SAEMark: Steering Personalized Multilingual LLM Watermarks with Sparse Autoencoders

Zhuohao Yu, Xingru Jiang, Weizheng Gu, Chang Gao,
Yidong Wang, Shikun Zhang, Wei Ye[†]
Peking University
zyu@stu.pku.edu.cn, wye@pku.edu.cn

Abstract

Watermarking LLM-generated text is critical for content attribution and misinformation prevention. However, existing methods compromise text quality, require white-box model access and logit manipulation —limitations that exclude API-based models and multilingual / domain-specific use cases. We propose SAEMARK, a user-specific watermarking method that embeds personalized signatures without altering logits. SAEMARK utilizes *Sparse Autoencoder (SAE)* to extract semantic features of generated texts and selects outputs matching watermark key-derived feature distributions via *rejection sampling*. This approach operates post-generation, preserving text quality while naturally generalizing to multilingual scenarios including programming languages. Experiments across 4 datasets shows SAEMARK's consistent accuracy, highest text quality among 8 baselines. SAEMark establishes a new paradigm: secure, personalizable watermarks that work out-of-the-box for all languages and closed-source LLMs for ethical and trustworthy AI systems. [1]

1 Introduction

Large language models (LLMs) have been deployed across diverse domains and languages, from creative writing to code synthesis (Brown et al., 2020; Guo et al., 2024). They are capable of generating natural text indistinguishable from human writing. While this capability is valuable, it also poses risks: misinformation, copyright infringement, content laundering or even harmful content. As these models become ubiquitous, the need for reliable content attribution grows increasingly urgent to address risks of misinformation propagation, copyright infringement, and content laundering. While watermarking – *embedding detectable signatures into generated text* – offers a promising solution, existing methods struggle to reconcile competing demands: preserving text quality, application on diverse domains and enabling fine-grained attribution while respecting user privacy.

Current approaches exhibit critical limitations that hinder practical deployment. The majority of existing methods focus on logit manipulation techniques (e.g. KGW (Kirchenbauer et al., 2023) and EXP (Aaronson and Kirchner, 2022)) that degrade generation quality through direct token probability distortion, while syntactic pattern injection (Atallah et al., 2002; Hou et al., 2023) fails to generalize across languages. Specialized methods like SWEET (Lee et al., 2024) achieve domain-specific detection but cannot handle multilingual text or code simultaneously. Most importantly, existing methods neglect the critical requirement of *fine-grained user attribution*—determining not just *whether text is machine-generated*, but precisely *which user originated it while preserving privacy*.

We address these challenges through SAEMARK, a novel framework that utilizes sparse autoencoder interpretability techniques for safe content attribution. Sparse autoencoders (Ng et al., 2011), typically used to analyze LLM internals, reveal that different language generations exhibit characteristic patterns in their activation feature concentrations. Our breakthrough stems from two key observations: (1) Different generations exhibit distinct sparse feature activation patterns in their hidden states, as revealed by SAE analysis (Huben et al., 2024), and (2) These patterns can be deliberately shaped through rejection sampling without accessing or altering the base LLM's weights or decoding algorithm. Through controlled experiments on the C4 dataset (Raffel et al., 2020), we demonstrate that SAE-derived feature concentrations follow predictable distributions that enable watermark embedding through sampling—unlike distribution-preserving methods like DIP (Wu et al.,

[1]We open-source all our code and datasets anonymously at: https://anonymous.4open.science/r/SAEMark/

2023) that maintain token probabilities at the cost of detection accuracy.

The SAEMARK framework operates through three coordinated mechanisms, illustrated in Figure 1. First, we extract language-agnostic semantic features using pre-trained SAEs on a small, Anchor LLM, leveraging their ability to capture syntax-invariant patterns across different languages and domains, overcoming the syntactic limitations of methods like SWEET (Lee et al., 2024). Second, we generate target distributions from hashed user credentials through a novel key-binding protocol that avoids content storage. Third, we select output candidates through rejection sampling that align generated text's feature concentrations with these targets, preserving the LLM's original output distribution. This approach fundamentally differs from prior work by operating on post-generation text features rather than manipulating generation dynamics—enabling API-compatible watermarking of closed-source models while supporting 1,200+ unique user signatures.

Our contributions are summarized as follows:

- A logit-free watermarking framework that achieve high detection accuracy and best preserves text quality while being compatible with API-based and open-source LLMs through post-generation processing.
- User-specific key-driven attribution that prevents sensitive data retention while supporting large-scale users in practical deployment.
- Multilingual generalization through domain-adaptive *unit* selection, validated on English, Chinese, code generation and instruction following with up to 99.7% detection accuracy.

To our knowledge, SAEMARK represents a novel, private watermarking solution capable of operating in regulated environments—its zero-knowledge design complies with data protection laws while providing robust attribution capabilities. By decoupling watermarking from generation, we provide a practical path toward accountable LLM deployment in open, multilingual ecosystems.

2 Preliminaries

2.1 Related Work

LLM watermarking is a technique to embed special patterns into the output of LLMs, and has traditionally been used to identify LLM generated text from human-written text (Jawahar et al., 2020). Different from *post-hoc detection* methods (Zellers et al., 2019) that analyze statistical patterns in existing text, language model watermarking aims to embed detectable signatures during generation (Kirchenbauer et al., 2023).

Early watermarking methods used syntactic transformations (Atallah et al., 2002) or machine translation artifacts (Venugopal et al., 2011), but with the rise of LLMs, strategies like **token-level manipulation** and **distribution preservation** became prominent. KGW (Kirchenbauer et al., 2023) introduced watermarking by biasing token selection with hashed context windows, while methods like SemStamp (Hou et al., 2023) and PersonaMark (Zhang et al., 2024) modify sentence-level patterns while maintaining distribution, though they face challenges with content that cannot be easily segmented into sentences. These approaches often rely on hypothesis testing, such as z-tests, to compare observed statistics with expected distributions under the null hypothesis of unwatermarked text (Kirchenbauer et al., 2023).

Existing approaches exhibit several limitations. Most methods *compromise generation quality* through direct manipulation of token probabilities (Kirchenbauer et al., 2023) or syntactic modifications (Atallah et al., 2002). The challenge of *language and domain generalization* remains largely unaddressed, with current techniques primarily optimized for English text and struggling to handle multilingual content or specialized domains like code (Lee et al., 2024). Notably, PersonaMark (Zhang et al., 2024) represents the only attempt at personalized watermarking to date, but its reliance on English-specific syntactic patterns and closed-source implementation makes its scalability and cross-lingual capabilities difficult to verify. Furthermore, it lacks critical analysis of performance scaling with increasing user counts—a crucial consideration for real-world deployment where systems must support thousands of users while maintaining detection accuracy. The broader challenge of enabling *robust user-specific attribution* while preserving privacy thus remains fundamentally unsolved in the field.

2.2 Sparse Autoencoders

Sparse Autoencoders (SAEs) are pre-trained interpretability tools that decompose LLM activations into human-understandable features (Bricken et al., 2023). For a given base model $\mathcal{M}$ and layer l, an

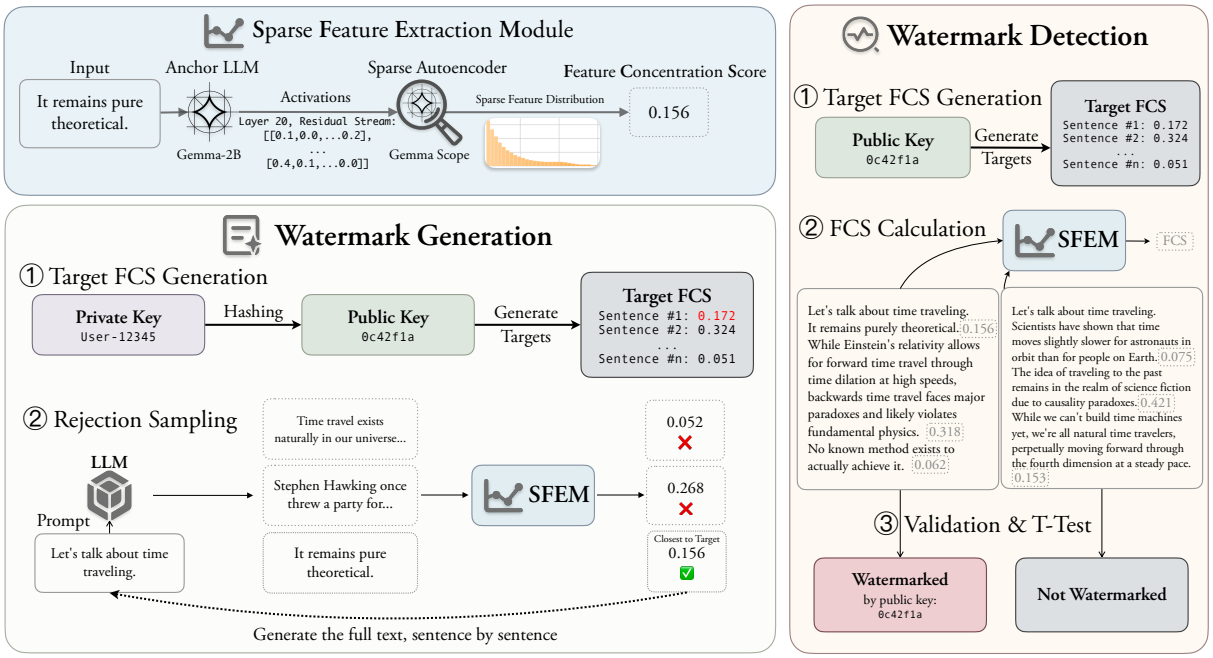

Figure 1: An overview of SAEMARK.

SAE processes hidden states $\mathbf{h}_t$ at position t as:

$$\mathbf{f}_t = \text{SAE}_l(\mathbf{h}_t) \tag{1}$$

where $\mathbf{f}_t \in \mathbb{R}^m$ is a sparse vector (typically $m \gg \dim(\mathbf{h}_t)$) with <5% active features. The SAE is trained through two objectives: 1) reconstruct original activations, and 2) enforce feature sparsity via L_1 regularization:

$$\mathcal{L} = \underbrace{\|\mathbf{h}_t - \text{Dec}(\mathbf{f}_t)\|^2}_{L_{rec}} + \lambda \underbrace{\|\mathbf{f}_t\|_1}_{L_{sparse}} \tag{2}$$

This training produces features that correspond to interpretable concepts (Bricken et al., 2023; Templeton et al., 2024). For instance, SAEs applied to Gemma 2B (Team et al., 2024)'s final transformer layer reveal features for "Python function definitions" and "Chinese proper nouns" (Lieberum et al., 2024).

Our watermarking leverages three key properties of pre-trained SAE features. First, **layer-specific** patterns capture distinct behaviors from different model layers. Second, **multilingual** activation allows the same features to fire for equivalent concepts across languages. Third, **sparsity** enables efficient analysis through few active features per token. These properties support language-agnostic watermark signatures through feature activation histograms:

$$\phi(\mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_t \mathbf{f}_t \odot \mathbf{m}_{pk} \tag{3}$$

where $\mathbf{m}_{pk}$ masks key-specific features determined by public key pk. The histogram $\phi(\mathbf{y})$ serves as the verifiable watermark signature, enabling detection through distribution alignment (Sec 3).

2.3 Task Definition

Personalized watermarking for LLMs establishes a cryptographic binding between generated text and user identity through two core mechanisms:

Generation Given a base language model $\mathcal{M}$, secret key $sk \in \mathcal{SK}$, and input prompt $\mathbf{x}$, the watermarking algorithm produces text $\mathbf{y}$ such that:

$$\mathbf{y} = \text{Mark}(\mathcal{M}, \mathbf{x}, sk) \tag{4}$$

where Mark operates through post-processing or sampling algorithms of $\mathcal{M}$'s outputs, without modifying $\mathcal{M}$'s parameters or generation rules.

Detection For any text $\mathbf{y}'$ and public key $pk \in \mathcal{PK}$, the detection algorithm determines:

$$\text{Detect}(pk, \mathbf{y}') \to \{0, 1\} \tag{5}$$

where 1 indicates $\mathbf{y}'$ contains a watermark traceable to pk, with verification requiring only public information.

The scheme must satisfy three fundamental properties. First, *key privacy* ensures computational infeasibility to derive sk from pk or watermarked texts. Second, *public verifiability* allows anyone to run Detect without secret data. Third, *collusion*

resistance prevents combining multiple pk's to remove or forge watermarks.

Our approach implements Mark through rejection sampling over $\mathcal{M}$'s natural outputs, selecting candidates whose hidden feature distributions align with sk-derived targets. The Detect function employs statistical hypothesis testing on these features using pk, as formalized in section 3. This design decouples watermarking from decoding algorithms, preserving $\mathcal{M}$'s output quality while enabling user-specific attribution.

3 Methodology

Our approach builds on an observation: *LLM outputs exhibit distinct patterns in their internal representations that can be captured through a fixed sparse autoencoder (SAE). These patterns, while preserving the semantic content of the text, provide a natural basis for embedding detectable signatures without compromising generation quality.*

The key insight lies in manipulating the *sparse feature concentration* of text, through sampling of generated outputs, rather than directly modifying models' weights or decoding process. We employ a compact Anchor LLM alongside its pretrained SAE to extract these sparse features from texts produced by the target LLM requiring watermarking. This approach enables watermarking that is both robust and compatible with black-box LLMs, while supporting user-specific personalized attribution.

3.1 Sparse Features as Watermark Basis

Given a token sequence T, let $\mathbf{a}_t \in \mathbb{R}^{d_{\text{model}}}$ denote the original activation vector at token position t in a specific layer of the Anchor LLM (where $d_{\text{model}} = 2048$ for Gemma-2B). The SAE with hidden dimension $d_{\text{sae}} = 16384$ produces sparse feature distribution:

$$\phi_t = \text{JumpReLU}(\mathbf{W}_e \mathbf{a}_t + \mathbf{b}_e) \in \mathbb{R}^{d_{\text{sae}}} \quad (6)$$

where $\mathbf{W}_e$ is the encoder weight matrix and JumpReLU zeros out all but the top feature activations.

After ϕ_t is calculated, we can obtain the set of indices corresponding to the most significant feature for each token by applying a masking operation and finding the argmax:

$$S = \{\text{argmax}_i(\phi_t \odot \mathbf{m})_i \mid t = 1, 2, \ldots, n\} \quad (7)$$

where $\mathbf{m}$ is a mask that excludes background frequent features (occurring in more than 60% of samples) and i denotes the index of ϕ_t, ranging from $[0, 16384)$.

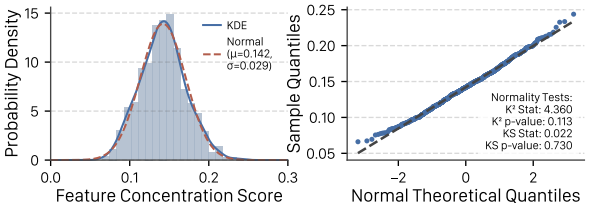

Figure 2: **Distribution analysis of FCS.** FCS distribution with density estimation (left) and Q-Q plot (right), with statistical tests supporting normality.

With S, the Feature Concentration Score (FCS) can be computed as follows:

$$\text{FCS}(T) = \frac{\sum_{t=1}^{n} \sum_{i \in S} \phi_{t,i}}{\sum_{t=1}^{n} \|\phi_t\|_1} \quad (8)$$

The FCS quantifies how sharply these activations focus on semantically salient features. More details about FCS calculation are provided in Appendix C.

Our empirical analysis of FCS distribution on 1000 LLM generated sentences on C4 dataset, shown in Figure 2, reveals that machine-generated sequences naturally follow a normal distribution (validated by D'Agostino's K² test, p=0.113, and Kolmogorov-Smirnov test, p=0.730). This well-behaved statistical property suggests that FCS varies naturally across different generations while maintaining consistent distributional characteristics, making FCS an effective measurement for watermarking through rejection sampling.

Algorithm 1: Watermark Generation

Input: Prompt c, private key k_{priv}, LLM G, SAE θ, units M, attempts K, hyperparameters μ, σ
Output: Watermarked text x^*
for *attempt* $\leftarrow 1$ **to** K **do**
  $x^* \leftarrow c$ // Initialize with prompt
  $k_{\text{pub}} \leftarrow \text{Hash}(k_{\text{priv}})$ // Generate public key
  $\{\tau_i\}_{i=1}^{M} \leftarrow \text{GenerateTargetFCS}(k_{\text{pub}}, \mu, \sigma, M)$ // Target sequence
  $\{\gamma_i\}_{i=1}^{M} \leftarrow \emptyset$ // Store achieved FCS values
  for $i \leftarrow 1$ **to** M **do**
    $\mathcal{X} \leftarrow \text{GenerateCandidates}(G, x^*, N = 50)$
    $x_{\text{best}} \leftarrow \text{argmin}_{x \in \mathcal{X}} |\text{ComputeFCS}(\theta(x)) - \tau_i|$
    $x^* \leftarrow x^* \oplus x_{\text{best}}$ // Append best candidate
    $\gamma_i \leftarrow \text{ComputeFCS}(\theta(x_{\text{best}}))$
  end
  if CheckAlignment($\{\tau_i\}, \{\gamma_i\}$) **then**
    return x^* // Return if conditions met
  end
end
return x^* // Return best attempt

3.2 Watermark Generation

Our watermarking process operates on textual *units* u_i (sentences for natural language, code segments for programming). Given hyperparameters μ, σ and private key k_{priv}, we:

1. Generate public key $k_{\text{pub}} = \text{Hash}(k_{\text{priv}})$. 2. For each *unit*, generate target FCS τ_i using k_{pub} as random seed. 3. Sample N candidates $\{x_j\}_{j=1}^{N}$ from LLM G. 4. Select $x_i^* = \text{argmin}_{x_j} |\gamma(x_j) - \tau_i|$ where γ computes FCS.

After generating M *units*, verify global alignment between target sequence $\{\tau_i\}_{i=1}^{M}$ and observed $\{\gamma_i\}_{i=1}^{M}$:

$$\begin{aligned} &\text{Range Similarity:} && R_{min} < \frac{\gamma_{\max} - \gamma_{\min}}{\tau_{\max} - \tau_{\min}} < R_{max} \\ &\text{Overlap Ratio:} && \frac{|\{i : \tau_i \in [\gamma_{\min}, \gamma_{\max}]\}|}{M} \geq O_{min} \end{aligned} \tag{9}$$

where $\gamma_{\max} = \max_i \gamma_i$, $\gamma_{\min} = \min_i \gamma_i$, and analogously for τ. We set default values for hyperparameters $R_{min} = 0.95$, $R_{max} = 1.05$, $O_{min} = 0.95$. The process repeats up to K attempts if conditions fail.

Let target FCS $\tau \in [\mu - 2\sigma, \mu + 2\sigma]$ and N candidates with FCS values $\gamma_j \sim \mathcal{N}(\mu, \sigma^2)$. For tolerance k, the probability of finding at least one candidate within $[(1-k)\tau, (1+k)\tau]$ satisfies:

$$\mathbb{P}(\exists j : |\gamma_j - \tau| \leq k\tau) \geq 1 - (1 - p_{\min})^N \tag{10}$$

where $p_{\min} = \Phi(\frac{(1+k)(\mu+2\sigma)-\mu}{\sigma}) - \Phi(\frac{(1-k)(\mu+2\sigma)-\mu}{\sigma})$ is the worst-case success probability.

Consider worst-case $\tau = \mu + 2\sigma$. For a single candidate:

$$\begin{aligned} p_{\min} &= \mathbb{P}((1-k)\tau \leq \gamma_j \leq (1+k)\tau) \\ &= \Phi\left(\tfrac{(1+k)(\mu+2\sigma)-\mu}{\sigma}\right) - \Phi\left(\tfrac{(1-k)(\mu+2\sigma)-\mu}{\sigma}\right) \\ &= \Phi(2(1+k) + k\mu/\sigma) - \Phi(2(1-k) - k\mu/\sigma) \end{aligned}$$

where Φ is the standard normal CDF. For N i.i.d. candidates, probability of at least one success is $1-(1-p_{\min})^N$. With empirically observed $\mu = 0.142$, $\sigma = 0.029$, and practical parameters $k = 0.1$ and $N = 50$, this gives success probability > 0.99 while using a smaller $N = 10$ would still yield > 0.61 success probability.

This bound ensures efficient watermarking—even with tight tolerance $k = 0.1$, using $N = 50$ candidates achieves 99.17% success rate per *unit*, while smaller batch sizes like $N = 20$ still maintain 85.32% success probability.

3.3 Watermark Detection

The detection process mirrors generation but focuses on statistical validation of FCS patterns. Given a text x and a set of public keys $\mathcal{K}_{\text{pub}}$, we first segment the input into domain-appropriate *units* and compute the observed FCS sequence $\{\gamma_j\}$ for each segment. For each candidate key $k_i \in \mathcal{K}_{\text{pub}}$, we generate the expected FCS sequence $\{\tau_j\}$ using the same process as in watermark generation.

Statistical validation combines sequence alignment (using criteria from generation) and significance testing. We employ Student's t-test to verify correlation between observed and expected FCS sequences. Under the null hypothesis $H_0 : \beta \leq 0$ for regression coefficient β in $\gamma_j = \beta\tau_j + \epsilon$, we consider a text watermarked if we can reject H_0 at significance level α (typically 0.05) with $t > t_{\alpha/2}$ and $p < \alpha$, while simultaneously satisfying the alignment conditions. This dual validation ensures robustness against both random matches and tampering attempts.

Algorithm 2: Watermark Detection

Input: Text x, public keys $\mathcal{K}_{\text{pub}}$, SAE θ, significance level α, hyperparameters μ, σ
Output: Detection result $d \in \mathcal{K}_{\text{pub}} \cup \{\emptyset\}$

```
{γ_j} ← [ComputeFCS(θ(s)) ∀s ∈ SegmentByDomain(x)]
D ← ∅ // Initialize detection results
foreach k_i ∈ K_pub do
    {τ_j} ← GenerateTargetFCS(k_i, μ, σ, |γ_j|) if CheckAlignment({τ_j}, {γ_j}) then
        t, p ← StudentTTest({γ_j}, {τ_j}) if t > t_{α/2} ∧ p < α then
            D ← D ∪ {(k_i, t)}
        end
    end
end
return argmax_{(k_i,t_i)∈D} t_i if D ≠ ∅ else ∅
```

4 Experiments

Our experiments aims to address key questions: (1) How *effective and accurate* is our method compared to existing methods? (2) How does our personalized watermark *scale with exponentially increasing user count*? (3) How *robust* is our method against *adversarial attacks*? (4) How *each component* contributes to the overall performance?

Table 1: **Dataset Statistics.** Characteristics of the multilingual benchmarks used in evaluation.

	C4 (Raffel et al., 2020)	**LCSTS** (Hu et al., 2015)	**MBPP** (Austin et al., 2021)	**PandaLM** (Wang et al., 2023)
# Samples	500	500	257†	169
Language	English	Chinese	Python	English
Task Type	Completion	Summarization	Code Generation	Instruction Following

†From test split of sanitized version of MBPP.

Table 2: **Main Results on Text Watermarking Detection.** We report detection performance at 1% false positive rate (FPR). Best results are in **bold** and second-best are underlined. All metrics are reported as percentages (%).

Method	C4 (English, 2020)			LCSTS (Chinese, 2015)			MBPP (Code, 2021)			PandaLM (Instruction, 2023)			
	Acc.↑	Rec.↑	F1↑	Acc.↑	Rec.↑	F1↑	Acc.↑	Rec.↑	F1↑	Quality↑	Acc.↑	Rec.↑	F1↑
KGW (2023)	99.2	99.6	99.2	99.1	98.8	99.1	65.4	31.9	48.0	41.5	**89.9**	**80.4**	**88.8**
EXP (2022)	99.5	99.6	99.5	**99.3**	99.4	**99.3**	57.8	16.7	28.4	23.2	79.3	59.4	74.2
UPV (2023a)	86.0	72.0	83.7	90.5	91.0	90.5	51.6	3.1	6.0	36.0	54.0	8.0	14.8
Unigram (2023)	98.8	98.6	98.8	98.2	97.0	98.2	65.4	31.9	48.0	35.3	53.3	7.2	13.4
DIP (2023)	96.0	92.6	95.9	97.7	96.2	97.7	60.7	22.6	36.5	36.5	81.5	63.8	77.5
Unbiased (2023b)	96.7	94.4	96.6	97.8	96.4	97.8	64.0	29.2	44.8	40.2	74.3	49.3	65.7
SynthID (2024)	98.2	97.2	98.2	97.6	96.2	97.6	62.5	26.1	41.0	36.0	81.2	63.0	77.0
SWEET (2024)	99.6	99.6	99.6	50.0	0.0	0.0	72.4	45.9	62.4	47.2	87.7	76.8	86.2
SAEMARK (OURS)	**99.7**	**99.8**	**99.7**	99.2	**99.6**	99.2	**74.5**	**50.2**	**66.3**	**67.6**	86.6	73.9	84.6

4.1 Experimental Setup

Datasets We evaluate on 4 diverse datasets as shown in Table 1. We select C4 and MBPP as they are widely adopted benchmarks for watermarking tasks, and include LCSTS as a representative Chinese dataset to validate multilingual capabilities. Together, these datasets cover both natural languages (English, Chinese) and programming language (Python), enabling comprehensive evaluation across different domains.

Metrics We report Accuracy, Recall and F1 values at 1% False Positive Rate (FPR) for all datasets. Additionally, we include PandaLM, which is an instruction following dataset specifically designed with LLM-as-a-judge to evaluate the quality of generated content. We use `GPT-4o` to perform pairwise comparison on the quality of watermarked outputs and un-watermarked outputs for all methods and report as Quality metric in our results.

Baselines We conduct extensive comparisons using the MarkLLM (Pan et al., 2024) toolkit, evaluating against 8 recent watermarking methods. Due to space constraints, detailed introduction of each baseline can be found in Appendix B.

Implementation We use Gemma-2B (Team et al., 2024) as our Anchor LLM and extract activations from layer 20, with Gemma Scope (Lieberum et al., 2024) serving as our SAE configured for 16,384 sparse features. For watermark application, we employ `Qwen-2.5-7B-Instruct` (Yang et al., 2024) as the backbone model. Additional hyperparameters and implementation details are provided in Appendix A.

4.2 Multilingual Detection Performance

Table 2 shows the watermark detection performance of different methods on 4 datasets, we report the Accuracy, F1 score, and AUC score given 1% False Positive Rate (FPR). For PandaLM, we report the percentage of the watermarked outputs that are not judged as degraded compared to the un-watermarked outputs by `GPT-4o` to evaluate the impact of watermarking on the quality of LLM generated content.

Our evaluation reveals two critical findings: (1) SAEMARK *consistently achieves superior detection accuracy while preserving text quality and supporting personalized attribution*, and (2) existing methods exhibit *fundamental limitations in code watermarking* that our approach alleviates through domain-adaptive feature alignment.

As Table 2 demonstrates, SAEMARK establishes new benchmarks with **99.7% F1** on English (C4) and **99.2%** on Chinese (LCSTS), outperforming baselines even on competitive fields where the detection accuracy is close to 100%. Notably, we surpass SWEET—a specialized code watermarker—by **3.9% F1** on MBPP despite our general-purpose design. This cross-domain superiority stems from SAE features that capture language-agnostic semantic patterns rather than surface-level token distributions.

While other methods struggle with language-specific performance cliffs, such as SWEET's 0% recall on Chinese, our SAEMark achieves balanced accuracy by using syntax-invariant SAE features. On the MBPP code generation dataset, where the average F1 score drops by 51% compared to text domains due to programming languages' lower entropy and rigid syntax, SAEMark still excels with a 66.3% F1 score, significantly outperforming alternatives like KGW at 48.0% and EXP at 28.4%. This highlights SAEMark's effectiveness in watermark embedding even within the constrained space of syntactically rigid code.

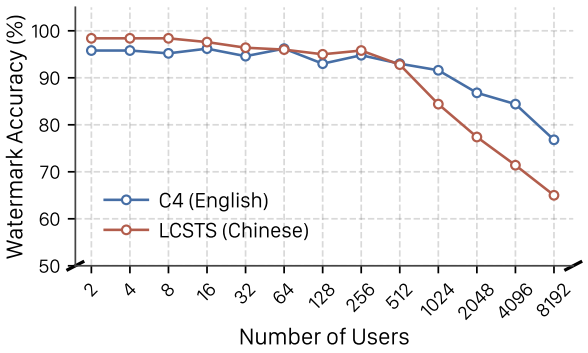

Figure 3: **Scaling analysis of SAEMark.** Watermark accuracy remains above **90%** for both C4 and LCSTS up to 512 users. Accuracy drops with the exponential increase of users given only 10 sentences for detection.

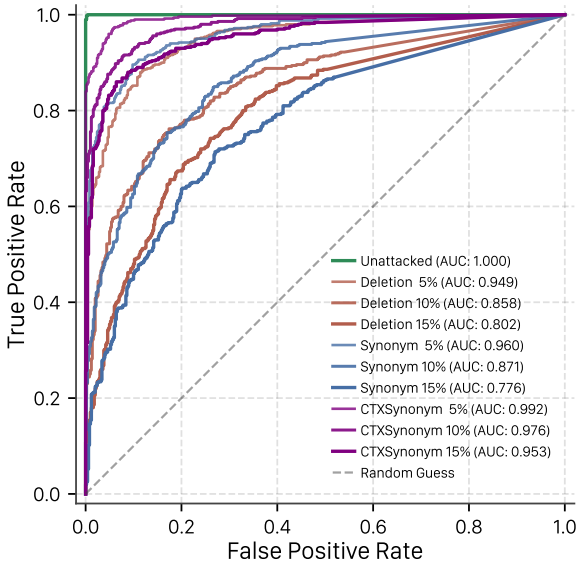

Figure 4: **Comparison of attacks on SAEMark.** ROC curves showing robust performance against three attack types with varying intensities (5%, 10%, 15%).

For PandaLM dataset, SAEMark shows its strength by achieving watermark quality score of 67.6. This score reflects SAEMark's capability to embed watermarks while preserving the quality of the original text effectively, suggesting that SAEMark manages to maintain a good balance between watermark detectability and text quality preservation. It indicates our approach's potential in offering a reliable solution that respects the integrity of the textual content, making it a promising option for applications where both watermarking effectiveness and text quality are critical considerations.

4.3 User-Specific Attribution

Real-world AI systems require watermarking that scales to many users while preserving privacy—storing user-generated content raises ethical and regulatory concerns. SAEMARK addresses this through private-public key binding that enables attribution without storing sensitive data.

Figure 3 shows SAEMARK maintains **>90% accuracy** for English and Chinese with up to 1024 different users, using only 10 generated sentences per detection. At 8192 users, accuracy remains practically viable at 75% for English and 65% for Chinese, outperforming random guessing by four orders of magnitude.

Unlike systems that store user texts for attribution, SAEMARK relies solely on public keys derived from private keys for detection. This approach ensures providers never store original user content while maintaining attribution capability.

Accuracy degrades gracefully despite exponentially increasing collision risk, thanks to our high-dimensional feature space ($\mathbb{R}^{16384}$). The 7% accuracy drop per order-of-magnitude user growth stems from widening feature concentration intervals, mitigated through dynamic range and overlap constraints (which is discussed in subsection 4.5). Practical deployments can tune detection strictness—requiring 8/10 matching sentences boosts accuracy to 99% at 1,024 users with minimal false attributions.

These results position SAEMARK as the first ethical watermarking solution suitable for regulated environments. By decoupling attribution from content storage, our method enables responsible auditing in public AI systems while respecting user privacy. We are first to carry out such experiments and we think this is a field that safe, ethical and responsible AI systems should take careful consideration about.

4.4 Adversarial Robustness Evaluation

Adversarial attacks pose significant threats to watermarking systems, and we evaluate SAEMARK against three different types of attacks: word deletion attacks (Deletion), basic synonym substitution attacks (Synonym), and context-aware synonym substitution (CTXSynonym), which represents one of the more sophisticated attacks. Our evaluation includes various substitution rates for a comprehensive assessment. These rates allow us to understand how well our method performs under differing levels of adversarial pressure.

As shown in Figure 4, our method demonstrates notable resilience across adversarial conditions.

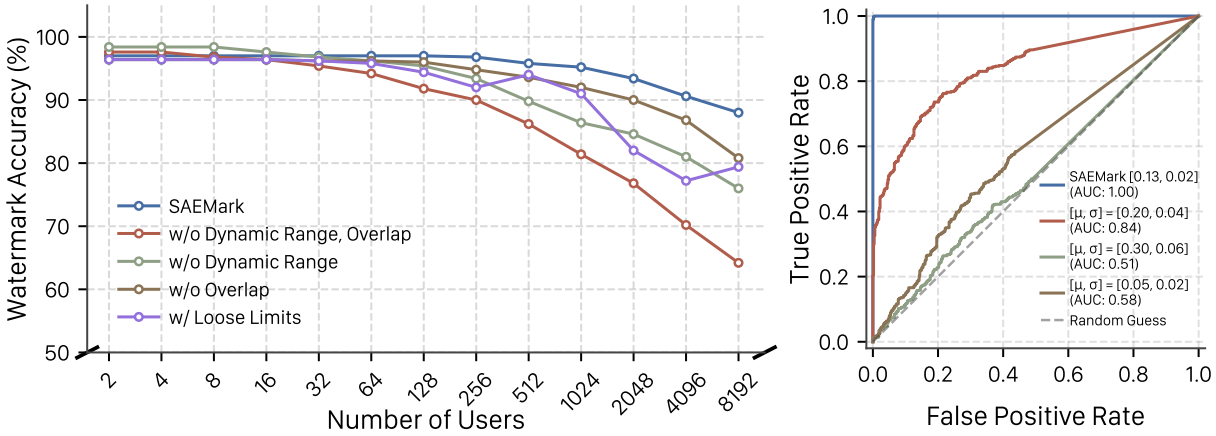

Figure 5: **Ablation studies on SAEMark.** Left: Scaling analysis of watermark accuracy with increasing user count under different constraint settings. Right: ROC curves for different target FCS distribution parameters (μ, σ). The default setting [0.13, 0.02] achieves optimal detection performance compared to alternative configurations.

For word deletion attacks, it maintains an AUC above **0.80** at a 15% deletion rate. In the case of synonym substitution attacks, our method achieves an AUC greater than **0.78** at 15% substitution.

Particularly noteworthy is the performance under context-aware synonym substitution. Our method achieves AUC of **0.992**, **0.976**, and **0.953** at 5%, 10%, and 15% substitution, respectively. These results indicate the robustness of SAE features in mitigating the impact of highly effective semantic-preserving edits.

4.5 Ablation Studies

We conduct comprehensive ablation studies validate effectiveness of core components in our method: (1) the necessity of dynamic range and overlap constraints for multi-user scaling, and (2) the impact of FCS distribution parameters on detection performance. Figure 5 illustrates these analyses. Due to space limits, we also perform ablation studies on background frequent feature masking but report the results in Appendix D.

The left subplot demonstrates how our constraints enable reliable scaling to large user bases. Removing either dynamic range constraint or overlap constraint causes significant accuracy degradation beyond 1,024 users. Without both mechanisms ("w/o Dynamic Range, Overlap"), the performance drops significantly to 64.2% at 8,192 users, which is notably lower than SAEMARK's accuracy of 88%. Specifically, the accuracy also drops to nearly 80% among 8,192 users, with loosen limits of $R_{\min} = 0.8$, $R_{\max} = 1.2$, and $O_{\min} = 0.8$. This validates our theoretical analysis that controlled feature concentration intervals are crucial for maintaining distinguishable watermarks at scale.

The right subplot examines detection sensitivity to FCS distribution parameters. Our default setting achieves optimal ROC performance, while alternative configurations show marked degradation. Notably, increasing the mean and variance [$\mu = 0.30$, $\sigma = 0.06$] leads to substantial performance loss, likely due to feature saturation in higher concentration regions. Conversely, an overly restrictive range [$\mu = 0.05$, $\sigma = 0.02$] limits the available feature space, reducing watermark capacity.

These results empirically validate two fundamental aspects of SAEMARK: (1) Dynamic range and overlap constraints are essential mechanisms for scalability over large amount of users, and (2) The theoretically derived optimal FCS distribution indeed maximizes detection performance, confirming our mathematical analysis.

5 Conclusion

SAEMARK introduces a fundamental shift in AI generated content detection through manipulating sparse autoencoder features. Our method achieves consistently high detection accuracy across languages and code while preserving text quality, demonstrating that model interpretability tools can be repurposed for ethical AI systems. By mapping user keys to activation patterns rather than surface features, we enable scalable attribution without content storage—supporting identifying thousands of users with high accuracy using a few sentences. This bridges the gap between technical watermarking and practical deployment constraints, offering a privacy-preserving solution that respects linguistic diversity while meeting regulatory requirements.

Limitations

We also found some limitations with our current approach. First, the method's effectiveness depends on SAE feature quality. But be noted that this does not affect the applicability of our algorithm on the base LLMs, since we only apply SAEs on the Anchor LLM and require only access to the output texts from the base LLM, and we have a lot of pretrained SAEs from the open-source community that exhibit strong performance in interpreting model outputs. Second, detection watermarks effectively requires open-ended generation tasks, making attribution challenging for very short outputs like multiple-choice problems that only contain option keys. However this is a universal challenge for all watermarking algorithms, since short texts inevitably contains less information and less space to inject additional signatures.

These constraints reflect tradeoffs in privacy-preserving watermarking. Future work could explore dynamic candidate pruning to address these limitations. Nevertheless, our experiments across 4 benchmarks suggest these constraints pose manageable practical impacts compared to the system's ethical advantages.

References

S. Aaronson and H. Kirchner. 2022. Watermarking gpt outputs. https://www.scottaaronson.com/talks/watermark.ppt.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Mikhail J Atallah, Victor Raskin, Christian F Hempelmann, Mercan Karahan, Radu Sion, Umut Topkara, and Katrina E Triezenberg. 2002. Natural language watermarking and tamperproofing. In *International workshop on information hiding*, pages 196–212. Springer.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Zhongze Cai, Shang Liu, Hanzhao Wang, Huaiyang Zhong, and Xiaocheng Li. 2024. Towards better statistical understanding of watermarking llms. *arXiv preprint arXiv:2403.13027*.

Liang Chen, Yatao Bian, Yang Deng, Shuaiyi Li, Bingzhe Wu, Peilin Zhao, and Kam-fai Wong. 2023. X-mark: Towards lossless watermarking through lexical redundancy. *arXiv preprint arXiv:2311.09832*.

Lingjie Chen, Ruizhong Qiu, Siyu Yuan, Zhining Liu, Tianxin Wei, Hyunsik Yoo, Zhichen Zeng, Deqing Yang, and Hanghang Tong. 2024. Wapiti: A watermark for finetuned open-source llms. *arXiv preprint arXiv:2410.06467*.

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

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/toy_model/index.html.

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

Varun Godbole, George E. Dahl, Justin Gilmer, Christopher J. Shallue, and Zachary Nado. 2023. Deep learning tuning playbook. Version 1.0.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT Press.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. 2023. On the learnability of watermarks for language models. *arXiv preprint arXiv:2312.04469*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, et al. 2024. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *arXiv preprint arXiv:2410.20526*.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554.

Lynette Hirschman and Robert Gaizauskas. 2001. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300.

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

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. *arXiv preprint arXiv:1506.05865*.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023a. Radar: Robust ai-text detection via adversarial learning. *Advances in Neural Information Processing Systems*, 36:15077–15095.

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

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

Robert Huben, Hoagy Cunningham, Logan Riggs, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Mingjia Huo, Sai Ashish Somayajula, Youwei Liang, Ruisi Zhang, Farinaz Koushanfar, and Pengtao Xie. 2024. Token-specific watermarking with enhanced detectability and semantic coherence for large language models. *arXiv preprint arXiv:2402.18059*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks VS Lakshmanan. 2020. Automatic detection of machine generated text: A critical survey. *arXiv preprint arXiv:2011.01314*.

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

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. In *Advances in Neural Information Processing Systems*, volume 36, pages 27469–27500. Curran Associates, Inc.

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2024. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4890–4911, Bangkok, Thailand. Association for Computational Linguistics.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. 2024. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*.

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

Aiwei Liu, Leyi Pan, Xuming Hu, Shu'ang Li, Lijie Wen, Irwin King, and Philip S Yu. 2023b. A private watermark for large language models. *arXiv preprint arXiv:2307.16230*.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024. An entropy-based text watermarking detection method. *arXiv preprint arXiv:2403.13485*.

Yiyang Luo, Ke Lin, and Chao Gu. 2024. Lost in overlap: Exploring watermark collision in llms. *arXiv preprint arXiv:2403.10020*.

Minjia Mao, Dongjun Wei, Zeyu Chen, Xiao Fang, and Michael Chau. 2024. A watermark for low-entropy and unbiased generation in large language models. *arXiv preprint arXiv:2405.14604*.

Piotr Molenda, Adian Liusie, and Mark JF Gales. 2024. Waterjudge: Quality-detection trade-off when watermarking large language models. *arXiv preprint arXiv:2403.19548*.

Alexander Nemecek, Yuzhou Jiang, and Erman Ayday. 2024. Topic-based watermarks for llm-generated text. *arXiv preprint arXiv:2404.02138*.

Andrew Ng et al. 2011. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024. o1 system card. https://cdn.openai.com/o1-system-card.pdf. Accessed: Dec 9, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, et al. 2024. Markllm: An open-source toolkit for llm watermarking. *arXiv preprint arXiv:2405.10051*.

Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. 2024. Attacking llm watermarks by exploiting their strengths. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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

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

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Lewis Tunstall, Leandro Von Werra, and Thomas Wolf. 2022. *Natural language processing with transformers*. " O'Reilly Media, Inc.".

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ashish Venugopal, Jakob Uszkoreit, David Talbot, Franz Josef Och, and Juri Ganitkevitch. 2011. Watermarking the outputs of structured prediction with an application in statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1363–1372.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Yidong Wang, Zhuohao Yu, Jindong Wang, Qiang Heng, Hao Chen, Wei Ye, Rui Xie, Xing Xie, and Shikun Zhang. 2024. Exploring vision-language models for imbalanced learning. *International Journal of Computer Vision*, 132(1):224–237.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, et al. 2023. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

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

Rui Xie, Zhengran Zeng, Zhuohao Yu, Chang Gao, Shikun Zhang, and Wei Ye. 2024. Codeshell technical report. *arXiv preprint arXiv:2403.15747*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Linyi Yang, Shuibai Zhang, Libo Qin, Yafu Li, Yidong Wang, Hanmeng Liu, Jindong Wang, Xing Xie, and Yue Zhang. 2022. Glue-x: Evaluating natural language understanding models from an out-of-distribution generalization perspective. *arXiv preprint arXiv:2211.08073*.

Linyi Yang, Shuibai Zhang, Zhuohao Yu, Guangsheng Bao, Yidong Wang, Jindong Wang, Ruochen Xu, Wei Ye, Xing Xie, Weizhu Chen, et al. 2023. Supervised knowledge makes large language models better in-context learners. *arXiv preprint arXiv:2312.15918*.

Wenjin Yao, Yidong Wang, Zhuohao Yu, Rui Xie, Shikun Zhang, and Wei Ye. 2024. Pure: Aligning llm via pluggable query reformulation for enhanced helpfulness. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8721–8744.

Zhuohao Yu, Chang Gao, Wenjin Yao, Yidong Wang, Wei Ye, Jindong Wang, Xing Xie, Yue Zhang, and Shikun Zhang. 2024a. Kieval: A knowledge-grounded interactive evaluation framework for large language models. *arXiv preprint arXiv:2402.15043*.

Zhuohao Yu, Chang Gao, Wenjin Yao, Yidong Wang, Zhengran Zeng, Wei Ye, Jindong Wang, Yue Zhang, and Shikun Zhang. 2024b. Freeeval: A modular framework for trustworthy and efficient evaluation of large language models. *arXiv preprint arXiv:2404.06003*.

Zhuohao Yu, Weizheng Gu, Yidong Wang, Zhengran Zeng, Jindong Wang, Wei Ye, and Shikun Zhang. 2024c. Outcome-refining process supervision for code generation. *arXiv preprint arXiv:2412.15118*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

Yuehan Zhang, Peizhuo Lv, Yinpeng Liu, Yongqiang Ma, Wei Lu, Xiaofeng Wang, Xiaozhong Liu, and Jiawei Liu. 2024. Personamark: Personalized llm watermarking for model protection and user attribution. *arXiv preprint arXiv:2409.09739*.

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

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. 2022. Distillation-resistant watermarking for model protection in nlp. *arXiv preprint arXiv:2210.03312*.

Xuandong Zhao, Chenwen Liao, Yu-Xiang Wang, and Lei Li. 2024. Efficiently identifying watermarked segments in mixed-source texts. *arXiv preprint arXiv:2410.03600*.

A Experimental Setup and Hyperparameter Details

This appendix provides a comprehensive description of the experimental setup, encompassing the hyperparameters and software configurations employed in this study.

A.1 Hyperparameters (SAEMARK)

The following hyperparameters were used for the SAEMARK:

- **Candidate Number (N):** 50. This parameter denotes the number of candidate sequences sampled from the LLM.
- **Unit Number (M):** 10. This specifies the number of discrete generation *units* produced by the model per attempt.
- **Attempt Number (K):** 15. This metric represents the maximum times that the algorithm attempts to get an alignment.

A.2 Model Configuration

The section outlines the hyperparameter by the model during generation.

- **Base Model:** `Qwen2.5-7B-Instruct`. This is the model on which the algorithm operates.
- **Sampling:** This algorithm enables the model to generate various candidates, for which the parameter do_sample is set to $True$.
- **Temperature:** This controls the randomness of the predictions by scaling the logits. The metric is set to 0.7.
- **Max New Tokens:** This specifies the maximum number of new tokens that the model can generate, which is 20 during generation.

B Introduction to baselines

KGW (Kirchenbauer et al., 2023) The Key-based Green-list Watermarking (KGW) algorithm is a modern approach for watermarking text generated by LLMs. This method builds upon the work of (Kirchenbauer et al., 2023), who introduced a watermarking scheme that divides the token set into 'red' and 'green' lists based on a secret key and previously generated tokens.

Key features of KGW include the bifurcation of the token set into 'red' and 'green' lists, the use of a random seed dependent on a secret key and hash of prior tokens, reweighting of token log-probabilities to favor green tokens, and the introduction of permutation-based reweight strategies. These elements work in concert to create an effective watermarking system that balances detectability with output quality preservation.

The approach offers a balance between watermark embedding and preservation of text quality, addressing challenges faced by previous watermarking methods.

Unigram (Zhao et al., 2023) The Unigram-Watermark and KGW algorithms, both designed for watermarking LLM-generated text, have distinct characteristics. Unigram-Watermark operates on individual tokens, using a consistent green list for each new token, while KGW employs a K-gram approach with varying green lists. Unigram-Watermark's simplicity offers enhanced robustness against editing attacks and requires minimal implementation overhead. This streamlined approach leads to potential efficiency gains in both watermark embedding and detection processes, setting it apart from the more complex K-gram nature of KGW.

SWEET (Lee et al., 2024) The Segment-Wise Entropy-based Embedding Technique (SWEET) is an innovative approach to watermarking code generated by large language models. SWEET addresses the challenge of maintaining code functionality while embedding detectable watermarks. It operates by selectively applying watermarking to high-entropy segments of the generated code, thereby preserving the overall code quality. This method significantly improves code quality preservation while outperforming baseline methods in detecting machine-generated code. SWEET achieves this by removing low-entropy segments during both the generation and detection of watermarks, effectively balancing the trade-off between detection capability and code quality degradation.

UPV (Liu et al., 2023a) The key feature of UPV is its use of separate neural networks for watermark generation and detection, addressing the limitation of shared key usage in previous methods. This separation allows for public verification without compromising the watermark's security. UPV employs shared token embedding parameters between the generation and detection networks, enabling efficient and accurate watermark detection. The

algorithm embeds small watermark signals into the LLM's logits during generation, similar to existing methods, but uniquely conceals the watermarking details in the detection process. This approach ensures high detection accuracy while maintaining computational efficiency, and significantly increases the complexity of forging the watermark, thus enhancing its security in public detection scenarios.

DIP (Wu et al., 2023) The Distribution-Preserving Watermarking (DIP) algorithm represents a significant advancement in watermarking techniques for large language models (LLMs). DIP's innovation is its ability to maintain the original token distribution of the LLM while embedding a watermark, addressing a critical limitation of previous methods. This distribution-preserving property is achieved through a novel permutation-based approach that reweights token probabilities without altering the overall distribution. DIP offers provable guarantees on distribution preservation, detectability, and resilience against text modifications. The algorithm employs a texture key generation mechanism that considers multiple previous tokens, enhancing its robustness. Notably, DIP maintains text quality comparable to the original LLM output, owing to its distribution-preserving nature.

Unbiased (Hu et al., 2023b) Unbiased watermarking and DIP watermarking are closely related concepts in the field of text watermarking for large language models (LLMs). Both approaches aim to embed watermarks while maintaining the original distribution of the LLM's output. The key distinction lies in their theoretical foundations and implementation. Unbiased watermarking ensures that the expectation of the watermarked distribution matches the original distribution, while DIP watermarking guarantees that the watermarked distribution is identical to the original for every input. In essence, unbiased watermarking can be viewed as a relaxed version of DIP watermarking. While unbiased watermarking allows for small deviations in individual instances, DIP watermarking maintains strict distribution preservation. This relationship highlights a spectrum of watermarking techniques, where unbiased methods offer a balance between practicality and distribution preservation, while DIP methods provide stronger theoretical guarantees at potentially higher computational costs.

SynthID (Dathathri et al., 2024) SynthID is an advanced watermarking method for large language models (LLMs) that builds upon previous work in generative text watermarking. The key innovation of SynthID lies in its use of Tournament sampling, which provides superior detectability compared to existing methods. This approach offers rigorous and customizable non-distortion properties, allowing for text quality preservation while maintaining effective watermarking. SynthID has been empirically validated, including through real user feedback from millions of chatbot interactions. Notably, the method introduces an algorithm to combine generative watermarking with speculative sampling, enabling efficient deployment in high-performance, large-scale production LLMs.

EXP (Aaronson and Kirchner, 2022) EXP employs a pseudorandom function $f_s()$ with a secret seed s known only to the model provider. Given previous tokens $w_1, ..., w_{t-1}$ and GPT's probability distribution $p_1, ..., p_K$ for the next token w_t, the algorithm generates real values $r_i \in [0, 1]$ using $f_s()$. EXP then selects the token i that maximizes r_i^{1/p_i}. To detect the watermark, it calculates $\sum_{t=1}^{T} \ln \frac{1}{1-r'_t}$ and compares it to a threshold. The scheme preserves the original token distribution while embedding a detectable watermark, with theoretical analysis showing distinct expected values for normal and watermarked text. The number of tokens required for reliable detection is $O(\frac{1}{\alpha^2} \log \frac{1}{\delta})$, where α is the average entropy per token and δ is the acceptable misclassification probability.

C Details of FCS Generation

This section elaborates on the methodology behind the generation of the Feature Concentration Score (FCS). The process is illustrated in Figure Figure 6, which outlines four key steps.

Extracting SAE Features for Each Token Given a token sequence T, we utilize SAE to derive an activation vector ϕ_t for each token position t. This vector, ϕ_t, embodies the representation of the token at position t with a dimensionality of 16,384.

Selecting the Most Significant Feature For every activation vector ϕ_t, our objective is to identify the most significant feature, which serves as a descriptor for the token at position t. This is achieved through applying the function $argmax_i(\phi_t \odot m)_i$, where m is a mask. The output of this function

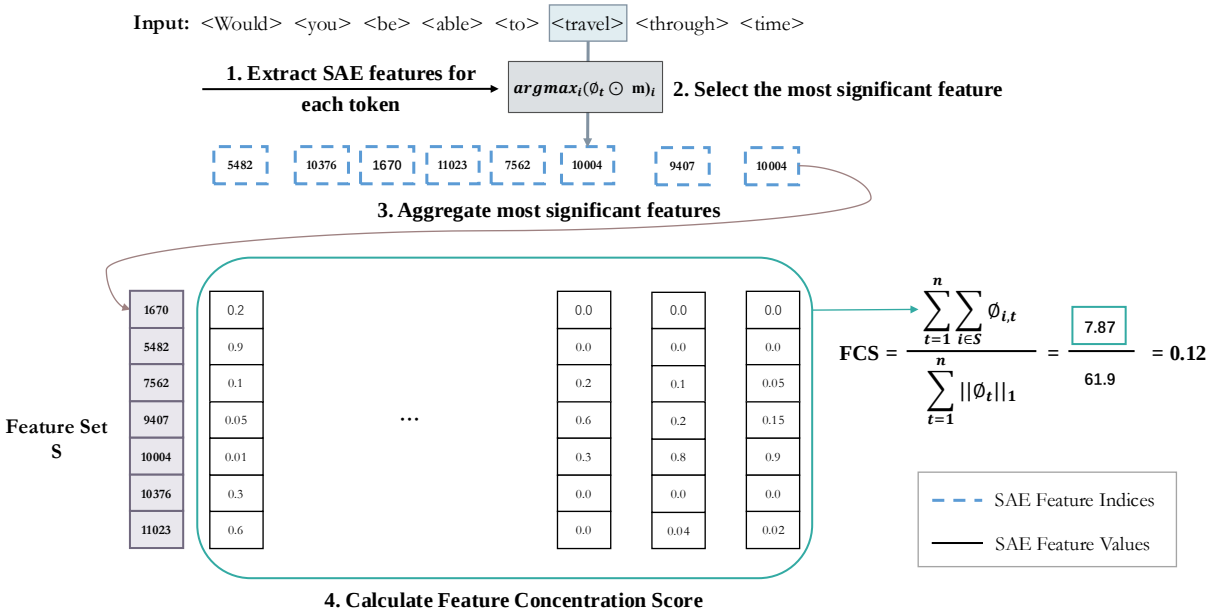

Figure 6: An example of Feature Concentration Score (FCS) calculation process.

Algorithm 3: ComputeFCS($\theta(T)$)

Input: Token sequence T, SAE θ for the entire sequence
Output: Feature Concentration Score (FCS)
$\Phi \leftarrow \theta(T)$, yielding activation vectors $\phi_1, \phi_2, ..., \phi_n$ for each token position in T;
$indices \leftarrow []$;
for $t = 1$ to n **do**
  ϕ_t is the activation vector for token at position t;
  $index \leftarrow \arg\max_i(\phi_t \odot m)_i$;
  Append $index$ to $indices$;
end
$featureSet \leftarrow set(indices)$, removing duplicates;
$featureSum \leftarrow 0$;
$totalNorm \leftarrow 0$;
for $t = 1$ to n **do**
  $tokenSignificance \leftarrow 0$;
  foreach $i \in featureSet$ **do**
    $tokenSignificance \leftarrow tokenSignificance + \phi_{t,i}$;
  end
  $featureSum \leftarrow featureSum + tokenSignificance$;
  $totalNorm \leftarrow totalNorm + ||\phi_t||_1$;
  // Accumulate significant features and norms
end
$FCS \leftarrow \frac{featureSum}{totalNorm}$;
// Calculate final FCS
return FCS;

yields the indices corresponding to the most prominent feature, denoted as "SAE Feature Indices" in Figure Figure 6.

Aggregating Most Significant Features As depicted in Figure Figure 6, each token's position t has its most significant feature. However, when summarizing the critical features of the entire sequence T, redundancies may occur. To address this, we employ a set operation to eliminate duplicate entries among the significant features, resulting in a unique collection termed as "Feature Set S".

Calculating Feature Concentration Score Upon obtaining the Feature Set S, we aim to quantify how these significant features contribute to the overall sequence T concerning SAE feature values. For each ϕ_t, we compute the sum of $\phi_{t,i}$, where i represents the index belonging to S. This aggregate score measures the contribution of significant features to individual tokens within T. Accumulating this metric across all tokens provides a global measure for the sequence.

To evaluate the total activation value of SAE features over the sequence T, we apply the $L1$ norm to each ϕ_t, obtaining the sum of absolute values for each token's feature vector. Summing these across all tokens yields the total SAE value for T. The Feature Concentration Score (FCS) is defined as the ratio of the accumulated contributions of significant features to the total SAE feature values.

The detailed steps for computing the FCS are

outlined in algorithm 3.

This score effectively captures the concentration of key features within a token sequence and is useful for applications in watermark embedding.

D Additional Experimental Results

D.1 Adversarial Robustness Evaluation

Word Deletion Attack In the main text, we conducted experiments using the "maintain content structure" version of the word deletion attack. However, the original word deletion attack involves splitting a paragraph and randomly removing words, which disrupts the structure that watermarking methods rely on, making it harder for the detection system to identify the watermark. To address this issue, we modified the attack to preserve structure while still performing word deletions. By maintaining the integrity of the structure, the attack bypasses watermark detection more effectively.

In our experimental results, we compare two versions of the word deletion attack. The "keep structure" method, represented in a darker color, shows more robust performance with higher AUC values (0.949 at $\epsilon = 0.05$ and 0.858 at $\epsilon = 0.1$). In contrast, the "not keep structure" method, shown in a lighter color, demonstrates a decline in performance, with AUC values dropping to 0.901 at $\epsilon = 0.05$ and 0.825 at $\epsilon = 0.1$. These results indicate that preserving the content structure during the attack strengthens the watermark's resistance, whereas random word deletions that disrupt the structure reduce detection accuracy.

As shown in the Figure 7, the "keep structure" method outperforms the "not keep structure" method in terms of AUC, demonstrating its effectiveness in watermark resistance.

Basic Synonym Substitution Attack Our study also examines "keeping structure" versus "not keeping structure" approaches in the context of basic synonym substitution attacks, which are less likely to disrupt the content's structural integrity.

Figure 8 shows ROC curves comparing model performance under different conditions, with the original non-structure-preserving method in lighter shades and the modified structure-preserving method in darker hues. The analysis reveals minimal differences in AUC values between the two, indicating similar model resilience to both forms of synonym substitution. Notably, the model demonstrates performance robustness that exceeds that observed in deletion attack scenarios, reflected by AUC scores that remain close to the baseline.

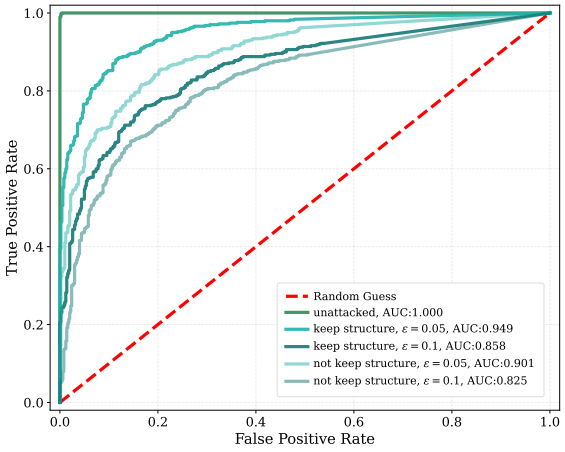

Figure 7: **Word Deletion on SAEMark** ROC curves highlighting the performance difference between "keep structure" and "not keep structure" methods under word deletion attacks with varying intensities (5%, 10%).

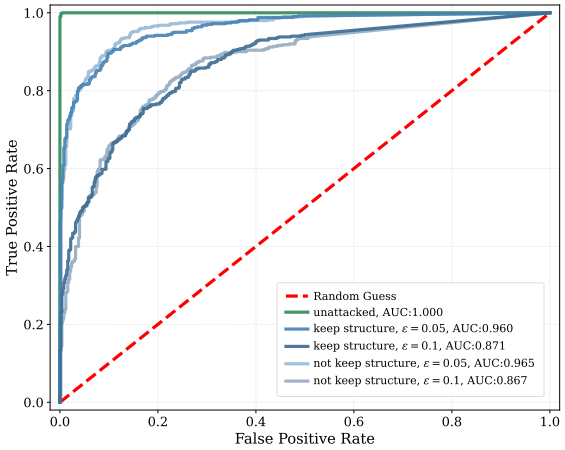

Figure 8: **Basic Synonym Substitution on SAEMark** ROC curves comparing "keep structure" and "not keep structure" methods under basic synonym substitution attacks at different intensities (5%, 10%).

Context-aware Synonym Substitution Attack Due to our algorithm's prominent performance against context-aware synonym attack. More intensities (20%, 30%, 40%, 50%) are carried upon these kinds of attacking.

The results of the context-aware watermarking method, shown in Figure 9 tested under this attack, demonstrate substantial robustness. Even with high substitution ratios—up to 50% token replacement—the AUC remains relatively high, highlighting the method's ability to maintain detection performance under significant adversarial pressure.

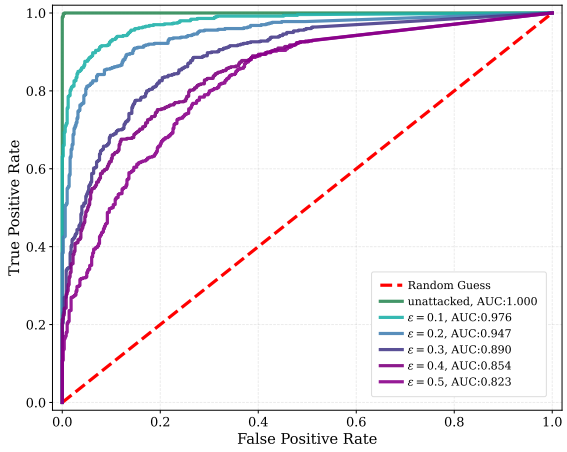

Figure 9: **Context-Aware Synonym Substitution on SAEMark** ROC curves comparing "keep structure" and "not keep structure" methods under basic synonym substitution attacks at different intensities (5%, 10%).

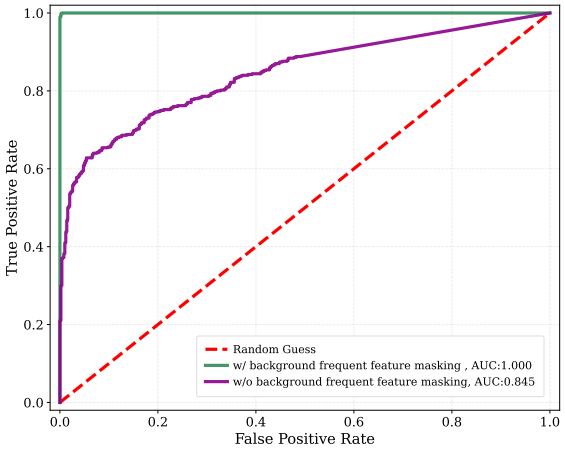

Figure 10: **Ablation on Background Frequent Feature Masking** The ROC curve compares the performance with and without background frequent feature masking.

The ROC curves further corroborate this, showing that the true positive rate remains consistently high across varying false positive levels, even as attack intensity increases. This demonstrates a well-balanced trade-off between true and false positives, ensuring reliable detection without excessive false alarms. These findings affirm that the watermarking method is both effective and robust, offering reliable protection against sophisticated attacks while maintaining strong detection accuracy.

D.2 Ablation Study on Background Frequent Features

In section 3, we utilize $\phi_t \odot m$, where m is a mask that excludes background frequent features.

In this section, we generate the Feature Concentration Score (FCS) without using m and conduct ROC experiments for further analysis. To evaluate the impact of background frequent feature masking on our model's performance, we performed an ablation study.

With background frequent feature masking in place, the model achieved an AUC of almost 1.0. Upon removing this masking, the AUC dropped to 0.85, as illustrated in Figure 10. This significant decrease demonstrates that background frequent feature masking plays a crucial role in our algorithm, emphasizing its importance for optimal performance.

D.3 Use Of AI Assistants

We employed AI assistants for two tasks: (1) generating routine code implementations and boilerplate functions, and (2) performing grammatical review and sentence-level editing of the manuscript. All AI-generated content underwent thorough manual review. The core research methodology, findings, and analysis remain entirely our own work.