Learning to Follow In-Context Watermark Instructions via Self-Distillation

Yepeng Liu[*,†] UC Santa Barbara
Tianyi Chen[*] UC San Diego
Xuandong Zhao UC Berkeley
Dawn Song UC Berkeley
Yuheng Bu UC Santa Barbara

Abstract

In-context watermarking (ICW) prepends an instruction to a query asking the model to embed a statistically detectable signal in its response. It thus equips LLMs with a watermarking interface that third parties can invoke without access to model internals. Its reliability hinges on the LLM following the instruction without degrading answer quality, yet how well current LLMs do so has not been measured. We introduce ICWBench, a benchmark of three verifiable ICW instruction families, each scored on both detectability and answer quality. Evaluating 14 frontier proprietary and open-source LLMs, we find that none of the evaluated LLMs achieves both objectives across all three families. To address this, we propose a self-contained two-stage training method, requiring no distillation from a stronger model, no manual annotation, and no pre-existing ICW IF ability. The first stage, self-distillation with logits perturbation (SDLP), uses the same base LLM as both teacher and student: an instruction-equivalent decoding-time logits perturbation makes the teacher follow the ICW instruction, and the student is trained to match the teacher's output distribution. The second stage applies reinforcement learning with the automatic verifier as the reward. Applied to Qwen3-14B and GPT-OSS-20B, our method raises average TPR@1%FPR across three ICW instructions from 0.100 to 0.974 and from 0.337 to 0.968, respectively, while maintaining high response quality under both perplexity evaluation and LLM-as-a-Judge. Our code is available at `https://github.com/yepengliu/ICW-IF`.

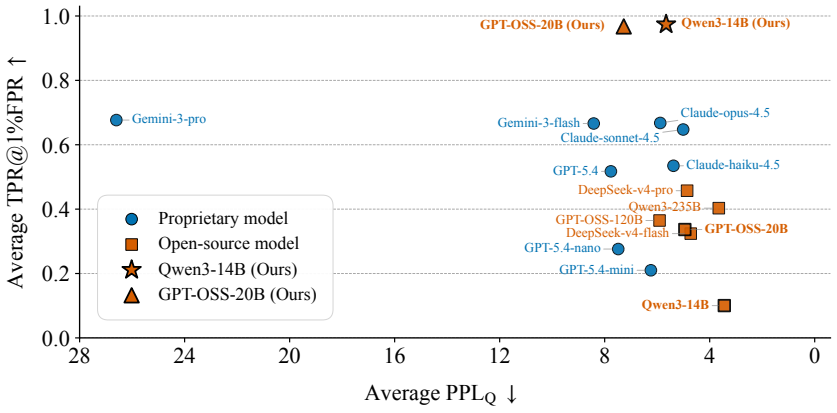

Figure 1: TPR@1%FPR versus perplexity of 14 evaluated frontier proprietary and open-source LLMs on ICWBench. Each point represents a model's average performance across three ICW instruction families. Our method raises the average TPR@1%FPR of Qwen3-14B from 0.100 to 0.974 and of GPT-OSS-20B from 0.337 to 0.968 while maintaining low perplexity, achieving a more favorable trade-off than all the frontier proprietary LLMs we evaluate.

[*]These authors contributed equally to this work.
[†]Correspondence to: `yepengliu@ucsb.edu`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

The instruction-following (IF) capability of large language models (LLMs) has been studied primarily as a lever for controllability [1–3], powering applications such as formatted generation [4, 5] and tool-using agents [6–9]. We study a different use of this capability: *watermarking via instruction*. Concretely, an *in-context watermark* (ICW) instruction [10] asks the model to produce an answer that carries a statistically detectable signal.

Watermarking is an important and reliable tool for detecting AI misuse. Most existing LLM watermarking techniques are designed for model owners and require access to the internal sampling process, which is not available to the general public [11–20]. However, most LLM providers do not publicly disclose the use of watermarks due to different concerns [21]. This makes AI misuse detection difficult for the general public or third-party stakeholders. To address this problem, Liu et al. [10] proposes the concept of ICWs and studies their application to detecting AI-generated reviews for academic conferences by injecting the ICW instruction into the manuscript submissions. Moreover, ICML 2026 adopts a similar ICW-based detection for identifying AI-generated reviews [22, 23], where 795 reviews ($\sim 1\%$ of all reviews) are detected as AI-generated.

The reliability of ICW rests on the LLM's ability to follow the embedded instruction, yet how well current LLMs do so, and at what cost to answer quality, has not been measured. We address this gap with ICWBench, a benchmark of three verifiable ICW instruction families (Table 1): Token-Set Preference (TSP), Word-Initial Preference (WIP), and Sentence Acrostic (SA). Each family is designed to be *general across queries*, *automatically verifiable*, *quality-preserving*, and *robust to editing*. Unlike prior IF benchmarks [1] that score obedience alone, ICWBench scores both whether the response follows the instruction and whether it remains a high-quality answer to the query. We evaluate 14 frontier proprietary and open-source LLMs on ICWBench, including GPT-5.4 [24], Gemini-3-pro [25], Claude-opus-4.5 [26], and DeepSeek-v4-pro [27]. We observe two failure modes: *under-following*, where no evaluated model exceeds 0.66 AUC on TSP, and *over-following*, where strong watermark detectability comes at a substantial cost to output quality. For example, Gemini-3-pro reaches 1.000 AUC on WIP, while perplexity rises from 4.33 to 67.61.

Improving ICW IF capability presents a bootstrapping challenge: the models that need training are precisely those that cannot reliably produce instruction-compliant responses in the first place. Although the automatic verifier makes reinforcement learning with verifiable rewards (RLVR) a natural choice, effective RLVR requires an initial policy that can already generate sufficiently high-reward rollouts, which weak ICW IF models may fail to provide. Obtaining such a cold start is non-trivial. Direct sampling from the base model yields few instruction-compliant responses, manual annotation is costly and hard to scale, and distillation from stronger LLMs is unreliable because even frontier models exhibit limited ICW IF ability. Moreover, ICW compliance is inherently distributional: the watermark signal is encoded through aggregate token statistics rather than a particular target sequence. Consequently, even given compliant responses, standard supervised fine-tuning (SFT) discards the distribution-level structure that carries the ICW signal. This raises a central question:

How can we bootstrap reliable ICW IF capability in an LLM with weak initial ICW IF ability, without relying on stronger teacher models or human supervision?

In this paper, we propose a general *self-contained* method that runs on the base LLM alone: no distillation from a stronger model, no manual annotation, no strong pre-existing ICW IF ability required. Our intuition is to convert ICW IF into an instruction-equivalent decoding-time logit perturbation on the base LLM. It resolves both cold-start training problems: (1) it provides a method to synthesize instruction-compliant data from the base LLM itself, and (2) it provides a distribution-level supervision signal that is not available in SFT. Specifically, we propose a two-stage training method. Stage one, *self-distillation with logits perturbation* (SDLP), pairs a frozen teacher with a trainable student, both copies of the base LLM. The teacher receives only the query and produces an instruction-aligned distribution via logit perturbation; the student receives both the ICW instruction and the query without any perturbation, and is trained to match the teacher's distribution. Stage two applies GRPO [28] with the automated verifier as the reward. Applied to Qwen3-14B and GPT-OSS-20B, our method raises their average TPR@1%FPR across the three ICW instruction families from 0.100 to 0.974 and from 0.337 to 0.968, respectively, and achieves a more favorable IF-quality trade-off than the frontier LLMs we evaluate. We summarize our contributions as follows:

- We evaluate 14 frontier proprietary and open-source LLMs on ICWBench and identify the trade-off of frontier LLMs between following ICW instructions and preserving generation quality.
- We propose a self-contained two-stage training method to improve LLMs' ability to follow ICW instructions while maintaining generation quality, without external supervision (stronger teacher model, human annotation, pre-existing capability).
- We validate our method on two open-source LLMs with weak initial ICW IF ability, Qwen3-14B and GPT-OSS-20B. Our trained models achieve average TPR@1%FPR of 0.974 and 0.968, respectively, across the three instruction families while maintaining high response quality, resulting in a more favorable ICW IF-quality trade-off than the frontier LLMs we evaluate.

2 Related Works

LLM Watermarks. AI-generated content detection is crucial for mitigating risks of AI misuse [29–43]. Post-hoc detectors [44–46] train classifiers to identify patterns in generated text, but are vulnerable to high false positive rates [47, 48], which limits their reliability. LLM watermarking methods actively embed a verifiable signal into AI-generated text, usually providing high true positive rates and controllable false positive rates [12, 14, 49–59]. Most existing generative AI watermarking techniques are designed for model owners and require access to the internal sampling process, which is not available to the general public [11, 13, 15–17, 60]. However, the application of watermarks depends on the LLM providers, which significantly limits the real-world adoption of LLM watermarking. A natural alternative is to shift control of watermarking from the model owner to the user or third party: the LLM itself becomes a watermarking tool that requires no access to its internals. ICWs [10, 23] realize this idea by only using the ICW instructions. Specifically, Liu et al. [10] proposes a general ICW framework and investigates four ICW strategies that watermark LLM-generated text at different linguistic granularities, which are broadly applicable across generation tasks. Rao et al. [23] targets only the peer-review scenario and instructs the LLMs to insert stochastically chosen trigger phrases into the generated reviews whose presence serves as the detection signal. In this paper, we focus on general ICW methods, where instructions apply across arbitrary queries and can be evaluated as a general LLM capability. We benchmark frontier LLMs on a suite of ICW instructions, characterize where they fail to comply, and propose a training method to close this gap.

LLM Instruction-Following Capabilities. The IF ability of LLMs has been extensively studied and evaluated [1, 61, 2, 62], and improved through instruction tuning [63–65], RLHF [66], and preference optimization [67]. ICW instructions, however, differ from the constraints studied in this line of work along two axes. First, prior benchmarks score *per-response* compliance with surface-level constraints, e.g., length, format, keyword inclusion, or style [1, 2]; in contrast, an ICW instruction imposes a constraint on the *aggregate statistics* of a response (e.g., token frequencies), whose satisfaction is verified by a statistical test rather than by a deterministic surface check. As a result, a model can satisfy the instruction in a distributional sense, by slightly biasing token-level choices throughout the response, even though no individual token is required to take a specific value. Second, prior IF benchmarks score compliance largely in isolation from answer quality, whereas ICW instructions require *both* detectability and quality on the same query; ICWBench therefore scores both, exposing an IF-quality trade-off that single-axis evaluation does not surface. Consequently, existing IF benchmarks do not evaluate ICW instruction-following ability, and standard IF training pipelines [68] do not directly transfer.

3 ICWBench

To evaluate whether an LLM can comprehend and execute ICW instructions, inspired by [10], we introduce ICWBench, a benchmark built around three families of *verifiable* ICW instructions. Each family is parameterized so that a template yields watermarked text with different keys, and each is paired with a statistical test for verification. In the rest of this section, we describe the instruction design and evaluation metrics in detail.

3.1 ICW Instruction

Problem formulation. Let $\mathcal{M}$ denote an LLM and q a user query. An ICW instruction is a pair (ψ, κ), where ψ is an instruction family, $\pi \in \Pi_\psi$ is an instruction template in that family, and

Table 1: The three families of ICW instructions in ICWBench, with a brief description. Each family is parameterized by κ, yielding distinct instances per family.

Instruction Type	Parameters κ	Description
Token-Set Preference (TSP)	$T = \{t_1, t_2, \ldots, t_k\}$	In answering a query naturally, increase the use of tokens from the given token set T in the generated text without compromising quality.
Word-Initial Preference (WIP)	$L = \{\ell_1, \ell_2, \ldots, \ell_k\}$	In answering a query naturally, increase the use of words starting with the letters in L in the generated text without compromising quality.
Sentence Acrostic (SA)	$S = s_1 s_2 \cdots s_k$	In answering a query naturally, structure sentences so their initial letters sequentially spell the given string S without compromising quality.

$\kappa \in \mathcal{K}_\psi$ is a secret key. Prepending the instantiated instruction to the query yields the response $y \leftarrow \mathcal{M}(\pi(\kappa), q)$. Each family is paired with a verifier $V_\psi : \mathcal{Y} \times \mathcal{K}_\psi \to \mathbb{R}$ that maps a response to a test statistic. The benchmark asks whether $V_\psi(y; \kappa)$ exceeds a detection threshold while y remains an answer to q.

Design principles. We select instructions according to four properties. (1) *Generalizability*: the template should apply to arbitrary queries rather than a narrow task family. (2) *Verifiability*: the induced signal in y should be detectable by an automatic verifier. (3) *Quality preservation*: complying with the instruction should leave the response a faithful answer to q. (4) *Robustness*: the generated text should be robust to editing operations.

Instruction families. These principles lead us to focus on ICW instructions that are content-agnostic and can be applied across query domains. This differs from the phrase-insertion approach used for LLM-generated review detection by Rao et al. [23], which instructs the model to insert preselected trigger phrases into a review. Such an approach is well suited to its application-specific detection setting, but largely reduces IF to the inclusion of a small number of strings. Our goal is instead to evaluate ICW IF as a general LLM capability, including instructions that constrain statistical or structural properties of the response while leaving its semantic content unconstrained. ICWBench instantiates three families, summarized in Table 1. The three families impose constraints at progressively coarser granularities: Token-Set Preference (TSP) biases which tokens may appear, Word-Initial Preference (WIP) biases the initial letter of each word, and Sentence Acrostic (SA) requires sentence-initial letters to spell a target string. All three are independent of the query domain and admit automatic statistical verification. As shown in [10], they cover distinct forms of ICW IF while satisfying our criteria of generalizability, verifiability, quality preservation, and robustness.

3.2 Evaluation Metrics

For a family ψ with parameter κ, let n be the number of scored units in the response y (tokens for TSP and word-initials for WIP), and let X count how many of these positions are instruction-aligned. Under the null hypothesis H_0 that y is independent of ICW instructions, let p_0 denote the probability that a scored unit is instruction-aligned. The verification score is the z-statistic

$$z \;=\; \frac{X - n\,p_0}{\sqrt{n\,p_0\,(1-p_0)}}. \tag{1}$$

A larger z provides stronger evidence against H_0 and thus that LLMs follow the instruction, and y carries the watermark signal. TSP and WIP admit this closed form in (1). SA instead scores a longest-common-subsequence match against a permutation null, which generalizes (1). We instantiate the details of (X, n, p_0) per family in Appendix A.1.

4 Learning to Follow ICW Instructions

The automatic verifier (1) of ICW instructions enables a natural reward signal for training LLMs to follow ICW instructions through reinforcement learning. However, effective RLVR typically requires an initial policy that can sample rollouts with high reward, a condition that base LLMs may not satisfy on ICW instructions (see Figure 1). Therefore, an effective cold-start training strategy is needed to give RLVR a better starting point. The cold-start training faces two challenges: (i) how to

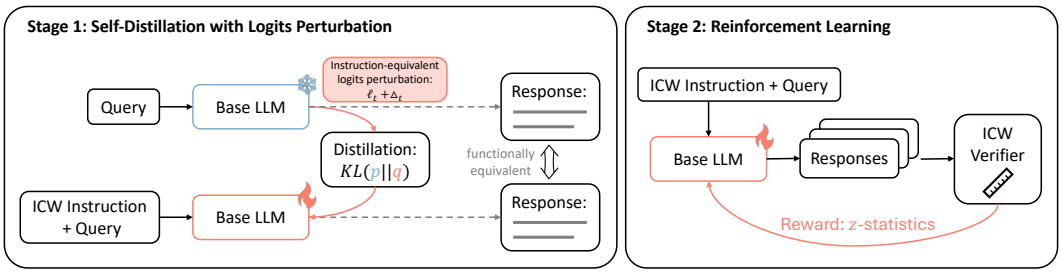

Figure 2: Our two-stage training workflow. Stage 1 (self-distillation with logits perturbation): A frozen base LLM, given only the query and with an ICW instruction-equivalent logits perturbation at decoding time, serves as the teacher; the student model (initially the same as the teacher model), conditioned on the ICW instruction and query, is trained to match the teacher's output distribution. Stage 2 (RL): The Stage-1 model is then refined by the reward of the automatic ICW verifier $V_\psi(\,\cdot\,;\kappa)$.

automatically synthesize high-quality training data; (ii) how to design a training method to effectively use the synthesized data.

To address these problems, we propose a two-stage approach (Figure 2). The first stage, *self-distillation with logits perturbation* (SDLP), bootstraps an initial policy that already produces aligned, high-quality rollouts, supplying RLVR with a better starting point. The second stage applies Group Relative Policy Optimization (GRPO) [28] on top of the SDLP-initialized policy, using (1) as the reward to further improve IF ability. We describe SDLP in Section 4.2 and RLVR in Section 4.3.

4.1 Training Data Synthesis

A high-quality ICW IF dataset is crucial for cold-start training, yet automatically synthesizing such data at scale is non-trivial. Direct sampling from the base LLM yields few instruction-aligned responses, since base LLMs do not reliably follow all ICW instructions. Distillation from a stronger LLM is also unreliable: frontier proprietary models also show limited ability to follow ICW instructions. The way forward comes from a property of ICW verification: it depends on the response's aggregate token statistics, not on a specific surface form.

The core idea is to enforce the IF at decoding time via instruction-equivalent logit perturbation, rather than rely on the base LLM to follow the ICW instruction. Concretely, given a query q, the base LLM $\mathcal{M}$ generates a response y autoregressively, one token y_t at a time. At each decoding step t, let $\boldsymbol{\ell}_t \in \mathbb{R}^{|\mathcal{V}|}$ denote the logits produced by $\mathcal{M}$ over vocabulary $\mathcal{V}$, conditioned on the prefix q and $y_{<t}$. We add an instruction-dependent perturbation $\boldsymbol{\Delta}_t(\psi;\kappa) \in \mathbb{R}^{|\mathcal{V}|}$ to $\boldsymbol{\ell}_t$ and sample the next token from the perturbed distribution:

$$y_t \;\sim\; \mathrm{softmax}(\boldsymbol{\ell}_t + \boldsymbol{\Delta}_t(\psi;\kappa))\,. \tag{2}$$

The perturbation $\boldsymbol{\Delta}_t$ is chosen to up-weight tokens that move y toward a higher verifier score $V_\psi(y;\kappa)$. For example, under the TSP family with parameter $T \subset \mathcal{V}$, a natural choice is

$$\boldsymbol{\Delta}_t[v] \;=\; \delta \cdot \mathbf{1}[v \in T], \qquad \delta > 0,$$

which uniformly boosts the logits of tokens in T and thereby raises the expected count of T-tokens in y, the very quantity scored by the z-statistic in (1). The concrete design of $\boldsymbol{\Delta}_t$ should be adapted to the specific ICW instruction family. For details of the perturbation design for three ICW instruction families, see Appendix A.1.

Repeating this procedure across different queries q and instruction parameters (ψ,κ) yields a synthetic cold-start dataset

$$\mathcal{D}_{\text{cs}} \;=\; \mathcal{D}_{H_1} \cup \mathcal{D}_{H_0} \;=\; \left\{ \left(\psi,\, \kappa,\, q^{(i)},\, y^{(i)}\right) \right\}_{i=1}^{N} \cup \left\{ \left(\hat{q}^{(i)},\, \hat{y}^{(i)}\right) \right\}_{i=1}^{M}. \tag{3}$$

The watermarked part $\mathcal{D}_{H_1}$ collects the perturbed samples: by construction, every $y^{(i)}$ follows the ICW instruction for (ψ,κ) scored by $V_\psi(\,\cdot\,;\kappa)$, where $\mathcal{M}$ does not see the instruction in its context. The unwatermarked part $\mathcal{D}_{H_0}$ collects pairs $(\hat{q}^{(i)},\hat{y}^{(i)})$, where $\hat{y}^{(i)}$ is text generated under null hypothesis. The evaluation of $\mathcal{D}_{\text{cs}}$ can be found in Appendix A.2, showing that it achieves high reward while maintaining good quality.

4.2 Self-Distillation with Logits Perturbation

A natural way to use $\mathcal{D}_{\text{cs}}$ is SFT on the realized sequences $y^{(i)}$. However, the ICW signal is encoded in the *distribution* of the response, spread across many token choices rather than any single emitted token. Standard SFT discards this distribution-level information at each decoding step.

We therefore distill the ICW distribution, rather than only the realized token. The remaining question is how to obtain a teacher distribution that already follows the ICW instruction. The logit perturbation $\boldsymbol{\Delta}_t(\psi;\kappa)$ from Section 4.1 directly defines such a distribution and can therefore serve as the teacher.

Teacher – student models setup. Both teacher and student models are copies of the base LLM $\mathcal{M}$. The teacher model remains frozen; the student model, denoted $\mathcal{M}_\theta$, is updated during training. Moreover, the two models receive different inputs at each decoding step t:

$$\boldsymbol{\ell}_t^{\mathcal{T}} \;\leftarrow\; \mathcal{M}\big(\cdot \,\big|\, q,\, y_{<t}\big)\,, \quad \boldsymbol{\ell}_t^{\mathcal{S}} \;\leftarrow\; \mathcal{M}_\theta\big(\cdot \,\big|\, \pi(\kappa),\, q,\, y_{<t}\big)\,.$$

Our core idea is to distill the teacher distribution into the student, so that given the ICW instruction $\pi(\kappa)$, the student learns to reproduce the distribution induced by the following perturbed decoding.

Self-distillation loss. The teacher distribution at step t is

$$p_t^{\mathcal{T}} \;=\; \mathrm{softmax}\Big(\boldsymbol{\ell}_t^{\mathcal{T}} + \boldsymbol{\Delta}_t\Big)\,, \qquad \boldsymbol{\Delta}_t \;=\; \begin{cases} \boldsymbol{\Delta}_t(\psi;\kappa) & \text{for samples in } \mathcal{D}_{H_1}, \\ \mathbf{0} & \text{for samples in } \mathcal{D}_{H_0}. \end{cases} \tag{4}$$

On $\mathcal{D}_{H_0}$, the target is the base LLM's natural output distribution (no instruction signal); on $\mathcal{D}_{H_1}$, it is the perturbed distribution. The student distribution is $p_t^{\mathcal{S}} = \mathrm{softmax}(\boldsymbol{\ell}_t^{\mathcal{S}})$. The SDLP loss is:

$$\mathcal{L}_{\text{SDLP}}(\theta) \;=\; \mathbb{E}_{(q,y)\sim\mathcal{D}_{\text{cs}}}\left[\sum_{t=1}^{|y|} \mathrm{KL}\big(\,\tilde{p}_t^{\mathcal{T}} \,\big\|\, \tilde{p}_t^{\mathcal{S}}\,\big)\right], \tag{5}$$

where $\mathrm{KL}\big(\tilde{p}_t^{\mathcal{T}} \big\| \tilde{p}_t^{\mathcal{S}}\big) = \sum_i \tilde{p}_t^{\mathcal{T}}(i) \log \frac{\tilde{p}_t^{\mathcal{T}}(i)}{\tilde{p}_t^{\mathcal{S}}(i)}$ is the KL divergence, and $\tilde{p}_t^{\mathcal{T}}, \tilde{p}_t^{\mathcal{S}}$ are top-k truncated distributions defined below.

Top-k truncation. In practice, low-probability tail of $p_t^{\mathcal{T}}$ contributes little meaningful training signal. Therefore, we restrict the KL term in (5) to the top-k tokens of the teacher distribution. Let $\mathcal{I}_t \;=\; \mathrm{TopK}\big(p_t^{\mathcal{T}},\, k\big) \;\subset\; \mathcal{V}$ denote the k token indices with the largest probability under $p_t^{\mathcal{T}}$. We restrict *both* $p_t^{\mathcal{T}}$ and $p_t^{\mathcal{S}}$ to this same support $\mathcal{I}_t$ and renormalize:

$$\tilde{p}_t^{\mathcal{T}}[v] \;=\; \frac{p_t^{\mathcal{T}}[v]\cdot \mathbf{1}[v\in\mathcal{I}_t]}{\sum_{v'\in\mathcal{I}_t} p_t^{\mathcal{T}}[v']}, \qquad \tilde{p}_t^{\mathcal{S}}[v] \;=\; \frac{p_t^{\mathcal{S}}[v]\cdot \mathbf{1}[v\in\mathcal{I}_t]}{\sum_{v'\in\mathcal{I}_t} p_t^{\mathcal{S}}[v']}. \tag{6}$$

4.3 Reinforcement Learning with Verifiable Rewards

After SDLP, the policy $\mathcal{M}_\theta$ learns to produce rollouts that follow ICW instructions and earn high rewards. We further improve $\mathcal{M}_\theta$ with GRPO [28], using the verifier $V_\psi(\,\cdot\,;\kappa)$ from Section 3.2 as the reward and rolling out only on $\mathcal{D}_{H_1}$. For each $(\psi,\kappa,q)\sim\mathcal{D}_{H_1}$, we sample G rollouts $\{y^{(i)}\}_{i=1}^{G}$ from the current policy $\mathcal{M}_{\theta_{\text{old}}}$, score them by $r_i = V_\psi(y^{(i)};\kappa)$, and compute group-relative advantages

$$A_i \;=\; \frac{r_i - \bar{r}}{\sigma_r}, \qquad \bar{r} \;=\; \frac{1}{G}\sum_{j=1}^{G} r_j, \qquad \sigma_r \;=\; \sqrt{\frac{1}{G}\sum_{j=1}^{G}(r_j-\bar{r})^2 + \varepsilon}. \tag{7}$$

Let $\rho_t^{(i)}(\theta) = \mathcal{M}_\theta\big(y_t^{(i)} \,\big|\, \pi(\kappa),\, q,\, y_{<t}^{(i)}\big) \,/\, \mathcal{M}_{\theta_{\text{old}}}\big(y_t^{(i)} \,\big|\, \pi(\kappa),\, q,\, y_{<t}^{(i)}\big)$ denote the per-token importance ratio. We maximize:

$$\mathcal{J}_{\text{RLVR}}(\theta) = \mathbb{E}_{(\psi,\kappa,q)\sim\mathcal{D}_{H_1}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y^{(i)}|}\sum_{t=1}^{|y^{(i)}|}\min\Big(\rho_t^{(i)}(\theta)A_i,\; \mathrm{clip}\big(\rho_t^{(i)}(\theta),\,1-\epsilon,\,1+\epsilon\big)A_i\Big)\right], \tag{8}$$

where ϵ is the clipping range.

5 Experiments

5.1 Experiment Settings

Models & Datasets & Baselines. *Evaluation queries.* We sample 500 queries from LFQA[1], a long-form question-answering dataset whose open-ended queries admit substantial response variation and thus expose the IF versus quality trade-off more clearly than short factual questions. The same 500 queries are reused across all three instruction families and all evaluated models.

Training data. The training set is built from LFQA queries disjoint from the 500 evaluation queries. For positives ($\mathcal{D}_{H_1}$), we use 3379, 1865, 2906 queries for TSP, WIP, and SA, respectively, and synthesize one instruction-aligned response per query using the data synthesis method in Section 4.1. For negatives ($\mathcal{D}_{H_0}$), we sample 1000, 800, 800 additional queries per family and generate a single response per query from the base model with the query alone as input. Together, the positive and negative examples form the cold-start dataset $\mathcal{D}_{\text{cs}}$ in (3).

Baselines & Models. We benchmark 14 frontier LLMs spanning two groups. *Proprietary*: GPT-5.4 [24], GPT-5.4-mini [69], GPT-5.4-nano [70], Gemini-3-pro [25], Gemini-3-flash [71], Claude-opus-4.5 [26], Claude-sonnet-4.5 [72], Claude-haiku-4.5 [73]. *Open-source*: DeepSeek-v4-pro, DeepSeek-v4-flash [27], Qwen3-235B, Qwen3-14B [74], GPT-OSS-120B, GPT-OSS-20B [75]. The same 14 models serve as the baselines against which we compare our trained models; alternative training methods (SFT, direct RL) are reported in the ablation study (Table 4). All models are queried with default decoding parameters and a generation cap of 600 new tokens.

Implementation Details. We train Qwen3-14B and GPT-OSS-20B, two open-source models with weak initial ICW IF ability among all LLMs we evaluate. All experiments are conducted on NVIDIA B200 GPUs and RTX PRO 6000 Blackwell GPUs. Full implementation details are in Appendix A.

Evaluation Metrics. We evaluate each model along two axes: ICW IF performance and response quality. All metrics are reported per instruction family. *ICW IF Performance:* For each query q, we collect a positive response y^+ generated under $\pi(\kappa)$ and q and a negative response y^- generated under q alone, and score both with the z-statistic in (1). We report ROC-AUC and TPR@1%FPR; the latter measures the true positive rate at a controlled 1% false positive rate. *Response Quality:* We assess response quality using both perplexity (PPL) and LLM-as-a-Judge. PPL is computed on y^+ using two external reference models, Nemotron-3-Super-120B [76] and Qwen3-235B, conditioned on the query q as the prefix; lower PPL indicates that the watermarked response remains a likely continuation of the query under these reference models. For LLM-as-a-Judge, we provide `gpt-4o-mini` with the query and generated response and ask it to assign an integer score from 1 to 5 along four dimensions: Relevance, Clarity, Informativeness, and Factual Correctness.

5.2 Main Results

5.2.1 Evaluation of frontier models on ICWBench

We evaluate the 14 frontier LLMs on ICWBench, reporting ICW IF performance in Table 2 and response quality in Table 3.

ICW IF Performance. IF performance varies sharply across both instruction families and models. SA is the easiest family: every evaluated LLM reaches AUC ≥ 0.83, and five proprietary models exceed TPR@1%FPR $= 0.98$. WIP is harder but tractable for some proprietary models: Gemini-3-pro, Gemini-3-flash, Claude-opus-4.5, and Claude-sonnet-4.5 all exceed TPR@1%FPR $= 0.94$, while other models stay below TPR@1%FPR $= 0.65$ (e.g., GPT-5.4 at 0.549) and 8 of them below 0.30 (e.g., DeepSeek-v4-pro at 0.260, GPT-OSS-120B at 0.166). TSP defeats every frontier model: the strongest is DeepSeek-v4-pro at TPR@1%FPR $= 0.120$, and the strongest TSP AUC across all evaluated LLMs is 0.658.

Text Quality. Across the three families, frontier models lack a controllable IF-quality trade-off. WIP shows the over-following failure mode. Gemini-3-pro pushes TPR@1%FPR to 1.000 but its

[1] `https://huggingface.co/datasets/vblagoje/lfqa`

Table 2: ICW instruction-following on ICWBench: ROC-AUC and TPR@1%FPR on each instruction family, for 14 frontier LLMs that were not trained for ICW IF and our two trained models. Averaged across the three families, our method raises TPR@1%FPR from 0.100 to 0.974 on Qwen3-14B and from 0.337 to 0.968 on GPT-OSS-20B. Answer quality for the same responses is reported separately in Table 3. † marks the two backbones we train.

	Models	TSP		WIP		SA	
		AUC ↑	TPR@1%FPR ↑	AUC	TPR@1%FPR	AUC	TPR@1%FPR
Proprietary	Gemini-3-pro	0.658	0.030	1.000	1.000	1.000	1.000
	Gemini-3-flash	0.646	0.040	0.998	0.958	1.000	1.000
	Claude-opus-4.5	0.597	0.022	0.998	0.983	1.000	0.998
	Claude-sonnet-4.5	0.487	0.011	0.996	0.948	1.000	0.983
	Claude-haiku-4.5	0.600	0.098	0.961	0.642	0.989	0.864
	GPT-5.4	0.538	0.020	0.933	0.549	0.998	0.983
	GPT-5.4-mini	0.527	0.020	0.692	0.059	0.913	0.551
	GPT-5.4-nano	0.462	0.000	0.529	0.004	0.973	0.824
Open-source	DeepSeek-v4-pro	0.579	0.120	0.841	0.260	0.999	0.992
	DeepSeek-v4-flash	0.576	0.030	0.832	0.277	0.957	0.665
	Qwen3-235B	0.591	0.020	0.837	0.252	0.995	0.937
	Qwen3-14B†	0.564	0.020	0.583	0.036	0.830	0.245
	GPT-OSS-120B	0.625	0.020	0.781	0.166	0.983	0.908
	GPT-OSS-20B†	0.575	0.010	0.568	0.015	0.998	0.985
Ours	**Qwen3-14B**	**0.993**	**0.941**	**0.999**	**0.996**	**0.999**	**0.985**
	GPT-OSS-20B	**0.983**	**0.922**	**0.992**	**0.989**	**0.999**	**0.992**

PPL_Q inflates to 67.61, roughly $10\times$ that of other models on WIP. The average LLM-as-a-Judge score drops to 3.94, the lowest among all evaluated models. TSP shows the opposite failure: PPL_Q stays in a normal range (2.78 – 7.21) and LLM-as-a-Judge scores remain high (4.58 – 4.98), but no model reaches AUC 0.66, so quality is preserved only because the constraint is not encoded at all. SA is the one ICW instruction where most models look balanced. This is because SA's constraint applies only at sentence-initial positions, so satisfying it leaves token distributions between sentences intact.

5.2.2 Evaluation of our method on ICWBench

We apply our method to two different open-source backbones with weak initial performance, Qwen3-14B and GPT-OSS-20B. The results are presented in Tables 2 and 3. Qwen3-14B has weak ICW IF ability across all three families, with TPR@1%FPR of 0.020, 0.036, and 0.245 on TSP, WIP, and SA, respectively. GPT-OSS-20B has near-zero TPR@1%FPR on TSP and WIP (0.010 and 0.015) and good performance on SA (0.985). After training, Qwen3-14B reaches TPR@1%FPR of 0.941/0.996/0.985, increasing its average TPR@1%FPR from 0.100 to 0.974. GPT-OSS-20B reaches 0.922/0.989/0.992, increasing its average from 0.337 to 0.968. The consistent ICW IF improvements across both backbones show that our method can effectively improve ICW IF ability across different model families.

These gains are achieved while maintaining high response quality. Across both perplexity and LLM-as-a-Judge, our trained models remain comparable to the original backbones, suggesting that the substantial gains in ICW IF are achieved without a degradation in overall response quality. The advantage is particularly clear on WIP: although Gemini-3-pro achieves near-perfect ICW IF, its response quality deteriorates substantially, with $\text{PPL}_Q/\text{PPL}_N$ of $67.61/53.51$ and an average judge score of 3.94. In comparison, our trained Qwen3-14B obtains $\text{PPL}_Q/\text{PPL}_N$ of $4.99/4.96$ and a judge score of 4.88, while GPT-OSS-20B obtains $7.15/6.36$ and 4.52, respectively. Current results use one instruction template per family, a different key for each sample, and queries drawn from LFQA. Appendix B reports ICW IF performance under a different instruction template, on two unseen query domains, and across held-out watermark keys.

Table 3: Response quality on ICWBench, measured on the same responses as Table 2. PPL_Q and PPL_N are perplexities computed by Qwen3-235B and Nemotron-3-Super-120B, two reference models from different families. R, C, I, and FC are LLM-as-a-Judge scores (`gpt-4o-mini`) for Relevance, Clarity, Informativeness, and Factual Correctness, each an integer in 1–5; Avg is their mean. Our trained models remain comparable to their backbones and to the frontier baselines on both metrics. † marks the two backbones we train.

	Models	TSP							WIP							SA						
		Perplexity ↓		LLM-as-a-Judge ↑					Perplexity ↓		LLM-as-a-Judge ↑					Perplexity ↓		LLM-as-a-Judge ↑				
		PPL_Q	PPL_N	R	C	I	FC	Avg	PPL_Q	PPL_N	R	C	I	FC	Avg	PPL_Q	PPL_N	R	C	I	FC	Avg
Proprietary	Gemini-3-pro	5.55	5.13	4.91	4.59	4.11	4.72	4.58	67.61	53.51	4.40	3.57	3.60	4.18	3.94	6.64	6.38	5.00	4.98	5.00	4.69	4.92
	Gemini-3-flash	3.35	3.30	5.00	4.86	4.96	4.56	4.85	13.62	11.86	4.98	4.57	4.63	4.47	4.66	8.28	7.81	5.00	4.87	5.00	4.57	4.86
	Claude-opus-4.5	4.07	3.92	4.92	4.88	4.87	4.72	4.85	7.04	6.34	5.00	4.99	4.98	4.71	4.92	6.54	5.80	5.00	4.97	5.00	4.71	4.92
	Claude-sonnet-4.5	3.49	3.53	4.89	4.85	4.83	4.64	4.80	6.53	5.88	5.00	4.99	4.99	4.73	4.93	5.01	4.48	5.00	4.98	4.99	4.77	4.93
	Claude-haiku-4.5	4.47	4.39	4.93	4.83	4.78	4.71	4.81	5.50	5.14	5.00	4.99	4.96	4.77	4.93	6.17	5.64	5.00	4.97	4.98	4.79	4.94
	GPT-5.4	6.75	6.23	5.00	4.99	4.99	4.93	4.98	8.21	7.54	5.00	4.97	4.98	4.92	4.97	8.33	7.47	5.00	4.94	4.99	4.94	4.97
	GPT-5.4-mini	5.55	5.16	5.00	4.99	4.97	4.94	4.98	5.94	5.44	4.99	4.99	4.95	4.91	4.96	7.23	6.32	5.00	4.98	4.95	4.92	4.96
	GPT-5.4-nano	7.21	6.33	4.99	4.98	4.99	4.94	4.98	7.24	6.46	4.98	4.98	4.95	4.96	4.97	8.00	7.01	5.00	4.92	4.99	4.92	4.96
Open-source	DeepSeek-v4-pro	3.81	3.65	4.97	4.98	4.94	4.76	4.91	4.43	4.36	4.99	4.98	4.89	4.76	4.91	6.34	5.64	5.00	4.97	4.99	4.76	4.93
	DeepSeek-v4-flash	3.08	3.24	5.00	4.97	4.96	4.71	4.91	4.87	4.74	4.98	4.95	4.75	4.68	4.84	6.22	5.87	4.99	4.89	4.76	4.56	4.80
	Qwen3-235B	2.78	4.27	5.00	4.96	4.99	4.72	4.92	3.98	5.75	4.97	4.96	4.88	4.82	4.91	4.20	5.84	4.99	4.94	4.98	4.80	4.92
	Qwen3-14B†	2.80	3.17	4.94	4.95	4.93	4.72	4.89	3.48	3.57	4.96	4.96	4.78	4.72	4.86	4.05	4.21	4.95	4.90	4.74	4.69	4.82
	GPT-OSS-120B	4.34	3.15	5.00	4.79	4.87	4.65	4.83	6.33	4.64	4.85	4.66	4.66	4.65	4.71	7.06	5.55	4.97	4.57	4.84	4.44	4.71
	GPT-OSS-20B†	4.06	3.19	4.95	4.86	4.91	4.65	4.84	5.43	4.30	4.67	4.51	4.51	4.43	4.53	5.35	4.44	4.97	4.79	4.95	4.53	4.81
Ours	**Qwen3-14B**	**5.74**	**5.72**	**4.97**	**4.88**	**4.94**	**4.60**	**4.85**	**4.99**	**4.96**	**4.98**	**4.95**	**4.97**	**4.61**	**4.88**	**6.25**	**6.28**	**4.89**	**4.86**	**4.75**	**4.70**	**4.80**
	GPT-OSS-20B	**7.29**	**7.10**	**4.88**	**4.82**	**4.79**	**4.65**	**4.79**	**7.15**	**6.36**	**4.58**	**4.51**	**4.59**	**4.39**	**4.52**	**7.38**	**6.48**	**4.90**	**4.80**	**4.76**	**4.44**	**4.73**

5.2.3 Robustness under different attacks

We evaluate robustness against three text perturbations applied to watermarked responses (Figure 3). *Paraphrase* rewrites the response text by an LLM. *Word replacement* substitutes 30% of words with synonyms. *Word deletion* drops 30% of words at random. Across different attacks, our trained models achieve state-of-the-art AUC on TSP. On WIP and SA, our trained models are competitive with the strongest baselines. Appendix C extends this evaluation to multi-round paraphrasing, back-translation, summarize-then-expand editing, and an RL-trained adaptive removal attack.

5.3 Ablation Study

Table 4: Ablation of training strategies on Qwen3-14B across three ICW instruction families. PPL is the mean of the two reference-model perplexities (Qwen3-235B and Nemotron-3-Super-120B), and Judge is the mean of the four LLM-as-a-Judge dimensions. SDLP produces a strong cold start across all three families, and RL further improves IF capability. * marks the SA run of Base + RL, whose detection scores come with visible reward hacking in text formatting.

Models	TSP				WIP				SA			
	AUC ↑	TPR@1%FPR ↑	PPL ↓	Judge ↑	AUC	TPR@1%FPR	PPL	Judge	AUC	TPR@1%FPR	PPL	Judge
Base Model	0.564	0.020	2.99	4.89	0.583	0.036	3.53	4.86	0.830	0.245	4.13	4.82
Base + SFT	0.868	0.382	3.63	4.89	0.997	0.939	3.53	4.70	0.518	0.006	3.17	4.88
Base + RL	0.565	0.199	4.25	4.75	0.519	0.008	3.16	4.79	0.999*	0.994*	3.89*	4.76*
Base + SDLP	0.968	0.822	4.88	4.83	0.994	0.925	4.10	4.88	0.978	0.849	7.90	4.63
Base + SDLP + RL	0.993	0.941	5.73	4.85	0.999	0.996	4.98	4.88	0.999	0.985	6.27	4.80

We ablate different training strategies against the same base model. Beyond our full method, we compare three alternatives (Table 4): SFT as cold start; RL, GRPO with the verifier reward applied directly on the base model with no cold start; and SDLP, SDLP only without RL.

SFT is instruction-dependent. SFT's effectiveness varies across the three families. On WIP, SFT is effective: TPR@1%FPR rises from 0.036 to 0.939. On TSP, the lift is partial. On SA, SFT is ineffective. SFT is therefore not a reliable cold-start strategy across ICW instructions.

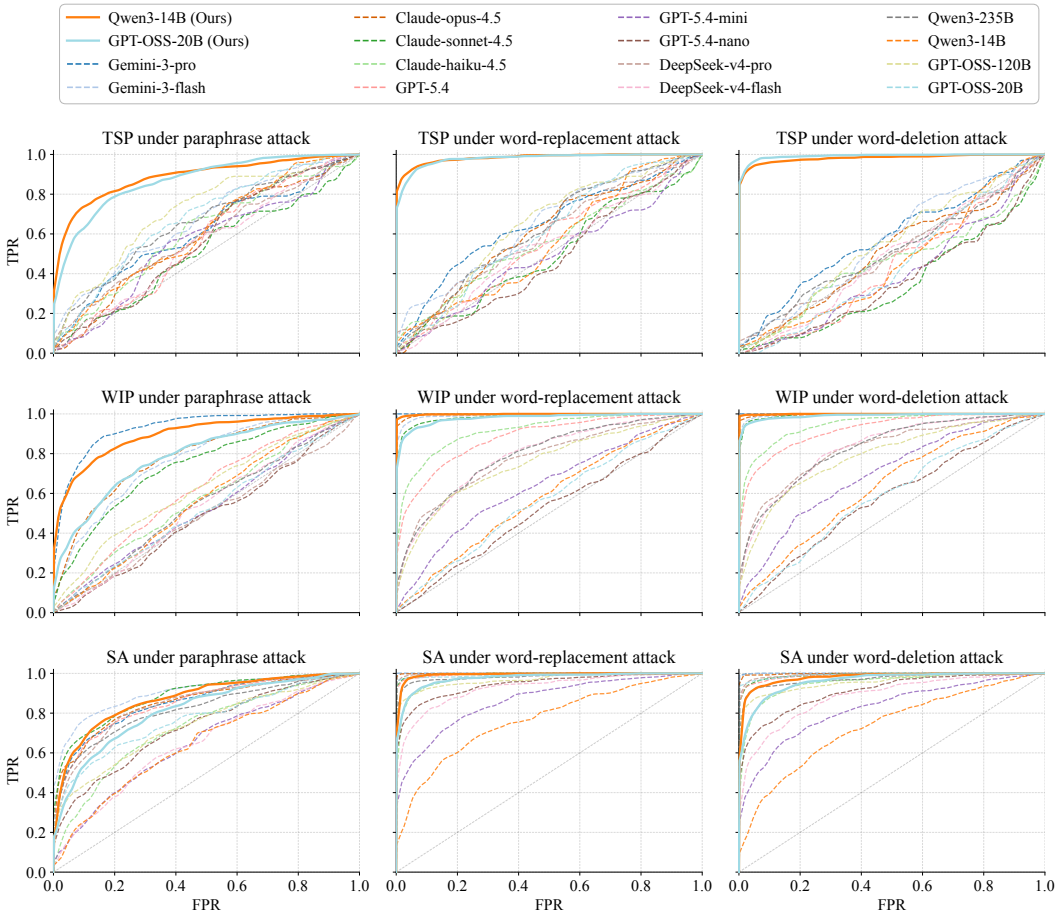

Figure 3: Robustness of ICWs under different attacks. Rows correspond to the three instruction families (TSP, WIP, SA) and columns correspond to the three attacks (paraphrase, word replacement, word deletion). Each panel overlays ROC curves of our trained Qwen3-14B and GPT-OSS-20B, and the 14 evaluated frontier proprietary and open-source LLMs on attacked responses.

RL alone is insufficient. Initialized directly from the base and trained with the verifier reward, RL shows limited effectiveness on both TSP and WIP. On SA, the run numerically reaches TPR@1%FPR 0.994, but we observe obvious reward hacking in which the model introduces unnatural formatting and word-splitting artifacts to place target letters at verifier-recognized sentence-initial positions, rather than satisfying the acrostic constraint through natural sentence construction.

SDLP consistently shows strong performance on all three ICW instructions. It demonstrates the effectiveness of SDLP in converting instruction-following into a decoding-time logit perturbation, which facilitates model training.

6 Conclusion

In this paper, we introduce ICWBench, a benchmark of three verifiable ICW instructions, to extensively evaluate 14 frontier proprietary and open-source LLMs. We observe the limitations of frontier LLMs in following ICW instructions: under-following and over-following. We propose an effective, scalable, and self-contained two-stage training method (SDLP and RLVR) that makes the ICW IF of Qwen3-14B and GPT-OSS-20B achieve a better IF-quality trade-off than evaluated frontier models on TSP and WIP, and matches the strongest proprietary baselines on SA. The effectiveness of our method shows that ICWs are a promising approach to equip LLMs with a watermarking interface that third parties can invoke without access to model internals, and we hope our benchmark and training method can facilitate future research in this direction.

Acknowledgments

The work of Yepeng Liu and Yuheng Bu was partially supported by NSF under Grant CCF-2543381.

References

[1] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

[2] Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, 2024.

[3] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

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

[5] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37: 62557–62583, 2024.

[6] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[7] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36: 68539–68551, 2023.

[8] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

[9] Yepeng Liu, Yu Huang, Yu-Xiang Wang, Yingbin Liang, and Yuheng Bu. Convexbench: Can llms recognize convex functions? *arXiv preprint arXiv:2602.01075*, 2026.

[10] Yepeng Liu, Xuandong Zhao, Christopher Kruegel, Dawn Song, and Yuheng Bu. In-context watermarks for large language models. *arXiv preprint arXiv:2505.16934*, 2025.

[11] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR, 2023.

[12] Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*, 2023.

[13] Yepeng Liu and Yuheng Bu. Adaptive text watermark for large language models. *arXiv preprint arXiv:2401.13927*, 2024.

[14] Haiyun He, Yepeng Liu, Ziqiao Wang, Yongyi Mao, and Yuheng Bu. Theoretically grounded framework for llm watermarking: A distribution-adaptive approach. *arXiv preprint arXiv:2410.02890*, 2024.

[15] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

[16] Scott Aaronson. Watermarking of large language models. https://simons.berkeley.edu/talks/scott-aaronson-ut-austin-openai-2023-08-17, 2023. Accessed: 2023-08.

[17] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023.

[18] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36, 2024.

[19] 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*, 2023.

[20] Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip S Yu. An unforgeable publicly verifiable watermark for large language models. In *The twelfth international conference on learning representations*, 2023.

[21] Yepeng Liu, Xuandong Zhao, Dawn Song, Gregory W Wornell, and Yuheng Bu. Position: Llm watermarking should align stakeholders' incentives for practical adoption. *arXiv preprint arXiv:2510.18333*, 2025.

[22] ICML 2026. On violations of llm review policies. https://blog.icml.cc/2026/03/18/on-violations-of-llm-review-policies/, 2026. Accessed: 2026-03-18.

[23] Vishisht Srihari Rao, Aounon Kumar, Himabindu Lakkaraju, and Nihar B Shah. Detecting llm-generated peer reviews. *PLoS One*, 20(9):e0331871, 2025.

[24] OpenAI. GPT-5.4, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.4.

[25] Google DeepMind. Gemini 3 Pro, 2025. URL https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/3-pro.

[26] Anthropic. Claude Opus 4.5, 2025. URL https://www.anthropic.com/news/claude-opus-4-5.

[27] DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. Technical report, DeepSeek, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf.

[28] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[29] Li An, Yujian Liu, Yepeng Liu, Yuheng Bu, Yang Zhang, and Shiyu Chang. A reinforcement learning framework for robust and secure llm watermarking. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7181–7198, 2026.

[30] Yepeng Liu, Xuandong Zhao, Dawn Song, and Yuheng Bu. Dataset protection via watermarked canaries in retrieval-augmented llms. *arXiv preprint arXiv:2502.10673*, 2025.

[31] Li An, Yujian Liu, Yepeng Liu, Yang Zhang, Yuheng Bu, and Shiyu Chang. Defending llm watermarking against spoofing attacks with contrastive representation learning. *arXiv preprint arXiv:2504.06575*, 2025.

[32] Yepeng Liu, Yiren Song, Hai Ci, Yu Zhang, Haofan Wang, Mike Zheng Shou, and Yuheng Bu. Image watermarks are removable using controllable regeneration from clean noise. *arXiv preprint arXiv:2410.05470*, 2024.

[33] Pei Yang, Yepeng Liu, Kelly Peng, Yuan Gao, and Yiren Song. Tokenpure: Watermark removal through tokenized appearance and structural guidance. *arXiv preprint arXiv:2512.01314*, 2025.

[34] Xianjun Yang, Liangming Pan, Xuandong Zhao, Haifeng Chen, Linda Petzold, William Yang Wang, and Wei Cheng. A survey on detection of LLMs-generated content. *arXiv preprint arXiv:2310.15654*, 2023.

[35] Xuandong Zhao, Sam Gunn, Miranda Christ, Jaiden Fairoze, Andres Fabrega, Nicholas Carlini, Sanjam Garg, Sanghyun Hong, Milad Nasr, Florian Tramer, et al. Sok: Watermarking for ai-generated content. *arXiv preprint arXiv:2411.18479*, 2024.

[36] Yapei Chang, Kalpesh Krishna, Amir Houmansadr, John Wieting, and Mohit Iyyer. Postmark: A robust blackbox watermark for large language models. *arXiv preprint arXiv:2406.14517*, 2024.

[37] Chaoyi Zhu, Jeroen Galjaard, Pin-Yu Chen, and Lydia Y Chen. Duwak: Dual watermarks in large language models. *arXiv preprint arXiv:2403.13000*, 2024.

[38] Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. {REMARK-LLM}: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1813–1830, 2024.

[39] 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.

[40] Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Protecting language generation models via invisible watermarking. In *International Conference on Machine Learning*, pages 42187–42199. PMLR, 2023.

[41] Hai Ci, Pei Yang, Yiren Song, and Mike Zheng Shou. Ringid: Rethinking tree-ring watermarking for enhanced multi-key identification. In *European conference on computer vision*, pages 338–354. Springer, 2024.

[42] Hanbo Huang, Yiran Zhang, Hao Zheng, Xuan Gong, Yihan Li, Lin Liu, Zhuotao Liu, and Shiyu Liang. Rlcracker: Evaluating the worst-case vulnerability of llm watermarks with adaptive rl attacks. *arXiv preprint arXiv:2509.20924*, 2025.

[43] Haiyun He, Yepeng Liu, Zhuoer Shen, Ziqiao Wang, Yongyi Mao, and Yuheng Bu. Fundamental trade-offs in multi-bit watermarking of stochastic processes. *arXiv preprint arXiv:2605.08826*, 2026.

[44] George Alexandru Adam, Alexander Cui, Edwin Thomas, Emily Napier, Nazar Shmatko, Jacob Schnell, Jacob Junqi Tian, Alekhya Dronavalli, Edward Tian, and Dongwon Lee. Gptzero: Robust detection of llm-generated texts, 2026. URL `https://arxiv.org/abs/2602.13042`.

[45] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International conference on machine learning*, pages 24950–24962. PMLR, 2023.

[46] Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. Radar: Robust ai-text detection via adversarial learning. *Advances in neural information processing systems*, 36:15077–15095, 2023.

[47] 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.

[48] Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. Gpt detectors are biased against non-native english writers. *Patterns*, 4(7), 2023.

[49] Xuandong Zhao, Lei Li, and Yu-Xiang Wang. Permute-and-flip: An optimally stable and watermarkable decoder for llms. *arXiv preprint arXiv:2402.05864*, 2024.

[50] Haiyun He, Yepeng Liu, Ziqiao Wang, Yongyi Mao, and Yuheng Bu. Distributional information embedding: A framework for multi-bit watermarking. *arXiv preprint arXiv:2501.16558*, 2025.

[51] Xiang Li, Feng Ruan, Huiyuan Wang, Qi Long, and Weijie J Su. A statistical framework of watermarks for large language models: Pivot, detection efficiency and optimal rules. *The Annals of Statistics*, 53(1):322–351, 2025.

[52] Xiang Li, Feng Ruan, Huiyuan Wang, Qi Long, and Weijie J Su. Robust detection of watermarks for large language models under human edits. *arXiv preprint arXiv:2411.13868*, 2024.

[53] Eva Giboulot and Teddy Furon. Watermax: breaking the llm watermark detectability-robustness-quality trade-off. *arXiv preprint arXiv:2403.04808*, 2024.

[54] Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2023.

[55] 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 e-prints*, pages arXiv–2401, 2024.

[56] Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang. Improved unbiased watermark for large language models. *arXiv preprint arXiv:2502.11268*, 2025.

[57] Xuehao Cui, Ruibo Chen, Yihan Wu, and Heng Huang. Mc 2 mark: Distortion-free multi-bit watermarking for long messages. *arXiv preprint arXiv:2602.14030*, 2026.

[58] Ruibo Chen, Yihan Wu, Xuehao Cui, Jingqi Zhang, and Heng Huang. More haste, less speed: Weaker single-layer watermark improves distortion-free watermark ensembles. *arXiv preprint arXiv:2602.11793*, 2026.

[59] Yihan Wu, Xuehao Cui, Ruibo Chen, and Heng Huang. Analyzing and evaluating unbiased language model watermark. *arXiv preprint arXiv:2509.24048*, 2025.

[60] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

[61] Norman Mu, Sarah Chen, Zifan Wang, Sizhe Chen, David Karamardian, Lulwa Aljeraisy, Basel Alomair, Dan Hendrycks, and David Wagner. Can llms follow simple rules? *arXiv preprint arXiv:2311.04235*, 2023.

[62] Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. Infobench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13025–13048, 2024.

[63] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. In *International conference on machine learning*, pages 22631–22648. PMLR, 2023.

[64] Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

[65] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508, 2023.

[66] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[67] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[68] Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *arXiv preprint arXiv:2406.13542*, 2024.

[69] OpenAI. GPT-5.4 mini, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.4-mini.

[70] OpenAI. GPT-5.4 nano, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.4-nano.

[71] Google DeepMind. Gemini 3 Flash, 2025. URL https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/3-flash.

[72] Anthropic. Claude Sonnet 4.5, 2025. URL https://www.anthropic.com/news/claude-sonnet-4-5.

[73] Anthropic. Claude Haiku 4.5, 2025. URL https://www.anthropic.com/news/claude-haiku-4-5.

[74] Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[75] OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

[76] NVIDIA. Nvidia nemotron 3: Efficient and open intelligence, 2025. URL https://arxiv.org/abs/2512.20856. White Paper.

[77] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artières, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138, 2015.

[78] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. WWW'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018 (WWW '18 Companion)*, pages 1941–1942, 2018.

[79] John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. Paraphrastic representations at scale. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 379–388, 2022.

[80] 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. In *International Conference on Machine Learning (ICML)*, 2024.

A Implementation Details

This appendix gives the per-family instantiation of the verifier V_ψ and the logit perturbation $\mathbf{\Delta}_t(\psi;\kappa)$ used by both the cold-start data synthesis (Section 4.1) and the SDLP teacher distribution (Section 4.2). It then reports the statistics of the cold-start dataset $\mathcal{D}_{\text{cs}}$, and lists the instruction templates together with the procedure that maps a seed to a key (Appendix A.3).

A.1 Verifier and Logit Perturbation per Instruction Family

Rather than using the full vocabulary, we use the English-token subset $\mathcal{V}_{\text{en}} = \{v \in \mathcal{V} : v \text{ is ASCII}\} \setminus \mathcal{V}_{\text{special}}$. For Qwen3, $|\mathcal{V}_{\text{en}}| = 88{,}492$, compared to $|\mathcal{V}| = 151{,}936$. We further denote by $\mathcal{V}_{\text{en}}^{\sqcup} \subset \mathcal{V}_{\text{en}}$ the subset whose surface form is a leading space followed by an ASCII letter (i.e. word-initial subwords); $|\mathcal{V}_{\text{en}}^{\sqcup}| = 41{,}547$. For $v \in \mathcal{V}_{\text{en}}^{\sqcup}$, let $\phi(v) \in \{\texttt{A}, \ldots, \texttt{Z}\}$ denote the (uppercased) first letter of v. We restrict to $\mathcal{V}_{\text{en}}$ rather than $\mathcal{V}$ to keep the context short: TSP serializes T as an explicit token list in the prompt with $|T| = \gamma\,|\mathcal{V}_{\text{en}}|$, so working over $\mathcal{V}_{\text{en}}$ instead of $\mathcal{V}$ significantly reduces the context overhead.

For each family, V_ψ instantiates the z-statistic in (1) by specifying the triple (X, n, p_0), and $\mathbf{\Delta}_t$ instantiates (2).

A.1.1 Token-Set Preference (TSP)

Parameter. $T \subseteq \mathcal{V}_{\text{en}}$, sampled by drawing a uniform random subset of size $\lfloor \gamma\,|\mathcal{V}_{\text{en}}| \rfloor$, where the subset ratio $\gamma \in (0,1)$ is a fixed hyperparameter. We use $\gamma = 0.2$ for evaluation and $\gamma \in \{0.1, 0.2, 0.3\}$ for data synthesis.

Verifier V_{TSP}. Following the unique-token construction of [11], let $\mathcal{U}(y) = \{v \in \mathcal{V}_{\text{en}} : v \text{ appears in } y\}$ be the set of unique tokens appearing in y. We set

$$X \;=\; |\mathcal{U}(y) \cap T|, \qquad n \;=\; |\mathcal{U}(y)|, \qquad p_0 \;=\; \gamma.$$

Under the null hypothesis H_0 (independence of y from the watermark key T), each unique token is in T with probability γ since T is a uniform random subset of $\mathcal{V}_{\text{en}}$, and z in (1) reduces to the standard binomial z-test.

Logit perturbation. The perturbation is independent of decoding state:

$$\mathbf{\Delta}_t^{\text{TSP}}[v] \;=\; \delta_{\text{TSP}} \cdot \mathbf{1}[v \in T]\,,$$

with strength $\delta_{\text{TSP}} = 3.0$ during synthesis.

A.1.2 Word-Initial Preference (WIP)

Parameter. $L \subset \{\texttt{A}, \ldots, \texttt{Z}\}$ with $|L| = 13$, sampled randomly from the alphabet with a key.

Verifier V_{WIP}. Let $\mathcal{U}^{\sqcup}(y) = \mathcal{U}(y) \cap \mathcal{V}_{\text{en}}^{\sqcup}$. We set

$$X \;=\; \big|\{v \in \mathcal{U}^{\sqcup}(y) \,:\, \phi(v) \in L\}\big|, \qquad n \;=\; \big|\mathcal{U}^{\sqcup}(y)\big|, \qquad p_0 \;=\; \sum_{\alpha \in L} \hat{f}_\alpha,$$

where $\hat{f}_\alpha \;=\; |\{v \in \mathcal{V}_{\text{en}}^{\sqcup} \,:\, \phi(v) = \alpha\}| / |\mathcal{V}_{\text{en}}^{\sqcup}|$ is the empirical fraction of word-initial English tokens whose first letter is α.

Logit perturbation. As with TSP, the perturbation is independent of decoding state:

$$\mathbf{\Delta}_t^{\text{WIP}}[v] \;=\; \delta_{\text{WIP}} \cdot \mathbf{1}[v \in \mathcal{V}_{\text{en}}^{\sqcup} \;\wedge\; \phi(v) \in L]\,,$$

with strength $\delta_{\text{WIP}} = 3.0$ during synthesis. Equivalently, $\mathbf{\Delta}_t^{\text{WIP}}$ uniformly adds logit perturbation for every word-initial English token whose first letter lies in the in-key letter set.

A.1.3 Sentence Acrostic (SA)

Parameter. $S = s_1 s_2 \cdots s_k$, drawn by sampling each character s_i uniformly from a 20-letter pool $\mathcal{P} =$ `ABCDEFGHILMNOPRSTUWY` (the alphabet minus {J, K, Q, V, X, Z}, whose word-initial frequency is below 1.5%). We use $|S| = 18$ for evaluation; for synthesis, $|S|$ is sampled uniformly in $\{18, 19, 20\}$ per training example.

Unlike TSP and WIP, SA constrains first letters at sentence-initial positions, which require text-level segmentation. We define an extractor $\Phi(y)$, which maps a response text y to the concatenation of the first ASCII letter at each detected sentence-initial position.

Verifier V_{SA}. The observed signal is the longest common subsequence (LCS) length

$$X \;=\; |\mathrm{LCS}(S,\, \Phi(y))|.$$

The null distribution is empirical: let ζ range over uniform random permutations of the multiset $\Phi(y)$, define the permutation null mean and variance

$$\mu_S \;=\; \mathbb{E}_\zeta[|\mathrm{LCS}(S,\zeta)|]\,, \qquad \sigma_S^2 \;=\; \mathrm{Var}_\zeta[|\mathrm{LCS}(S,\zeta)|]\,,$$

and report the z-statistic

$$z \;=\; \frac{X - \mu_S}{\sigma_S}.$$

This generalizes (1): under the Bernoulli null used by TSP and WIP, $(np_0,\ np_0(1-p_0)) = (\mu, \sigma^2)$ admits a closed form, while SA has no such closed form because LCS depends on the whole sequence rather than the count alone. We estimate (μ_S, σ_S) with $N = 1000$ permutations of $\Phi(y)$. Using LCS rather than per-position equality makes the verifier robust to extra sentences: if $|\Phi(y)| > k$, the secret string S may still match $\Phi(y)$ as a subsequence anywhere, with mismatches and excess letters not penalized.

Logit perturbation. Intuitively, we treat the secret $S = s_1 \cdots s_k$ as a queue of letters to emit: at each new sentence start, we bias tokens whose first letter is the head of the queue, and pop the queue either when the model successfully emits that letter or fails in three consecutive sentences. Thus, the perturbation is *stateful*.

State. Per request we keep two integer counters: (1) a target index $\tau \in \{0, \ldots, k\}$, the index of the next letter of S we want to emit; (2) a fail counter $f \in \{0, 1, 2, 3\}$, the number of sentences that have failed to match the current target letter s_τ. Both start at 0.

When to perturb logits. A decoding step t is marked as *active* for logits perturbation iff (i) $\tau < k$ (the secret S is not yet fully emitted) and (ii) the generated text $y_{<t}$ ends a sentence which can be detected by the extractor Φ.

What logit bias to apply. Let $\phi(v)$ be the first ASCII letter of v. When the step is *active*, we bias every token whose first letter is the current target:

$$\boldsymbol{\Delta}_t^{\mathrm{SA}}[v] \;=\; \delta_{\mathrm{SA}} \cdot \mathbf{1}[\phi(v) = s_\tau] \qquad \text{(active step)},$$

and $\boldsymbol{\Delta}_t^{\mathrm{SA}} = \mathbf{0}$ otherwise.

Table 5: Cold-start dataset $\mathcal{D}_{\mathrm{cs}}$ statistics by instruction family. z-scores and perplexities are computed on each split independently; H_1 samples carry the in-context watermark, H_0 samples do not.

	Size		Mean z		Perplexity	
Family	$\mathcal{D}_{H_1}$	$\mathcal{D}_{H_0}$	$\mathcal{D}_{H_1}$	$\mathcal{D}_{H_0}$	$\mathcal{D}_{H_1}$	$\mathcal{D}_{H_0}$
TSP	3,379	1,000	7.86	−1.07	4.18	3.20
WIP	1,865	800	9.68	−2.06	4.35	3.20
SA	2,906	800	7.91	−0.09	3.86	3.21

A.2 Cold-Start Dataset $\mathcal{D}_{\text{cs}}$ Statistics

This section reports the size and z-statistics of the cold-start dataset $\mathcal{D}_{\text{cs}} = \mathcal{D}_{H_1} \cup \mathcal{D}_{H_0}$ defined in (3). Each $(\psi, \kappa, q^{(i)}, y^{(i)}) \in \mathcal{D}_{H_1}$ is generated by sampling $y^{(i)} \sim \mathcal{M}(\cdot \mid q^{(i)})$ with the family-specific perturbation $\mathbf{\Delta}_t(\psi; \kappa)$ injected at decoding time, and each $(\hat{q}^{(i)}, \hat{y}^{(i)}) \in \mathcal{D}_{H_0}$ is generated from $\mathcal{M}$ on a clean query with no perturbation. All queries are drawn from the LFQA training split, disjoint from the 500 evaluation queries.

Table 5 reports $|\mathcal{D}_{H_1}|$, $|\mathcal{D}_{H_0}|$, and the average verifier score on $\mathcal{D}_{H_1}$. For all three families, mean z on $\mathcal{D}_{H_1}$ is well above the detection threshold at 1%FPR, confirming that $\mathcal{D}_{\text{cs}}$ contains a strong watermark signal.

A.3 Instruction Templates and Key Instantiation

This section gives the prompt text of the three ICW instruction families and the procedure that maps a seed to a watermark key. For every query, the paired negative response is generated from the query alone, so a positive/negative pair differs only by the presence of the instruction.

TSP instruction template

```
<green>
{green_tokens}
</green>

Respond to the user query. Seamlessly incorporate as many tokens
from <green> as possible without compromising text quality.
```

WIP instruction template

```
<green>
{green_letters}
</green>

<red>
{red_letters}
</red>

Given the <green> and <red> letter lists, respond to the user query
with clarity, accuracy, informativeness, and relevance. Favor words
beginning with letters from <green> and minimize those beginning with
letters from <red>. Never reveal the <green> and <red> letter lists
in your reply.
```

SA instruction template

```
You will receive a SECRET STRING and a user query. Answer the query
naturally and helpfully. While doing so, structure the answer so the
first letters of your sentences can spell out the secret string in
order. Treat the acrostic as a soft guide that you follow whenever it
does not hurt the response.

Rules:
1. Track the next unmatched letter in the secret string as the target
   letter. It starts at the first letter and only advances when a
   sentence successfully starts with it.
2. Prefer to start each new sentence with the target letter. If
   starting with the target letter would clearly hurt the response
   quality, write a non-matching sentence instead. After three
```

```
   consecutive misses, drop that letter and advance to the next.
3. Once the secret string is fully consumed, continue answering
   naturally with no further letter constraints.
4. Write in plain narrative prose. Do not visually highlight first
   letters in any way.
```

In every family the key is a deterministic function of a seed, so an instruction instance is described by the triple (family, seed, length or ratio parameter). Table 6 states the mapping. Training and evaluation draw from disjoint seed pools, so no evaluation key is seen during training.

Table 6: Key instantiation per instruction family. Training and evaluation use disjoint seeds.

Family	**Key**	**Instantiation from a seed**	**Training instances**
TSP	Token set $T \subseteq \mathcal{V}_{\text{en}}$	Sample $T \subseteq \mathcal{V}_{\text{en}}$ tokens uniformly from $\mathcal{V}$ with the seed, $\gamma \in (0, 1)$.	One seed per example; γ drawn from $\{0.1, 0.2, 0.3\}$.
WIP	Letter set L	Sample $\lvert L\rvert$ letters uniformly from the alphabet with the seed.	One seed per example; $\gamma = 0.5$.
SA	Secret string S	Sample $\lvert S\rvert$ letters with a seeded RNG and concatenate them into a string.	One seed per example; $\lvert S\rvert$ drawn uniformly from $\{18, 19, 20\}$.

B Generalization of ICW Instruction Following

Section 3.1 describes an ICW instance by an instruction π, a key κ, and a query q. Training draws different keys, uses one instruction template per family and takes queries from LFQA. We therefore vary all three at evaluation time: Appendix B.1 reports how detection varies across evaluation keys, drawn from seeds disjoint from training; Appendix B.2 replaces the instruction template; and Appendix B.3 replaces the query domain. All results here use our trained Qwen3-14B; we refer to it as Ours and to the untrained Qwen3-14B as Base.

B.1 Detectability across watermark keys

κ and ψ are two parameters used by the RL reward, since that reward is $V_\psi(\,\cdot\,;\kappa)$. Our evaluation uses keys generated from seeds that are disjoint from those used during training, ensuring that the reported performance does not reflect memorization of specific keys. Table 7 reports the distribution of the verifier z-statistic over the evaluation set. The median sits between 4.29 and 4.98 across the three families.

Table 7: Distribution of the verifier z-statistic across watermark keys, for our trained Qwen3-14B on the evaluation queries. Each response uses a key drawn from a seed disjoint from those used in training.

Family	Mean z	SD	p_5	Median z	p_{95}
TSP	4.22	1.55	1.84	4.29	6.53
WIP	4.27	1.47	1.75	4.32	6.49
SA	4.95	1.20	3.09	4.98	6.89

B.2 Held-out instruction templates

We rewrote each family's instruction into a template distinct from the one used during training, keeping the task identical. Table 8 reports the result. Under the reconstructed template, our method retains high detectability across all three families, and shows stable improvement over the base model.

Table 8: ICW IF performance under a held-out instruction template, on the LFQA evaluation queries. Ours is our trained Qwen3-14B, Base is the untrained Qwen3-14B, and Δ is Ours $-$ Base.

Template		TSP AUC ↑	WIP AUC ↑	SA AUC ↑
Original	Base	0.564	0.583	0.830
	Ours	0.993	0.999	0.999
	Δ	+0.429	+0.416	+0.169
Reconstructed	Base	0.463	0.588	0.788
	Ours	0.997	0.980	0.977
	Δ	+0.534	+0.392	+0.189

B.3 Unseen query domains

We evaluate on two query sets that are different from training: BioASQ [77] for the biomedical domain and FiQA [78] for the finance domain. Table 9 reports the result. The gap between Ours and Base is preserved outside the LFQA distribution the model was trained on.

Table 9: ICW IF performance on two unseen query domains, under the original instruction template.

Query set	**Model**	TSP AUC ↑	WIP AUC ↑	SA AUC ↑
LFQA (training domain)	Base	0.564	0.583	0.830
	Ours	0.993	0.999	0.999
BioASQ (biomedical)	Base	0.585	0.502	0.817
	Ours	0.941	0.996	0.993
FiQA (finance)	Base	0.584	0.510	0.842
	Ours	0.949	0.999	0.996

C Robustness under Stronger and Adaptive Attacks

Figure 3 evaluates paraphrase, word replacement, and word deletion. This section adds attacks that edit the response more aggressively, and one attack trained specifically to remove watermarks. As in Appendix B, all results use our trained Qwen3-14B.

C.1 Multi-round paraphrasing and back-translation

We apply three further edits. *Multi-round paraphrasing* paraphrases the response for one, two, or three successive rounds. *Back-translation* translates the response into French and back into English. *Summarize-then-expand* summarizes the response and then expands the summary back to full length. Table 10 reports detection AUC.

After three rounds of paraphrasing the watermark stays detectable, at AUC 0.853, 0.872, and 0.796 for TSP, WIP, and SA. Back-translation costs at most 0.065 AUC on any family. Summarize-then-expand hurts SA the most, cutting it to 0.597; on TSP and WIP it stays at 0.757 and 0.887.

Table 10: Detection AUC of our trained Qwen3-14B under three stronger edits.

Family	No attack	Paraphrase $\times 1$	Paraphrase $\times 2$	Paraphrase $\times 3$	Back-translation	Summarize-expand
TSP	0.993	0.890	0.911	0.853	0.932	0.757
WIP	0.999	0.897	0.962	0.872	0.977	0.887
SA	0.999	0.873	0.850	0.796	0.979	0.597

C.2 Adaptive attack

We evaluate against RLCracker [42], an RL-trained removal attack effective against logit-based and sampling-based watermarks.

The attack needs watermarked text to train on. That requirement is harder to satisfy for ICWs than for conventional watermarks: a logit-based or sampling-based watermark is embedded by the model

provider, so an attacker can collect many watermarked outputs from one system, whereas an ICW is introduced by a third-party user. We nonetheless grant the attacker 100 ICW-bearing samples.

We measure two quantities. The evasion success rate (ESR) is the fraction of attacked responses that fall below the detection threshold set at 1% FPR. Semantic utility is measured by P-SP [79], the cosine similarity between averaged subword embeddings trained on paraphrase data, and by an LLM judge that scores on a 1–5 scale how much of the reference answer's substance survives (Coverage) and whether what survives is unchanged in meaning (Fidelity).

Table 11 reports both together. Under the loosest constraint reported (Coverage ≥ 2), RLCracker evades detection on more than 97% of responses, but it pays for this with semantic damage: P-SP falls from about 0.96 to about 0.78 and Coverage from about 4.95 to about 3.5. Once the attacked response is required to retain content, its advantage narrows. At Coverage ≥ 4 its ESR drops to 0.261/0.373/0.378, and at Coverage ≥ 4.5 to 0.011/0.370/0.073, at which point it no longer exceeds plain LLM paraphrasing on any family. The same pattern holds under a P-SP constraint. These findings are consistent with Zhang et al. [80], which establishes that a strong (robust) watermark is theoretically impossible when attackers are allowed to alter semantic content without restriction.

Table 11: RLCracker [42] against LLM paraphrasing on our trained Qwen3-14B. Semantic utility is averaged over attacked responses. ESR is the fraction of responses that evade the 1% FPR detection threshold, restricted to responses meeting the stated utility constraint. Lower ESR is better for the defender. Coverage 5: every substantive claim, entity, number, and qualification is present; 4: one or two minor details missing; 3: several substantive points missing but the main answer is still delivered; 2: most substance gone.

	Attack	**Semantic utility**			**ESR ↓ at Coverage**				**ESR ↓ at P-SP**			
		P-SP ↑	Cov. ↑	Fid. ↑	≥ 2	≥ 3	≥ 4	≥ 4.5	≥ 0.6	≥ 0.7	≥ 0.8	≥ 0.9
TSP	LLM paraphrase	0.960	4.945	4.992	0.549	0.549	0.546	0.541	0.549	0.546	0.546	0.532
	RLCracker	0.785	3.412	4.239	0.983	0.888	0.261	0.011	0.952	0.860	0.473	0.022
WIP	LLM paraphrase	0.965	4.957	4.996	0.370	0.370	0.370	0.370	0.370	0.370	0.370	0.364
	RLCracker	0.801	3.556	4.338	0.978	0.919	0.373	0.370	0.969	0.880	0.560	0.039
SA	LLM paraphrase	0.967	4.944	4.976	0.392	0.392	0.392	0.387	0.389	0.389	0.389	0.384
	RLCracker	0.762	3.605	4.235	0.986	0.941	0.378	0.073	0.969	0.843	0.300	0.003

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please see Abstract and Introduction Sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see Experiments Section.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A].

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all the information needed to reproduce the main experimental results in the Experiments Section and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will provide open access to the data and code, along with sufficient instructions to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all the training and test details in the Experiments Section and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Detection results include z-statistic.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please see the Experiments Section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that our research conforms to it in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses both potential positive and negative societal impacts of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

Justification: The paper does not include the release of data or models that have a high risk for misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly credit the creators or original owners of assets used in the paper.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not introduce new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not include crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.