WATERMOE: EXPERT-ROUTING-BASED WATERMARKING FOR HIGH FIDELITY AND EFFICIENCY

Anonymous authors
Paper under double-blind review

ABSTRACT

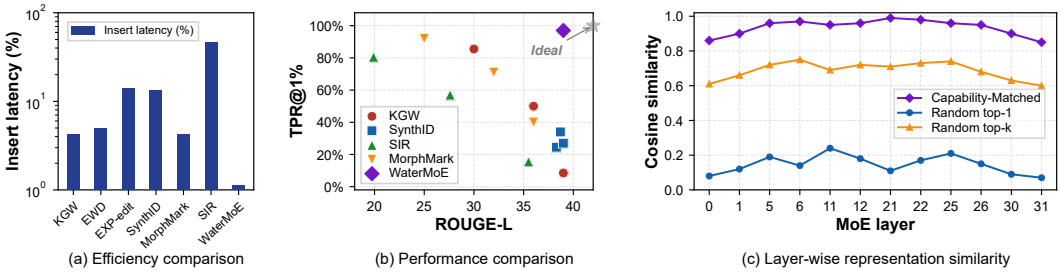
Large language models (LLMs) have achieved remarkable success but raise growing concerns about content provenance and misuse, motivating the need for reliable watermarking techniques. However, these techniques have rarely been adopted in practice mainly for two reasons: i) severely degraded model performance, and ii) additional inference overhead. To confirm the problem, we construct a comprehensive benchmark spanning different generation tasks to systematically evaluate 9 representative watermarking methods. We found almost all existing methods are designed for text fluency, but not for restricted and complicated tasks, and their overhead prevents them from deployment in latency-critical systems. To address i) and ii), we propose an LLM watermarking scheme *WaterMoE* for the growingly popular Mixture-of-Experts (MoE) LLMs. WaterMoE embeds watermarking signals through controlled perturbation into the expert selection at each router, which accumulates to token selection shift at the final output. In contrast to watermarking as a post-processing token-sampling approach, WaterMoE embeds watermark within the inference loop incurring negligible quality degradation and computational overhead. Extensive experiments demonstrate that our method achieves a fidelity performance close to the unwatermarked and consistently outperforms state-of-the-art watermarking methods on the benchmark, with up to $4\times$ speedup, incurring merely 1% additional inference latency compared to native generation. The results demonstrate the capability of WaterMoE to be deployed in real-world tasks.

1 INTRODUCTION

In recent years, Large Language Models (LLMs) have achieved remarkable success across a wide range of applications, including medical consultation (Wang et al., 2025a; Yuan et al., 2024), information retrieval (Labruna et al., 2025; Zhu et al., 2025; Ghodratnama & Zakershahrak, 2023), code generation (Li et al., 2022; Wang & Chen, 2023; Yang et al., 2025b; Jiang et al., 2026) and autonomous agents (Huang et al., 2024; Liu et al., 2023c; Chu et al., 2025; Li et al., 2026). However, alongside their growing capabilities, the potential misuse of LLMs for generating misinformation, deceptive content, and malicious text has raised serious concerns about the integrity of the ecosystem (Liang et al., 2026; Huang et al., 2026). This has led to an increasing demand for reliable techniques to distinguish machine-generated text from human-written content, among which LLM watermarking has emerged as a promising direction.

Existing LLM watermarking techniques largely follow a common token-sampling-based paradigm where the model's next-token distribution is subtly manipulated to encode identifiable patterns. For example, KGW (Kirchenbauer et al., 2023) partitions the vocabulary into disjoint subsets by random hashing and increases the logits of tokens in a designated subset; SynthID-text (Dathathri et al., 2024) adopts a tournament sampling strategy to inject watermark signals without introducing distributional distortion. While being effective for post-hoc detection, these methods inherently introduce overhead in the critical decoding path, as illustrated in Fig. 1(a). Modern LLM serving systems are optimized for high-throughput, low-latency generation, where even lightweight per-token operations—such as hash-based partitioning or probability reweighting—can trigger frequent CPU–GPU synchronization and data transfer, accumulating with sequence length and introducing substantial latency and system overhead. At the same time, these approaches rely on explicitly altering token distributions, which inevitably interferes with the model's natural generation behavior. This interference is particularly

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Efficiency comparison (b) Performance comparison (c) Layer-wise representation similarity

Figure 1: (a) Efficiency comparison on Mixtral-8×7B between WaterMoE and baseline watermarking algorithms. We report per-token insertion latency overhead (%) compared to the native inference. (b) Performance comparison among different watermarking methods. The x-axis denotes text generation quality measured by ROUGE-L, while the y-axis represents watermark detectability measured by TPR@1%. The trade-offs are obtained by varying watermark strength. (c) Layer-wise representation similarity between the capability-matched experts.

detrimental in delicate real-world tasks such as summarization, instruction following, code generation, etc. In those restricted settings, valid outputs are sparse in the token space and thus token probability shifting caused by watermarking tends to drive the generated text to deviate from correct or optimal outputs. As shown in Fig. 1(b), for a summarization task, existing watermarking methods exhibit a clear 'impossible' region at the top-right corner, corresponding to simultaneously high detection rates and high fidelity. Even unbiased methods can suffer from weak watermark signals due to the constraint of preserving unbiasedness.

Therefore, a fundamental question arises: *can we design a watermarking mechanism that achieves high efficiency, fidelity and detection rate at the same time?*

To answer the question, we propose *WaterMoE*, an expert-routing-based watermarking approach for Mixture-of-Experts (MoE) LLMs. MoE models have been increasingly adopted in modern high-performance language models (Cai et al., 2024; Zhu et al., 2024), where inference is carried out through sparse expert activation and routing decisions. We are motivated by the inherent redundancy in MoE models that allows different experts to be chosen so that each input token takes a flexible combination of model weights, leading to a varied output distribution (Rajbhandari et al., 2022; Skliar et al., 2024). Hence without altering the model's generation strategy, the expert selection brings in 'controlled randomness' in the output distribution: at each routing gate, top-k experts are potentially chosen, allowing stable and detectable internal patterns to be expressed through subtle biases at the routing level, while maintaining generation capability. Building on this observation, we integrate watermark signals directly into the MoE routing logic, making the watermark an intrinsic component of the model's inference process.

Our contributions are summarized as follows. 1) **Real-world evaluation of watermarking**: we systematically reveal the limitations of existing watermarking methods under realistic workloads, and construct a comprehensive benchmark spanning summarization, question answering, code generation, reasoning tasks and instruction following to evaluate the watermark detection rate, efficiency, and task performance. 2) **Structural watermarking via MoE routing**: we propose **WaterMoE**, a watermarking approach that embeds signals into expert routing. Extensive experiments show that WaterMoE improves detection under strict false positive constraints by 12.1% at TPR@1% over the strongest baseline, without degradation from unwatermarked task performance. Our method achieves up to 4× speedup in watermark embedding compared to state-of-the-art methods, incurring merely 1% latency overhead compared to native generation, demonstrating strong practicality.

2 Preliminaries and Related Work

Mixture-of-Experts. MoE architectures have recently become a dominant scaling paradigm for large language models, where sparse expert activation enables substantial parameter scaling while maintaining bounded inference cost. In a typical MoE layer, the standard feed-forward network is replaced by multiple parallel expert networks, and a router dynamically selects a small subset of experts for each token according to their routing scores.

Formally, given the hidden representation $\mathbf{h}_\ell$ at layer ℓ, the router produces routing scores $\mathbf{s}_\ell \in \mathbb{R}^K$, where K denotes the number of experts in the layer. The top-k experts $\mathcal{E}_\ell(\mathbf{h}_\ell)$ are selected according to these scores, which are then normalized via softmax over the selected experts to compute the output of the MoE layer:

$$\mathbf{s}_\ell = \mathrm{Router}_\ell(\mathbf{h}_\ell), \;\; p_\ell(i \mid \mathbf{h}_\ell) = \frac{\exp(s_{\ell,i})}{\sum_{j \in \mathcal{E}_\ell(\mathbf{h}_\ell)} \exp(s_{\ell,j})}, \;\; \mathrm{MoE}_\ell(\mathbf{h}_\ell) = \sum_{i \in \mathcal{E}_\ell(\mathbf{h}_\ell)} p_\ell(i \mid \mathbf{h}_\ell) f_i(\mathbf{h}_\ell),$$

where $f_i(\cdot)$ denotes the i-th expert network and $p_\ell(i \mid \mathbf{h}_\ell)$ represents its normalized routing weight.

Due to the over-parameterized nature of MoE architectures, different experts may learn correlated representations, leading to substantial function overlaps. Recent studies have verified such redundancy from multiple perspectives: multiple experts can be merged or compressed with minimal performance loss (Li et al., 2024); only a small subset of experts are essential for maintaining model capability (Zeng et al., 2024). Thus functionally similar experts can be dynamically replaced during inference with limited impact on generation quality (Skliar et al., 2024).

As shown in Figure 1(c), replacing the routed expert with one having a closely matched routing score (difference < 0.2) preserves high layer-wise similarity, whereas random replacement causes substantial shifts. This suggests that experts with similar routing scores are functionally similar, supporting that the routing path could be moderately altered with negligible fidelity loss.

LLM watermarking. Existing LLM watermarking methods mainly embed watermark signals by shifting the token probability distribution during autoregressive decoding. KGW (Kirchenbauer et al., 2023) partitions the vocabulary via N-gram hashing and biases generation toward a selected subset of tokens. EWD (Lu et al., 2024) improves detection performance through entropy-weighted detection. SynthID (Dathathri et al., 2024) proposes a non-distortionary watermarking method based on tournament sampling, while MorphMark (Wang et al., 2025b) introduces an adaptive watermarking strategy. However, most existing approaches rely on post-hoc perturbations of token logits, which directly intervene in the autoregressive generation process and introduce two major challenges in practical deployment: text quality degradation and additional inference overhead. By modifying the token sampling distribution, token-level watermarking methods may deviate from the native LLM output distribution, conflicting with real-world requirements such as factual correctness, strict formatting, or even program execution. As shown in Fig. 1(b), although unbiased watermarking avoids such distributional distortion, its restricted watermark signal can become insufficient on difficult or highly constrained tasks. Existing approaches therefore often face a trade-off between watermark detectability and generation quality. Moreover, these token-level perturbations are performed in the decoding step lying on the critical generation path. Even lightweight operations, such as token hashing, probability reweighting, or constrained sampling, may introduce unnecessary CPU–GPU synchronization and data transfer, adding non-negligible latency in large-scale serving systems. Such interaction also disrupts the highly optimized GPU-resident decoding pipeline, limiting the practical efficiency of existing watermarking methods.

In contrast to token-level watermarking methods, WaterMoE embeds watermark signals into the expert routing mechanism by introducing a lightweight perturbation to expert selection. Hence WaterMoE avoids restricting the token sampling space and largely preserves the original output distribution. Furthermore, the routing bias is seamlessly integrated into the existing MoE inference pipeline, resulting in negligible insertion overhead while maintaining high-quality generation.

3 WaterMoE: Watermarking via Expert Routing

Our design is motivated by the flexibility in MoE models' routing mechanism: experts with similar routing scores often possess comparable functional capabilities. Hence a small routing bias can shift expert selection which gradually amplifies through layers leading to subtle, but traceable variations in the final output distribution. The proposed framework is shown in Fig. 2.

3.1 Watermark Embedding through Routing Bias

To enable efficient watermark embedding at scale, we introduce a *Green Expert Map* $\mathcal{M}_\ell$ that is randomly generated offline, and deterministically specifies the green experts for each MoE layer. The

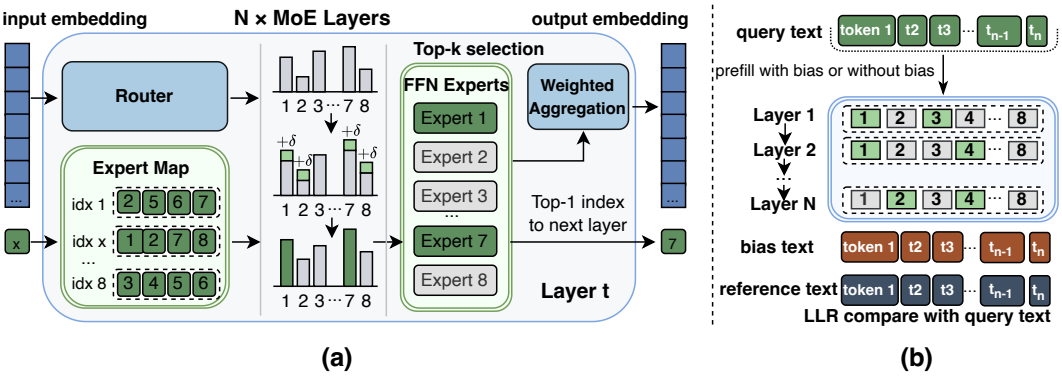


Figure 2: Overview of the WaterMoE framework. (a) Watermark embedding via routing bias. A lightweight additive bias is applied to the routing scores of green experts, subtly steering top-k expert selection. (b) Watermark detection via likelihood test. The presence of watermark signals is identified by a token-level likelihood test between biased and reference models, where consistent shifts reveal statistically detectable watermark patterns.

green experts refer to those experts perturbed by the injected biases. Let $e_{\ell-1} \in \{1, \ldots, K\}$ denote the top-1 selected expert index from layer $\ell - 1$. The green expert set at layer ℓ is then determined as:

$$\mathcal{G}_\ell = \mathcal{M}_\ell(e_{\ell-1}), \quad \text{s.t.} \quad \mathcal{G}_\ell \subseteq \{1, \ldots, K\}, \quad |\mathcal{G}_\ell| = K/2. \tag{1}$$

For the first MoE layer, we skip this procedure due to the absence of a preceding expert. We utilize a binary indicator vector $\mathbf{g}_\ell \in \{0, 1\}^K$ to denote the selected green set:

$$g_{\ell,i} = \begin{cases} 1, & i \in \mathcal{G}_\ell, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The routing bias for layer ℓ is defined by $\delta\, \mathbf{g}_\ell$, where $\delta > 0$ signaling the watermark strength. Hence the biased routing logits are:

$$\tilde{\mathbf{s}}_\ell = \mathbf{s}_\ell + \delta\, \mathbf{g}_\ell. \tag{3}$$

Since δ is small, routing changes occur only between experts with close routing scores, which exhibit similar capabilities as validated in Figure 1(c), thereby preserving model fidelity. In practice, the green expert map is precomputed offline and stored as a tensor aligned with MoE layers. During inference, the routing bias is applied via a single element-wise addition (equation 3), which is fully GPU-parallelizable and introduces negligible latency overhead.

During autoregressive generation, the routing bias is consistently applied to every token across all MoE layers. As a result, the expert selection process is subtly biased toward green experts. Although the token logits are not explicitly modified, the perturbed routing alters the internal representations that ultimately induce subtle but consistent shifts in the output distribution, enabling statistical detection using standard hypothesis testing methods (e.g., Z-test).

3.2 Likelihood-Based Watermark Detection via Reference Calibration

For watermark detection, the key idea is to detect the distributional shift caused by routing biases. We use an unwatermarked model to estimate the intrinsic likelihood of each token as a reference. Then we calibrate token-level probabilities through disentangling routing-induced preference from the native generations, thereby isolating the watermark signals.

Given a sequence $\mathbf{x} = (x_1, \ldots, x_T)$, we evaluate it using a *single model* under two routing configurations: i) a biased mode with the watermarking enabled, and ii) a reference mode without routing bias. Importantly, both configurations share the same model parameters and differ only in the expert routing path, thus needing no two separate model deployments.

For each token, we compute the log likelihood difference as:

$$\Delta_t = \ell_t^{\text{bias}} - \ell_t^{\text{base}}, \quad \ell_t^{\text{bias}} = \log P_{\text{bias}}(x_t \mid x_{<t}), \quad \ell_t^{\text{base}} = \log P_{\text{base}}(x_t \mid x_{<t}), \tag{4}$$

where P_{bias} and P_{base} denote the conditional distributions of x_t under configuration i) and ii). A higher ℓ_t^{base} indicates that the token is highly consistent with the context. Δ_t measures the additional preference induced by the routing bias for the token. We aggregate token-level signals of which the log likelihood difference exceeds threshold τ:

$$S(\mathbf{x};\tau) = \frac{1}{T}\sum_{t=1}^{T} \Delta_t \mathbb{I}(\Delta_t > \tau)\,, \tag{5}$$

Following the standard calibration protocol adopted in MarkLLM (Pan et al., 2024) and prior watermarking studies, we estimate the expectation $\mu_0(\tau)$ and standard deviation $\sigma_0(\tau)$ of $S(\mathbf{x};\tau)$, and calibrate it by:

$$Z(\mathbf{x}) = \frac{S(\mathbf{x};\tau) - \mu_0(\tau)}{\sigma_0(\tau)}. \tag{6}$$

A sequence is flagged as watermarked if $Z(\mathbf{x})$ exceeds a threshold.

Design rationale WaterMoE is motivated by the asymmetric requirements of watermark embedding and verification. During generation, watermarking is applied to every decoding step and therefore must preserve both model fidelity and inference efficiency, particularly for tasks with strict output constraints. Verification, in contrast, is invoked only when provenance needs to be examined and is typically less latency-sensitive. Based on this asymmetry, WaterMoE places the primary design emphasis on lightweight and minimally invasive watermark embedding, while shifting additional computation and model evaluation to the verification stage. This trade-off is favorable in practical deployment, as occasional verification overhead is preferable to persistent quality degradation or latency overhead during generation.

WaterMoE also maintains high fidelity because the routing perturbation is deliberately small and mainly changes the selection among experts with similar routing scores and comparable functional outputs. Consequently, the induced changes in hidden representations and final token distributions remain limited, preserving the model's original generation behavior.

4 Experimental Results

To comprehensively evaluate the effectiveness of WaterMoE, we organize our experiments around the following research questions. *RQ1 Detection Performance:* Can WaterMoE achieve reliable and highly accurate detection across different models and tasks? *RQ2 Generation Quality:* How does WaterMoE affect the generation quality? *RQ3 Efficiency:* What is the computational overhead of watermark embedding and detection compared to existing methods? *RQ4 Robustness:* How robust is WaterMoE under realistic adversarial attacks? *RQ5 Routing Preservation:* Does WaterMoE Preserve the Original MoE Routing Behavior?

4.1 Setup

All experiments are implemented based on the MarkLLM (Pan et al., 2024) repository and conducted on NVIDIA H200 GPUs. We consider two representative MoE models: Mixtral-8×7B-Instruct-v0.1 (Jiang et al., 2024)and Qwen3-30B-A3B-Instruct (Yang et al., 2025a), both of which adopt sparse expert routing mechanisms. We compare WaterMoE against a set of representative watermarking baselines, as summarized in Table 2. For WaterMoE, we set as default $\delta = 0.2$ and $\tau = 0$ across all experiments. The complete experimental setup and hyperparameter configurations are provided in Appendix G.

We construct a benchmark suite spanning diverse LLM evaluation settings, organized into three tiers by task complexity, as illustrated in Table 1: *Low-*, *Moderate-*, and *High-Complexity* tasks. This tiered design enables a systematic assessment of watermark performance under varying difficulty and constraint levels. Notably, the low-complexity setting where $\delta = 0.8$ is included to ensure fair comparison with existing watermarking methods.

Table 1: Overview of evaluated datasets grouped by task complexity. 'Len.(In/Ans)' refers to the average length of input question and reference answer.

Complexity	Source Data	ID	Task	Metric	Language	Len. (In/Ans)
Low	*(Text Gen.)*					
	C4 (Raffel et al., 2020)	1-1	Language Modeling	PPL	English	22.14 / 151.01
	Booksum (Kryściński et al., 2022)	1-2	Long-Context Generation	PPL	English	45.53 / 541.54
Moderate	*(Sum. and QA)*					
	Multinews (Fabbri et al., 2019)	2-1	Multi-Doc Sum.	Rouge-L	English	1464.55 / 220.50
	ELI5 (Fan et al., 2019)	2-2	Long-Form QA	Rouge-L	English	38.98 / 261.31
High	*(Code Gen.)*					
	APPS (Hendrycks et al., 2021)	3-1	Competitive Coding	PASS@1	Python	313.52 / 108.80
	CodeContests (Li et al., 2022)	3-2	Competitive Coding	PASS@1	Python/C#/Java	383.45 / 176.75
	(CoT Reasoning)					
	GSM8K (Cobbe et al., 2021)	3-3	Math Word Problem	Accuracy	English	54.39 / 151.66
	MMLU (Hendrycks et al., 2020)	3-4	Multiple Choice QA	Accuracy	English	54.35 / 256.53
	(Alignment)					
	IFEval (Zhou et al., 2023)	3-5	Instruction Following	Rule-based Judge	English	37.42 / 1502.53
	WritingBench (Wu et al., 2025)	3-6	Creative Generation	LLM-based Score	English	482.04 / 2512.34

4.2 RQ1 & RQ2: Detection Performance and Generation Quality

Performance on High-Complexity Tasks. Table 2 reports watermarking performance across highly constrained domains which impose rigid structural and logical requirements, making watermark injection particularly challenging. Under a strict false positive rate (TPR@1%), baseline methods (e.g., EWD, SynthID) exhibit notable degradation in detectability, often falling below 70% on reasoning and coding benchmarks such as GSM8K, MMLU, and CodeContests. EXPEdit performs poorly in such settings, as it operates at the sample level, where the space of valid edits is severely constrained in high-complexity tasks, making watermark injection difficult. In contrast, WaterMoE achieves consistently strong detection performance, exceeding 90% on challenging datasets like APPS and CodeContests, and reaching 100.0% on alignment benchmarks (IFEval, WritingBench). Importantly, this high detectability does not come at the cost of generation quality: while conventional methods (e.g., KGW, SynthID) incur substantial accuracy drops (over 15 points) on strict tasks, WaterMoE maintains performance closely aligned with the unwatermarked model across all datasets, effectively mitigating the detectability–quality trade-off under rigid constraints. The results for moderate-complexity tasks are provided in Appendix E.

Performance on Low-Complexity Tasks. Table 3 presents results on low-complexity, open-ended generation tasks (C4 and Booksum). While such tasks are generally friendly to watermarking, the detection performance varies across model architectures. On Mixtral, most methods achieve near-perfect detection, but noticeable degradation occurs on Qwen under strict false positive constraints (TPR@1%), where baselines such as MorphMark, SynthID, and KGW exhibit significant drops. In contrast, WaterMoE maintains consistent 100.0% detection rate across all models and datasets, demonstrating strong model-independent stability. Importantly, linguistic fluency is not sacrificed: while some baselines (e.g., EXPEdit, EWD) substantially raise perplexity, WaterMoE achieves highly competitive PPLs among watermarked models, closely aligning with the unwatermarked baseline. These results indicate that WaterMoE effectively preserves fluency while ensuring reliable detectability in unconstrained generation settings.

4.3 RQ3: Efficiency Analysis

As shown in Fig. 3(a), WaterMoE exhibits a distinct efficiency profile between watermark insertion and detection. On the insertion side, WaterMoE introduces only **1.1%** additional latency, substantially lower than all baselines. This efficiency stems from our GPU-resident expert mapping design, which directly injects lightweight routing biases without token-level hashing, repeated candidate generation, or additional model inference, thereby introducing minimal disruption to the original generation pipeline.

Detection, in contrast, is inherently more heterogeneous across watermarking methods. Existing approaches may require extensive hashing operations Kirchenbauer et al. (2023); Dathathri et al. (2024); Wang et al. (2025b), additional model inference Lu et al. (2024); Hu et al. (2024); Kuditipudi et al. (2023), or separately trained detector models Liu et al. (2024). WaterMoE also incurs a relatively higher detection cost, since verification requires recovering the expert-routing evidence from the

Table 2: Watermarking performance on high-complexity tasks. WaterMoE maintains near-native generation quality on these highly rigid tasks while consistently achieving state-of-the-art detectability. Here, Accuracy and PASS@1 denote the percentage (%) of correct answers. AUC indicates the area under the ROC curve for detection performance.

Category: Code Generation (Model: Qwen)								
Algorithm	**Dataset: APPS**				**Dataset: CodeContests**			
	TPR@1%	**TPR@5%**	**AUC**	**PASS@1**	**TPR@1%**	**TPR@5%**	**AUC**	**PASS@1**
Unwatermarked	-	-	-	65.4	-	-	-	42.0
EWD (Lu et al., 2024)	73.0	92.0	97.3	27.8	40.0	54.0	92.3	42.5
EXPEdit (Kuditipudi et al., 2023)	6.0	6.0	41.3	2.0	2.0	2.0	39.9	6.1
KGW (Kirchenbauer et al., 2023)	63.0	72.0	94.0	42.2	38.0	42.0	79.4	34.4
MorphMark (Wang et al., 2025b)	71.0	82.0	96.3	42.5	46.0	50.0	83.3	42.2
SynthID (Dathathri et al., 2024)	80.0	87.0	92.0	41.1	64.0	78.0	89.2	40.5
Unbiased (Hu et al., 2023)	53.0	63.0	86.5	58.9	41.0	56.0	82.2	41.4
UPV (Liu et al., 2023a)	62.0	78.0	92.3	35.4	32.0	36.0	62.8	41.0
WaterMoE	**90.0**	**98.0**	**99.6**	**61.9**	**92.0**	**98.0**	**99.6**	**44.2**

Category: Strict Alignment (Model: Qwen)								
Algorithm	**Dataset: IFEval**				**Dataset: WritingBench**			
	TPR@1%	**TPR@5%**	**AUC**	**Accuracy**	**TPR@1%**	**TPR@5%**	**AUC**	**Score**
Unwatermarked	-	-	-	88.5	-	-	-	91.3
EWD	97.0	100.0	99.7	87.7	100.0	100.0	100.0	91.3
EXPEdit	61.0	61.0	83.5	60.1	30.0	30.0	69.4	57.0
KGW	60.0	80.0	93.0	87.7	88.0	98.0	98.7	90.8
MorphMark	50.0	77.0	91.6	88.3	51.0	88.0	96.3	91.1
SynthID	99.0	99.0	99.9	41.1	100.0	100.0	100.0	91.0
Unbiased	80.0	85.0	95.6	88.3	54.0	70.0	83.7	91.1
UPV	20.0	60.0	53.1	87.7	1.0	1.0	54.4	91.5
WaterMoE	**100.0**	**100.0**	**100.0**	**88.3**	**100.0**	**100.0**	**100.0**	**91.6**

Category: Chain-of-Thought Reasoning (Model: Qwen)								
Algorithm	**Dataset: GSM8K**				**Dataset: MMLU**			
	TPR@1%	**TPR@5%**	**AUC**	**Accuracy**	**TPR@1%**	**TPR@5%**	**AUC**	**Accuracy**
Unwatermarked	-	-	-	72.0	-	-	-	61.0
EWD	79.0	83.0	94.8	53.0	85.0	93.0	98.9	54.0
EXPEdit	11.0	22.0	67.2	33.0	3.0	14.0	42.2	59.0
KGW	69.0	78.0	95.6	54.0	67.0	81.0	96.1	49.0
MorphMark	44.0	49.0	85.6	68.0	43.0	47.0	84.9	62.0
SynthID	70.0	82.0	93.9	68.0	82.0	86.0	94.8	47.0
Unbiased	58.0	62.0	83.9	66.0	69.0	80.0	89.3	56.0
UPV	58.0	72.0	93.2	38.0	61.0	80.0	95.3	47.0
WaterMoE	**91.0**	**95.0**	**99.2**	**68.0**	**94.0**	**96.0**	**99.6**	**63.0**

generated sequence. Nevertheless, the detection process is performed non-autoregressively through two prefill passes.

Importantly, watermark insertion and detection have fundamentally different deployment frequencies. Insertion is executed for every generated token during online inference and therefore directly affects user-facing latency, whereas detection is typically invoked only after generation and much less frequently, such as during provenance or ownership verification. WaterMoE therefore deliberately shifts computational cost away from the latency-critical generation path toward the less frequent detection stage. This design achieves a favorable trade-off among **fidelity, insertion efficiency, and detection cost**: WaterMoE preserves the original generation behavior with negligible insertion overhead, while accepting a moderate verification cost that remains practical for offline or occasional detection.

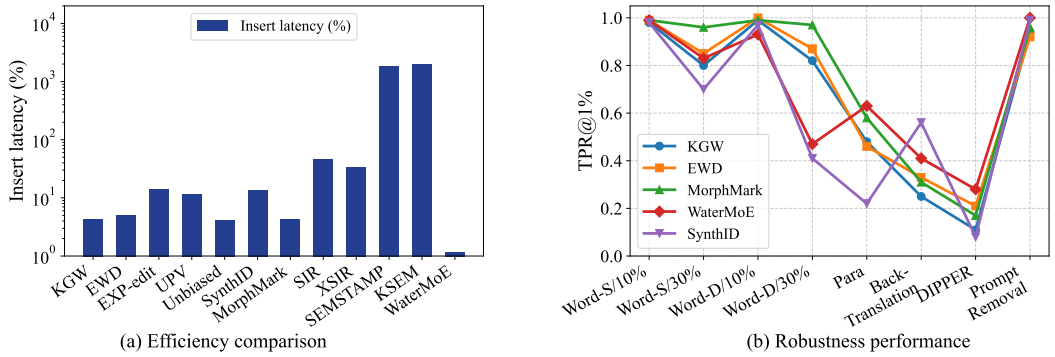
4.4 RQ4: Robustness under Adversarial Attacks

As shown in Fig. 3(b), WaterMoE demonstrates strong robustness across various attacks. Under word substitution attacks, WaterMoE achieves competitive performance compared with other methods. We further conduct an additional API-assisted substitution attack with approximately 30% token replacement, where WaterMoE still maintains a 98% TPR@1% detection rate. In contrast, aggressive word deletion causes a larger degradation, as it disrupts semantic and contextual coherence, weakening the watermark signal.

Table 3: Watermarking performance on low-complexity tasks. Detectability metrics (TPR and AUC) are reported as percentages (higher is better), while text quality is evaluated by Perplexity (PPL, lower is better). The best results are highlighted in bold.

Dataset: C4								
Algorithm	**Mixtral-8x7B**				**Qwen**			
	TPR@1%	**TPR@5%**	**AUC**	**PPL ↓**	**TPR@1%**	**TPR@5%**	**AUC**	**PPL ↓**
Unwatermarked	-	-	-	6.18	-	-	-	5.22
EWD (Lu et al., 2024)	100.0	100.0	100.0	8.16	94.0	100.0	99.7	6.76
EXPEdit (Kuditipudi et al., 2023)	93.5	96.0	98.7	8.49	98.0	98.5	99.4	8.79
KGW (Kirchenbauer et al., 2023)	100.0	100.0	100.0	7.80	82.0	88.0	98.2	6.61
MorphMark (Wang et al., 2025b)	100.0	100.0	100.0	6.85	87.0	95.5	98.4	6.51
SIR (Liu et al., 2023b)	100.0	100.0	100.0	7.65	-	-	-	-
SynthID (Dathathri et al., 2024)	100.0	100.0	100.0	**6.12**	86.5	90.0	96.8	6.44
Unbiased (Hu et al., 2023)	100.0	100.0	100.0	6.72	96.0	99.0	99.9	6.39
UPV (Liu et al., 2023a)	100.0	100.0	100.0	7.10	81.0	90.0	97.2	6.27
XSIR (He et al., 2024)	99.5	100.0	99.9	7.68	-	-	-	-
WaterMoE	**100.0**	**100.0**	**100.0**	6.63	**100.0**	**100.0**	**100.0**	**5.48**

Dataset: Booksum								
Algorithm	**Mixtral-8x7B**				**Qwen**			
	TPR@1%	**TPR@5%**	**AUC**	**PPL ↓**	**TPR@1%**	**TPR@5%**	**AUC**	**PPL ↓**
Unwatermarked	-	-	-	6.94	-	-	-	5.74
EWD	100.0	100.0	100.0	9.04	98.0	98.0	99.8	7.35
EXPEdit	99.0	99.0	99.9	10.50	100.0	100.0	99.9	11.31
KGW	100.0	100.0	100.0	8.45	84.5	92.0	98.0	7.42
MorphMark	100.0	100.0	100.0	7.86	89.0	95.5	98.6	7.30
SIR	100.0	100.0	100.0	8.08	-	-	-	-
SynthID	100.0	100.0	100.0	**6.80**	92.0	94.0	98.7	7.47
Unbiased	100.0	100.0	100.0	7.56	93.0	99.0	99.8	7.42
UPV	100.0	100.0	100.0	7.99	67.5	85.0	92.8	7.52
XSIR	100.0	100.0	100.0	8.59	-	-	-	-
WaterMoE	**100.0**	**100.0**	**100.0**	7.50	**100.0**	**100.0**	**100.0**	**6.37**



(a) Efficiency comparison (b) Robustness performance

Figure 3: (a) Efficiency comparison. We report per-token insertion latency overhead (%) on Mixtral-8×7B (C4) across different watermarking methods. (b) Robustness performance of different methods under word substitution, word deletion, and paraphrasing attacks on Mixtral-8×7B over the C4 dataset.

For semantic-preserving attacks, including DIPPER-based paraphrasing, back-translation, and GLM-4.5 rewriting, WaterMoE consistently ranks among the best-performing methods and achieves 63% TPR@1% under strong paraphrasing. Prompt removal attacks further show that WaterMoE does not rely on input prompts during detection. Since WaterMoE relies on model-side signals, its robustness is influenced by the quality of attacked texts; however, highly destructive attacks that harm text usability are less practical in real-world scenarios.

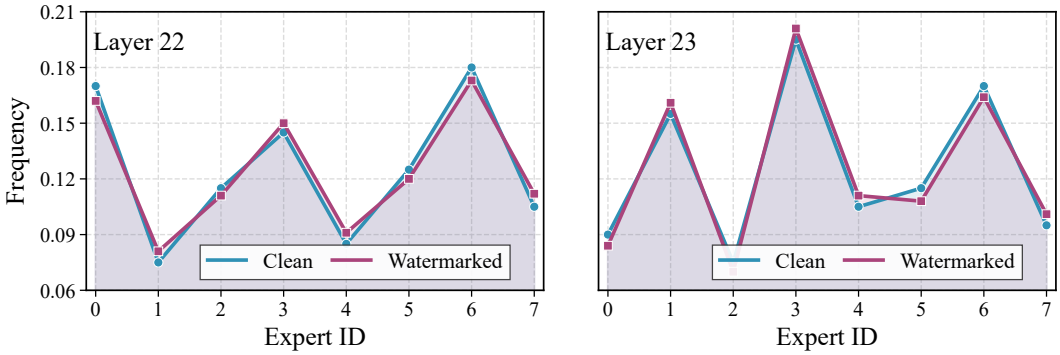


Figure 4: Expert activation frequencies of the clean and watermarked models in two representative watermarked MoE layers. The representative results show nearly identical routing distributions with only minor changes in expert selection.

4.5 RQ5: Routing Consistency

To investigate the impact of WaterMoE on the original routing behavior, we compare the expert activation distributions between the clean and watermarked models on 100 clean inputs. We report Layers 22 and 23 as two representative examples, with each distribution normalized over the eight experts.

As shown in Figure 4, the clean and watermarked models exhibit highly consistent expert activation patterns. The routing distributions achieve cosine similarities of 0.9989 and 0.9990 in Layers 22 and 23, respectively. The mean absolute frequency change is only 0.006 for both layers, with maximum deviations of 0.008 and 0.007. This shows that WaterMoE does not substantially redistribute tokens across experts, but only introduces small local changes around the original routing decisions. Therefore, the overall expert utilization pattern of the original MoE model is largely preserved, limiting the impact of watermark injection on the model's native computation.

We further examine whether these local routing changes affect internal representations. WaterMoE uses a small routing bias, so routing changes mainly occur between experts with close routing scores rather than arbitrary experts. As shown in Fig. 1(c), such capability-matched expert replacements maintain high representation similarity, with cosine similarities around 0.95 or higher across different MoE layers, whereas random replacements cause substantially larger deviations. This indicates that even when WaterMoE changes an expert selection, the alternative expert tends to produce a similar representation. Consequently, WaterMoE preserves both the global routing distribution and the intermediate representations, explaining its negligible impact on downstream model performance.

5 Limitations

WaterMoE currently requires access to the corresponding watermarked model and routing configuration for detection, making it primarily suitable for provider-side verification. This dependence also limits fully black-box third-party verification. Developing a model-independent detection mechanism is therefore an important direction for future work.

6 Conclusion

Watermarking for LLMs remains challenging due to performance degradation and inference overhead, particularly in complex and constrained tasks. In this work, we systematically identify these limitations through a comprehensive benchmark and propose WaterMoE, a watermarking scheme tailored for MoE-based LLMs. By integrating watermark signals into the routing process, WaterMoE achieves strong detectability while preserving model quality and efficiency. Extensive experiments demonstrate its consistent advantages over existing methods, highlighting its practicality for real-world deployment.

AI USE STATEMENT

In this work, generative AI tools were used to assist with language polishing, manuscript drafting and revision, formatting, and literature search. All AI-assisted outputs, including retrieved information and suggested text, were critically reviewed, verified, and revised by the authors. The authors take full responsibility for the accuracy, integrity, and final content of the paper.

ETHICS STATEMENT

This work studies watermarking techniques for identifying AI-generated text. The proposed method is intended to support content provenance and responsible AI deployment. While watermarking may potentially be used for monitoring or attribution purposes, our method does not require access to private user data or personally identifiable information. All experiments are conducted on publicly available datasets and language models. We do not involve human subjects or collect sensitive personal information.

REPRODUCIBILITY STATEMENT

We provide implementation details, hyperparameter settings, and additional experimental results in the appendix. The source code will be made publicly available upon acceptance.

REFERENCES

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts. *Authorea Preprints*, 2024.

Zhendong Chu, Shen Wang, Jian Xie, Tinghui Zhu, Yibo Yan, Jinheng Ye, Aoxiao Zhong, Xuming Hu, Jing Liang, Philip S Yu, et al. Llm agents for education: Advances and applications. *arXiv preprint arXiv:2503.11733*, 2, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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.

Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1074–1084, 2019.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. Eli5: Long form question answering. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 3558–3567, 2019.

Samira Ghodratnama and Mehrdad Zakershahrak. Adapting llms for efficient, personalized information retrieval: Methods and implications. In *International conference on service-oriented computing*, pp. 17–26. Springer, 2023.

Thibaud Gloaguen, Nikola Jovanović, Robin Staab, and Martin Vechev. Black-box detection of language model watermarks. *arXiv preprint arXiv:2405.20777*, 2024.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4115–4129, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021.

Abe 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. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4067–4082, 2024a.

Abe Hou, Jingyu Zhang, Yichen Wang, Daniel Khashabi, and Tianxing He. k-semstamp: A clustering-based semantic watermark for detection of machine-generated text. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1706–1715, 2024b.

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

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. Unbiased watermark for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45408–45436, 2024.

Hanbo Huang, Xuan Gong, Yiran Zhang, Hao Zheng, and Shiyu Liang. Rlspoofer: A lightweight evaluator for llm watermark spoofing resilience. *arXiv preprint arXiv:2604.11546*, 2026.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, 35(2): 1–72, 2026.

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*, pp. 17061–17084. PMLR, 2023.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization. In *Findings of the association for computational linguistics: EMNLP 2022*, pp. 6536–6558, 2022.

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

Tiziano Labruna, Jon Ander Campos, and Gorka Azkune. When to retrieve: Teaching llms to utilize information retrieval effectively. In *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing-Natural Language Processing in the Generative AI Era*, pp. 623–632, 2025.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy. In *International Conference on Learning Representations*, volume 2024, pp. 14234–14256, 2024.

Yu Li, Lehui Li, Zhihao Wu, Qingmin Liao, Jianye Hao, Kun Shao, and Fengli Xu. Agentswift: Efficient llm agent design via value-guided hierarchical search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31843–31851, 2026.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Yuqing Liang, Jiancheng Xiao, Wensheng Gan, and Philip S Yu. Watermarking techniques for large language models: A survey. *Artificial Intelligence Review*, 59(2):74, 2026.

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

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

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip Yu. An unforgeable publicly verifiable watermark for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 8291–8307, 2024.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*, 2023c.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. An entropy-based text watermarking detection method. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11724–11735, 2024.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, et al. Markllm: An open-source toolkit for llm watermarking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 61–71, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.

Andrii Skliar, Ties van Rozendaal, Romain Lepert, Todor Boinovski, Mart Van Baalen, Markus Nagel, Paul Whatmough, and Babak Ehteshami Bejnordi. Mixture of cache-conditional experts for efficient mobile device inference. *arXiv preprint arXiv:2412.00099*, 2024.

Jianxun Wang and Yixiang Chen. A review on code generation with llms: Application and evaluation. In *2023 IEEE International Conference on Medical Artificial Intelligence (MedAI)*, pp. 284–289. IEEE, 2023.

Wenxuan Wang, Zizhan Ma, Zheng Wang, Chenghan Wu, Jiaming Ji, Wenting Chen, Xiang Li, and Yixuan Yuan. A survey of llm-based agents in medicine: How far are we from baymax? *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10345–10359, 2025a.

Zongqi Wang, Tianle Gu, Baoyuan Wu, and Yujiu Yang. Morphmark: Flexible adaptive watermarking for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4842–4860, 2025b.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, et al. Writingbench: A comprehensive benchmark for generative writing. *arXiv preprint arXiv:2503.05244*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Zezhou Yang, Sirong Chen, Cuiyun Gao, Zhenhao Li, Xing Hu, Kui Liu, and Xin Xia. An empirical study of retrieval-augmented code generation: Challenges and opportunities. *ACM Transactions on Software Engineering and Methodology*, 34(7):1–28, 2025b.

Dong Yuan, Eti Rastogi, Gautam Naik, Sree Prasanna Rajagopal, Sagar Goyal, Fen Zhao, Bharath Chintagunta, and Jeffrey Ward. A continued pretrained llm approach for automatic medical note generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 565–571, 2024.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. Adamoe: Token-adaptive routing with null experts for mixture-of-experts language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6223–6235, 2024.

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.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. Llama-moe: Building mixture-of-experts from llama with continual pre-training. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 15913–15923, 2024.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. Large language models for information retrieval: A survey. *ACM Transactions on Information Systems*, 44(1):1–54, 2025.

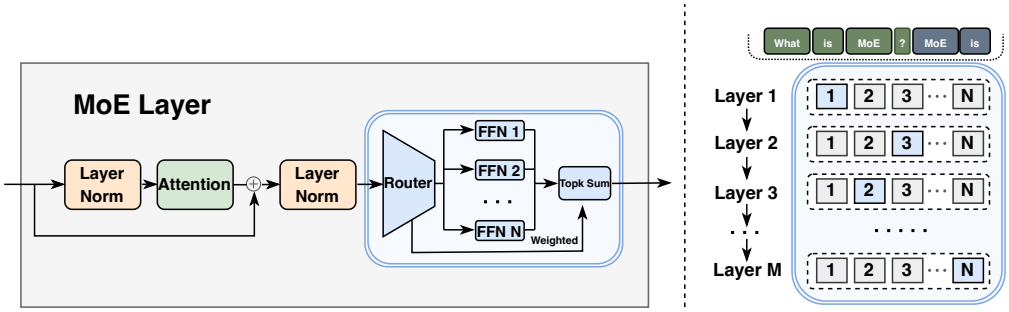


Figure 5: Detailed Framework of MoE.

A Related Work

Semantic-level watermarking. Recent approaches explore embedding watermark signals at the semantic level. SIR Liu et al. (2023b) employs learned encoders to map semantically similar inputs to similar watermark patterns, improving robustness to paraphrasing. However, this design introduces additional system complexity and incurs non-trivial latency due to sentence-level encoding. Moreover, by conditioning watermark signals on semantic representations, such methods implicitly constrain the feasible token space during generation, leading to similar limitations as token-level approaches in restricting the solution space.

Subsequent methods such as SEMSTAMP Hou et al. (2024a) and KSEMSTAMP Hou et al. (2024b) adopt a different strategy by discretizing semantic representations via locality-sensitive hashing or clustering (e.g., k-means), and enforcing constraints on the semantic space of the next sentence. In practice, these methods rely on repeated resampling until a candidate satisfying the watermark constraint is found. While effective in controlled settings, this mechanism introduces substantial computational overhead, often requiring tens of resampling iterations per sentence, which leads to significant latency.

More critically, the imposed semantic constraints further shrink the already limited feasible solution space in many real-world tasks, such as summarization and question answering, where correctness and coherence impose strict requirements. Empirical evidence shows that even with up to 50 resampling attempts, SEMSTAMP frequently fails to embed watermarks successfully on datasets such as MultiNews and ELI5 benchmarks. These limitations suggest that existing semantic-level watermarking methods are not well-suited for practical deployment in constraint-heavy generation scenarios.

B Detailed Background on Mixture-of-Experts (MoE)

B.1 Overview

Mixture-of-Experts (MoE) is a sparsely-activated neural architecture designed to scale model capacity without incurring proportional computational cost. Instead of using a single dense feed-forward network (FFN) in each Transformer layer, MoE replaces it with a set of K parallel expert networks and a routing mechanism that dynamically selects a subset of experts for each token.

As illustrated in Fig. 5, an MoE layer consists of three key components:

- **Router (Gating Network):** computes routing scores for each expert.
- **Experts:** a set of independent feed-forward networks.
- **Aggregation:** combines outputs from selected experts via weighted summation.

This sparse activation enables MoE models to achieve extremely large parameter counts while keeping the per-token computation comparable to standard dense models.

B.2 FORMULATION

Given the hidden representation $\mathbf{h}_\ell \in \mathbb{R}^d$ at layer ℓ, the router produces a score vector:

$$\mathbf{s}_\ell = \mathrm{Router}_\ell(\mathbf{h}_\ell), \quad \mathbf{s}_\ell \in \mathbb{R}^K, \tag{7}$$

where K is the number of experts.

Based on these scores, only the top-k experts are selected:

$$\mathcal{E}_\ell(\mathbf{h}_\ell) = \mathrm{TopK}(\mathbf{s}_\ell, k). \tag{8}$$

The routing weights are then normalized via softmax over the selected experts:

$$p_\ell(i \mid \mathbf{h}_\ell) = \frac{\exp(s_{\ell,i})}{\sum_{j \in \mathcal{E}_\ell(\mathbf{h}_\ell)} \exp(s_{\ell,j})}, \quad i \in \mathcal{E}_\ell(\mathbf{h}_\ell). \tag{9}$$

The final output of the MoE layer is:

$$\mathrm{MoE}_\ell(\mathbf{h}_\ell) = \sum_{i \in \mathcal{E}_\ell(\mathbf{h}_\ell)} p_\ell(i \mid \mathbf{h}_\ell) \cdot f_i(\mathbf{h}_\ell), \tag{10}$$

where $f_i(\cdot)$ denotes the i-th expert network.

B.3 INTEGRATION WITH TRANSFORMER

In modern large language models, MoE layers replace the standard FFN sub-layer in Transformer blocks. A typical Transformer layer with MoE can be written as:

$$\mathbf{h}'_\ell = \mathbf{h}_\ell + \mathrm{Attention}(\mathrm{LN}(\mathbf{h}_\ell)), \tag{11}$$

$$\mathbf{h}_{\ell+1} = \mathbf{h}'_\ell + \mathrm{MoE}_\ell(\mathrm{LN}(\mathbf{h}'_\ell)), \tag{12}$$

where $\mathrm{LN}(\cdot)$ denotes Layer Normalization.

B.4 ROUTING STRATEGIES

Different routing strategies are used in practice:

Top-k Routing The most common strategy selects the top-k experts per token. Typical choices include:

- $k = 1$ (Switch Transformer): maximum efficiency
- $k \geq 2$ (GShard, DeepSeek): better performance with modest overhead

B.5 COMPUTATIONAL EFFICIENCY

Although the total number of parameters grows linearly with K, the actual computation per token depends only on k:

$$\text{FLOPs} \approx O(k \cdot d^2), \tag{13}$$

which is comparable to a dense FFN when $k \ll K$.

Thus, MoE achieves:

- **High capacity** (large parameter count)
- **Low compute cost** (sparse activation)

C REPRESENTATION STABILITY UNDER ROUTING PERTURBATION

We analyze how the routing bias introduced by WaterMoE affects the output representation of an MoE layer. Unlike an expectation-based analysis over random Expert Maps, the following results hold for any fixed Green Expert Map and any fixed green-expert assignment.

C.1 Notations, Scope, and Assumptions

Consider an MoE layer with K experts. For a fixed hidden representation $\mathbf{h}$, let

$$s_i \in \mathbb{R}, \qquad i = 1, \dots, K, \tag{14}$$

denote the original routing logits. Let $g_i \in \{0, 1\}$ indicate whether expert i is a green expert. After injecting a routing bias of magnitude $\delta > 0$, the routing logit becomes

$$\widetilde{s}_i = s_i + \delta g_i. \tag{15}$$

Let

$$\mathcal{E} = \mathrm{TopK}(\mathbf{s}, k), \qquad \widetilde{\mathcal{E}} = \mathrm{TopK}(\widetilde{\mathbf{s}}, k) \tag{16}$$

denote the selected expert sets before and after bias injection. Following the actual top-k routing mechanism used by the evaluated MoE models, the selected routing weights are renormalized within the active expert set:

$$w_i = \frac{\exp(s_i)}{\sum_{j \in \mathcal{E}} \exp(s_j)}, \qquad i \in \mathcal{E}, \tag{17}$$

and

$$\widetilde{w}_i = \frac{\exp(\widetilde{s}_i)}{\sum_{j \in \widetilde{\mathcal{E}}} \exp(\widetilde{s}_j)}, \qquad i \in \widetilde{\mathcal{E}}. \tag{18}$$

The corresponding MoE outputs are

$$\mathbf{y} = \sum_{i \in \mathcal{E}} w_i f_i(\mathbf{h}), \qquad \widetilde{\mathbf{y}} = \sum_{i \in \widetilde{\mathcal{E}}} \widetilde{w}_i f_i(\mathbf{h}). \tag{19}$$

Scope of the analysis. A routing perturbation may in principle replace an unrestricted number of experts simultaneously resulting in difficulties in analysis. Hence we made a few mild assumptions according to the observations in practice.

Specifically, with a small routing bias, more than 95% of the observed routing events incur no shift in the active expert set or only a single expert replacement near the top-k selection threshold. This is consistent with the fact that the top-ranked experts are typically far away from the boundary, whereas routing changes are concentrated around experts whose routing scores are close to the k-th selection threshold. Accordingly, we analyze the following two dominant cases:

1. the selected expert set remains unchanged; and
2. exactly one expert near the top-k threshold is replaced.

Simultaneous replacement of multiple experts is possible and is not ruled out by our method. Such cases account for only a small fraction of the observed routing events and would require tracking interactions among multiple switched experts. We therefore leave them out for the analysis. Consequently, the bounds derived below should be interpreted as *conditional perturbation bounds for the empirically dominant routing regime*, rather than as worst-case guarantees over all possible routing configurations. We use the following assumptions.

Assumption 1 (Bounded expert output). There exists a finite constant $B > 0$ such that

$$\|f_i(\mathbf{h})\| \le B, \qquad \forall i \in \{1, \dots, K\}. \tag{20}$$

Assumption 2 (Empirically validated similarity of near-tied experts). Consider a routing event in the single-replacement regime, where expert a is selected before bias injection and is replaced by expert b afterward. Such a replacement can occur only when

$$0 \le s_a - s_b \le \delta. \tag{21}$$

For these actually switched, near-boundary expert pairs, we assume that their outputs for the current hidden representation satisfy

$$\|f_b(h) - f_a(h)\|_2 \le \epsilon_f B, \tag{22}$$

where B is the expert-output bound in Assumption 1 and ϵ_f is an empirically bounded discrepancy.

This assumption is supported by the experiment in Fig. 1(c) displaying similarity in the representation for capability-matched experts. For expert pairs that are actually exchanged under WaterMoE routing perturbations, we directly compare their output representations. The switched expert outputs exhibit cosine similarities above 0.95 in the dominant cases, while their normalized ℓ_2 discrepancies remain bounded as described in equation 22.

We emphasize that the routing score condition $0 \leq s_a - s_b \leq \delta$ does *not* by itself imply functional similarity. It only characterizes which experts can cross the top-k selection boundary under a bias of magnitude δ. The similarity assumption in equation 22 is an independently validated empirical property of the switched expert pairs.

C.2 Routing Weight Perturbation on a Fixed Expert Set

We first analyze how the routing weights change when the active expert set is fixed. For any fixed expert set $\mathcal{A}$, define

$$w_i^{\mathcal{A}} = \frac{\exp(s_i)}{\sum_{j\in\mathcal{A}} \exp(s_j)}, \qquad i \in \mathcal{A}, \tag{23}$$

and its biased counterpart

$$\overline{w}_i^{\mathcal{A}} = \frac{\exp(s_i + \delta g_i)}{\sum_{j\in\mathcal{A}} \exp(s_j + \delta g_j)}, \qquad i \in \mathcal{A}. \tag{24}$$

Lemma 1 (Fixed-set routing-weight perturbation). Let

$$P_G^{\mathcal{A}} = \sum_{\substack{i\in\mathcal{A}\\ g_i=1}} w_i^{\mathcal{A}} \tag{25}$$

denote the original normalized routing mass assigned to green experts within $\mathcal{A}$. Then

$$\left\|\overline{\mathbf{w}}^{\mathcal{A}} - \mathbf{w}^{\mathcal{A}}\right\|_1 = 2\frac{(e^\delta - 1)P_G^{\mathcal{A}}(1 - P_G^{\mathcal{A}})}{1 + (e^\delta - 1)P_G^{\mathcal{A}}}, \tag{26}$$

and consequently

$$\left\|\overline{\mathbf{w}}^{\mathcal{A}} - \mathbf{w}^{\mathcal{A}}\right\|_1 \leq 2\tanh\left(\frac{\delta}{4}\right). \tag{27}$$

Proof. Let

$$D_{\mathcal{A}} = 1 + (e^\delta - 1)P_G^{\mathcal{A}}. \tag{28}$$

The biased routing weights can be written as

$$\overline{w}_i^{\mathcal{A}} = \begin{cases} \dfrac{e^\delta}{D_{\mathcal{A}}} w_i^{\mathcal{A}}, & g_i = 1, \\ \dfrac{1}{D_{\mathcal{A}}} w_i^{\mathcal{A}}, & g_i = 0. \end{cases} \tag{29}$$

Summing the positive changes over green experts and the negative changes over non-green experts gives

$$\left\|\overline{\mathbf{w}}^{\mathcal{A}} - \mathbf{w}^{\mathcal{A}}\right\|_1 = 2\frac{(e^\delta - 1)P_G^{\mathcal{A}}(1 - P_G^{\mathcal{A}})}{1 + (e^\delta - 1)P_G^{\mathcal{A}}}. \tag{30}$$

Maximizing the right-hand side over $P_G^{\mathcal{A}} \in [0, 1]$ gives

$$\left(P_G^{\mathcal{A}}\right)^\star = \frac{1}{1 + e^{\delta/2}}, \tag{31}$$

and therefore

$$\left\|\overline{\mathbf{w}}^{\mathcal{A}} - \mathbf{w}^{\mathcal{A}}\right\|_1 \leq 2\tanh\left(\frac{\delta}{4}\right). \tag{32}$$

For $\delta \ll 1$,

$$2\tanh\left(\frac{\delta}{4}\right) = \frac{\delta}{2} + O(\delta^3). \tag{33}$$

□

C.3 Cosine Perturbation Bound

We next relate a Euclidean representation perturbation to the resulting cosine-similarity degradation.

Lemma 2 (Cosine perturbation). Let

$$\widetilde{\mathbf{y}} = \mathbf{y} + \mathbf{d} \tag{34}$$

and suppose that

$$\frac{\|\mathbf{d}\|}{\|\mathbf{y}\|} \le r < 1. \tag{35}$$

Then

$$\cos(\mathbf{y}, \widetilde{\mathbf{y}}) \ge \sqrt{1 - r^2}, \tag{36}$$

and consequently

$$1 - \cos(\mathbf{y}, \widetilde{\mathbf{y}}) \le 1 - \sqrt{1 - r^2} = O(r^2). \tag{37}$$

Proof. Let

$$\mathbf{u} = \frac{\mathbf{y}}{\|\mathbf{y}\|} \tag{38}$$

and decompose

$$\mathbf{d} = \|\mathbf{y}\| \left(t\mathbf{u} + \mathbf{v}\right), \qquad \mathbf{v} \perp \mathbf{u}, \tag{39}$$

where

$$t^2 + \|\mathbf{v}\|^2 \le r^2. \tag{40}$$

Then

$$\widetilde{\mathbf{y}} = \|\mathbf{y}\| \left((1 + t)\mathbf{u} + \mathbf{v}\right), \tag{41}$$

and

$$\cos^2(\mathbf{y}, \widetilde{\mathbf{y}}) = \frac{(1 + t)^2}{(1 + t)^2 + \|\mathbf{v}\|^2}. \tag{42}$$

Using $\|\mathbf{v}\|^2 \le r^2 - t^2$, we obtain

$$\cos^2(\mathbf{y}, \widetilde{\mathbf{y}}) \ge \frac{(1 + t)^2}{1 + 2t + r^2}. \tag{43}$$

The right-hand side is minimized at $t = -r^2$, yielding

$$\cos^2(\mathbf{y}, \widetilde{\mathbf{y}}) \ge 1 - r^2. \tag{44}$$

Since $r < 1$ implies $1 + t > 0$, the cosine similarity is positive, and hence

$$\cos(\mathbf{y}, \widetilde{\mathbf{y}}) \ge \sqrt{1 - r^2}. \tag{45}$$

The quadratic bound follows from the Taylor expansion of $\sqrt{1 - r^2}$ around $r = 0$. □

C.4 Case I: Unchanged Top-k Expert Set

We first consider the case in which the routing bias changes only the normalized routing weights but not the active experts.

Theorem 1 (Unchanged top-k expert set). Suppose

$$\widetilde{\mathcal{E}} = \mathcal{E}. \tag{46}$$

Let

$$\kappa = \frac{B}{\|\mathbf{y}\|} \tag{47}$$

and define

$$R_0 = 2\kappa \tanh\left(\frac{\delta}{4}\right). \tag{48}$$

If $\mathbf{y} \neq \mathbf{0}$ and $R_0 < 1$, then

$$\cos(\mathbf{y}, \widetilde{\mathbf{y}}) \ge \sqrt{1 - R_0^2}, \tag{49}$$

and

$$1 - \cos(\mathbf{y}, \widetilde{\mathbf{y}}) \le 1 - \sqrt{1 - R_0^2}. \tag{50}$$

In particular, if κ remains bounded and $\delta \ll 1$, then

$$1 - \cos(\mathbf{y}, \widetilde{\mathbf{y}}) = O(\delta^2). \tag{51}$$

Proof. Since $\widetilde{\mathcal{E}} = \mathcal{E}$, Lemma 1 can be applied directly with $\mathcal{A} = \mathcal{E}$. Therefore,

$$\|\widetilde{\mathbf{y}} - \mathbf{y}\| = \left\| \sum_{i \in \mathcal{E}} (\widetilde{w}_i - w_i) f_i(\mathbf{h}) \right\| \tag{52}$$

$$\leq B \sum_{i \in \mathcal{E}} |\widetilde{w}_i - w_i| \tag{53}$$

$$\leq 2B \tanh\left(\frac{\delta}{4}\right). \tag{54}$$

Dividing by $\|\mathbf{y}\|$ gives

$$\frac{\|\widetilde{\mathbf{y}} - \mathbf{y}\|}{\|\mathbf{y}\|} \leq 2\kappa \tanh\left(\frac{\delta}{4}\right) = R_0. \tag{55}$$

Since $R_0 < 1$, Lemma 2 yields equation 49 and equation 50. Finally, $R_0 = O(\delta)$ for bounded κ, giving

$$1 - \cos(\mathbf{y}, \widetilde{\mathbf{y}}) = O(\delta^2). \tag{56}$$

□

C.5 Case II: A Single Boundary-Expert Replacement

We next consider the dominant case in which one expert near the top-k boundary is replaced:

$$\mathcal{E} = \mathcal{C} \cup \{a\}, \qquad \widetilde{\mathcal{E}} = \mathcal{C} \cup \{b\}, \tag{57}$$

where $\mathcal{C}$ contains the common $k - 1$ experts.

Lemma 3 (Proximity of switched boundary experts). If expert a is selected before bias injection and is replaced by expert b after applying the bias, then the switch requires

$$g_a = 0, \qquad g_b = 1, \tag{58}$$

and

$$0 \leq s_a - s_b \leq \delta. \tag{59}$$

Proof. Since $a \in \mathcal{E}$ and $b \notin \mathcal{E}$ before bias injection,

$$s_a \geq s_b. \tag{60}$$

For b to replace a after bias injection, it is necessary that

$$s_b + \delta g_b \geq s_a + \delta g_a. \tag{61}$$

Therefore,

$$0 \leq s_a - s_b \leq \delta(g_b - g_a). \tag{62}$$

Because $g_a, g_b \in \{0, 1\}$, a strict order reversal requires $g_a = 0$ and $g_b = 1$, which gives

$$0 \leq s_a - s_b \leq \delta. \tag{63}$$

□

Interpretation. A routing bias of magnitude δ can only replace an expert whose routing logit lies within δ of the original top-k boundary. Thus, WaterMoE does not arbitrarily redirect tokens to experts with substantially lower original routing scores.

To separate the effect of routing-weight perturbation from the effect of expert replacement, we define the intermediate weights obtained by applying the bias while temporarily retaining the original expert set:

$$\widehat{w}_i = \frac{\exp(s_i + \delta g_i)}{\sum_{j \in \mathcal{E}} \exp(s_j + \delta g_j)}, \qquad i \in \mathcal{E}, \tag{64}$$

and

$$\widehat{\mathbf{y}} = \sum_{i \in \mathcal{E}} \widehat{w}_i f_i(\mathbf{h}). \tag{65}$$

Lemma 4 (Weight change caused by one normalized replacement). Under the single-replacement setting in equation 57,

$$\sum_{i\in\mathcal{C}} |\widetilde{w}_i - \widehat{w}_i| + |\widetilde{w}_b - \widehat{w}_a| \le 2\tanh\left(\frac{\delta}{4}\right). \tag{66}$$

Proof. Let

$$u_i = s_i + \delta g_i, \qquad Z_{\mathcal{C}} = \sum_{i\in\mathcal{C}} e^{u_i}, \qquad x = e^{u_a}. \tag{67}$$

From Lemma 3,

$$0 \le u_b - u_a = \delta - (s_a - s_b) \le \delta. \tag{68}$$

Define

$$q = e^{u_b - u_a}, \tag{69}$$

so that

$$1 \le q \le e^{\delta}. \tag{70}$$

The relevant weights can then be written as

$$\widehat{w}_a = \frac{x}{Z_{\mathcal{C}} + x}, \qquad \widetilde{w}_b = \frac{qx}{Z_{\mathcal{C}} + qx}, \tag{71}$$

while for each $i \in \mathcal{C}$,

$$\widehat{w}_i = \frac{e^{u_i}}{Z_{\mathcal{C}} + x}, \qquad \widetilde{w}_i = \frac{e^{u_i}}{Z_{\mathcal{C}} + qx}. \tag{72}$$

This replacement is equivalent to increasing the logit of one normalized routing slot by

$$\gamma = \log q \in [0, \delta]. \tag{73}$$

Applying Lemma 1 to this one-slot perturbation gives

$$\sum_{i\in\mathcal{C}} |\widetilde{w}_i - \widehat{w}_i| + |\widetilde{w}_b - \widehat{w}_a| \le 2\tanh\left(\frac{\gamma}{4}\right) \le 2\tanh\left(\frac{\delta}{4}\right). \tag{74}$$

□

Theorem 2 (Single expert replacement). Suppose

$$\mathcal{E} = \mathcal{C} \cup \{a\}, \qquad \widetilde{\mathcal{E}} = \mathcal{C} \cup \{b\}, \tag{75}$$

and Assumption 2 holds:

$$\|f_b(\mathbf{h}) - f_a(\mathbf{h})\| \le \epsilon_f B. \tag{76}$$

Let

$$\kappa = \frac{B}{\|\mathbf{y}\|} \tag{77}$$

and define

$$R_1 = \kappa\left[4\tanh\left(\frac{\delta}{4}\right) + \widetilde{w}_b\epsilon_f\right]. \tag{78}$$

If $\mathbf{y} \neq \mathbf{0}$ and $R_1 < 1$, then

$$\cos(\mathbf{y}, \widetilde{\mathbf{y}}) \ge \sqrt{1 - R_1^2}, \tag{79}$$

and

$$1 - \cos(\mathbf{y}, \widetilde{\mathbf{y}}) \le 1 - \sqrt{1 - R_1^2}. \tag{80}$$

In the small-perturbation regime, for bounded κ,

$$1 - \cos(\mathbf{y}, \widetilde{\mathbf{y}}) = O\left(\delta + \widetilde{w}_b\epsilon_f\right)^2. \tag{81}$$

Proof. We first bound the difference between the original output $\mathbf{y}$ and the intermediate output $\widehat{\mathbf{y}}$. Since the active expert set is fixed at $\mathcal{E}$, Lemma 1 gives

$$\|\widehat{\mathbf{y}} - \mathbf{y}\| \leq 2B \tanh\left(\frac{\delta}{4}\right). \tag{82}$$

Next, the difference between the final output and the intermediate output can be decomposed as

$$\begin{aligned}\widetilde{\mathbf{y}} - \widehat{\mathbf{y}} = &\sum_{i\in\mathcal{C}} (\widetilde{w}_i - \widehat{w}_i) f_i(\mathbf{h}) + (\widetilde{w}_b - \widehat{w}_a) f_a(\mathbf{h}) \\ &+ \widetilde{w}_b \left(f_b(\mathbf{h}) - f_a(\mathbf{h})\right).\end{aligned} \tag{83}$$

Using Lemma 4, Assumption 1, and Assumption 2,

$$\|\widetilde{\mathbf{y}} - \widehat{\mathbf{y}}\| \leq B \left[\sum_{i\in\mathcal{C}} |\widetilde{w}_i - \widehat{w}_i| + |\widetilde{w}_b - \widehat{w}_a|\right] + \widetilde{w}_b \epsilon_f B \tag{84}$$

$$\leq B \left[2 \tanh\left(\frac{\delta}{4}\right) + \widetilde{w}_b \epsilon_f\right]. \tag{85}$$

Combining equation 82 and equation 85 gives

$$\|\widetilde{\mathbf{y}} - \mathbf{y}\| \leq B \left[4 \tanh\left(\frac{\delta}{4}\right) + \widetilde{w}_b \epsilon_f\right]. \tag{86}$$

Dividing by $\|\mathbf{y}\|$ yields

$$\frac{\|\widetilde{\mathbf{y}} - \mathbf{y}\|}{\|\mathbf{y}\|} \leq R_1. \tag{87}$$

Since $R_1 < 1$, Lemma 2 gives equation 79 and equation 80. Finally,

$$4 \tanh\left(\frac{\delta}{4}\right) = \delta + O(\delta^3), \tag{88}$$

which gives

$$1 - \cos(\mathbf{y}, \widetilde{\mathbf{y}}) = O\left(\delta + \widetilde{w}_b \epsilon_f\right)^2. \tag{89}$$

□

Interpretation. With one boundary-expert replacement, the representation perturbation contains two components: an $O(\delta)$ change caused by normalized routing-weight perturbation and an additional $\widetilde{w}_b \epsilon_f$ term caused by replacing one expert with a near-tied expert. Under the empirically validated local-similarity assumption, their combined first-order representation perturbation produces only a second-order cosine-similarity loss.

C.6 Summary and Discussion

The two dominant routing cases can be summarized as

$$\widetilde{\mathcal{E}} = \mathcal{E} \quad \Longrightarrow \quad 1 - \cos(\mathbf{y}, \widetilde{\mathbf{y}}) = O(\delta^2), \tag{90}$$

and

$$\left|\mathcal{E} \triangle \widetilde{\mathcal{E}}\right| = 2 \quad \Longrightarrow \quad 1 - \cos(\mathbf{y}, \widetilde{\mathbf{y}}) = O\left(\delta + \widetilde{w}_b \epsilon_f\right)^2. \tag{91}$$

These bounds are deterministic for any fixed Green Expert Map and therefore do not rely on averaging over randomly sampled maps. They show that, in the dominant routing regimes observed in our experiments, WaterMoE introduces a first-order perturbation to the normalized routing weights while causing only a second-order degradation in the direction of the MoE representation.

The analysis is local to a single MoE layer and is conditional on the empirically validated similarity of near-tied experts. It should therefore be interpreted as a theoretical explanation for the observed representation stability, rather than as a global guarantee on the final token distribution. The end-to-end preservation of model capability is further supported by the layer-wise representation analysis and downstream-task experiments reported in the main text.

D End-to-End Perturbation Bound to Token Selection

The preceding analysis establishes local perturbation bounds for the output of a single MoE layer under the two routing regimes that dominate WaterMoE inference: unchanged expert selection and a single boundary-expert replacement. We now study how such local perturbations propagate through multiple Transformer layers and affect the final vocabulary distribution.

A general analysis allowing an arbitrary number of simultaneous expert replacements at every layer would require tracking all possible changes in active expert sets together with their interactions with subsequent routing decisions. We do not claim that such multi-expert replacements are impossible. Instead, we deliberately restrict the following analysis to the empirically dominant routing regime, which accounts for more than 95% of the observed routing events.

Therefore, the following end-to-end bound is conditional on the considered routing trajectory remaining within the no-replacement or single-boundary- replacement regime at the analyzed routing events. The result should not be interpreted as a worst-case bound for the rare multi-replacement cases.

Per-layer perturbation bound. Consider a Transformer with L layers. Let

$$h_\ell \quad \text{and} \quad \widetilde{h}_\ell \tag{92}$$

denote the hidden representations after layer ℓ under the original and watermarked routing configurations, respectively, with

$$h_0 = \widetilde{h}_0. \tag{93}$$

Let the corresponding clean and watermarked Transformer blocks be denoted by F_ℓ and $\widetilde{F}_\ell$, such that

$$h_\ell = F_\ell(h_{\ell-1}), \qquad \widetilde{h}_\ell = \widetilde{F}_\ell(\widetilde{h}_{\ell-1}). \tag{94}$$

For the same block input $h_{\ell-1}$, the difference

$$\eta_\ell := \left\| \widetilde{F}_\ell(h_{\ell-1}) - F_\ell(h_{\ell-1}) \right\| \tag{95}$$

represents the perturbation newly introduced by WaterMoE at layer ℓ.

From the single-boundary-replacement analysis above,

$$\eta_\ell \le B_\ell \left[4 \tanh\left(\frac{\delta}{4}\right) + \widetilde{w}_{b,\ell}\epsilon_{f,\ell} \right]. \tag{96}$$

Since $0 \le \widetilde{w}_{b,\ell} \le 1$, we obtain the uniform bound

$$\eta_\ell \le \overline{\eta}_\ell := B_\ell \left[4 \tanh\left(\frac{\delta}{4}\right) + \epsilon_{f,\ell} \right]. \tag{97}$$

This bound also covers the unchanged-expert-set case, for which the tighter single-layer bound derived previously applies.

Local propagation assumption. To characterize how an existing representation perturbation propagates through subsequent Transformer blocks, we assume that each watermarked block is locally Lipschitz along the clean–watermarked representation trajectory. Specifically, for each layer ℓ, there exists a finite constant Λ_ℓ such that

$$\left\| \widetilde{F}_\ell(u) - \widetilde{F}_\ell(v) \right\| \le \Lambda_\ell \|u - v\| \tag{98}$$

for representations u and v in the local neighborhood traversed by the clean and watermarked forward passes.

This is a local rather than global smoothness condition. In particular, top-k routing is discrete at expert-selection boundaries; the assumption is therefore applied only along the local perturbation trajectory considered above, where routing changes are dominated by unchanged or single-boundary replacement cases.

Define the accumulated representation error after layer ℓ as

$$e_\ell := \|\widetilde{h}_\ell - h_\ell\|. \tag{99}$$

Then, using equation 94,

$$\begin{aligned} e_\ell &= \left\|\widetilde{F}_\ell(\widetilde{h}_{\ell-1}) - F_\ell(h_{\ell-1})\right\| \\ &= \left\|\widetilde{F}_\ell(\widetilde{h}_{\ell-1}) - \widetilde{F}_\ell(h_{\ell-1}) + \widetilde{F}_\ell(h_{\ell-1}) - F_\ell(h_{\ell-1})\right\| \\ &\le \left\|\widetilde{F}_\ell(\widetilde{h}_{\ell-1}) - \widetilde{F}_\ell(h_{\ell-1})\right\| + \left\|\widetilde{F}_\ell(h_{\ell-1}) - F_\ell(h_{\ell-1})\right\| \\ &\le \Lambda_\ell e_{\ell-1} + \eta_\ell \\ &\le \Lambda_\ell e_{\ell-1} + \overline{\eta}_\ell. \end{aligned} \tag{100}$$

The equation 100 separates the total perturbation at each layer into two components: the propagation of previously accumulated perturbations and the new perturbation introduced by the routing bias at the current layer.

Since $e_0 = 0$, recursively expanding equation 100 gives

$$\begin{aligned} e_L \le \overline{\eta}_L + \Lambda_L\overline{\eta}_{L-1} + \Lambda_L\Lambda_{L-1}\overline{\eta}_{L-2} + \cdots \\ + \left(\prod_{j=2}^{L}\Lambda_j\right)\overline{\eta}_1. \end{aligned} \tag{101}$$

Equivalently,

$$\boxed{e_L \le \sum_{\ell=1}^{L}\overline{\eta}_\ell \prod_{j=\ell+1}^{L}\Lambda_j} \tag{102}$$

where the empty product for $\ell = L$ is defined as 1.

Substituting equation 97 yields

$$e_L \le \sum_{\ell=1}^{L} B_\ell\left[4\tanh\left(\frac{\delta}{4}\right) + \epsilon_{f,\ell}\right]\prod_{j=\ell+1}^{L}\Lambda_j. \tag{103}$$

For $\delta \ll 1$,

$$4\tanh\left(\frac{\delta}{4}\right) = \delta + O(\delta^3), \tag{104}$$

and hence the routing-induced contribution to the final hidden-state perturbation remains first-order in δ, modulated by the subsequent layer-wise propagation factors.

Propagation to vocabulary logits. Let the final hidden representation be transformed by the final normalization and LM head as

$$r = \mathrm{LN}_f(h_L), \qquad z = W_U r + b, \tag{105}$$

where $z \in \mathbb{R}^{|\mathcal{V}|}$ denotes the vocabulary-logit vector. The corresponding watermarked quantities are $\widetilde{r}$ and $\widetilde{z}$.

Assume that the final normalization is locally Lipschitz with constant L_f:

$$\|\widetilde{r} - r\| \le L_f\|\widetilde{h}_L - h_L\| = L_f e_L. \tag{106}$$

Let $w_v^\top$ denote the row of W_U corresponding to token $v \in \mathcal{V}$. Then

$$\begin{aligned} |\widetilde{z}_v - z_v| &= \left|w_v^\top(\widetilde{r} - r)\right| \\ &\le \|w_v\|_2\|\widetilde{r} - r\|_2 \\ &\le L_f\|w_v\|_2 e_L. \end{aligned} \tag{107}$$

Define

$$C_{\text{head}} := L_f \max_{v \in \mathcal{V}} \|w_v\|_2. \tag{108}$$

It follows that

$$\boxed{\|\widetilde{z} - z\|_\infty \le C_{\text{head}} e_L} \tag{109}$$

and therefore, by equation 102,

$$\boxed{\|\widetilde{z} - z\|_\infty \le C_{\text{head}} \sum_{\ell=1}^{L} \overline{\eta}_\ell \prod_{j=\ell+1}^{L} \Lambda_j}. \tag{110}$$

Substituting the uniform single-layer bound gives

$$\boxed{\|\widetilde{z} - z\|_\infty \le C_{\text{head}} \sum_{\ell=1}^{L} B_\ell \left[4 \tanh\left(\frac{\delta}{4}\right) + \epsilon_{f,\ell}\right] \prod_{j=\ell+1}^{L} \Lambda_j}. \tag{111}$$

The equation 111 directly connects the routing perturbation introduced by WaterMoE to the maximum perturbation of any vocabulary logit.

Greedy-decoding stability. Let

$$v^\star = \arg\max_{v \in \mathcal{V}} z_v \tag{112}$$

be the token selected by the clean model, and define its logit margin as

$$m := z_{v^\star} - \max_{v \ne v^\star} z_v. \tag{113}$$

Let

$$\varepsilon_{\text{tok}} := C_{\text{head}} \sum_{\ell=1}^{L} \overline{\eta}_\ell \prod_{j=\ell+1}^{L} \Lambda_j. \tag{114}$$

From equation 110,

$$|\widetilde{z}_v - z_v| \le \varepsilon_{\text{tok}}, \qquad \forall v \in \mathcal{V}. \tag{115}$$

For any competing token $v \ne v^\star$,

$$\begin{aligned} \widetilde{z}_{v^\star} - \widetilde{z}_v &\ge (z_{v^\star} - \varepsilon_{\text{tok}}) - (z_v + \varepsilon_{\text{tok}}) \\ &= (z_{v^\star} - z_v) - 2\varepsilon_{\text{tok}} \\ &\ge m - 2\varepsilon_{\text{tok}}. \end{aligned} \tag{116}$$

Therefore, if

$$\boxed{m > 2\varepsilon_{\text{tok}},} \tag{117}$$

then

$$\widetilde{z}_{v^\star} > \widetilde{z}_v, \qquad \forall v \ne v^\star, \tag{118}$$

and consequently

$$\boxed{\arg\max_{v \in \mathcal{V}} \widetilde{z}_v = \arg\max_{v \in \mathcal{V}} z_v.} \tag{119}$$

Thus, under the local propagation assumptions above, the per-layer routing perturbations introduced by WaterMoE accumulate according to equation 102, induce a bounded perturbation of the final vocabulary logits, and preserve the selected token whenever the clean model's logit margin exceeds twice the resulting end-to-end perturbation bound.

Sampling-distribution stability. The preceding result gives an exact sufficient condition for preserving the greedy token. In practical generation, however, tokens are often sampled from the model distribution rather than selected by a deterministic argmax. We therefore further characterize how the bounded logit perturbation affects the sampling distribution.

Let

$$p_v = \frac{\exp(z_v/T)}{\sum_{u\in\mathcal{V}}\exp(z_u/T)}, \qquad \widetilde{p}_v = \frac{\exp(\widetilde{z}_v/T)}{\sum_{u\in\mathcal{V}}\exp(\widetilde{z}_u/T)}, \tag{120}$$

where $T > 0$ denotes the sampling temperature. From equation 115, we have

$$|\widetilde{z}_v - z_v| \le \epsilon_{\text{tok}}, \qquad \forall v \in \mathcal{V}. \tag{121}$$

Let $d_v = \widetilde{z}_v - z_v$. Then

$$\frac{\widetilde{p}_v}{p_v} = \frac{\exp(d_v/T)}{\mathbb{E}_{u\sim p}\left[\exp(d_u/T)\right]}. \tag{122}$$

Since $|d_v| \le \epsilon_{\text{tok}}$,

$$e^{-\epsilon_{\text{tok}}/T} \le \mathbb{E}_{u\sim p}\left[\exp(d_u/T)\right] \le e^{\epsilon_{\text{tok}}/T}, \tag{123}$$

and therefore

$$e^{-2\epsilon_{\text{tok}}/T} \le \frac{\widetilde{p}_v}{p_v} \le e^{2\epsilon_{\text{tok}}/T}. \tag{124}$$

Equivalently, for every token,

$$|\log\widetilde{p}_v - \log p_v| \le \frac{2\epsilon_{\text{tok}}}{T}. \tag{125}$$

Moreover, the total-variation distance between the clean and watermarked sampling distributions satisfies

$$D_{\text{TV}}\left(p, \widetilde{p}\right) \le \tanh\left(\frac{\epsilon_{\text{tok}}}{T}\right). \tag{126}$$

Hence, in the small-perturbation regime,

$$D_{\text{TV}}\left(p, \widetilde{p}\right) = O\left(\frac{\epsilon_{\text{tok}}}{T}\right). \tag{127}$$

Therefore, for any subset of candidate tokens $\mathcal{A} \subseteq \mathcal{V}$,

$$\left|\Pr_{\widetilde{p}}(v \in \mathcal{A}) - \Pr_{p}(v \in \mathcal{A})\right| \le \tanh\left(\frac{\epsilon_{\text{tok}}}{T}\right). \tag{128}$$

Thus, even when the greedy-token margin condition is not satisfied, a small end-to-end logit perturbation still guarantees that the full sampling distribution remains close to the original one.

Interpretation: controlled perturbation as watermark evidence. The goal of WaterMoE is not to make the watermarked model mathematically identical to the original model. A strictly zero perturbation would also remove the statistical evidence required for watermark detection. Instead, WaterMoE operates in a controlled nonzero-perturbation regime.

The preceding analysis shows that small routing perturbations induce bounded changes in the final hidden representation and vocabulary logits. When the clean-model logit margin is sufficiently large, the greedy token is preserved exactly. More generally, under probabilistic sampling, the resulting token distribution remains close to the original distribution in total variation, with

$$D_{\text{TV}}(p, \widetilde{p}) \le \tanh\left(\frac{\epsilon_{\text{tok}}}{T}\right). \tag{129}$$

At the same time, the perturbation is generally nonzero. The equation 125 shows that it induces small token-level log-likelihood shifts between the biased and reference routing configurations. These small but systematic likelihood shifts are precisely the statistical evidence accumulated by the WaterMoE detector.

Therefore, the theoretical objective is not zero loss, but a favorable perturbation regime in which the routing-induced deviation is sufficiently small to preserve model behavior while remaining sufficiently consistent to be statistically detectable. This explains how WaterMoE can simultaneously maintain high generation fidelity and embed a measurable watermark signal.

E PERFORMANCE ON MODERATE-COMPLEXITY TASKS

Table 4 shows the detection-text quality trade-off on both MultiNews and ELI5. On the Mixtral-8×7B model, which is harder to achieve balance, WaterMoE achieves substantially higher detection rates under strict thresholds while also attaining the highest AUC and maintaining Rouge-L comparable to the unwatermarked baseline. A similar trend holds on ELI5, where WaterMoE achieves the best AUC and Rouge-L, with strong detection performance. Overall, it achieves strong detectability without sacrificing text quality across models and tasks.

Table 4: Watermarking performance on Moderate-Complexity Tasks. WaterMoE consistently achieves the highest AUC and TPR while maintaining competitive Rouge-L scores.

Dataset: MultiNews								
Algorithm	**Mixtral-8×7B**				**Qwen**			
	TPR@1%	**TPR@5%**	**AUC**	**Rouge-L**	**TPR@1%**	**TPR@5%**	**AUC**	**Rouge-L**
Unwatermarked	-	-	-	38.1	-	-	-	40.6
EWD Lu et al. (2024)	71.0	84.5	94.2	37.8	91.5	95.5	99.3	40.1
EXPEdit Kuditipudi et al. (2023)	41.5	61.5	91.7	35.3	36.0	48.0	77.6	31.5
KGW Kirchenbauer et al. (2023)	21.0	49.5	86.5	36.3	98.5	100.0	100.0	40.3
MorphMark Wang et al. (2025b)	26.0	41.0	88.2	36.4	77.0	93.5	98.9	40.5
SIR Liu et al. (2023b)	21.5	36.0	85.0	35.9	-	-	-	-
SynthID Dathathri et al. (2024)	8.0	17.5	63.7	38.0	98.0	99.0	99.5	**40.5**
Unbiased Hu et al. (2023)	74.5	82.5	93.1	37.7	13.0	52.0	73.8	40.5
UPV Liu et al. (2023a)	17.0	29.5	87.4	34.2	3.5	27.0	84.3	40.3
XSIR He et al. (2024)	8.0	19.5	67.5	35.3	-	-	-	-
WaterMoE	**83.5**	**86.5**	**99.6**	**38.3**	**100.0**	**100.0**	**100.0**	40.4
Dataset: ELI5								
Algorithm	**Mixtral-8×7B**				**Qwen**			
	TPR@1%	**TPR@5%**	**AUC**	**Rouge-L**	**TPR@1%**	**TPR@5%**	**AUC**	**Rouge-L**
Unwatermarked	-	-	-	31.2	-	-	-	28.8
EWD	**87.0**	91.0	98.6	29.3	100.0	100.0	100.0	28.5
EXPEdit	28.5	28.5	64.9	29.4	32.0	32.0	69.8	26.1
KGW	78.0	89.5	97.6	28.8	100.0	100.0	100.0	28.5
MorphMark	49.5	64.5	90.8	29.3	92.0	96.5	99.5	28.5
SIR	63.0	75.5	93.7	28.6	-	-	-	-
SynthID	43.0	67.0	92.1	30.2	99.5	100.0	100.0	28.5
Unbiased	84.0	89.5	96.4	29.8	100.0	100.0	100.0	28.0
UPV	60.5	80.5	95.1	28.4	95.5	100.0	99.9	28.2
XSIR	47.0	60.5	87.5	27.6	-	-	-	-
WaterMoE	81.0	**97.5**	**98.8**	**30.5**	**100.0**	**100.0**	**100.0**	**28.8**

F THREE-ENT AND SEMANTIC ENTROPY ANALYSIS

As illustrated in Fig. 6, we compare the trigram entropy (three-ent) and semantic entropy of texts generated by different methods. Our proposed WaterMoE produces entropy values that are closest to those of the non-watermarked model. In contrast, other watermarking methods lead to noticeable reductions in entropy.

This observation indicates that WaterMoE better preserves the diversity and richness of the generated text, maintaining natural generation characteristics while embedding the watermark.

G WATERMARKING SETUP FOR REPRODUCIBILITY

G.1 WATERMARK HYPERPARAMETER CONFIGURATION

To ensure reproducibility, we summarize the watermark-specific hyperparameters for all algorithms under different task complexity levels in Table 5. Each cell reports the key hyperparameter used for the corresponding setting.

The watermark-strength hyperparameters vary across task complexity levels because weak watermark injection leads to very low detection rates on high-complexity tasks. We therefore use stronger watermark settings for several baselines in these scenarios. For existing methods, watermark strength

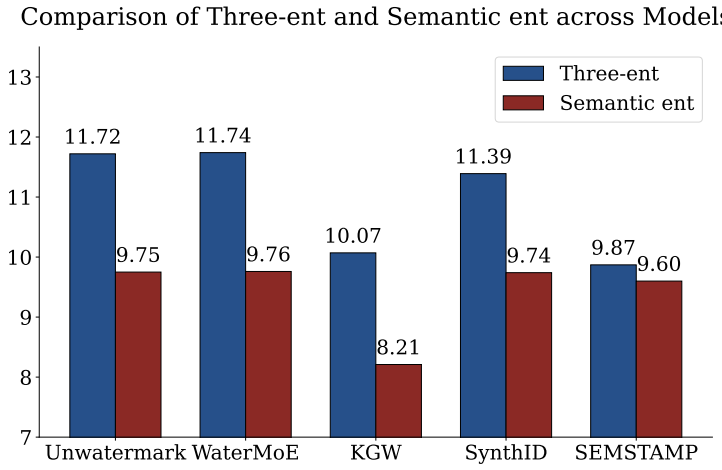


Figure 6: Trigram entropy and semantic entropy comparison. WaterMoE preserves entropy closest to the non-watermarked model, indicating better text diversity.

Table 5: Watermark hyperparameters across different task complexity levels. EXPEdit and Unbiased are distortion-free algorithms and thus do not require injection strength parameters (denoted by -). SIR and XSIR are not applicable to high-complexity tasks as they are incompatible with the Qwen model.

Algorithm	**Low**	**Moderate**	**High**
KGW	$\delta = 2$	$\delta = 2$	$\delta = 5$
SynthID	`num_leaves=2`	`num_leaves=2`	`num_leaves=8`
EWD	$\delta = 2$	$\delta = 2$	$\delta = 5$
EXPEdit	-	-	-
MorphMark	$k_{\text{exp}} = 1.3$	$k_{\text{exp}} = 1.3$	$k_{\text{exp}} = 7$
Unbiased	-	-	-
UPV	$\delta = 2$	$\delta = 2$	$\delta = 4$
SIR	$\delta = 1$	$\delta = 1$	-
XSIR	$\delta = 1$	$\delta = 1$	-
WaterMoE	$\delta = 0.8$	$\delta = 0.2$	$\delta = 0.2$

generally exhibits a trade-off with generation quality: increasing the strength improves detectability but often degrades text quality and downstream task performance. In contrast, WaterMoE maintains both high detection performance and strong generation quality across different task complexities, even with a small routing bias.

G.2 Detection Calibration and Evaluation Protocol

Watermark detection requires calibrating the null distribution of the detection statistic and determining the decision threshold corresponding to a target false-positive rate (FPR). Following the standard evaluation protocol adopted by MarkLLM and prior LLM watermarking studies, we use a held-out calibration set that is strictly separated from the final evaluation set.

Calibration set construction. For each dataset, we randomly sample 500 examples that are excluded from the final evaluation subset and use them exclusively for detector calibration. Therefore, the calibration and evaluation subsets are strictly disjoint. Within each model–dataset setting, the same 500 underlying calibration examples are shared across all watermarking methods. Each method computes its own detection statistic on this common calibration set and derives its corresponding decision threshold from the resulting null distribution. Using an identical calibration set across methods eliminates differences in calibration data as a confounding factor and enables a controlled and fair comparison among watermarking algorithms.

For WaterMoE, we compute the sequence-level detection statistic $Z(x)$ defined in Eq. (6). We distinguish the token-level filtering threshold τ in Eq. (5) from the sequence-level decision threshold used to classify a complete sequence. Throughout all experiments, we fix $\tau = 0$. The 500 held-out calibration samples are used to estimate the null-distribution statistics $\mu_0(\tau)$ and $\sigma_0(\tau)$, and subsequently determine the sequence-level decision threshold.

Decision-threshold estimation. For a target false-positive rate α, we denote the corresponding sequence-level decision threshold by γ_α. The threshold γ_α is estimated exclusively from the held-out calibration negatives such that the empirical false-positive rate on the calibration set corresponds to the target operating point. We use $\alpha = 0.01$ and $\alpha = 0.05$ to obtain the thresholds for reporting TPR@1% and TPR@5%, respectively. Once calibrated, γ_α is frozen and applied to the held-out evaluation samples without further adjustment. In particular, no sample from the final evaluation subset is used to estimate $\mu_0(\tau)$, $\sigma_0(\tau)$, or γ_α.

Formally, given the calibration set $\mathcal{D}_{\text{cal}} = \{x_i\}_{i=1}^{N_{\text{cal}}}$ with $N_{\text{cal}} = 500$, the null-distribution statistics are estimated as

$$\hat{\mu}_0(\tau) = \frac{1}{N_{\text{cal}}} \sum_{i=1}^{N_{\text{cal}}} S(x_i; \tau), \tag{130}$$

and

$$\hat{\sigma}_0(\tau) = \sqrt{\frac{1}{N_{\text{cal}} - 1} \sum_{i=1}^{N_{\text{cal}}} \left(S(x_i; \tau) - \hat{\mu}_0(\tau)\right)^2}. \tag{131}$$

The calibrated statistic is then

$$Z(x) = \frac{S(x; \tau) - \hat{\mu}_0(\tau)}{\hat{\sigma}_0(\tau)}. \tag{132}$$

For each target FPR α, γ_α is determined from the empirical null distribution of $Z(x)$ on $\mathcal{D}_{\text{cal}}$. A test sequence is classified as watermarked if

$$Z(x) > \gamma_\alpha. \tag{133}$$

Evaluation subsets. To evaluate watermarking performance over a broad range of tasks while keeping the computational cost manageable for the large number of evaluated baselines, we randomly sample different numbers of evaluation examples according to task complexity. For low-complexity and moderate-complexity tasks, we use 200 test examples for each model–dataset pair. For high-complexity tasks, we use 100 test examples for each model–dataset pair. These samples are used exclusively for final evaluation and never participate in detector calibration.

Cross-dataset threshold stability. The dataset-specific calibration protocol described above follows MarkLLM and prior watermarking studies and is applied identically to WaterMoE and all baseline methods to ensure a fair comparison. In practical deployment, however, recalibrating the detector for every downstream data distribution may be undesirable. A deployment-friendly watermark detector should ideally maintain a relatively stable decision threshold across heterogeneous domains.

We therefore additionally examine the cross-dataset variation of the calibrated decision thresholds. Empirically, WaterMoE exhibits substantially smaller threshold variation across datasets than representative token-level watermarking baselines. Across the evaluated datasets, the calibrated thresholds of WaterMoE vary by less than 10%, whereas representative KGW-series baselines exhibit variations exceeding 30%. This suggests that the routing-induced likelihood signal of WaterMoE is relatively stable across heterogeneous task distributions and is therefore more amenable to a fixed-threshold deployment setting.

Nevertheless, we do not exploit this additional advantage in the main comparison. Instead, following MarkLLM and prior work, we independently calibrate the decision threshold for each dataset for both WaterMoE and all baseline methods. This ensures that all methods are evaluated under the same dataset-specific calibration protocol and avoids favoring WaterMoE through a different threshold-selection strategy.

H Dataset and Global Prompts

To ensure evaluation consistency and standardize model behavior across diverse tasks, we define a specific "Global Prompt" for each dataset. These prompts serve as the foundational instructions that dictate the model's reasoning style and output constraints, effectively minimizing performance variance caused by prompt phrasing.

The complete set of these standardized instructions, categorized by task type and dataset, is presented in Figure 7. These prompts are prepended to each individual query to guide the model's generation process throughout our experiments.

Dataset	Task Type	Global Prompt
MultiNews	Summarization	Please summarize the following article:
ELI5	Open-domain QA	Please answer the question:
APPS	Code Generation	Please output only the final Python code solution. Do not include any explanation or reasoning.
CodeContests	Code Generation	Please output only the final Python code solution. Do not include any explanation or reasoning.
GSM8K	Math Reasoning	Please reason step by step, and you MUST provide a final answer. The final answer MUST be explicitly written in the format \boxed{}.
MMLU	Multiple Choice	The following is a multiple-choice question. First, reason through the problem step by step, analyzing each option. Then, conclude with your final answer using ONLY one letter from A, B, C, or D, placed inside \boxed{}. Do not include any text inside the box other than the single letter.

Figure 7: Detailed Global Prompts used across various datasets to standardize model instructions and output formats.

I Configuration of Attack Prompts

In this section, we present the prompts used for different types of watermark attacks. In addition to the document-level paraphrasing attack, we specifically design token-level substitution.

Token-level Substitution Attack. The prompt for the token-level substitution attack explicitly instructs the model to replace a specific ratio of words with their synonyms while maintaining the original semantic meaning and logical order as much as possible, as shown in Figure 8.

> Please rewrite the following text by substituting words.
> **Requirement:** Replace exactly {ratio}% of the words while keeping the overall semantics, sentence order, and logic unchanged.
> **Strategy:**
> The primary goal is to achieve the {ratio}% substitution rate through synonym replacement. If a high substitution rate is requested, prioritize reaching the target ratio even if semantic coherence cannot be fully guaranteed. If a low substitution rate is requested, ensure that {100-ratio}% of the text remains strictly unchanged. Output only the rewritten text without any additional explanation.
>
> {text}

Figure 8: Prompt used for the token-level substitution attack.

Document-level Paraphrasing Attack. For the document-level paraphrasing attack, we instruct the model to rewrite the entire text output, simulating a real-world scenario of semantic restructuring, as depicted in Figure 9.

Please rewrite the following text (Only return the rewritten text):

{text}

Figure 9: Prompt used for the document-level paraphrasing attack.

Table 6: Key-specific detection under different negative sources. The positive samples are WaterMoE texts generated by Mixtral-8×7B using the target Expert Map. Each row independently replaces the negative set while keeping the detector and decision threshold unchanged.

Negative source	**TPR (%)** ↑	**FPR (%)** ↓
Human-written C4 text	100.0	0.0
Non-watermarked Mixtral-8×7B outputs	100.0	0.0
Non-watermarked Qwen outputs	100.0	0.0
WaterMoE outputs generated by Qwen	100.0	0.0
KGW-watermarked text	100.0	0.0
WaterMoE outputs with incorrect Expert Maps	100.0	0.0

J Key Specificity and Negative-Source Generalization

WaterMoE detects whether a text exhibits the routing preference induced by a specific Green Expert Map, rather than identifying which model generated the text. In our detector, the reference configuration mainly estimates the intrinsic likelihood of each token under natural routing. Comparing this likelihood with that under the target biased routing configuration removes the text's intrinsic fluency and retains the additional preference induced by the watermark:

$$\Delta_t = \log P_{\text{bias}}(x_t \mid x_{<t}; M) - \log P_{\text{ref}}(x_t \mid x_{<t}), \tag{134}$$

where M denotes the target Green Expert Map. Consequently, fluent text may receive high likelihood under both configurations, but only text generated with the target map should exhibit a consistent positive routing-induced shift.

To verify that the detector does not merely distinguish between generators, we construct multiple negative sets from substantially different sources. Specifically, we separately use human-written C4 text, non-watermarked Mixtral-8×7B outputs, non-watermarked Qwen outputs, WaterMoE outputs generated by Qwen, and KGW-watermarked text as negative samples. We additionally evaluate WaterMoE texts generated with independently sampled incorrect Expert Maps. In every experiment, the positive set consists of Mixtral-8×7B outputs generated using the target Expert Map, and the detection configuration always uses this target map.

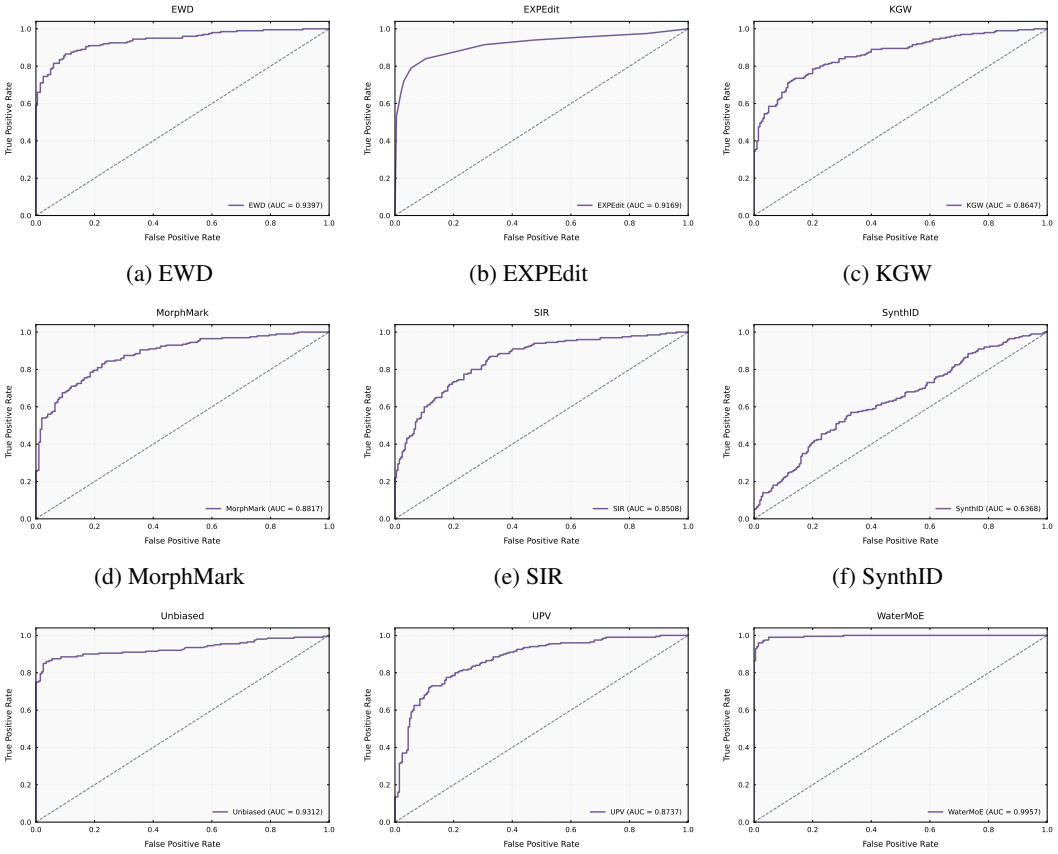
As shown in Table 6, WaterMoE achieves a 100% true-positive rate and a 0% false-positive rate for every negative source. In particular, texts watermarked using another model or an incorrect Expert Map are not falsely attributed to the target watermark. These results demonstrate that the detector captures the map-specific routing bias rather than generic model characteristics, text fluency, or the presence of an arbitrary watermark.

K ROC-AUC Curves of Watermarking Methods on MultiNews.

As shown in Fig.10, we present the ROC-AUC curves of all evaluated watermarking algorithms on the MultiNews task using Mixtral-8x7B.

L Examples of Prompts and Generated Texts across Different Tasks

To provide a qualitative understanding of watermark behavior across different task types, we present representative examples from three categories: summarization, code generation, and alignment. For



(a) EWD (b) EXPEdit (c) KGW

(d) MorphMark (e) SIR (f) SynthID

(g) Unbiased (h) UPV (i) WaterMoE

Figure 10: ROC-AUC curves of different watermarking methods on MultiNews (Mixtral-8x7B).

each task, we show the input prompt together with the corresponding outputs generated with and without watermarking. These examples highlight how WaterMoE embeds watermark signals while preserving task fidelity. In particular, the comparisons illustrate that the watermarked outputs remain semantically consistent with the unwatermarked ones, even under strict task constraints such as structured code generation and format-sensitive alignment tasks.

M Stealthiness Analysis

Beyond detectability by authorized parties, a desirable watermark should remain statistically indistinguishable from unwatermarked text under general-purpose analysis. We evaluate this property using two representative black-box statistical detectors Gloaguen et al. (2024), namely R-G detection and FIXED detection, which are designed to identify common watermarking artifacts, including distributional bias and diminished generation diversity. As shown in Table 7, existing methods are detectable under at least one setting, while WaterMoE consistently achieves high p-values across all cases. This indicates that its outputs remain statistically indistinguishable from the unwatermarked text, demonstrating strong stealthiness.

Summarization Task **Prompt**

Please summarize the following article: Passage 1:
The early Monday declared an unlawful assembly on the streets surrounding City Hall and ordered Occupy L.A. protesters to immediately disperse or face arrest. Police officers carrying batons, plastic handcuffs and non-lethal weapons lined up on 1st Street directly outside headquarters in anticipation of possible arrests of protesters who were standing in the streets. "It is not our intent to clear the park at this time," an officer said over a loudspeaker...

Watermark text

In Philadelphia, on Sunday, November 30, a 5 p.m. deadline imposed by Mayor Michael Nutter for Occupy Philly to leave Dilworth Plaza expired without any police removal. Roughly 100 protesters continued to stay peacefully, chanting and distributing food, as law enforcement kept a low-key and composed presence. Chief Inspector Joseph Sulivan highlighted collaboration and nonviolence, noting that conflict was not the goal and that the city was seeking a solution.Although authorities intended to move forward with a $50 million renovation of Dilworth including new facilities and fencing, authorities did not act immediately. A number of tents had already been taken down, and some demonstrators were preparing to move to Occupy locations in Washington, D.C.Meanwhile, city officials and observers, such as activist Ellen Rogovinhar, called for restraint and voiced support for the movement's message. Public figures, including Russell Simmons, also appealed to Mayor Nutter to refrain from using force. The situation stayed calm throughout, with no arrests reported.

Unwatermark text

In Philadelphia, a deadline set by Mayor Michael Nutter for Occupy Philly to vacate Dilworth Plaza by 5 p.m. on Sunday, November 30, passed without a police eviction. About 1,00 of protesters remained peacefully, chanting and sharing food, while police maintained a calm presence. Police Chief Inspector Joseph Sulivan emphasized cooperation and nonviolence, stating that confrontation was not desired and that the city was working toward a resolution. Although the city planned to begin a $50-million renovation of Dilworth, including new facilities and fencing, authorities did not act immediately. Many tents were already being dismantled, with some protesters planning to relocate to Occupy sites in Washington, D.C. City officials and observers, including activist Ellen Rogovinhar, urged restraint and expressed support for the movement's message, while public figures like Russell Simmons urged Mayor Nutter to avoid force. No arrests were made, and the situation remained peaceful.

Figure 11: An example of the Summarization task from the Multi-News dataset. The texts are generated using the WaterMoE watermark algorithm. In the "Watermark text" section, words with a green background indicate tokens that were altered by the watermark algorithm, while words with a gray background indicate unchanged tokens that remain identical to those in the "Unwatermark text".

Coding Task **Prompt**

Please output only the final Python code solution. Do not include any explanation or reasoning.Write a Python solution for the following problem:
You have a long fence which consists of n sections. Unfortunately, it is not painted, so you decided to hire q painters to paint it. i-th painter will paint all sections x such that $l_i \leq x \leq r_i$.
Unfortunately, you are on a tight budget, so you may hire only q - 2 painters. Obviously, only painters you hire will do their work.
You want to maximize the number of painted sections if you choose q - 2 painters optimally. A section is considered painted if at least one painter paints it...

Watermark text

```
n, q = map(int, input().split())
segs = [tuple(map(int, input().split())) for _ in range(q)]
ans = 0
for i in range(q):
    for j in range(i + 1, q):
        # difference array to mark coverage
        diff = [0] * (n + 2)
        for k, (l, r) in enumerate(segs):
            # skip removed painters
            if k != i and k != j:
                diff[l] += 1
                diff[r + 1] -= 1
        cur = cnt = 0
        for x in range(1, n + 1):
            cur += diff[x]
            if cur > 0: cnt += 1
        ans = max(ans, cnt)
print(ans)
```

Unwatermark text

```
n, q = map(int, input().split())
painters = []
for _ in range(q):
    l, r = map(int, input().split())
    painters.append((l, r))
max_painted = 0
# Try removing every pair of painters
for i in range(q):
    for j in range(i + 1, q):
        # Create a set of painted sections when painters i and j are removed
        painted = set()
        for k in range(q):
            if k != i and k != j:
                l, r = painters[k]
                for section in range(l, r + 1):
                    painted.add(section)
        max_painted = max(max_painted, len(painted))
print(max_painted)
```

Figure 12: An example of the Coding task from the APPS dataset. The text demonstrates the prompt and the corresponding outputs generated with and without the WaterMoE watermark algorithm.

Table 7: Stealthiness evaluation measured by p-values of statistical watermark detectors. A p-value below 0.05 indicates watermark detection.

Method	Mixtral-8×7B		Qwen	
	R-G Detection	FIXED Detection	R-G Detection	FIXED Detection
Unwatermarked	1.00	0.94	1.00	0.869
KGW-series	0.00	0.94	0.00	0.94
SynthID	0.06	0.94	0.04	0.94
EXP	1.00	1.8×10^{-6}	1.00	1.5×10^{-7}
WaterMoE	0.92	0.94	1.00	0.94

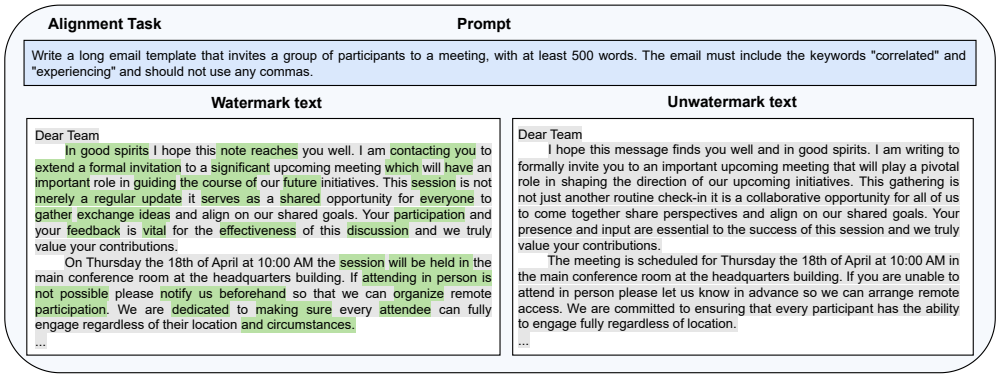


Figure 13: An example of the Alignment task from the IFEval dataset. The text illustrates the prompt along with the watermarked (using WaterMoE) and unwatermarked generation results, demonstrating adherence to the length and formatting constraints.