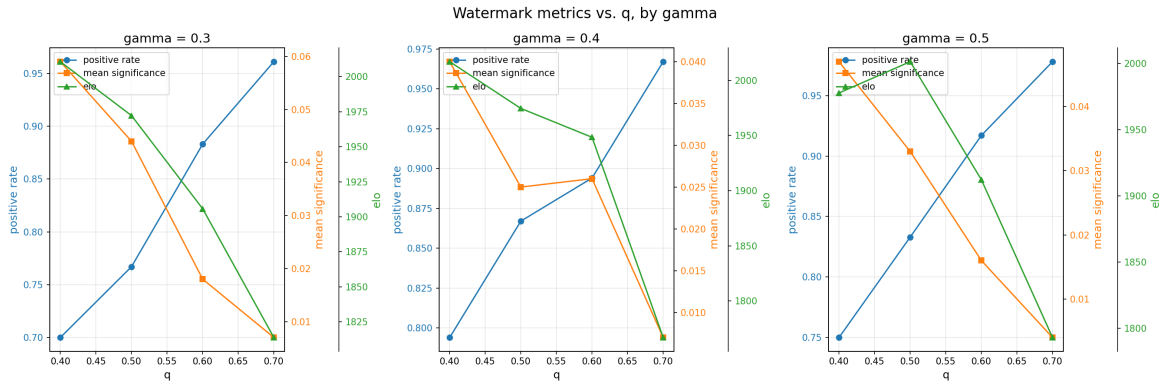

Figure 2: Each q, γ datapoint comes from one tournament of 1,260 matches; 140 matches for each of the 9 references. p-values are calculated after 7 matches; true positive rates and mean-p-value are estimated by averaging across all the references for a particular q, γ.

Model Evaluation

We downloaded weights for LCZero from Maia[4] which gave us nine reference models with Elos in $\{1100,\ 1200,\ \ldots,\ 1900\}$. To evaluate the watermarked mode, the target model is pitted against all of these reference models over the course of several matches. The number of points the target model scores over the reference models is recorded (1 for wins, 0.5 for draws, 0 for losses) and used to compute the Elo, which is the number R that satisfies the equation

$$\sum_i \frac{n_i}{1 + 10^{(R_i - R)/400}} = \sum_i n_i s_i \qquad (1)$$

where here n_i is the number of games played against reference i, R_i is reference i's known Elo, and s_i is the score of the target's match against reference i.

Implementation Details

For the policy, we used LCZero's default weights. These weights provide a policy that are typically used for a Monte Carlo Tree Search (MCTS); we use the policy network on its own without doing a tree search. If we act on the policy greedily (picking the move with highest probability) this default model has $\approx 2{,}200$ Elo.

Nucleus Sampling Procedure: We used $q = 0.6$ for the strongest model but results for other values of q are available in the Results section. *Coloring procedure:* After experimenting with several values, we settled on $\gamma = 0.5$; experiments with different values of γ are in the Results section. To hash current states into a coloring, do the following.

(i) We first flatten the board state into an observation $v \in \mathbb{R}^{65}$.[5]

(ii) The bytes from this board state in memory are summed with a private key that is fixed in advance and hashed using SHA-256 to generate a local seed for this state.

(iii) The local seed is used to generate a vector $r \in [0,1]^{|Q|}$ and we take any index r_i which is $\leq \gamma$ to color the ith action in Q as green.

Selection Procedure: We experimented with several values before settling on $\delta = 0.6$.

Evaluation: We evaluated the watermarked model's Elo by playing 20 matches against each of the 9 reference models. We estimated the true positive rate of detection by playing 100 games of 7-matches each, where each game yields one p-value of detection as described in the Watermark Detection section.

Results

Watermarking these chess models always involves a tradeoff between model performance and watermarking signal. We can achieve higher watermarking signal by boosting the advantage of actions marked green (making δ larger). The smaller we make q, the more likely we are to pick optimal options but

[4] https://www.maiachess.com

[5] An 8×8 board state appended with the current player. Each of the first 64 elements is a number representing pieces.

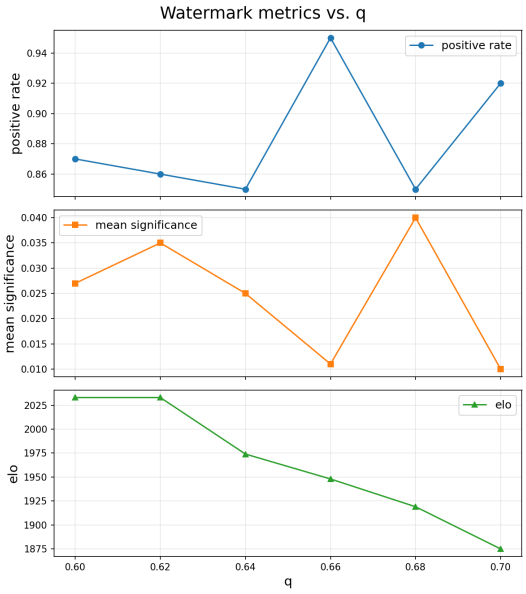

Figure 3: The true-positive rate for watermark detection on 7-matches, average observed statistical significance, and model elo for watermark models with $q \in \{0.62, 0.64, \ldots, 0.7\}$. We have fixed $\gamma = 0.4$, $\delta = 0.6$, and $\ell = 1$ here.

the less likely it will be that a δ boost will affect our choice. If q is larger, we are more apt to pick poorer moves but there will be more freedom to boost green moves. We tried to achieve a compromise between these two.

As can be seen in Figure 3, Elos, positivity rates, and average observed significance varies widely across trials, but even small increases in q can drop the model's Elo substantially. Figure 3's trials were conducted with $\ell = 1$; this forces Q to have at least one member when picking actions. But even for large values of q, it is still possible Q is left with only one element, which leaves no room for watermarking.

To improve watermarking signal, we tried $\ell = 2$ to force Q to have at least two moves. The results are shown in (Figure 2). As a function of γ (Figure 2), we can see the tradeoff between model performance and watermarking signal more clearly. The highlight of these trials is $\gamma = 0.5$ and $q = 0.6$, which attained an Elo of 1912, a power of 0.917, and an average p-value of 0.016.