FastDetectGPT: Understanding Sampling-based Zero-shot AI-generated Text Detection

Jose Garcia[1], Krystof Bobek[1], Mara Dragomir[1], Mahdi Rahimi[1], Salvador Torpes[1]

[1]University of Amsterdam

Contact: { jose.garcia.carrillo, krystof.bobek, mara.dragomir, mahdi.rahimi, salvador.baptista.torpes } @student.uva.nl

Abstract

DetectGPT introduced a zero-shot, sampling-based method for detecting machine-generated text by exploiting the local curvature of a model's log-likelihood function. Fast-DetectGPT later improved this method, matching performance at far lower cost. In this work, we reproduce Fast-DetectGPT and extend evaluation to new source models (DeepSeek R1 Distill Llama, Microsoft Phi 2, Mistral 7B, GPT-4o mini). We assess performance on three datasets: XSum, HC3, and Scottish Gaelic from XL-Sum as an under-represented language. Additionally, we propose a novel adversarial setting: the Mid-K perturbation attack, which forces the generator to avoid high-probability tokens at random positions during decoding. Our results show that Fast-DetectGPT remains highly effective and can generalize well across diverse models, datasets, and languages, but that decoding perturbations reduce its accuracy, revealing a key vulnerability. The original code and our extensions are available on GitHub: https://github.com/salvatorpes/zero-shot-llm-detection.

1 Introduction

Large language models (LLMs) such as ChatGPT (Brown et al., 2020), have transformed content creation in journalism, education, and research. However, their fluency raises concerns about misuse, especially in the generation of fake news (Ahmed et al., 2021), fraudulent reviews (Adelani et al., 2019), and plagiarism (Lee et al., 2023). As these models become increasingly better, distinguishing machine- from human-written text is difficult even for experts (Shahid et al., 2022), highlighting the need for reliable automated detection methods.

Detection approaches fall into two groups: *supervised classifiers* and *zero-shot methods*. Supervised detectors rely on fine-tuning language models with labeled human and machine-written examples to learn distinguishing patterns (Mitrović et al., 2023). Although effective within their training domains, such models tend to generalize poorly to unseen models or domains (**?**). In contrast, zero-shot methods (Su et al., 2023) leverage pre-trained language models without additional training, relying on generalizable statistical or probabilistic clues instead. This makes them more robust to domain and model variations.

Zero-shot detectors operate by identifying intrinsic statistical properties of text, such as word ranking, token likelihood, and entropy. The key assumption is that machine-generated and human-written texts show different statistical properties. For example, LLMs typically assign higher log-likelihoods to their own outputs than to human-written passages, as they favour more probable sequences and tokens during decoding. Therefore, by designing algorithms that measure internal text properties and compare them against a threshold, we can classify them as machine-generated or human-written.

2 Methodology

We frame detection as *zero-shot binary classification*: given a candidate passage, determine whether it was written by a human or generated by a language model. We build on DetectGPT and Fast-DetectGPT, extending to new models, datasets, and introducing a novel adversarial attack.

2.1 Methods

Baseline methods compare internal model properties against a threshold to decide if text is human- or machine-generated.

DetectGPT (Mitchell et al., 2023) introduced a sampling-based criterion for detection. Its central hypothesis is that model-generated text variations tend to exhibit lower log-probabilities than the original, placing them in regions of negative curvature within the source model's log-likelihood function. DetectGPT generates multiple perturba-

tions of a candidate passage using a masked language model (e.g., T5), then scores both the original and its perturbations with the source model. If the original scores higher than its perturbations, it is likely machine-generated; otherwise human-written. While effective, this method is computationally expensive, as it requires scoring a large number of perturbations per passage (often more than 100).

Fast-DetectGPT replaces DetectGPT's perturb-and-score pipeline with a single-pass estimate of the conditional probability curvature. Framed as a sequential token selection problem, the hypothesis is that language models approximate the modal human distribution, favouring safer, higher-probability tokens, whereas individuals choose more diverse words. Consequently, model-generated text tends to lie near local maxima of the conditional probability function (positive curvature), while human text remains closer to flat regions (curvature near zero).

Operationally, Fast-DetectGPT samples alternative token choices independently at each position (conditioned on the observed prefix), computes their conditional probabilities under a scoring model, and aggregates these into a curvature statistic. All alternatives are produced once and scored in a single batch, avoiding per-perturbation scoring. Accuracy matches DetectGPT with far better efficiency.

$$\sum_i (\log p_\theta(\tilde{x}_i) - \tilde{\mu})/\tilde{\sigma} > \epsilon \rightarrow x \text{ is machine} \quad (1)$$

In eq. (1), $\tilde{x}_i$ are perturbations of the candidate passage x generated by a sampling model $q_\varphi(\tilde{x}|x)$, p_θ is the scoring model, $\tilde{\mu}$ and $\tilde{\sigma}$ are the sample mean and standard deviation of the perturbation log-probabilities under p_θ, and ϵ is a tunable threshold. The passage x is classified as machine-generated if its standardized log-probability exceeds ϵ. This formulation is also valid for DetectGPT when $q_\varphi(\tilde{x}|x)$ corresponds to its perturbation distribution. The pseudocode for both algorithms is presented in algorithm 1 and algorithm 2.

2.2 Models

Fast-DetectGPT employs three main model components: the source model, which generates candidate text by conditioning on a human-written prefix (first 30 tokens) and decoding from it; the sampling model which produces conditional token distributions used to generate perturbations $\tilde{x}_i$ of each candidate passage x; and the scoring model scores log-probabilities.

Detection experiments are conducted under two configurations: *white-box* (where source and scoring models are identical) and *black-box* (a more realistic setting).

2.3 Datasets

The models are evaluated on synthetic datasets created by prompting the source with the first 30 human tokens. The model then generates continuations up to an average of 230 total tokens, producing the necessary machine-generated text samples.

We use two human-written datasets to construct these new synthetic ones: XSum (Narayan et al., 2018), a large collection of documents and their summaries, and HC3 (Guo et al., 2023), a dataset of human and GPT-3.5-generated answers to the same questions (using only the human responses). From each, up to 500 human texts generate corresponding synthetic continuations.

2.4 Mid-K Attack

To evaluate detector robustness, we introduce the Mid-K Attack, a perturbation strategy that alters the token selection process during generation. At a fixed number of randomly selected positions, the top-K most probable tokens in the model's next-token distribution are banned (logits set to $-\infty$), forcing sampling from lower-probability tokens.

Mid-K has three parameters: P (the number of positions per sequence where the ban is applied), K (number of top tokens excluded at each position), and S (the first generated-token index eligible for perturbation).

For each sequence, we compute the maximum number of generated tokens $M = \texttt{max\_length} - \texttt{prompt\_length}$ and select up to P positions from $\{S, \ldots, M\}$. At each chosen position, we set the top-K logits to $-\infty$ and sample from the remaining options.

3 Experiments

Our experiments evaluate Fast-DetectGPT's performance and robustness against three baseline methods from the original study (Likelihood, Entropy, and LogRank), across architectures, datasets, and adversarial settings. We report AUROC and run all experiments in the *white-box* setting. DetectGPT was run only once due to computational limits (see Table 1).

3.1 Reproducibility Study

We first assess the reproducibility and generalization of Fast-DetectGPT across different model architectures and datasets. We chose three small open-weight models of varying sizes: DeepSeek R1 Distill Llama 8B (DeepSeek-AI, 2025), Microsoft Phi-2 (2.7B parameters) (Javaheripi et al., 2023), and Mistral 7B Instruct v0.2 (Jiang et al., 2023). These models were evaluated on the datasets mentioned above, XSum and HC3. This tests stability across domains, lengths, formats, and model sizes.

3.2 Closed-Weight API Model

To further test the method's applicability, in addition to the three open-weight models, we evaluate Fast-DetectGPT on a closed-weight API model: GPT-4o mini. Unlike open-weight models, GPT-4o mini only gives access to the top-20 token probabilities per token. While this limitation prevents a direct comparison with the open-weight models, the goal of this experiment is to analyze how restricted access to token probabilities affects detection performance.

3.3 Low-Resource Language

We also investigate Fast-DetectGPT's robustness in low-resource language settings. For this, we employ the XL-Sum dataset (Hasan et al., 2021), an extension of XSum covering multiple languages, and focus on the Scottish Gaelic subset. This experiment aims to determine whether Fast-DetectGPT can maintain detection accuracy when applied to languages with limited pre-training data and distinctive linguistic characteristics.

3.4 Adversarial Attack

Finally, we evaluate the robustness of all detectors under the Mid-K adversarial attack introduced in Section **??**. The objective is to quantify how detection accuracy degrades as attack strength increases.

We conduct all adversarial experiments using the GPT-2 model on the XSum dataset, since the goal is to analyze the impact of Mid-K parameters rather than variations in models or datasets, and since we had to consider computational limitations. For each example, the generator is prompted with the first 30 tokens and produces a continuation of up to 200 tokens. The Mid-K attack is applied during decoding, while the detection pipeline (Fast-DetectGPT) remains unchanged, ensuring that any observed performance degradation is attributable only to the attack.

We explore the following parameter ranges: $P \in \{1, 5, 10, 20, 30, 50, 60, 70, 80, 90, 100, 150\}$, $K \in \{1, 2, 5, 10, 20\}$, and fixed $S = 5$. Each (P, K) pair corresponds to one experiment, where Mid-K is applied at up to P positions starting from token index S. In total, 60 parameter combinations are tested, covering a wide spectrum of perturbation strengths.

Method	XSum			
	R1-8B	**Phi-2**	**Mistral**	**Avg.**
Likelihood	0.9999	0.8782	0.9665	0.9482
Entropy	0.1645	0.5411	0.4558	0.3871
LogRank	**1.0000**	0.9051	0.9596	0.9549
DetectGPT	0.8560	-	-	-
Fast-DetectGPT	**1.0000**	**0.9769**	**0.9989**	**0.9919**

Method	HC3			
	R1-8B	**Phi-2**	**Mistral**	**Avg.**
Likelihood	0.8391	0.7139	0.7486	0.7672
Entropy	0.3521	0.4358	0.4235	0.4038
LogRank	0.8275	0.7224	0.7393	0.7631
Fast-DetectGPT	**0.9710**	**0.8347**	**0.8963**	**0.9007**

Table 1: AUROC results on XSum and HC3 across three new models. Fast-DetectGPT consistently outperforms all other methods.

4 Results

4.1 Reproducibility

Table 1 shows that Fast-DetectGPT achieves the highest performance across all tested models and datasets. Among the baselines, Likelihood and LogRank follow closely on XSum, while Entropy consistently performs worst. The only available comparison with DetectGPT (R1-8B on XSum) shows that it lags behind all methods except Entropy.

Moreover, model size appears to have a mild positive effect on detection performance, with larger architectures achieving slightly higher AUROC scores. However, these differences are not significant enough to suggest any lack of robustness or scalability in Fast-DetectGPT.

Performance differences are more pronounced across datasets. Most methods, including Fast-DetectGPT, perform better on XSum than on HC3, possibly due to variations in dataset size, topic distribution, or stylistic diversity. Interestingly, Entropy exhibits the opposite trend, performing marginally better on HC3. Despite these varia-

tions, Fast-DetectGPT consistently outperforms all baselines, demonstrating its robustness across both datasets and model architectures.

4.2 API Model

Method	GPT 4o mini		
	XSum	HC3	**Avg.**
Fast-DetectGPT	**0.8051**	0.6026	0.7039

Table 2: AUROC results for the API experiment over the two datasets, only on Fast-DetectGPT. Performance is lower than with open-weight models.

Table 2 summarizes the AUROC results for the GPT-4o mini API experiments. Overall, performance is substantially lower than for the open-weight counterparts, particularly on the HC3 dataset. This decline is likely caused by the API's restriction to only return the top-20 token probabilities, which alters the underlying probability distribution available to the detector and limits the accuracy of its curvature-based computations.

Method	Underrepresented language			
	R1-8B	**Phi-2**	**Mistral**	**Avg.**
Likelihood	**1.0000**	0.9976	0.9972	0.9983
Entropy	0.0010	0.0968	0.2730	0.1236
LogRank	**1.0000**	**0.9984**	0.9974	**0.9986**
Fast-DetectGPT	0.9994	0.9825	**0.9995**	0.9938

Table 3: AUROC results for the under-represented language experiment. Surprisingly, Fast-DetectGPT performs slightly worse than the LogRank baseline.

4.3 Low-resource Language

Table 3 presents the results for the Scottish Gaelic experiment. All methods except Entropy remain strong, matching or slightly exceeding English results. In contrast, the Entropy baseline performs notably worse, indicating that it may be more sensitive to linguistic differences or data sparsity in low resource settings.

4.4 Mid-K Attack

Figure 2 shows that AUROC steadily declines as the number of perturbed positions (P) increases. The unperturbed baseline achieves an AUROC of 0.9941, near-perfect baseline. However, even moderate perturbations significantly reduce detection capability. For example, with $P = 50, K = 3$, AUROC drops to 0.5287, and with $P = 60, K = 2$, it decreases further to 0.512. These perturbations barely alter the text but make detection nearly random, suggesting that Fast-DetectGPT is highly sensitive to subtle sampling noise in the generation process.

Perplexity. As illustrated in Figure 1 in Appendix **??**, mean perplexity increases with the strength of the perturbation, reflecting greater sampling uncertainty and linguistic variation.

Summary. Overall, the Mid-K attack reveals that small probabilistic manipulations during decoding can sharply degrade detector accuracy without significantly altering the text. This highlights a key vulnerability of curvature-based zero-shot detectors and emphasizes the importance of robustness evaluation under adversarial conditions.

5 Conclusion

This study extended and evaluated the Fast-DetectGPT framework for machine-generated text detection across new models, datasets, and adversarial conditions. Experiments show Fast-DetectGPT consistently outperforms zero-shot baselines like Likelihood, Entropy, and LogRank, achieving near-perfect detection performance in most settings. The method also generalizes well across model architectures, dataset domains, and even to low-resource languages, confirming its robustness and reproducibility.

However, our findings also highlight important limitations. When applied to closed-weight API models like GPT-4o mini, where only limited probability information is accessible, Fast-DetectGPT's effectiveness drops significantly, revealing a dependency on full token distributions. Furthermore, adversarial experiments with the Mid-K attack show that even minimal perturbations to the decoding process can sharply degrade detection accuracy, highlighting the need for detectors that are resistant to probabilistic manipulations.

In summary, Fast-DetectGPT provides a fast, effective, and generalizable approach to identifying AI-generated text, but its sensitivity to incomplete probability access and sampling perturbations suggests that future work should focus on improving robustness.

References

David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. 2019. Generating sentiment-preserving

fake online reviews using neural language models and their human- and machine-based detection. *Preprint*, arXiv:1907.09177.

Alim Al Ayub Ahmed, Ayman Aljabouh, Praveen Kumar Donepudi, and Myung Suh Choi. 2021. Detecting fake news using machine learning : A systematic literature review. *Preprint*, arXiv:2102.04458.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *Preprint*, arXiv:2301.07597.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, and 1 others. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Jooyoung Lee, Thai Le, Jinghui Chen, and Dongwon Lee. 2023. Do language models plagiarize? In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 3637–3647. ACM.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature.

Sandra Mitrović, Davide Andreoletti, and Omran Ayoub. 2023. Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text. *Preprint*, arXiv:2301.13852.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *Preprint*, arXiv:1808.08745.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Wajiha Shahid, Yiran Li, Dakota Staples, Gulshan Amin, Saqib Hakak, and Ali Ghorbani. 2022. Are you a cyborg, bot or human?—a survey on detecting fake news spreaders. *IEEE Access*, 10:27069–27083.

Jinyan Su, Terry Yue Zhuo, Di Wang, and Preslav Nakov. 2023. DetectLLM: Leveraging log rank information for zero-shot detection of machine-generated text. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Implementation

The implementation of our methods is based on the original code provided by the authors of Fast-DetectGPT [1]. We introduced all necessary modifications to adapt it to our new introduced models, datasets, Mid-K adversarial attack in a new repository [2].

B Speed Comparison

Despite the use of DetectGPT being rather limited, there is one additional insight that could be extracted from its single use: the inference time. Running DetectGPT on XSum with the DeepSeek R1 model took 7 hours, 47 minutes and 43 seconds, which is approximately 118.9 times slower than Fast-DetectGPT which took only 3 minutes and 56 seconds for the same setting. This proves the claim that Fast-DetectGPT is more efficient by multiple orders of magnitude, but stays far behind the 340x speed-up reported in the original paper.

C Additional Results

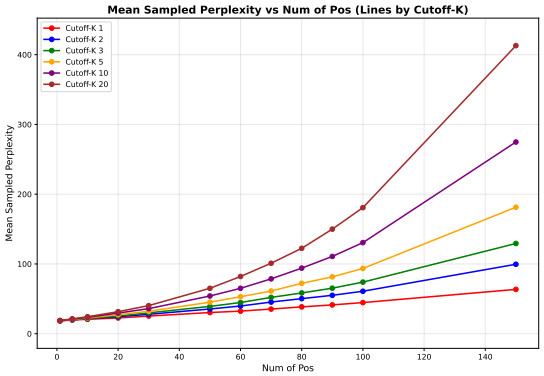

Figure 1: Mean sampled perplexity vs number of perturbations made. Presented per each cutoff tested. The trend follows inverse curve to the model performance. Measured on facebook/opt-125m (Zhang et al., 2022).

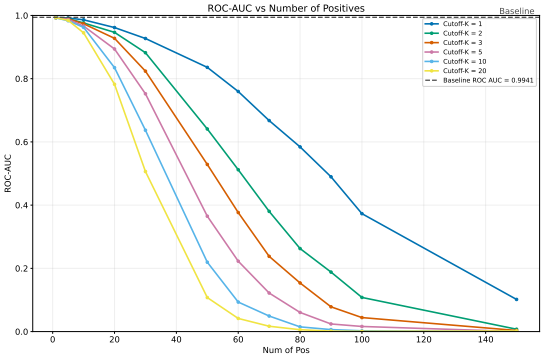

Figure 2: AUROC vs number of perturbed positions, shown separately for different cutoff values. The dashed line indicates baseline performance without perturbation. AUROC decreases as the number of perturbations increases, with larger cutoffs accelerating the degradation. Results are reported for GPT-2 (124M) (Radford et al., 2019).

[1] https://github.com/baoguangsheng/fast-detect-gpt

[2] https://github.com/salvatorpes/zero-shot-llm-detection

Table 4: Example texts under Mid-K attack with GPT-2 (124M) on XSum.

Perturbation Results:

Pos / Limit	Text
Human	"It was an oversight," Palmer told BBC Sport.
No perturbation	"It was an honest and open challenge over the length of time," he said.
50 / 5	"It was an honest and open challenge over the length of time," he tells us.
150 / 1	"It was an honest and open challenge over exactly how this particular scheme could have become so costly to operate after it closed in 2001, we still do to this present day," said the report said
150 / 20	"It was an inappropriate response by the HL [Herodoplyosis Unit to complain that player Eddie Gabboni ran red before half goal]; in such situations players wear distinctive black shorts (most usually found throughout tournaments by Uprod on other social programs at venues as in RTS).

D Algorithms

Algorithm 1 Fast-DetectGPT machine-generated text detection.

Input: passage x, sampling model q_φ, scoring model p_θ, and decision threshold ϵ
Output: True – probably machine-generated, False – probably human-written.

function FASTDETECTGPT(x, q_φ, p_θ)
  $\tilde{x}_i \sim q_\varphi(\tilde{x}|x), i \in [1..N]$ ▷ Conditional sampling
  $\tilde{\mu} \leftarrow \frac{1}{N}\sum_i \log p_\theta(\tilde{x}_i|x)$ ▷ Estimate the mean
  $\tilde{\sigma}^2 \leftarrow \frac{1}{N-1}\sum_i (\log p_\theta(\tilde{x}_i|x) - \tilde{\mu})^2$ ▷ Estimate the variance
  $\hat{\mathrm{d}}_x \leftarrow (\log p_\theta(x) - \tilde{\mu})/\tilde{\sigma}$ ▷ Estimate conditional probability curvature
  return $\hat{\mathrm{d}}_x > \epsilon$ ▷ Compare with threshold
end function

Algorithm 2 DetectGPT machine-generated text detection

Input: passage x, source model p_θ, perturbation function q, number of perturbations k, decision threshold ϵ
$\tilde{x}_i \sim q(\cdot \mid x), i \in [1..k]$ ▷ mask spans, sample replacements
$\hat{\mu} \leftarrow \frac{1}{k}\sum_i \log p_\theta(\tilde{x}_i)$ ▷ approximate expectation in Eq. 1
$\hat{\mathrm{d}}_x \leftarrow \log p_\theta(x) - \hat{\mu}$ ▷ estimate d (x, p_θ, q)
$\tilde{\sigma}_x^2 \leftarrow \frac{1}{k-1}\sum_i (\log p_\theta(\tilde{x}_i) - \hat{\mu})^2$ ▷ variance for normalization
if $\frac{\hat{\mathrm{d}}_x}{\tilde{\sigma}_x} > \epsilon$ **then**
  return `true` ▷ probably model sample
else
  return `false` ▷ probably not model sample