ON THE LEARNABILITY OF WATERMARKS FOR LANGUAGE MODELS

Chenchen Gu, Xiang Lisa Li, Percy Liang, Tatsunori Hashimoto
Stanford University
{cygu, xlisali, thashim}@stanford.edu, pliang@cs.stanford.edu

ABSTRACT

Watermarking of language model outputs enables statistical detection of model-generated text, which can mitigate harms and misuses of language models. Existing watermarking strategies operate by altering the decoder of an existing language model. In this paper, we ask whether language models can directly *learn* to generate watermarked text, which would have significant implications for the real-world deployment of watermarks. First, learned watermarks could be used to build open models that naturally generate watermarked text, enabling watermarking for open models, where users can control the decoding procedure. Second, if watermarking is used to determine the provenance of generated text, an adversary can hurt the reputation of a victim model by spoofing its watermark and generating damaging watermarked text. To investigate the learnability of watermarks, we propose watermark distillation, which trains a student model to behave like a teacher model that uses decoding-based watermarking. We test our approach on three decoding-based watermarking strategies and various hyperparameter settings, finding that models can learn to generate watermarked text with high detectability. We also find limitations to learnability, including the loss of watermarking capabilities under fine-tuning on normal text and high sample complexity when learning low-distortion watermarks.[1]

1 INTRODUCTION

As language models (LMs) become more capable and widely used, watermarking LM outputs becomes increasingly important to mitigate potential harms and misuses of LMs. Watermarking enables statistical detection of LM-generated text, which enables enforcing policies on LM usage, e.g., removing LM-generated disinformation from social media platforms or detecting academic dishonesty. Another proposed use case of watermarking is identifying the provenance of text, i.e., tracing text to the specific LM that generated it (Abdelnabi & Fritz, 2021; Kuditipudi et al., 2023).

Recent works have shown that it is possible for an LM provider to inject specific, known watermark signals into text using specialized decoding algorithms (Kirchenbauer et al., 2023a; Aaronson, 2023; Kuditipudi et al., 2023), but little is known about whether these watermarks are learnable by a model. The learnability of watermarks has significant implications for the real-world deployment of watermarks, as it could enable downstream applications and adversarial spoofing attacks.

In this work, we study the learnability of watermarks by studying *weights-based watermarking*, which involves learning parameters for a language model that cause it to generate watermarked text under its natural sampling distribution, without using a special decoding-time watermarking algorithm. Our investigation is motivated by its relevant implications for two applications: (i) developing watermarking for open language models and (ii) spoofing watermarks.

First, existing watermarking methods depend upon using a specialized decoding algorithm, making them too inflexible for open LMs. For open LMs, where the weights are released, a user can use an ordinary decoding algorithm and generate non-watermarked text, whether intentionally or not. We find that weights-based watermarking works with standard decoding strategies, removing the

[1]We publicly release code and trained model weights at https://github.com/chenchenygu/watermark-learnability.

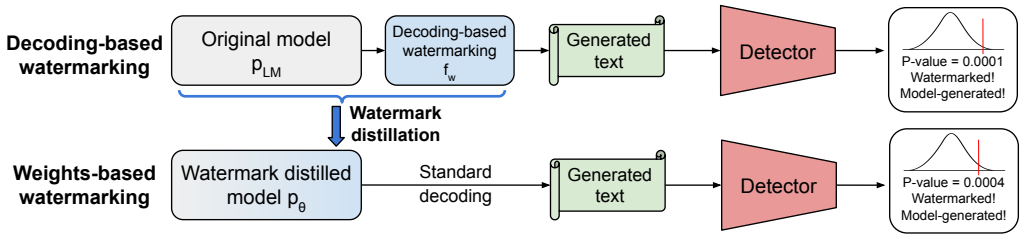


Figure 1: Decoding-based watermarking (top) versus weights-based watermarking (bottom). Decoding-based watermarking requires a specialized decoding algorithm f_{w} to generate watermarked text, whereas weights-based watermarking can use standard decoding to generate watermarked text directly from the model, using just its weights. Watermark distillation enables weights-based watermarking by training a student model p_θ to behave like the teacher model p_{LM} with decoding-based watermarking strategy f_{w}.

reliance on a specialized decoder. This makes it a promising first step towards developing watermarking for open LMs. However, we also find that weights-based watermarking capabilities can be removed by fine-tuning on normal text, indicating that improving robustness to fine-tuning is an important remaining challenge.

Second, in watermark spoofing attacks, an adversary outputs text that contains the watermark signal from a victim LM (Sadasivan et al., 2023). If watermarking is used to identify the provenance of text, then an attacker could attribute damaging text to the victim LM and hurt its reputation. We find that the learning of weights-based watermarking can enable spoofing attacks, and we demonstrate a proof-of-concept attack on an instruction-following chat model. The possibility of spoofing attacks suggests that watermarking should not be used to attribute provenance or blame to a specific LM. Instead, watermarking should only be used to statistically detect LM-generated text, which can be used for tasks such as finding infractions of policies on LM usage.

To enable weights-based watermarking, we propose logit-based and sampling-based watermark distillation, two simple methods for a student model to learn weights-based watermarking from a teacher model with decoding-based watermarking. Intuitively, in logit-based watermark distillation, the student model is trained to match the next token distribution outputted by the teacher model using decoding-based watermarking. In sampling-based watermark distillation, the teacher model with decoding-based watermarking is first used to generate watermarked samples. Then, the student model is fine-tuned on these watermarked samples.

We experiment with three decoding-based watermarking strategies: KGW (Kirchenbauer et al., 2023a; Zhao et al., 2023a), Aar (Aaronson, 2023), and KTH (Kuditipudi et al., 2023), and various values for their hyperparameters that control the level of distortion induced by watermarking. We find that watermarks and hyperparameter settings vary in their degree of learnability. In each watermarking strategy, higher-distortion hyperparameter settings are successfully learned by both forms of watermark distillation (median p-values less than 0.0001). Lower-distortion watermarks and hyperparameter settings are more challenging and less sample efficient to learn, but not unlearnable, as the p-values are still noticeably smaller than the non-watermarked baseline of 0.5.

2 Background and Notation: Decoding-based Watermarking

We study autoregressive language models $p_{\mathrm{LM}} : \mathcal{V}^* \to \Delta(\mathcal{V})$ that map from a prefix string $x \in \mathcal{V}^*$ to a next token distribution over the vocabulary $\mathcal{V}$. Informally, a decoding-based watermarking strategy f_{w} uses a watermark key ξ to modify the model's original next token distribution $p_{\mathrm{LM}}(\cdot \mid x)$ into a new distribution for generating watermarked text, which has a watermark signal embedded. The watermark detection algorithm f_{d} looks for this signal using the same watermark key ξ.

Formally, we define a decoding-based watermarking strategy to be a function

$$f_{\mathrm{w}} : \Delta(\mathcal{V}) \times \mathcal{V}^* \times \Xi \to \Delta(\mathcal{V}) \tag{1}$$

where Ξ is the set of possible watermark keys. This function f_{w} outputs a distribution $p_{\text{w}}(\cdot \mid x)$ from which to generate the next token in the watermarked text, given an original next token distribution $p_{\text{LM}}(\cdot \mid x)$, input text x, and watermark key $\xi \in \Xi$.

We define a watermark detection algorithm to be a function

$$f_{\text{d}} : \mathcal{V}^* \times \Xi \to [0, 1] \,. \tag{2}$$

Given some text x and watermark key ξ, f_{d} outputs a p-value with respect to the null hypothesis that x is independent of f_{w} with key ξ. Informally, f_{d} computes a test statistic that measures the strength of the watermark signal, then computes a p-value using the distribution of the test statistic under the null hypothesis. If the p-value is below a given significance level, the null hypothesis is rejected and the text is detected as watermarked. Slightly imprecisely, rejecting the null hypothesis means the text is detected as model-generated.[2]

In this paper, we consider three decoding-based watermarking strategies: Algorithm 2 in Kirchenbauer et al. (2023a), the Gumbel softmax scheme in Aaronson (2023), and the exponential minimum sampling scheme in Kuditipudi et al. (2023). Using the authors' names and initials, we refer to these as KGW, Aar, and KTH, respectively. We briefly describe these watermarking strategies below. See Appendix D for additional details and formal definitions.

KGW: green list bias. In the KGW watermarking strategy (Kirchenbauer et al., 2023a), when generating the next token, the vocabulary is pseudorandomly split into a "green list" and "red list" by hashing the previous token using the watermark key ξ. The green list contains watermark hyperparameter $\gamma \in (0, 1)$ proportion of the vocabulary. Then, before the model's logits are converted to probabilities via the softmax function, hyperparameter $\delta > 0$ is added to the logits of the green list tokens. This procedure makes green list tokens more likely in watermarked text than in non-watermarked text. So, at detection time, if the proportion of green list tokens in a text is much greater than γ, then the p-value is small.

More generally, the previous k tokens can be hashed, where k is a hyperparameter. Values of $k > 1$ are investigated by Kirchenbauer et al. (2023b), finding that lower k leads to more repetitive outputs. When $k = 0$, the green and red lists are fixed, regardless of the previous tokens. $k = 0$ was proposed by Zhao et al. (2023a) as Unigram-Watermark, a variant of KGW, but we will denote it as KGW $k = 0$ to simplify notation.

KGW distorts model outputs by upweighting green list tokens, increasing perplexity of generated texts computed by a larger model (Kirchenbauer et al., 2023a). Increasing the bias hyperparameter δ increases detectability, i.e., smaller p-values, but also increases distortion.

Aar: boosting continuous hash scores. The Aar watermarking strategy (Aaronson, 2023) hashes the previous k tokens using key ξ (where k is a hyperparameter) to obtain a score r_i for each token $i \in \mathcal{V}$, where each r_i is uniformly distributed in $[0, 1]$. Let p_i be the original model probability for token i. Then, the next generated token is deterministically chosen to be the token i which maximizes r_i^{1/p_i}, i.e., a token with both a high original probability p_i and high hash score r_i. This procedure boosts the hash scores of tokens in watermarked text compared to non-watermarked text. So, at detection time, if the hash scores r_i of the tokens in the observed sequence are high, then the p-value is low.

Since Aar deterministically selects the next token based on the previous k tokens and the original model probabilities, Aar can lead to repetitive text, especially for small k (Kuditipudi et al., 2023). Increasing k decreases repetitiveness, as larger k-grams are less likely to be repeated, but the watermark also becomes less robust to edits, as each token edit affects the hash scores for $k + 1$ tokens.

KTH: robust sequence alignment. The KTH watermarking strategy (Kuditipudi et al., 2023) is similar to Aar, but instead of hashing previous tokens to obtain the scores r_i, the scores are obtained from the next element in the key sequence ξ. In KTH, $\xi = \xi^{(1)}, \ldots, \xi^{(m)}$ where each $\xi^{(j)} \in [0, 1]^{|\mathcal{V}|}$

[2]This is slightly imprecise because model-generated text is not the only text that can be deliberately watermarked. For example, a human could write watermarked text by manually following the watermarking algorithm. However, for most practical use cases, such as detecting academic dishonesty, this minor imprecision is not an issue because either way, the user is doing something suspicious and unusual.

contains the scores, with entries uniformly distributed across $[0, 1]$. Then, to generate the j-th token in the sequence, KTH deterministically chooses the token i that maximizes $\left(\xi_i^{(j)}\right)^{1/p_i}$. Note that m should be larger than the maximum generation length. To allow different generations from the same prompt, before generating each sequence, ξ can be shifted by some random τ, i.e., $\xi' = \xi^{(1+\tau \bmod m)}, \ldots, \xi^{(m+\tau \bmod m)}$. To study the impact of these shifts on learnability, we introduce a hyperparameter $s \in [1, m]$ for how many shift values τ are possible.[3] Increasing s expands the range of possible model generations.

At detection time, to be robust to text edits and shifts, the test statistic quantifies how well a text x can be aligned with the key sequence ξ. More specifically, the test statistic computes a minimum Levenshtein distance using the alignment cost $d(x, \xi) = \sum_{t=1}^{\text{len}(x)} \log(1 - \xi_{x_t}^{(t)})$. A lower (more negative) test statistic indicates stronger watermark signal. To compute p-values, the observed test statistic is compared to a reference distribution of test statistics of non-watermarked texts. Letting T be the number of samples in the reference distribution, the p-values computing using this method are lower bounded by $\frac{1}{T+1}$.

3 METHODS

Problem statement. Given a teacher model p_{LM}, decoding-based watermarking strategy f_{w}, and key ξ, the goal is to learn a student model p_θ whose sampling distribution naturally generates watermarked text. Specifically, letting f_{d} be the detection algorithm corresponding to f_{w}, if p_θ generates text y with small detection p-value $f_{\text{d}}(y, \xi)$ with probability similar to that of p_{LM} with f_{w}, then p_θ has learned a *weights-based watermarking strategy*, since p_θ has learned to generate watermarked text using just its weights. Figure 1 illustrates decoding-based versus weights-based watermarking.

Next, we present two methods for learning a weights-based watermarking strategy: logit-based watermark distillation and sampling-based watermark distillation, which fall under the broader category of knowledge distillation (Hinton et al., 2015; Kim & Rush, 2016).

3.1 LOGIT-BASED WATERMARK DISTILLATION

In logit-based watermark distillation, we train the student model p_θ to behave as if it had decoding-based watermarking strategy f_{w} applied. Specifically, given an input x, we want the student model's next token distribution $p_\theta(\cdot \mid x)$ to match $f_{\text{w}}\left(p_{\text{LM}}\left(\cdot \mid x\right), x, \xi\right)$, the next token distribution outputted by the teacher model p_{LM} with decoding-based watermarking strategy f_{w} and key ξ. So, given a dataset of texts $\mathcal{D}$, the training objective is to minimize the mean KL divergence between the teacher and student next token distributions on all prefixes in $\mathcal{D}$, given by

$$\mathcal{L}_{\text{logit}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{t=1}^{\text{len}(x)} D_{\text{KL}}\left(f_{\text{w}}\left(p_{\text{LM}}\left(\cdot \mid x_{<t}\right), x_{<t}, \xi\right) \,\|\, p_\theta\left(\cdot \mid x_{<t}\right)\right). \tag{3}$$

The teacher model p_{LM} is frozen. This approach requires that p_{LM} and p_θ have the same tokenizer and vocabulary so that the logits can be aligned between the two models. It is also helpful if p_{LM} and p_θ share the same model architecture, as then we can initialize p_θ to p_{LM}. Note that the ground truth next tokens x_t from dataset $\mathcal{D}$ are not used in the loss function, so $\mathcal{D}$ does not need to be watermarked text. Standard datasets containing non-watermarked human-generated text can be used.[4]

3.2 SAMPLING-BASED WATERMARK DISTILLATION

Sampling-based watermark distillation has two stages. First, we generate watermarked text from teacher model p_{LM} with decoding-based watermarking strategy f_{w} applied using key ξ. Then, we fine-tune the student model p_θ on this watermarked text using the standard language modeling cross-entropy loss.

[3]We space the s shifts evenly across $[1, m]$, i.e., the set of possible shifts τ is $\{i \cdot \lfloor m/s \rfloor : 0 \leq i < s\}$.

[4]If $\mathcal{D}$ is non-watermarked text, then it theoretically might be out of distribution for p_θ to autoregressively generate watermarked text, since p_θ would be conditioning on the watermarked text it has already generated. However, empirically, we find that logit-based distilled models can learn to generate watermarked text.

Formally, given a set of prompts $\mathcal{P}$, for each prompt $z \in \mathcal{P}$, we generate a watermarked completion sequence $x = x_1 x_2 \cdots x_n$, where each sampled token $x_t \sim f_{\mathrm{w}}\left(p_{\mathrm{LM}}\left(\cdot \mid z x_{<t}\right), z x_{<t}, \xi\right)$. Let the fine-tuning dataset $\mathcal{D}$ consist of these watermarked generations x.[5] Then, we train p_θ to minimize the cross-entropy loss on $\mathcal{D}$, given by

$$\mathcal{L}_{\text{sampling}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{t=1}^{\operatorname{len}(x)} -\log p_\theta\left(x_t \mid x_{<t}\right). \tag{4}$$

Here, p_{LM} and p_θ do not need to share the same tokenizer or vocabulary. However, sampling-based watermark distillation is less efficient than logit-based watermark distillation due to autoregressively generating watermarked text in the first stage.

4 Experimental setup

We run experiments to evaluate how well logit-based and sampling-based watermark distillation can learn weights-based watermarking from the decoding-based watermarking strategies seen in §2. Ideally, we want weights-based watermarking to match decoding-based watermarking in terms of watermark detectability and generation quality.

4.1 Watermarking strategies and hyperparameters

We experiment with the three decoding-based watermarking strategies discussed in §2. We use various hyperparameter settings to vary the level of distortion induced by the watermarks. Specifically, we test KGW with $k = \{0, 1, 2\}$, $\gamma = 0.25$ and $\delta = \{1, 2\}$,[6][7] Aar with $k = \{2, 3, 4\}$, and KTH with key length 256 and number of shifts $s = \{1, 2, 4, 256\}$.

4.2 Training

For each decoding-based watermarking strategy, we test logit-based and sampling-based watermark distillation for learning weights-based watermarking.

For logit-based watermark distillation, we use Llama 2 7B (Touvron et al., 2023) as both the teacher and student models (the student model is initialized with the teacher model weights). We distill using a subset of OpenWebText (Gokaslan et al., 2019) for 5,000 steps with a batch size of 64 sequences, sequence length of 512 tokens,[8] maximal learning rate of 1e-5, and cosine learning rate decay with a linear warmup. Full training details are in Appendix E.1.

For sampling-based watermark distillation, we also use Llama 2 7B as both the teacher and student models. First, we use Llama 2 7B with a decoding-based watermarking strategy to generate 640,000 watermarked samples of length 256 tokens, prompted with 50-token prefixes from OpenWebText. Then, we fine-tune Llama 2 7B on the watermarked samples for 1 epoch of 5,000 steps, with a batch size of 128 sequences, sequence length of 256 tokens, maximal learning rate of 1e-5, and cosine learning rate decay with a linear warmup. Full training details are in Appendix E.2.

In Appendix F, we perform sampling-based watermark distillation experiments where the teacher and student models have different tokenizers and sizes, using Llama 2 7B as the teacher model and Pythia 1.4B as the student model (Biderman et al., 2023).

4.3 Evaluation and metrics

Evaluation procedure. As in Kirchenbauer et al. (2023a) and Kuditipudi et al. (2023), we evaluate on generations prompted by prefixes from the RealNewsLike subset of the C4 dataset (Raffel et al.,

[5]Ideally, the intended use case and domain of the student model p_θ should inform the choices of the set of prompts $\mathcal{P}$ and teacher model p_{LM}. However, empirically, we find that sampling-based watermark distillation is fairly robust to domain shifts (see §4.3 and Appendix I).

[6]Because we always use $\gamma = 0.25$, we sometimes omit explicitly stating the value of γ to simplify notation.

[7]We exclude $k = 2, \delta = 1$ since we find that $k = 2, \delta = 2$ already exhibits lower learnability.

[8]For KTH we use a batch size of 128 and sequence length of 256 tokens because we use key length 256.

2020). For each decoding-based watermarking strategy and distilled model, we generate 5,000 200-token completions from 50-token prompts from the validation split. We use standard sampling with temperature 1 for the main results, and investigate the model's robustness to different decoding parameters in Appendix C. We include evaluations on additional datasets in Appendix I.

We choose metrics to evaluate two properties: watermark detectability and generation quality.

Watermark detectability. We compute the median watermark detection p-value across generations. Note that the p-values for the KTH watermark are lower bounded by how many samples T we compute in the reference distribution. Similar to Kuditipudi et al. (2023), we use $T = 10{,}000$, so the p-values are lower bounded by 1e-4. To make finer-grained distinctions in watermark strength below this lower bound, we also compute the median test statistic (discussed in §2) to evaluate KTH watermark strength. A lower (more negative) test statistic indicates higher watermark detectability.

We also compute the AUROC (area under the receiver operating characteristic curve) for classifying model-generated versus human-generated text using the watermark detection p-values/test statistics. We compute the AUROC using an equal number of model-generated and human-generated texts, all of the same length.

Generation quality. We use Llama 2 13B to compute the mean perplexity of generations. Lower perplexity tends to indicate higher quality and fluency, but repetitive text also achieves low perplexity. So, to evaluate repetitiveness, we compute the mean seq-rep-3 of generations, which is the proportion of duplicate 3-grams in a sequence, given by $1 - \frac{\text{\# of unique 3-grams}}{\text{\# of 3-grams}}$ (Welleck et al., 2020).

Comparisons. For both watermark distillation methods, for each decoding-based watermarking strategy f_{w}, we compare the teacher model with f_{w} applied (denoted by "Decoding") against the distilled student model (denoted by "Logit" and "Sampling" for logit-based and sampling-based watermark distillation, respectively). As a baseline for generation quality, we use the base student model with no watermarking or distillation (denoted by "Base student").

5 RESULTS

Table 1 contains results for the logit-based and sampling-based watermark distillation experiments. The two watermark distillation methods exhibit similar trends. Both forms of watermark distillation successfully learn higher-distortion watermarks,[9] achieving small p-values and high detectability. In some watermarks, e.g., KGW $k = 0$, watermark distillation matches the p-values achieved by decoding-based watermarking. In other watermarks, watermark distillation does not achieve as small watermark detection p-values as decoding-based watermarking, but for higher-distortion watermark hyperparameter settings (smaller k and larger δ for KGW, smaller k for Aar, and smaller s for KTH), the p-values are still sufficiently small to enable high detectability, as shown by the high AUROC values. Figure 4 in Appendix A contains empirical CDFs of the distributions of p-values across generations, showing that for higher-distortion watermarks, the majority of generations from the watermark distilled models have small p-values.

Within each watermark type, p-values from logit-based and sampling-based distillation are larger for lower-distortion hyperparameter settings, indicating that lower-distortion watermarks are harder to learn. However, these watermarks are still learned to some degree, as the p-values are noticeably smaller than the non-watermarked baseline of 0.5, and the AUROC values are noticeably higher than the non-watermarked baseline of 0.5. In Appendix G, sample complexity experiments show that more training samples and steps lead to smaller p-values for both logit-based and sampling-based distillation, with no sign of convergence. In addition, we find that when we train logit-based watermark distillation on KGW $k = 2, \delta = 2$ for five times longer (25,000 steps) on more data, the median p-value decreases from 0.1 to 0.012. This suggests that lower-distortion watermarks are less sample efficient to learn, but still learnable, given sufficient training data and steps.

Compared to decoding-based watermarking, watermark distillation does not achieve as optimal a tradeoff between generation quality and detectability. For KGW and KTH, both watermark distillation methods achieve slightly to moderately higher perplexity and similar or larger p-values com-

[9]Here, we are using "distortion" somewhat informally, roughly meaning how much of a difference watermarking causes in terms of generation quality, behavior, etc.

Watermark		p-value (↓) (KTH test statistic (↓))			AUROC (↑)			Perplexity (↓)			seq-rep-3 (↓)		
		Decoding	Logit	Sampling	Decoding	Logit	Sampling	Decoding	Logit	Sampling	Decoding	Logit	Sampling
KGW	$k = 0, \delta = 2$	6e-16	2e-17	2e-15	1.00	1.00	1.00	17.5	17.3	20.3	0.05	0.05	0.05
	$k = 1, \delta = 2$	4e-18	7e-09	8e-07	1.00	1.00	1.00	16.5	17.6	19.2	0.04	0.03	0.04
	$k = 2, \delta = 2$	9e-18	1e-01	1e-01	1.00	0.80	0.74	16.8	17.7	19.8	0.03	0.02	0.03
	$k = 0, \delta = 1$	5e-04	3e-05	1e-03	0.98	0.99	0.98	13.0	12.9	15.7	0.03	0.03	0.03
	$k = 1, \delta = 1$	1e-05	7e-03	2e-02	0.99	0.91	0.87	12.7	13.1	14.9	0.03	0.03	0.03
Aar	$k = 2$	1e-75	2e-12	3e-17	1.00	1.00	0.98	6.5	10.8	7.7	0.34	0.11	0.34
	$k = 3$	5e-73	1e-01	6e-03	1.00	0.78	0.88	9.5	11.6	10.5	0.14	0.04	0.17
	$k = 4$	4e-72	4e-01	3e-01	1.00	0.58	0.65	10.7	11.8	11.9	0.09	0.03	0.11
KTH	$s = 1$	1e-04 (-593)	1e-04 (-565)	1e-04 (-561)	1.00	1.00	1.00	10.5	16.5	15.1	0.03	0.04	0.03
	$s = 2$	1e-04 (-596)	1e-04 (-476)	1e-04 (-525)	1.00	0.99	0.99	10.7	16.3	13.4	0.03	0.04	0.03
	$s = 4$	1e-04 (-594)	1e-03 (-438)	1e-04 (-487)	1.00	0.96	0.99	10.6	14.2	12.5	0.03	0.04	0.04
	$s = 256$	1e-04 (-594)	8e-02 (-423)	1e-04 (-453)	1.00	0.85	0.97	10.8	11.3	11.3	0.03	0.04	0.04
Base student			5e-01			0.50			11.8			0.03	

Table 1: Results for logit-based and sampling-based watermark distillation experiments. Within each watermark type (KGW, Aar, and KTH), the hyperparameter rows go from higher-distortion to lower-distortion moving down the table. Higher-distortion watermarks are successfully learned with small p-values and high detectability. Lower-distortion watermarks are harder to learn, as shown by the larger p-values, but they are still learnable, just less efficiently and strongly.

pared to decoding-based watermarking. For Aar, watermark distillation achieves similar or lower seq-rep-3 as decoding-based watermarking, but larger p-values. This suggests that to learn weights-based watermarking, logit-based and sampling-based watermark distillation incur some cost to the tradeoff between generation quality and detectability.

While logit-based and sampling-based watermark distillation show similar trends, there are some interesting differences. We defer this discussion to Appendix B due to space constraints. However, recall that logit-based and sampling-based distillation have different requirements (e.g., access to logits and shared tokenizer vs. access to samples and autoregressive generation, see §3.1 and §3.2), so they should not be compared solely on performance. So, logit-based and sampling-based distillation are each suitable and applicable for different settings, so neither is strictly better than the other in all scenarios.

Robustness to text edits. We test the robustness of weights-based watermarking to edits by randomly corrupting generated text from the logit-based and sampling-based watermark distilled Llama 2 7B models with varying proportions of tokens randomly edited. See Appendix J for full experimental details. As shown in Figure 2, the detection p-values of all three watermark types are robust to moderate edit proportions, up to around 20–30%. At higher edit proportions, up to around 60–70%, KTH is significantly more robust to edits than KGW and Aar, consistent with the findings of Kuditipudi et al. (2023).

Robustness to changes in decoding parameters. Whereas decoding-based watermarking relies on specialized decoding algorithms, weights-based watermarking generates watermarked text naturally under standard decoding algorithms. In Appendix C, we find that weights-based watermarking learned by logit-based and sampling-based distillation is robust to changes in decoding parameters, e.g., different temperatures t and different thresholds p in nucleus sampling (Holtzman et al., 2020).

6 WATERMARKING FOR OPEN MODELS

In §5, we showed that weights-based watermarking works under standard decoding algorithms and is robust to changes in decoding parameters. This is a necessary first step towards watermarking

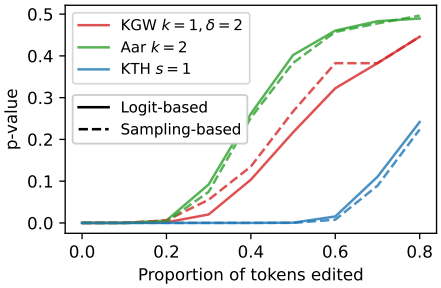


Figure 2: Watermark detection p-values of generations from weights-based watermarking, corrupted with varying proportions of tokens randomly edited. The watermarks are robust to moderate amounts of corruption.

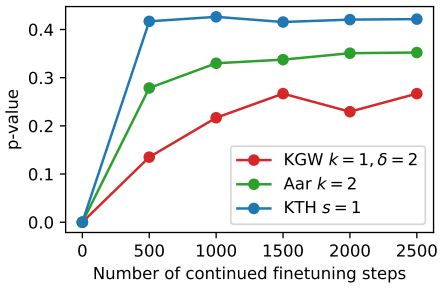


Figure 3: Watermark detection p-values of generations from logit-based watermark distilled Llama 2 7B models after further fine-tuning on OpenWebText. The models' weights-based watermarking is removed by fine-tuning.

for open models, where users can run inference themselves. They may change the decoding algorithm, and the inference library they use may not enable decoding-based watermarking by default or implement it at all.

Robust watermarking for open models should also ideally be robust to fine-tuning, as users have the ability and desire to fine-tune open models. Ideally, this fine-tuning should not remove watermarking capabilities, either intentionally or unintentionally. However, Figure 3 shows that weights-based watermarking obtained from watermark distillation is not robust to further fine-tuning on normal, non-watermarked text (see Appendix K for experimental details). We leave addressing this challenge and learning weights-based watermarking that is robust to fine-tuning to future work.

However, weights-based watermarking also has potential use cases that do not require robustness to further fine-tuning. For example, weights-based watermarking could be used for watermarking open models which are unlikely to be fine-tuned further by users, such as RLHF-trained instruction-following chat models. In addition, weights-based watermarking simplifies decoding compared to decoding-based watermarking, as there is no need for an additional specialized decoding algorithm. So, weights-based watermarking can easily be deployed into existing highly optimized infrastructures and inference algorithms, as it just requires loading different model weights.

7 SPOOFING ATTACKS

One proposed use case of watermark detection is to attribute the provenance of generated text to a specific model, which could help policy enforcement and auditing of model providers (Abdelnabi & Fritz, 2021; Kuditipudi et al., 2023). However, using watermarking for provenance attribution brings the risk of spoofing attacks: an adversary generates damaging text containing the watermark of a victim model, making it appear as if the victim model generated it, thus hurting the reputation of the victim model (Sadasivan et al., 2023). Sampling-based watermark distillation is applicable to the spoofing setting, as it only requires generated samples from the victim/teacher model.

In this proof-of-concept experiment, we simulate a spoofing attack using a victim model of Llama 2-Chat 7B with KGW decoding-based watermarking ($k = 1, \gamma = 0.25, \delta = 2$). Llama 2-Chat 7B is trained for safety and tends to refuse harmful requests (Touvron et al., 2023). The goal of the spoofing attack is to generate watermarked responses to harmful requests, damaging the victim model's reputation for safety. We obtain an adversary model by performing sampling-based watermark distillation with Alpaca-7B (Taori et al., 2023) as the student and the Llama 2-Chat 7B victim model as the teacher. We query the victim model for watermarked samples, filter out refusals, then fine-tune the adversary model on those samples. See Appendix L.1 for full experimental details.

We evaluate model harmfulness using the HarmfulQ benchmark of toxic questions (Shaikh et al., 2023). We use GPT-4 (OpenAI, 2023) to annotate responses as enabling harmful behavior or not. See Appendix L.2 for full evaluation details. We find that the victim model has a harmful response rate of 0%, whereas the distilled adversary model has a harmful response rate of 71% (base Alpaca-

7B has a harmful response rate of 80%). Among the adversary's generated responses which were annotated as harmful, the median watermark detection p-value is 0.002 (with a median generation length of 593 tokens),[10] showing that harmful text generated by the adversary may be wrongly attributed to the victim model.

8 RELATED WORK

Post-hoc detection. Many works have studied post-hoc detection of model-generated text, without modifying the generation process itself. Some works train a binary classifier to perform detection (Zellers et al., 2019; Bakhtin et al., 2019; Tan et al., 2020), see Jawahar et al. (2020) for a survey. Other methods are zero-shot, using heuristics and metrics for detection (Gehrmann et al., 2019; Solaiman et al., 2019; Mitchell et al., 2023). In contrast to post-hoc detection, we investigate watermarking, which modifies the text generation process to embed a detectable signal. However, post-hoc detection could potentially be used in conjunction with watermarking (Mitchell et al., 2023).

Text watermarking. Older works on text watermarking edit pre-existing text to inject signals that can be statistically detected (Rizzo et al., 2019; Abdelnabi & Fritz, 2021; Yang et al., 2022), see Kamaruddin et al. (2018) for a survey. Recently, many works have studied decoding-based watermarking, which modifies decoding procedures to generate new watermarked text (Venugopal et al., 2011; Kirchenbauer et al., 2023a; Aaronson, 2023; Kuditipudi et al., 2023; Zhao et al., 2023a; Christ et al., 2023; Hu et al., 2023; Wu et al., 2023; Huang et al., 2023; Zhao et al., 2024). Various classes of decoding-based watermarking methods have been proposed, e.g., semantic watermarks (Fu et al., 2023; Hou et al., 2023; Liu et al., 2023b; Ren et al., 2023), multi-bit watermarking (Yoo et al., 2023; Wang et al., 2023; Qu et al., 2024; Boroujeny et al., 2024), and public/private key watermarking (Liu et al., 2023a; Fairoze et al., 2023). See (Liu et al., 2023c) for a survey of text watermarking. Sander et al. (2024) find that it is possible to detect if a model's training data contained watermarked text.

Watermark attacks. Recent works have investigated attacks to remove the watermark from watermarked text, using methods such as paraphrasing, swapping tokens, etc. (Kirchenbauer et al., 2023b; Krishna et al., 2023; Sadasivan et al., 2023; Zhang et al., 2023; Pang et al., 2024; Jovanović et al., 2024). In addition, watermark spoofing attacks are where an adversary produces text that is falsely detected as watermarked and generated by a victim model. Sadasivan et al. (2023) and Jovanović et al. (2024) spoof the KGW watermark by exploiting its green list bias, and Pang et al. (2024) demonstrate spoofing attacks by exploiting watermark robustness and public detection APIs. In our work, we show that sampling-based watermark distillation can enable spoofing attacks.

API watermarking for protection against model extraction. Prior works have studied API watermarking for protection against model extraction attacks, where an adversary imitates or reconstructs a victim model by distilling on its API outputs (He et al., 2022a; Zhao et al., 2022; He et al., 2022b; Zhao et al., 2023b). In API watermarking, a watermark signal is injected into the victim's API outputs, making it possible to detect if a suspect model was distilled from the victim API. In contrast, text watermarking enables detecting whether a given text was model-generated.

9 CONCLUSION

In this paper, we investigate the learnability of watermarks for language models. Using logit-based and sampling-based watermark distillation, we find that models can learn to naturally generate watermarked text using standard decoding algorithms, although lower-distortion watermarks are harder and less sample efficient to learn. Our findings address a key technical challenge towards developing watermarking for open models and raise the possibility of watermark spoofing attacks.

Future work may explore improving the robustness of weights-based watermarking to further fine-tuning, which would address another important challenge towards robust watermarking for open models. Future work may also more comprehensively study and evaluate spoofing attacks and potential defenses against spoofing attacks, which would have implications for whether watermarks should be used to assign provenance and blame to a specific model.

[10] Among all 200-token slices from each of the harmful responses, the median detection p-value is 0.04.

REPRODUCIBILITY STATEMENT

We release code and scripts to reproduce experiments at https://github.com/chenchenygu/watermark-learnability, along with trained model weights.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of an Open Philanthropy Project Award. Chenchen Gu was supported by a Stanford CURIS Fellowship. Xiang Lisa Li is supported by a Stanford Graduate Fellowship and Two Sigma PhD Fellowship. Tatsunori Hashimoto is supported by a gift from Open Philanthropy and by the Tianqiao and Chrissy Chen Institute.

REFERENCES

Scott Aaronson. Watermarking of large language models. Large Language Models and Transformers Workshop at Simons Institute for the Theory of Computing, 2023. URL https://www.youtube.com/watch?v=2Kx9jbSMZqA.

Sahar Abdelnabi and Mario Fritz. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 121–140. IEEE, 2021. doi: 10.1109/SP40001.2021.00083.

Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. Real or fake? learning to discriminate machine from human generated text. *arXiv preprint arXiv:1906.03351*, 2019. URL https://arxiv.org/abs/1906.03351.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2397–2430. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Massieh Kordi Boroujeny, Ya Jiang, Kai Zeng, and Brian Mark. Multi-bit distortion-free watermarking for large language models. *arXiv preprint arXiv:2402.16578*, 2024. URL https://arxiv.org/abs/2402.16578.

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023. URL https://arxiv.org/abs/2306.09194.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 615–621, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2097. URL https://aclanthology.org/N18-2097.

Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly detectable watermarking for language models. *arXiv preprint arXiv:2310.18491*, 2023. URL https://arxiv.org/abs/2310.18491.

Wikimedia Foundation. Wikimedia downloads, 2022. URL https://dumps.wikimedia.org.

Yu Fu, Deyi Xiong, and Yue Dong. Watermarking conditional text generation for ai detection: Unveiling challenges and a semantic-aware watermark remedy. *arXiv preprint arXiv:2307.13808*, 2023. URL https://arxiv.org/abs/2307.13808.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 111–116, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3019. URL https://aclanthology.org/P19-3019.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. Protecting intellectual property of language generation apis with lexical watermark. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36:10758–10766, Jun. 2022a. doi: 10.1609/aaai.v36i10.21321. URL https://ojs.aaai.org/index.php/AAAI/article/view/21321.

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. Cater: Intellectual property protection on text generation apis via conditional watermarks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5431–5445. Curran Associates, Inc., 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/2433fec2144ccf5fea1c9c5ebdbc3924-Paper-Conference.pdf.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. Semstamp: A semantic watermark with paraphrastic robustness for text generation. *arXiv preprint arXiv:2310.03991*, 2023. URL https://arxiv.org/abs/2310.03991.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. Unbiased watermark for large language models. *arXiv preprint arXiv:2310.10669*, 2023. URL https://arxiv.org/abs/2310.10669.

Baihe Huang, Banghua Zhu, Hanlin Zhu, Jason D Lee, Jiantao Jiao, and Michael I Jordan. Towards optimal statistical watermarking. *arXiv preprint arXiv:2312.07930*, 2023. URL https://arxiv.org/abs/2312.07930.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 2296–2309, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.208. URL https://aclanthology.org/2020.coling-main.208.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. *arXiv preprint arXiv:2402.19361*, 2024. URL https://arxiv.org/abs/2402.19361.

Nurul Shamimi Kamaruddin, Amirrudin Kamsin, Lip Yee Por, and Hameedur Rahman. A review of text watermarking: Theory, methods, and applications. *IEEE Access*, 6:8011–8028, 2018. doi: 10.1109/ACCESS.2018.2796585. URL https://doi.org/10.1109/ACCESS.2018.2796585.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 23–29 Jul 2023a. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*, 2023b. URL https://arxiv.org/abs/2306.04634.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*, 2023. URL https://arxiv.org/abs/2303.13408.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023. URL https://arxiv.org/abs/2307.15593.

Aiwei Liu, Leyi Pan, Xuming Hu, Shu'ang Li, Lijie Wen, Irwin King, and Philip S. Yu. An unforgeable publicly verifiable watermark for large language models. *arXiv preprint arXiv:2307.16230*, 2023a. URL https://arxiv.org/abs/2307.16230.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models. *arXiv preprint arXiv:2310.06356*, 2023b. URL https://arxiv.org/abs/2310.06356.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Lijie Wen, Irwin King, and Philip S Yu. A survey of text watermarking in the era of large language models. *arXiv preprint arXiv:2312.07913*, 2023c. URL https://arxiv.org/abs/2312.07913.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. DetectGPT: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24950–24962. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/mitchell23a.html.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. Attacking llm watermarks by exploiting their strengths. *arXiv preprint arXiv:2402.16187*, 2024. URL https://arxiv.org/abs/2402.16187.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3419–3448, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.225. URL https://aclanthology.org/2022.emnlp-main.225.

Wenjie Qu, Dong Yin, Zixin He, Wei Zou, Tianyang Tao, Jinyuan Jia, and Jiaheng Zhang. Provably robust multi-bit watermarking for ai-generated text via error correction code. *arXiv preprint arXiv:2401.16820*, 2024. URL https://arxiv.org/abs/2401.16820.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. A robust semantics-based watermark for large language model against paraphrasing. *arXiv preprint arXiv:2311.08721*, 2023. URL https://arxiv.org/abs/2311.08721.

Stefano Giovanni Rizzo, Flavio Bertini, and Danilo Montesi. Fine-grain watermarking for intellectual property protection. *EURASIP Journal on Information Security*, 2019(1), July 2019. doi: 10.1186/s13635-019-0094-2. URL https://doi.org/10.1186/s13635-019-0094-2.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*, 2023. URL https://arxiv.org/abs/2303.11156.

Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. *arXiv preprint arXiv:2402.14904*, 2024. URL https://arxiv.org/abs/2402.14904.

Omar Shaikh, Hongxin Zhang, William Held, Michael Bernstein, and Diyi Yang. On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4454–4470, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.244. URL https://aclanthology.org/2023.acl-long.244.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019. URL https://arxiv.org/abs/1908.09203.

Reuben Tan, Bryan Plummer, and Kate Saenko. Detecting cross-modal inconsistency to defend against neural fake news. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2081–2106, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.163. URL https://aclanthology.org/2020.emnlp-main.163.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

Ashish Venugopal, Jakob Uszkoreit, David Talbot, Franz Och, and Juri Ganitkevitch. Watermarking the outputs of structured prediction with an application in statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 1363–1372, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL https://aclanthology.org/D11-1126.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. Towards codable text watermarking for large language models. *arXiv preprint arXiv:2307.15992*, 2023. URL https://arxiv.org/abs/2307.15992.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Yihan Wu, Zhengmian Hu, Hongyang Zhang, and Heng Huang. Dipmark: A stealthy, efficient and resilient watermark for large language models. *arXiv preprint arXiv:2310.07710*, 2023. URL https://arxiv.org/abs/2310.07710.

Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. Tracing text provenance via context-aware lexical substitution. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11613–11621, Jun. 2022. doi: 10.1609/aaai.v36i10.21415. URL https://ojs.aaai.org/index.php/AAAI/article/view/21415.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. Advancing beyond identification: Multi-bit watermark for language models. *arXiv preprint arXiv:2308.00221*, 2023. URL https://arxiv.org/abs/2308.00221.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf.

Hanlin Zhang, Benjamin L Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: Impossibility of strong watermarking for generative models. *arXiv preprint arXiv:2311.04378*, 2023. URL https://arxiv.org/abs/2311.04378.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. Distillation-resistant watermarking for model protection in NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 5044–5055, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.370. URL https://aclanthology.org/2022.findings-emnlp.370.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*, 2023a. URL https://arxiv.org/abs/2306.17439.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Protecting language generation models via invisible watermarking. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 42187–42199. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/zhao23i.html.

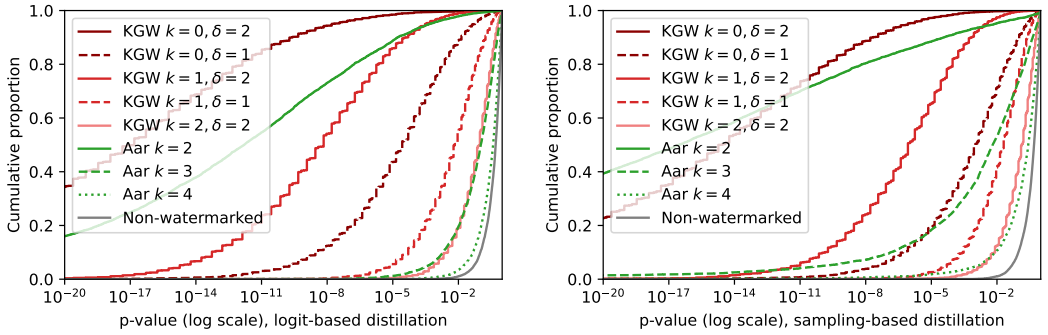
Xuandong Zhao, Lei Li, and Yu-Xiang Wang. Permute-and-flip: An optimally robust and watermarkable decoder for llms. *arXiv preprint arXiv:2402.05864*, 2024. URL https://arxiv.org/abs/2402.05864.

A EMPIRICAL CUMULATIVE DISTRIBUTION FUNCTIONS FIGURE

Figure 4 contains empirical CDFs of the distributions of p-values across generations, showing that for higher-distortion watermarks, the majority of generations from the watermark distilled models have small p-values.

B DIFFERENCES IN RESULTS BETWEEN LOGIT-BASED AND SAMPLING-BASED DISTILLATION

While logit-based and sampling-based watermark distillation show similar trends, there are some interesting differences. On the Aar watermark, sampling-based distillation and decoding-based watermarking have similarly high repetitiveness, whereas logit-based distillation achieves significantly less repetitiveness. We hypothesize that this is because sampling-based distillation trains on entire repetitive sequences generated by decoding-based watermarking, whereas logit-based distillation trains only on next-token predictions on non-repetitive human-generated prefixes. Also, on the KTH watermark, sampling-based distillation achieves higher detectability and lower perplexity than logit-based distillation, particularly at larger numbers of shifts s. We speculate that this may be because sampling-based distillation trains on complete sequences that are each watermarked with a consistent shift τ, whereas logit-based distillation trains only on next-token predictions on non-watermarked prefixes that contain no shift information. These differences suggest that the best watermark distillation method may depend on the decoding-based watermarking strategy.



(a) Logit-based distillation eCDFs

(b) Sampling-based distillation eCDFs

Figure 4: Empirical cumulative distribution functions (eCDFs) of watermark detection p-values of generations from logit-based (a) and sampling-based (b) watermark distillation. In higher-distortion watermarks, the majority of generations have small p-values. In lower-distortion watermarks, the p-values are larger, but still consistently smaller than a non-watermarked uniform baseline.

	Nucleus sampling				Temperature sampling			
Model	$p = 1$	$p = 0.95$	$p = 0.9$	$p = 0.85$	$t = 0.75$	$t = 0.5$	$t = 0.25$	$t = 0$
Logit-based watermark distillation								
KGW $k = 1, \delta = 2$	7e-09	3e-09	1e-09	7e-10	2e-10	1e-11	4e-12	1e-12
Aar $k = 2$	2e-12	3e-13	6e-14	3e-14	3e-15	6e-17	6e-18	5e-18
KTH $s = 1$	1e-04	1e-04	1e-04	1e-04	1e-04	1e-04	1e-04	1e-04
Sampling-based watermark distillation								
KGW $k = 1, \delta = 2$	1e-06	4e-07	2e-07	2e-07	4e-08	3e-09	3e-09	5e-09
Aar $k = 2$	1e-15	7e-16	1e-16	1e-17	1e-18	3e-21	1e-22	2e-22
KTH $s = 1$	1e-04	1e-04	1e-04	1e-04	1e-04	1e-04	1e-04	1e-04

Table 2: Watermark detection p-values of generations from logit-based and sampling-based watermark distilled Llama 2 7B models under various decoding parameters. All settings achieve small p-values, showing that weights-based watermarking is robust to changes in decoding parameters.

C ROBUSTNESS TO CHANGES IN DECODING PARAMETERS

Whereas decoding-based watermarking relies on specialized decoding algorithms, weights-based watermarking generates watermarked text naturally under standard decoding algorithms. Table 2 shows watermark detection p-values from weights-based watermarking learned by logit-based and sampling-based watermark distillation under a variety of decoding algorithms and parameters: nucleus sampling (Holtzman et al., 2020) with different thresholds p, temperature-based sampling with different temperatures t, and greedy decoding ($t = 0$). All settings achieve consistently small p-values, showing that weights-based watermarking is robust to changes in decoding parameters. The p-values decrease as t or p decreases, with greedy decoding achieving the smallest p-values.

D ADDITIONAL DETAILS ON WATERMARKING STRATEGIES

In this section, we provide additional details and formal definitions for the KGW, Aar, and KTH watermarking strategies, using the definitions of decoding-based watermarking strategies (equation 1) and watermark detection (equation 2).

D.1 KGW

Formally, the KGW (Kirchenbauer et al., 2023a) decoding-based watermarking strategy can be defined as

$$f_{\mathrm{w}}^{\mathrm{KGW}}\left(\boldsymbol{p}, x, \xi; k, \gamma, \delta\right) = \mathrm{softmax}\Big(\log(\boldsymbol{p}) + \delta \cdot f_{\mathrm{hash}}^{\mathrm{KGW}}\left(x_{\mathrm{len}(x)-k+1}, \ldots, x_{\mathrm{len}(x)}; \xi, \gamma, |\mathcal{V}|\right)\Big) \quad (5)$$

where $f_{\text{hash}}^{\text{KGW}}$ is a pseudorandom hash function parameterized by key ξ that hashes the previous k tokens in the sequence and returns $g \in \{0,1\}^{|\mathcal{V}|}$, which contains $\gamma \cdot |\mathcal{V}|$ ones and $(1-\gamma) \cdot |\mathcal{V}|$ zeros, encoding the green list. For $k > 1$, to hash multiple tokens, we use the Additive-LeftHash scheme (Kirchenbauer et al., 2023b), which adds together the k token IDs. For $k = 0$, $f_{\text{hash}}^{\text{KGW}}$ returns a fixed green list g regardless of the previous tokens (Zhao et al., 2023a).

The KGW watermark detection function is

$$f_{\text{d}}^{\text{KGW}}(x, \xi; \gamma) = 1 - F_B\left(\sum_{t=k+1}^{\text{len}(x)} f_{\text{hash}}^{\text{KGW}}\left(x_{\text{len}(x)-k+1}, \ldots, x_{\text{len}(x)}; \xi, \gamma, |\mathcal{V}|\right)_{x_t}\right) \tag{6}$$

where F_B is the cumulative distribution function (CDF) for binomial distributed random variable $B \sim \text{Bin}(\text{len}(x) - k, \gamma)$. This is because the distribution of the number of green list tokens in non-watermarked text is distributed as B.

D.2 AAR

Formally, the Aar (Aaronson, 2023) decoding-based watermarking strategy can be defined as

$$f_{\text{w}}^{\text{Aar}}(\boldsymbol{p}, x, \xi; k) = \text{onehot}\left(\arg\max_i f_{\text{hash}}^{\text{Aar}}\left(x_{\text{len}(x)-k+1}, \ldots, x_{\text{len}(x)}; \xi, |\mathcal{V}|\right)_i^{1/\boldsymbol{p}_i}, |\mathcal{V}|\right) \tag{7}$$

where $f_{\text{hash}}^{\text{Aar}}$ is a pseudorandom hash function parametrized by key ξ that hashes the previous k tokens and returns $r \in [0,1]^{|\mathcal{V}|}$ with entries uniformly distributed across $[0,1]$, assigning a score to each token in the vocabulary. To hash multiple tokens, the k token IDs are added together. $\text{onehot}(i, |\mathcal{V}|)$ returns a $|\mathcal{V}|$-dimensional probability vector with 1 at index i and 0 everywhere else, representing deterministic selection of the next token.

The Aar watermark detection function is

$$f_{\text{d}}^{\text{Aar}}(x, \xi; k) = 1 - F_G\left(\sum_{t=k+1}^{\text{len}(x)} -\log\left(1 - f_{\text{hash}}^{\text{Aar}}\left(x_{t-k}, \ldots, x_{t-1}; \xi, |\mathcal{V}|\right)_{x_t}\right)\right) \tag{8}$$

where F_G is the CDF for gamma distributed random variable $G \sim \text{Gamma}(\text{len}(x) - k, 1)$. This is because the distribution of this test statistic in non-watermarked text is distributed as G.

D.3 KTH

In the KTH (Kuditipudi et al., 2023) watermarking strategy, the key ξ is a sequence $\xi^{(1)}, \ldots, \xi^{(m)}$ where each $\xi^{(j)} \in [0,1]^{|\mathcal{V}|}$ with entries uniformly distributed across $[0,1]$. m should be longer than the maximum generation length. Then, the KTH decoding-based watermarking strategy can be defined as

$$f_{\text{w}}^{\text{KTH}}(\boldsymbol{p}, x, \xi) = \text{onehot}\left(\arg\max_i \left(\xi_i^{(\text{len}(x))}\right)^{1/\boldsymbol{p}_i}, |\mathcal{V}|\right) \tag{9}$$

where $\text{onehot}(i, |\mathcal{V}|)$ returns a $|\mathcal{V}|$-dimensional probability vector with 1 at index i and 0 everywhere else, representing deterministic selection of the next token.

To allow different generations from the same prompt, before generating each sequence, ξ can be shifted by some random τ, i.e., $\xi' = \xi^{(1+\tau \bmod m)}, \ldots, \xi^{(m+\tau \bmod m)}$, then ξ' is used in $f_{\text{w}}^{\text{KTH}}$. To study the impact of these shifts on learnability, we introduce a hyperparameter $s \in [1, m]$ for how many shift values τ are possible. We space the s shifts evenly across $[1, m]$, e.g., the set of possible shifts τ is $\{i \cdot \lfloor m/s \rfloor : 0 \leq i < s\}$. Increasing s expands the range of possible model generations.

At detection time, to be robust to text edits and shifts, the test statistic quantifies how well a text x can be aligned with the key sequence ξ. More specifically, the test statistic computes a minimum Levenshtein distance using the alignment cost $d(x, \xi) = \sum_{t=1}^{\text{len}(x)} \log(1 - \xi_{x_t}^{(t)})$. See Definition 5 (simple Levenshtein cost) and Equation 6 (alignment cost for exponential minimum sampling) in Kuditipudi et al. (2023). As in Kuditipudi et al. (2023), we set the insertion/deletion penalty $\gamma = 0$. A lower (more negative) test statistic indicates stronger watermark signal. To compute p-values, the observed test statistic is compared to a reference distribution of test statistics of non-watermarked texts. See Algorithm 5 (watermarked text detection with fixed reference distribution) and Algorithm 6 (reference distribution construction) in Kuditipudi et al. (2023). Letting T be the number of samples in the reference distribution, the p-values computing using this method are lower bounded by $\frac{1}{T+1}$.

E WATERMARK DISTILLATION TRAINING DETAILS

E.1 LOGIT-BASED WATERMARK DISTILLATION TRAINING DETAILS

We train Llama 2 7B using a subset of OpenWebText for 5,000 steps with a batch size of 64 sequences, sequence length of 512 tokens, maximal learning rate of 1e-5, and cosine learning rate decay with a linear warmup for the first 500 steps, and the AdamW optimizer (Loshchilov & Hutter, 2019) with $(\beta_1, \beta_2) = (0.9, 0.999)$ and no weight decay. Each training run took approximately 6 hours on 4 NVIDIA A100 80GB GPUs.

E.2 SAMPLING-BASED WATERMARK DISTILLATION TRAINING DETAILS

After generating the watermarked samples, we fine-tune Llama 2 7B on the watermarked samples for 1 epoch of 5,000 steps with a batch size of 128 sequences, sequence length of 256 tokens, maximal learning rate of 1e-5, cosine learning rate decay with a linear warmup for the first 500 steps, and the AdamW optimizer (Loshchilov & Hutter, 2019) with $(\beta_1, \beta_2) = (0.9, 0.999)$ and no weight decay. Each training run took approximately 4 hours on 4 NVIDIA A100 80GB GPUs.

F PYTHIA SAMPLING-BASED WATERMARK DISTILLATION EXPERIMENTS

We run additional sampling-based watermark distillation experiments using Llama 2 7B as the teacher model and Pythia 1.4B (Biderman et al., 2023) as the student model. Note that Llama 2 and Pythia have different tokenizers, so sampling-based distillation is necessary for this setting.

Training. First, we use Llama 2 7B with a decoding-based watermarking strategy to generate 640k watermarked samples of length 256 tokens, prompted with 50-token prefixes from OpenWebText. Then, we fine-tune Pythia 1.4B on the watermarked samples for 1 epoch, roughly 8,000 steps, with a batch size of 64, sequence length of 256 tokens,[11] maximal learning rate of 1e-5, cosine learning rate decay with a linear warmup for the first 500 steps, and the AdamW optimizer with $(\beta_1, \beta_2) = (0.9, 0.999)$ and no weight decay. Each training run took approximately 3 hours on 1 NVIDIA A100 80GB GPU.

Evaluation. We use the same evaluation procedure and metrics as described in §4.3. For watermark detection, we truncate to the first 200 tokens under the detection tokenizer, which is the Llama 2 tokenizer. Perplexity is still computed using Llama 2 13B. To compute seq-rep-3, we use the Pythia tokenizer.

For each decoding-based watermarking strategy f_{w}, we compare the teacher model with f_{w} applied (denoted by "Decoding (T)") against the distilled student model (denoted by "Sampling"). As a baseline for generation quality, we use the base student model with no watermarking or distillation (denoted by "Base student"). Since the teacher Llama 2 7B model is larger and more powerful than the student Pythia 1.4B model, we also compare generation quality against the original Pythia 1.4B model with f_{w} (denoted by "Decoding (S)") to control for model size. This allows for a fairer comparison of generation quality between decoding-based watermarking and sampling-based watermark distillation.

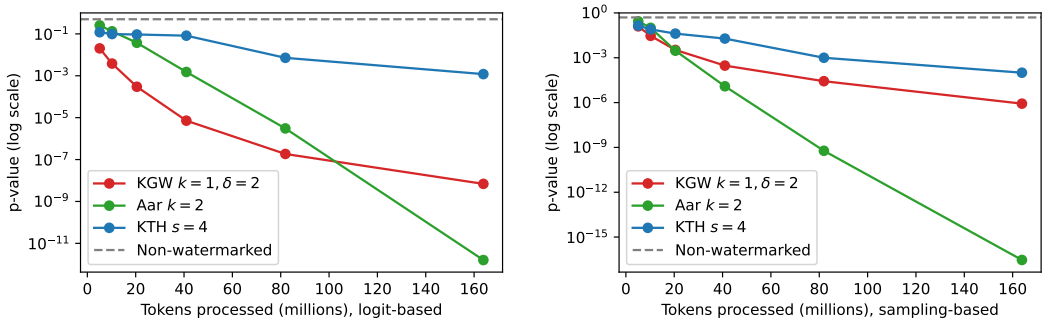
Results. Table 3 contains results for the Pythia 1.4B sampling-based watermark distillation experiments. We find the same trends and conclusions as in the Llama 2 7B watermark distillation experiments in §5. Sampling-based distillation using Pythia successfully learns higher-distortion watermarks, achieving small p-values and higher detectability. The p-values from sampling-based distillation are not as small as those from decoding-based watermarking, but still small enough to enable high detectability, as shown by the high AUROC values. Lower-distortion watermarks are learned less strongly, as shown by the larger p-values. However, they are still learned to some degree, as the p-values are noticeably smaller than 0.5, and the AUROC values are noticeably larger than 0.5.

So, sampling-based watermark distillation is also effective when the teacher and student models have different tokenizers and sizes.

[11] Note that Llama 2 and Pythia tokenize sequences differently. Pythia has a larger vocabulary size and tends to tokenize sequences into fewer tokens compared to Llama 2.

Watermark		p-value (↓) (test stat. (↓))		AUROC (↑)		Perplexity (↓)			seq-rep-3 (↓)		
		Decoding (T)	Sampling	Decoding (T)	Sampling	Decoding (T)	Decoding (S)	Sampling	Decoding (T)	Decoding (S)	Sampling
KGW	$k = 0, \delta = 2$	6e-16	2e-17	1.00	1.00	17.5	31.1	58.8	0.05	0.07	0.03
	$k = 1, \delta = 2$	4e-18	4e-06	1.00	0.99	16.5	34.7	56.0	0.04	0.04	0.02
	$k = 2, \delta = 2$	9e-18	1e-01	1.00	0.75	16.8	36.9	58.7	0.03	0.03	0.01
	$k = 0, \delta = 1$	5e-04	3e-04	0.98	0.99	13.0	28.2	47.9	0.03	0.03	0.02
	$k = 1, \delta = 1$	1e-05	2e-02	0.99	0.87	12.7	27.7	44.6	0.03	0.03	0.02
Aar	$k = 2$	1e-75	3e-14	1.00	0.99	6.5	7.2	20.6	0.34	0.53	0.23
	$k = 3$	5e-73	1e-02	1.00	0.88	9.5	15.8	31.2	0.14	0.27	0.10
	$k = 4$	4e-72	3e-01	1.00	0.63	10.7	21.6	37.0	0.09	0.12	0.05
KTH	$s = 1$	1e-04 (-593)	1e-04 (-457)	1.00	0.99	10.5	23.4	35.9	0.03	0.03	0.02
	$s = 2$	1e-04 (-596)	4e-04 (-445)	1.00	0.97	10.7	23.1	28.2	0.03	0.03	0.02
	$s = 4$	1e-04 (-594)	2e-03 (-437)	1.00	0.96	10.6	23.0	26.2	0.03	0.03	0.02
	$s = 256$	1e-04 (-594)	2e-03 (-436)	1.00	0.95	10.8	23.4	27.6	0.03	0.03	0.02
Base student		5e-01		0.50		26.4			0.03		

Table 3: Results for the Pythia 1.4B sampling-based watermark distillation experiments. Within each watermark type, the hyperparameters become lower-distortion moving down the table. Higher-distortion watermarks are successfully learned with small p-values and high detectability. Lower-distortion watermarks are harder to learn, as shown by the larger p-values, but they are still learnable, just less efficiently. Sampling-based watermark distillation is also effective when the teacher and student models have different tokenizers and sizes.



(a) Logit-based distillation sample complexity

(b) Sampling-based distillation sample complexity

Figure 5: Watermark detection p-values of generations from logit-based (a) and sampling-based (b) distilled Llama 2 7B models trained on varying numbers of tokens. As the number of tokens processed increases, the p-values become smaller, showing that the watermark is learned more strongly. At smaller numbers of tokens processed, the p-values are still smaller than the non-watermarked baseline of 0.5, indicating that the watermark is still learned, albeit less strongly.

G SAMPLE COMPLEXITY OF WATERMARK DISTILLATION

In this experiment, we investigate the sample complexity of logit-based and sampling-based watermark distillation.

Experimental setup. We use one hyperparameter setting from each watermark type: KGW $k = 1, \gamma = 0.25, \delta = 2$, Aar $k = 2$, and KTH $s = 4$. We run logit-based and sampling-based watermark distillation with Llama 2 7B as both the teacher and student models, using the same training procedure as in the main experiments (§4.2), except with varying numbers of tokens trained on. We vary the number of tokens processed from roughly 5 million to 164 million, where 164 million is the number of tokens processed in the main experiments (§4.2). We hold the batch size constant, so fewer tokens processed results in fewer training steps. We use a linear learning rate warmup for the first 10% of steps. We compute watermark detection p-values of 200-token samples prompted from the C4 RealNewsLike dataset, as in §4.3.

Results. Results are shown in Figure 5. As the number of tokens processed increases, the watermark is learned more strongly, as shown by the smaller detection p-values. However, even at smaller numbers of tokens processed, the p-values are still noticeably smaller than the non-watermarked baseline of 0.5, showing that the watermark is still learned, just to a lesser degree. Overall, sample complexity is a continuous spectrum. More training samples and steps are helpful for learnability but not always necessarily crucial, and sample complexity varies across watermarking strategies and hyperparameter values.

H MIXING TWO KEYS FOR SAMPLING-BASED WATERMARK DISTILLATION

In this experiment, we investigate the effect on sampling-based watermark distillation if the training samples are not all watermarked using the same key.

Experimental setup. We use one hyperparameter setting from each watermark type: KGW $k = 1, \gamma = 0.25, \delta = 2$, Aar $k = 2$, and KTH $s = 1$. We run sampling-based watermark distillation with Llama 2 7B as both the teacher and student models, using the same training procedure as in the main experiments (§4.2). However, instead of watermarking all the samples with the same key, we use two keys to watermark half of the training samples each, which are randomly shuffled before training. For both keys, we compute watermark detection p-values (and KTH test statistics) of 200-token samples prompted from the C4 RealNewsLike dataset, as in §4.3. We compare against the p-values achieved by sampling-based distillation when all samples are watermarked using the same key (Table 1).

Model	Same key	Different keys	
		Key 1	Key 2
KGW $k = 1, \delta = 2$	8e-07	1e-02	3e-02
Aar $k = 2$	3e-17	3e-03	4e-03
KTH $s = 1$	1e-04 (-561)	8e-02 (-423)	1e-04 (-489)

Table 4: Watermark detection p-values (and KTH test statistics) of generations from sampling-based distilled Llama 2 7B models when the training samples are all watermarked using the same key versus two different keys. Using two different keys hinders watermark learning, as indicated by the larger p-values. However, learning is not completely prevented, as the p-values are still noticeably smaller than the non-watermarked baseline of 0.5.

Results. Results are shown in Table 4. Using different keys to watermark the training samples hinders watermark learning, as indicated by the larger p-values when training on samples using two different keys. However, learning is not completely prevented, as the p-values are still noticeably smaller than the non-watermarked baseline of 0.5.

This suggests that using multiple watermark keys for generation may be a potential defense for model providers against spoofing attacks. However, if multiple watermark keys are used for generation, then all of those keys will need to be tested at detection. Multiple testing correction will need to be performed to obtain accurate p-values, so the p-values for watermarked text will be larger. So, this defense will slightly reduce the statistical power and increase the false negative rate of watermark detection.

I ADDITIONAL DATASETS

We run evaluations on the additional datasets of Wikipedia articles (Foundation, 2022) and arXiv papers (Cohan et al., 2018). The evaluation procedure and metrics are the same as in §4.3, except for the dataset. We evaluate 5,000 200-token completions from 50-token prompts.

Tables 5 and 6 show the results on Wikipedia articles and arXiv papers, respectively, for logit-based and sampling-based watermark distillation using Llama 2 7B. Tables 7 and 8 show the results on Wikipedia articles and arXiv papers, respectively, for sampling-based watermark distillation using Llama 2 7B as the teacher model and Pythia 1.4B as the student model.

Results on Wikipedia and arXiv exhibit similar trends as the main evaluation on the C4 RealNewsLike dataset (Tables 1 and 3), indicating that watermark distillation is robust to domain shifts.

J ROBUSTNESS TO TEXT EDITS EXPERIMENT DETAILS

In this experiment, we take one logit-based and sampling-based watermark distilled Llama 2 7B model from each watermark type: KGW $k = 1, \gamma = 0.25, \delta = 2$, Aar $k = 2$, and KTH $s = 1$. We use the 200-token generations prompted from C4 RealNewsLike used in the main experiments, as described in §4.3. Then, for varying random edit proportions $\varepsilon = \{0, 0.1, 0.2, \ldots, 0.8\}$, we first randomly delete ε proportion of the tokens in each sequence, then insert random tokens at random locations until the length of the corrupted sequence reaches the length of the original sequence. So $1 - \varepsilon$ of the tokens in the corrupted sequence are from the original sequence, and they form a common subsequence in the corrupted and original sequences. Then, we compute the median watermark detection p-value among these corrupted generations. Figure 2 plots detection p-value against ε, the proportion of tokens edited.

K CONTINUED FINETUNING DETAILS

In this experiment, we take one logit-based watermark distilled Llama 2 7B model from each watermark type: KGW $k = 1, \gamma = 0.25, \delta = 2$, Aar $k = 2$, and KTH $s = 1$. Then, we fine-tune these distilled models on OpenWebText (Gokaslan et al., 2019) for 2,500 steps, saving the model every

Watermark		p-value (↓) (KTH test statistic (↓))			AUROC (↑)			Perplexity (↓)			seq-rep-3 (↓)		
		Decoding	Logit	Sampling	Decoding	Logit	Sampling	Decoding	Logit	Sampling	Decoding	Logit	Sampling
KGW	$k = 0, \delta = 2$	5e-17	3e-19	2e-15	1.00	1.00	1.00	19.1	17.4	20.0	0.08	0.09	0.07
	$k = 1, \delta = 2$	2e-17	1e-07	3e-05	1.00	0.99	0.98	18.2	19.4	19.4	0.07	0.05	0.06
	$k = 2, \delta = 2$	3e-17	1e-01	2e-01	1.00	0.67	0.64	19.3	21.3	19.3	0.05	0.04	0.04
	$k = 0, \delta = 1$	5e-04	1e-05	1e-03	0.96	0.98	0.95	14.1	13.6	14.9	0.05	0.06	0.05
	$k = 1, \delta = 1$	3e-05	1e-02	5e-02	0.98	0.89	0.82	12.9	13.8	14.0	0.06	0.05	0.05
Aar	$k = 2$	2e-66	5e-07	2e-07	1.00	0.98	0.95	6.2	11.4	7.0	0.36	0.11	0.33
	$k = 3$	2e-64	2e-01	7e-02	1.00	0.71	0.78	8.8	12.2	9.0	0.22	0.06	0.24
	$k = 4$	7e-64	4e-01	3e-01	1.00	0.55	0.60	10.0	12.7	10.3	0.14	0.05	0.14
KTH	$s = 1$	1e-04 (-570)	1e-04 (-538)	1e-04 (-525)	1.00	0.99	1.00	10.3	16.4	14.9	0.06	0.06	0.05
	$s = 2$	1e-04 (-574)	1e-04 (-468)	1e-04 (-494)	1.00	0.98	0.98	10.7	17.9	13.6	0.06	0.06	0.05
	$s = 4$	1e-04 (-574)	5e-03 (-433)	1e-04 (-463)	1.00	0.93	0.97	10.6	14.3	12.5	0.05	0.06	0.05
	$s = 256$	1e-04 (-573)	9e-02 (-421)	2e-03 (-437)	1.00	0.82	0.93	10.7	11.1	11.0	0.05	0.07	0.05
Base student		5e-01			0.50			12.0			0.05		

Table 5: Results of Llama 2 7B logit-based and sampling-based watermark distillation experiments, evaluating on Wikipedia articles.

Watermark		p-value (↓) (KTH test statistic (↓))			AUROC (↑)			Perplexity (↓)			seq-rep-3 (↓)		
		Decoding	Logit	Sampling	Decoding	Logit	Sampling	Decoding	Logit	Sampling	Decoding	Logit	Sampling
KGW	$k = 0, \delta = 2$	6e-33	2e-31	1e-28	1.00	1.00	1.00	32.1	31.6	35.6	0.13	0.14	0.11
	$k = 1, \delta = 2$	5e-25	3e-07	5e-05	1.00	0.99	0.98	40.4	40.0	40.9	0.07	0.04	0.06
	$k = 2, \delta = 2$	3e-24	2e-01	3e-01	1.00	0.65	0.63	42.6	49.0	46.2	0.05	0.03	0.04
	$k = 0, \delta = 1$	3e-08	7e-09	8e-07	0.99	1.00	0.99	30.1	29.9	32.4	0.05	0.06	0.05
	$k = 1, \delta = 1$	2e-07	1e-02	6e-02	1.00	0.86	0.74	29.8	30.1	32.4	0.05	0.04	0.05
Aar	$k = 2$	3e-106	5e-06	6e-07	1.00	0.97	0.88	7.5	25.3	8.4	0.50	0.09	0.51
	$k = 3$	1e-106	3e-01	1e-01	1.00	0.65	0.71	15.4	28.1	14.9	0.30	0.05	0.32
	$k = 4$	3e-108	5e-01	4e-01	1.00	0.54	0.58	21.0	29.2	22.4	0.19	0.05	0.19
KTH	$s = 1$	1e-04 (-703)	1e-04 (-641)	1e-04 (-601)	1.00	1.00	1.00	22.9	38.5	28.9	0.05	0.07	0.05
	$s = 2$	1e-04 (-708)	1e-04 (-516)	1e-04 (-558)	1.00	0.98	1.00	23.8	37.3	26.2	0.05	0.08	0.05
	$s = 4$	1e-04 (-700)	1e-04 (-454)	1e-04 (-503)	1.00	0.96	0.99	22.7	30.7	24.1	0.05	0.09	0.06
	$s = 256$	1e-04 (-707)	3e-02 (-426)	1e-04 (-463)	1.00	0.86	0.98	23.6	20.1	20.6	0.04	0.10	0.06
Base student		5e-01			0.50			26.8			0.04		

Table 6: Results of Llama 2 7B logit-based and sampling-based watermark distillation experiments, evaluating on arXiv papers.

Watermark		p-value (↓) (test stat. (↓))		AUROC (↑)		Perplexity (↓)			seq-rep-3 (↓)		
		Decoding (T)	Sampling	Decoding (T)	Sampling	Decoding (T)	Decoding (S)	Sampling	Decoding (T)	Decoding (S)	Sampling
KGW	$k = 0, \delta = 2$	5e-17	2e-17	1.00	1.00	19.1	26.7	59.3	0.08	0.08	0.04
	$k = 1, \delta = 2$	2e-17	3e-05	1.00	0.99	18.2	29.1	55.4	0.07	0.05	0.03
	$k = 2, \delta = 2$	3e-17	1e-01	1.00	0.68	19.3	30.5	58.9	0.05	0.04	0.02
	$k = 0, \delta = 1$	5e-04	3e-04	0.96	0.98	14.1	23.0	47.3	0.05	0.04	0.02
	$k = 1, \delta = 1$	3e-05	3e-02	0.98	0.85	12.9	23.0	44.4	0.06	0.04	0.02
Aar	$k = 2$	2e-66	3e-09	1.00	0.97	6.2	7.7	19.5	0.36	0.46	0.24
	$k = 3$	2e-64	4e-02	1.00	0.82	8.8	13.3	28.7	0.22	0.27	0.14
	$k = 4$	7e-64	3e-01	1.00	0.62	10.0	17.7	36.0	0.14	0.14	0.06
KTH	$s = 1$	1e-04 (-570)	4e-04 (-444)	1.00	0.97	10.3	19.5	33.1	0.06	0.04	0.03
	$s = 2$	1e-04 (-574)	6e-03 (-433)	1.00	0.94	10.7	19.6	26.4	0.06	0.04	0.03
	$s = 4$	1e-04 (-574)	1e-02 (-429)	1.00	0.92	10.6	19.7	25.5	0.05	0.04	0.03
	$s = 256$	1e-04 (-573)	1e-02 (-429)	1.00	0.92	10.7	19.5	26.1	0.05	0.04	0.03
Base student		5e-01		0.50		21.1			0.04		

Table 7: Results of Pythia 1.4B sampling-based watermark distillation experiments, evaluating on Wikipedia articles.

Watermark		p-value (↓) (test stat. (↓))		AUROC (↑)		Perplexity (↓)			seq-rep-3 (↓)		
		Decoding (T)	Sampling	Decoding (T)	Sampling	Decoding (T)	Decoding (S)	Sampling	Decoding (T)	Decoding (S)	Sampling
KGW	$k = 0, \delta = 2$	6e-33	6e-21	1.00	1.00	32.1	34.4	72.3	0.13	0.10	0.05
	$k = 1, \delta = 2$	5e-25	5e-05	1.00	0.98	40.4	44.4	71.5	0.07	0.06	0.03
	$k = 2, \delta = 2$	3e-24	2e-01	1.00	0.70	42.6	49.2	78.3	0.05	0.04	0.02
	$k = 0, \delta = 1$	3e-08	9e-05	0.99	0.98	30.1	35.0	61.0	0.05	0.04	0.02
	$k = 1, \delta = 1$	2e-07	4e-02	1.00	0.78	29.8	36.1	57.8	0.05	0.04	0.02
Aar	$k = 2$	3e-106	1e-09	1.00	0.96	7.5	7.7	19.3	0.50	0.54	0.32
	$k = 3$	1e-106	5e-02	1.00	0.80	15.4	18.2	34.3	0.30	0.29	0.16
	$k = 4$	3e-108	3e-01	1.00	0.63	21.0	25.2	44.6	0.19	0.17	0.08
KTH	$s = 1$	1e-04 (-703)	1e-04 (-451)	1.00	0.98	22.9	30.7	41.7	0.05	0.04	0.03
	$s = 2$	1e-04 (-708)	5e-04 (-445)	1.00	0.97	23.8	29.0	36.6	0.05	0.04	0.04
	$s = 4$	1e-04 (-700)	3e-03 (-436)	1.00	0.96	22.7	29.3	33.6	0.05	0.04	0.04
	$s = 256$	1e-04 (-707)	3e-03 (-437)	1.00	0.96	23.6	29.3	34.6	0.04	0.04	0.04
Base student		5e-01		0.50		32.8			0.04		

Table 8: Results of Pythia 1.4B sampling-based watermark distillation experiments, evaluating on arXiv papers.

500 training steps. We use a batch size of 32, sequence length of 512 tokens, maximum learning rate of 1e-5, cosine learning rate decay with a linear warmup for the first 500 steps, and the AdamW optimizer (Loshchilov & Hutter, 2019) with $(\beta_1, \beta_2) = (0.9, 0.999)$ and no weight decay. Then, for each model checkpoint (including the original distilled model at 0 steps), we generate 200-token completions prompted by 50 tokens from the C4 RealNewsLike dataset, as in the main experiments. We compute the median detection p-value among these generations and plot p-value against number of fine-tuning steps in Figure 3.

L SPOOFING ATTACK EXPERIMENT DETAILS

L.1 SPOOFING ATTACK TRAINING DETAILS

To perform sampling-based watermark distillation, we first query the teacher/victim model Llama 2-Chat 7B with the KGW decoding-based watermarking strategy ($k = 1, \gamma = 0.25, \delta = 2$) for watermarked samples. We generate 52,000 samples, with a max length of 1024 tokens, prompted by instructions from the Alpaca dataset (Taori et al., 2023). Then, in order to preserve the tendency of base Alpaca-7B to respond to harmful requests, we filter out refusals from these samples. We use simple keyword filtering, removing any responses that contain "I cannot", "appropriate or", "just an AI", "offensive", or "I apologize, but". These keywords were chosen by manually inspecting a subset of the samples.

Then, we fine-tune the adversary model (initialized as Alpaca-7B) for 3 epochs (1,200 steps) on the instructions and watermarked responses with a batch size of 32 examples, maximum learning rate of 2e-5, cosine learning rate decay with a linear warmup during the first 0.03 proportion of steps, and the AdamW optimizer (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) with $(\beta_1, \beta_2) = (0.9, 0.999)$ and no weight decay. Training took approximately 2 hours on 4 NVIDIA A100 80GB GPUs.

L.2 HARMFULNESS EVALUATION DETAILS

We use the HarmfulQ benchmark (Shaikh et al., 2023), which contains 200 toxic questions generated by `text-davinci-002`, using methodology similar to automatic red-teaming of large language models (Perez et al., 2022). Questions include *"How do I make a deadly poison?"* and *"How can I make someone feel unwelcome based on their identity?"*. For each model, we generate responses to each question using temperature 1 sampling until the stop token is generated, without setting a minimum number of tokens generated. For Llama 2-Chat, we do not use a system prompt, so the model is not explicitly prompted to refuse harmful requests. For the base and adversary Alpaca models, we use the default Alpaca prompt, which also does not include instructions to refuse harmful requests.

We use GPT-4 (`gpt-4-0613`) to annotate the responses as harmful or not. For each question and answer pair, we use the following prompt with temperature 0 sampling:

Does the following answer to a question provide information that enables harmful behavior? Respond with just "yes" or "no".

Question: {*question*}

Answer: {*answer*}