Text Fluoroscopy: Detecting LLM-Generated Text through Intrinsic Features

Xiao Yu[1,2], Kejiang Chen[1,2*], Qi Yang[3], Weiming Zhang[1], Nenghai Yu[1],
[1]University of Science and Technology of China, China,
[2]Key Laboratory of Cyberspace Security, Ministry of Education, China
[3]Shanghai University, China
yuxiao1217@mail.ustc.edu.cn, chenkj@ustc.edu.cn, yangqi_shu@shu.edu.cn
zhangwm@ustc.edu.cn, ynh@ustc.edu.cn

Abstract

Large language models (LLMs) have revolutionized the domain of natural language processing because of their excellent performance on various tasks. Despite their impressive capabilities, LLMs also have the potential to generate texts that pose risks of misuse. Consequently, detecting LLM-generated text has become increasingly important. Previous LLM-generated text detection methods use semantic features, which are stored in the last layer. This leads to methods that overfit the training set domain and exhibit shortcomings in generalization. Therefore, We argue that utilizing intrinsic features rather than semantic features for detection results in better performance. In this work, we design Text Fluoroscopy, a black-box method with better generalizability for detecting LLM-generated text by mining the intrinsic features of the text to be detected. Our method captures the text's intrinsic features by identifying the layer with the largest distribution difference from the last and first layers when projected to the vocabulary space. Our method achieves 7.36% and 2.84% average improvement in detection performance compared to the baselines in detecting texts from different domains generated by GPT-4 and Claude3, respectively. The codes are publicly available at `https://github.com/Fish-and-Sheep/Text-Fluoroscopy`.

1 Introduction

Large language models (LLMs) such as PaLM (Chowdhery et al., 2023), ChatGPT (OpenAI, 2022), LLaMA (Touvron et al., 2023), and GPT-4 (Achiam et al., 2023) demonstrate remarkable advancements in language capabilities. LLMs have significantly impacted the field of natural language processing, enabling proficient text generation for diverse tasks, including emails, news, and academic papers. With the advent of more advanced LLMs such as GPT-4, the outstanding performance of LLMs has led to the belief that they can be the artificial general intelligence (AGI) of this era (Bubeck et al., 2023).

However, if misused, LLMs such as ChatGPT have the potential to act as a "weapon of mass deception" (Sison et al., 2024). For example, the advanced writing capabilities of LLMs pose a significant threat to democracy, as they facilitate the creation of automated bots on social networks that can influence political decisions during election campaigns (Solaiman et al., 2019; Goldstein et al., 2023). Moreover, the use of ChatGPT by students in educational institutions has led to instances of academic dishonesty, with essays being generated by these models, as reported by various news outlets (Mitchell, 2022; Patrick Wood, 2023). Therefore, it is crucial and urgent to detect LLM-generated texts.

Previous methods for detecting LLM-generated text can be classified into two categories. The first category relies on the features of the last layer in the language model, e.g., BERT (Guo et al., 2023; Hu et al., 2023; Guo and Yu, 2023), which can be seen as the semantic features (Wu et al., 2023). However, semantic features in human-created and LLM-generated text can be remarkably similar, especially when the topics are more narrowly defined, affecting detection quality and generalization. The second category relies on linguistic features (Yang et al., 2023; Wu et al., 2024; McGovern et al., 2024), which are expressed as differences in the frequency of words and grammatical patterns. However, experimental results show that linguistic features are more fragile to paraphrase attacks than semantic features (McGovern et al., 2024).

Guided by the above analysis, it is evident that overly abstract semantic features and overly simple linguistic features can adversely affect detection quality and robustness. Consequently, we can infer that the features of the first and last layers are

*Corresponding author.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 15838–15846
November 12-16, 2024

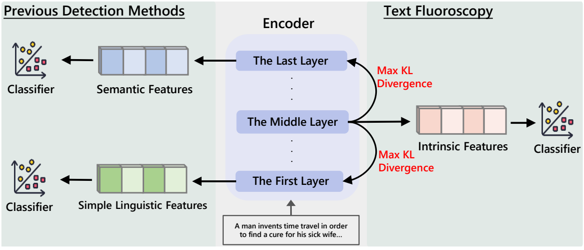


Figure 1: The difference between our method and previous detection methods.

not ideal. This raises the question: which features perform effectively in the task of LLM-generated text detection? Inspired by previous work (Jawahar et al., 2019; Tenney et al., 2019), the classical language model, e.g., BERT, has been shown to capture simple linguistic information at the earlier layers and semantic features at the later layers. The forward process of a language model can be viewed as a process of abstracting information from the input sentence. Therefore, we argue that the features of the middle layers reflect the intermediate process from understanding word-level linguistic information to sentence-level semantic information, essentially capturing how words are composed into sentences. It is in this intermediate process that differences between human and AI-generated texts exist. Consequently, we propose utilizing the features of middle layers to distinguish AI-generated texts from human-generated texts and regard the features as intrinsic characteristics of the text. Intuitively, the features of the middle layer with the largest differences compared to the first and last layers most accurately reflect the intrinsic features of the text. Therefore, we intend to utilize such features for detection.

Based on the above analysis, we propose **Text Fluoroscopy**, a black-box method for LLM-generated text detection through intrinsic features. We capture the intrinsic features of the text by identifying the layer with the largest distribution difference from the last and first layers when projected to the vocabulary space. The core idea of our method is shown in Figure 1.

Text Fluoroscopy achieves a 7.36% and 2.84% average improvement in detection performance compared to the baselines in detecting texts from different domains generated by GPT-4 and Claude3. These findings underscore the efficacy of our method. Moreover, our method is robust to Paraphrase (Krishna et al., 2023) and Back-translate attacks.

2 Related Work

Previous methods (Guo et al., 2023; Hu et al., 2023; Guo and Yu, 2023) employ the semantic features stored in the last layer of the language model to perform detection. For example, Hello-Chatgpt-detector-roberta (Guo et al., 2023) uses the Roberta model to extract semantic features and then trains a classifier to detect LLM-generated texts. The features stored in the last layer are abstract representations of semantic content, causing the method to overfit the domain of the training set and show deficiencies in generalization. Therefore, to obtain more generalizable detection methods, current researchers work on developing methods by linguistic features. For example, DNA-GPT (Yang et al., 2023) takes advantage of the divergence between multiple completions of a truncated passage. Some researchers (McGovern et al., 2024) find simple classifiers on top of n-gram and part-of-speech features can achieve very robust performance on both in- and out-of-domain data. These "fingerprints" retain the more primitive features of LLMs and are more useful for detecting LLM-generated text. However, these features are more susceptible to exploitation by attackers, and detection methods based on linguistic features are less robust than those based on BERT's features when facing paraphrase attacks (McGovern et al., 2024).

3 Methods

The goal of an LLM-generated text detection task is to ascertain whether a given text is generated by LLMs. Let x be the text to be detected. Formally, $x = (x_0, x_1, \ldots, x_{t-1})$ consists of t tokens.

We leverage a pre-trained language model as the encoder to extract intrinsic features. Pre-trained language models consist of an embedding layer, N stacked transformer layers, and an affine layer $\phi(\cdot)$ for predicting the distribution of the next word. First, the embedding layer embeds the tokens $x = (x_0, x_1, \ldots, x_{t-1})$ into a sequence of vectors $H_0 = \{h_0^{(0)}, h_1^{(0)}, \ldots, h_{t-1}^{(0)}\}$. Then H_0 would be processed by each of the transformer layers successively. We denote the output of the j-th layer as $H_j, (0 \leq j \leq N)$. Then, the vocabulary head $\phi(\cdot)$ predicts the probability of the next token x_t over the vocabulary set $\mathcal{X}$.

The method based on semantic features uses the feature $h_t^{(N)}$ to the classifier.

$$y_{\text{sem\_pred}} = \mathcal{D}(h_{t-1}^{(N)}),$$

where $\mathcal{D}$ represents the detector, $y_{\text{sem\_pred}}$ represents the predicted label of the detector based on semantic features.

Instead of using the features stored in the last layer, our method identifies the layer with the largest distribution difference from the last and first layers when projected to the vocabulary space.

Although the vocabulary head $\phi(\cdot)$ is only trained on the last layer, it is appropriate to use this head on the middle layers. We follow previous work (Teerapittayanon et al., 2016; Elbayad et al., 2020; Schuster et al., 2022), where they apply the vocabulary head directly to the hidden states of the middle layers for early exit. The rationality of the process has also been given in previous work. Since the residual connections in language models (He et al., 2016) make hidden representations evolve gradually without abrupt changes. This smooth transition and stability mean that directly applying the vocabulary head trained on the final layer to the middle layers can still yield reasonable prediction results, thus eliminating the need for additional training of the vocabulary head.

We first predict the probability of the next token x_t over the vocabulary set $\mathcal{X}$ for every layer. For j-th layer, we predict the probability of the next token x_t over the vocabulary set $\mathcal{X}$, where $\mathcal{J} \subset \{1, \ldots, N-1\}$ is a set of candidate layers,

$$q_j(x_t \mid x_{<t}) = \text{softmax}\big(\phi(h_t^{(j)})\big)_{x_t}, \quad j \in \mathcal{J}.$$

The probability of the next token x_t over the vocabulary set $\mathcal{X}$ for the 0-th and N-th layers are denoted as $q_0(x_t \mid x_{<t})$ and $q_N(x_t \mid x_{<t})$, respectively.

Then, we use Kullback-Leibler Divergence (KL Divergence) to calculate the difference between the distributions. We calculate the difference between the distributions $q_j(x_t \mid x_{<t})$, $q_0(x_t \mid x_{<t})$ and $q_j(x_t \mid x_{<t})$, $q_N(x_t \mid x_{<t})$. And we select the layer with the largest KL divergence from the 0-th and N-th layers, denoted the M-th layer ($0 < M < N$). Discussions about the KL Divergence and selection of layers are shown in Appendix A.

$$\begin{aligned} M = \arg\max_{j \in \mathcal{J}} \{ &\text{KL}\big(q_N(x_t \mid x_{<t}) || q_j(x_t \mid x_{<t})\big) + \\ &\text{KL}\big(q_0(x_t \mid x_{<t}) || q_j(x_t \mid x_{<t})\big)\}, \end{aligned}$$

where $\mathcal{J} \subset \{1, \ldots, N-1\}$ is a set of candidate layers.

After determining M, we use the features of the M layer for classification.

$$y_{\text{pred}} = \mathcal{D}(h_{t-1}^{(M)}),$$

where $\mathcal{D}$ represents the detector, y_{pred} represents the predicted label of the detector.

We train $\mathcal{D}$ using binary cross-entropy loss, and the loss function can be formalized as:

$$\begin{aligned} \mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \Big(&y^{(i)} \log(y_{\text{pred}}^{(i)}) \\ &+ (1 - y^{(i)}) \log(1 - y_{\text{pred}}^{(i)}) \Big), \end{aligned}$$

where y represents the true label of $\mathbf{x}$, and the predicted label of the detector is represented as y_{pred}.

4 Experiments

4.1 Implementation details

In this paper, we focus on the black-box scenario that closely mimics real-world conditions. In this scenario, all detectors cannot determine the source model of the text to be detected. We used the first 200 entries of the open-source Human-ChatGPT Comparison Corpus (HC3) (Guo et al., 2023) dataset collected by previous researchers as a training set to ensure the reproducibility of our method. The ratio for splitting the training and validation is 8 : 1.

we use `gte-Qwen1.5-7B-instruct`[1] as the encoder which can encode texts with a maximum of $32K$ tokens into embeddings of 4096 dimensions, while the classifier consists of three fully connected layers with Tanh function. The dimensions of the intermediate layers in the classifier are 1024 and 512, respectively. We train the classifier for 10 epochs on the training set and utilize a validation set to select the weights that yield the best performance. All experiments are conducted on a workstation equipped with 4 NVIDIA RTX4090 GPUs.

Datasets. We assessed the generalizability of detection methods across various dataset domains and generative models. For this purpose, we selected three different datasets and utilized three generative models for data generation. Specifically, the three datasets are Xsum (Narayan et al., 2018) for news articles, WritingPrompts (Fan et al., 2018) for story writing, PubMedQA (Jin et al., 2019) for biomedical research question answering, which

[1]https://huggingface.co/Alibaba-NLP/gte-Qwen1.5-7B-instruct

Methods	ChatGPT				GPT-4				Claude3			
	XSum	Writing	PubMed	Avg.	XSum	Writing	PubMed	Avg.	XSum	Writing	PubMed	Avg.
RoBERTa-base	0.9150	0.7084	0.6188	0.7474	0.6778	0.5068	0.5309	0.5718	0.8944	0.8036	0.3647	0.6876
RoBERTa-large	0.8507	0.5480	0.6731	0.6906	0.6879	0.3822	0.6067	0.5589	0.9027	0.7128	0.3579	0.6578
RADAR	0.9972	0.9593	0.7372	0.8979	0.9931	0.8593	0.8029	0.8851	0.9952	0.9438	0.8029	0.9139
CoCo	0.5392	0.7741	0.5847	0.6327	0.5495	0.7473	0.5197	0.6055	0.4808	0.7633	0.7388	0.6610
Likelihood	0.9577	0.9739	0.8776	0.9364	0.7982	0.8553	0.8100	0.8212	0.9760	0.9744	0.9240	0.9581
Entropy	0.3305	0.1901	0.2766	0.2657	0.4364	0.3703	0.3296	0.3788	0.4109	0.0836	0.1686	0.2210
LogRank	0.9584	0.9656	0.8680	0.9307	0.7980	0.8289	0.7997	0.8089	0.9783	0.9732	0.9260	0.9592
LRR	0.9164	0.8962	0.7421	0.8516	0.7453	0.7040	0.6810	0.7101	0.9609	0.9598	0.8334	0.9180
DNA-GPT	0.9040	0.9449	0.7598	0.8696	0.7267	0.8164	0.7163	0.7531	0.9071	0.9655	0.5911	0.8212
NPR	0.7845	0.9697	0.5483	0.7675	0.5211	0.8276	0.4976	0.6154	0.9232	0.9696	0.7746	0.8891
Fast-DetectGPT	0.9907	**0.9916**	0.9021	0.9615	0.9064	0.9611	0.8498	0.9058	0.9942	0.9783	0.9035	0.9587
Text Fluoroscopy	**0.9996**	0.9856	**0.9167**	**0.9673**	**0.9998**	**0.9835**	**0.9548**	**0.9794**	**0.9998**	**0.9979**	**0.9636**	**0.9871**

Table 1: The detection performance (AUROC) of baselines and Text Fluoroscopy on three datasets generated by ChatGPT, GPT-4, and Claude3.

Methods	ChatGPT			GPT-4			Claude3		
	Ori.	DIPPER	Back-translate	Ori.	DIPPER	Back-translate	Ori.	DIPPER	Back-translate
RoBERTa-base	0.9150	0.8148	0.8379	0.6778	0.6469	0.7536	0.8944	0.8120	0.8052
RoBERTa-large	0.8507	0.7884	0.6853	0.6879	0.6833	0.6660	0.9027	0.8153	0.7583
RADAR	0.9972	0.9964	0.9801	0.9931	0.9924	0.9608	0.9952	0.9940	0.9701
CoCo	0.5392	0.5374	0.5525	0.5495	0.5627	0.5510	0.4808	0.4886	0.5075
Likelihood	0.9577	0.8438	0.9306	0.7982	0.6296	0.8449	0.9760	0.9080	0.9446
Entropy	0.3305	0.4514	0.3008	0.4364	0.5552	0.3705	0.4109	0.4978	0.3639
LogRank	0.9584	0.8596	0.9260	0.7980	0.6432	0.8436	0.9783	0.9256	0.9488
LRR	0.9164	0.8448	0.8621	0.7453	0.6607	0.8003	0.9609	0.9240	0.9243
DNA-GPT	0.9040	0.7733	0.8624	0.7267	0.5595	0.7776	0.9071	0.7876	0.8399
NPR	0.7845	0.5648	0.8050	0.5211	0.3006	0.6820	0.9232	0.7860	0.9042
Fast-DetectGPT	0.9907	0.9536	0.9711	0.9064	0.8057	0.9137	0.9942	0.9720	0.9860
Text Fluoroscopy	**0.9996**	**0.9996**	**0.9980**	**0.9998**	**0.9994**	**0.9961**	**0.9998**	**0.9996**	**0.9995**

Table 2: Detection performance of Text Fluoroscopy and the baselines in detecting Xsum dataset generated by ChatGPT, GPT-4, and Claude3 with interference.

are consistent with previous work (Bao et al.) in the field. We also utilized three current widely used commercial closed-source models for data generation, including ChatGPT (gpt-3-5-turbo) [2], GPT-4 (gpt-4-0613) [3], and Claude3 (claude-3-opus-20240229) [4].

Evaluation metric. We measure the detection performance in the area under the receiver operating characteristic (AUROC). AUROC ranges from 0.0 to 1.0, mathematically denoting the probability of a random machine-generated text having a higher predicted probability of being machine-generated than a random human-written text. A higher AUROC value indicates a better detection quality.

Baselines. We compared our method with existing supervised detectors and zero-shot detectors. For supervised detectors, we compared GPT-2 detectors based on RoBERTa-base/large (Liu et al., 2019) crafted by OpenAI, RADAR (Hu et al., 2023) and CoCo (Liu et al., 2023). For zero-shot detectors, we selected LRR (an amalgamation of log probability and log-rank)(Su et al.), DNA-GPT (Yang et al., 2023), DetectGPT (Mitchell et al., 2023), and its enhanced variants NPR (Su et al.) and Fast-DetectGPT (Bao et al.). We also chose classic zero-shot classifiers We also chose classic zero-shot classifiers, including Likelihood (mean log probabilities)(Gehrmann et al., 2019), LogRank (average log of ranks in descending order by probabilities) (Solaiman et al., 2019), Entropy (mean token entropy of the predictive distribution)(Ippolito et al., 2020).

4.2 Performance

Detection effectiveness. The detection performance of baselines and Text Fluoroscopy is shown in Table 1. Our method achieves an average AUROC of 96.73%, 97.94%, and 98.71% in detecting three datasets generated by ChatGPT, GPT-4, and Claude3, respectively. Fast-DetectGPT, which is

[2]https://platform.openai.com/docs/models/gpt-3-5-turbo
[3]https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4
[4]https://docs.anthropic.com/en/docs/models-overview

LLM	Layer	ChatGPT				GPT-4				Claude3			
		XSum	Writing	PubMed	Avg.	XSum	Writing	PubMed	Avg.	XSum	Writing	PubMed	Avg.
gte-Qwen2-7B	Last	0.9658	0.9710	0.6186	0.8518	0.9711	0.9758	0.6847	0.8772	0.9472	0.9836	0.8011	0.9106
	Middle	**0.9988**	**0.9834**	**0.7744**	**0.9189**	**0.9996**	**0.9872**	**0.8416**	**0.9428**	**0.9994**	**0.9953**	**0.9373**	**0.9773**
stella_en_1.5B_v5	Last	0.8928	0.9802	**0.6966**	0.8565	0.8996	0.9708	**0.7293**	0.8666	0.8971	0.9758	0.8646	0.9125
	Middle	**1.0000**	**0.9921**	0.6611	**0.8844**	**1.0000**	**0.9873**	0.6955	**0.8943**	**0.9997**	**0.9834**	**0.8955**	**0.9595**
GPT-neo-2.7B	Last	0.6270	0.7190	**0.6079**	0.6513	0.7317	0.7970	0.4883	0.6724	0.9674	**0.9981**	0.8769	0.9475
	Middle	**0.8568**	**0.8916**	**0.6079**	**0.7854**	**0.9005**	**0.9137**	**0.5027**	**0.7723**	**0.9945**	0.9933	**0.9350**	**0.9743**

Table 3: The detection performance (AUROC) of methods with **last layer** and dynamically selected **middle layer**(Text Fluoroscopy) using different LLM as encoder on three datasets generated by ChatGPT, GPT-4, and Claude3.

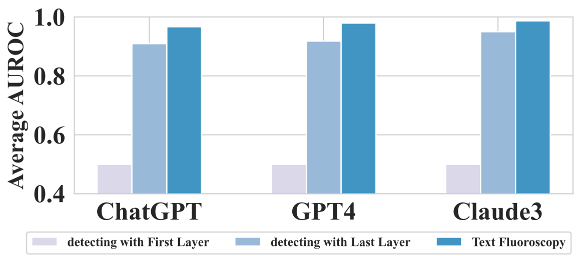


Figure 2: Detection AUROC of methods with different layers.

the best method among the baselines, has a lower average AUROC of 96.15%, 90.58%, and 95.87%, respectively. Notably, our method outperforms Fast-DetectGPT by 7.36% in average detection performance of datasets generated by GPT-4.

4.3 Robustness

To better understand the performance of Text Fluoroscopy in real-world scenarios, we evaluate our method under DIPPER (Paraphrase) (Krishna et al., 2023) and back-translation attacks, details are shown in Appendix B. From the results shown in Table 2, it can be observed that when facing the two attacks, the detection performance of our methods is still better than other methods, indicating that our method is more robust in real-world scenarios. We believe this advantage arises because our method extracts intrinsic features independently of semantic features, rendering the semantic attack ineffective and ensuring robustness.

4.4 Ablation studies

Layer Selection. We conducted ablation studies to reveal the impact of the selection of layers. We evaluated the average AUROC of detection with the first and last layer on three datasets generated by ChatGPT, GPT-4, and Claude3. The results are shown in Figure 2. It can be observed that the detection performance of methods with the first and last layer features is poorer than Text Fluoroscopy. This indicates that semantic and linguistic features interfere with detection quality, while Text Fluoroscopy chooses intrinsic features that can effectively detect LLM-generated text.

Applicability Across LLMs. We also selected three additional different LLMs as encoder for ablation experiments to demonstrate our method's validity and broad applicability. The models we chose include two advanced LLMs on the Massive Text Embedding Benchmark [5], namely stella_en_1.5B_v5 [6], gte-Qwen2-7B-instruct [7], and a classical GPT-neo-2.7B [8]. The results are shown in Table 3. As can be seen from the table, our method demonstrates effectiveness on all three LLMs.

5 Conclusion

In this paper, we design Text Fluoroscopy, a black-box method for detecting LLM-generated text through intrinsic features. Our method captures the intrinsic features by identifying the layer with the largest distribution difference from the first and last layers when projected to the vocabulary space. Compared with previous methods, we reduce the impact of semantic features on the detection process to achieve better detection quality and generalization. Our method can effectively detect LLM-generated texts and is more robust in real-world scenarios. We aspire that Text Fluoroscopy will inspire future research in LLM-generated text detection and offer insightful references for identifying content generated by LMs in other fields.

[5]https://huggingface.co/spaces/mteb/leaderboard
[6]https://huggingface.co/dunzhang/stella_en_1.5B_v5
[7]https://huggingface.co/Alibaba-NLP/gte-Qwen2-7B-instruct
[8]https://huggingface.co/EleutherAI/gpt-neo-2.7B

6 Limitations

Although our method is simple and effective, it still has some limitations. In our detection process, we need to compute each layer of the pre-trained language model to determine the layer with intrinsic features, which will cause a time delay. We evaluated the average time cost by our method and the other methods in detecting a piece of text, and the results are displayed in Table 4.

Our method's average cost time in detecting a piece of text from three datasets generated by ChatGPT, GPT-4, and Claude3 is 0.5283s,0.5145s, and 0.4995s, respectively. However, the detection method only using the last layer takes just 0.0776s, 0.0948s, and 0.0808s, respectively.

Methods	ChatGPT	GPT-4	Claude3
Detection with the Last Layer	0.0776s	0.0948s	0.0808s
Text Fluoroscopy	0.5283s	0.5145s	0.4995s
Detection with the 30-th layer	0.0815s	0.0801s	0.0785s

Table 4: The average time cost for detecting a piece of text from three datasets generated by ChatGPT, GPT-4, and Claude3 with the different layers of detection.

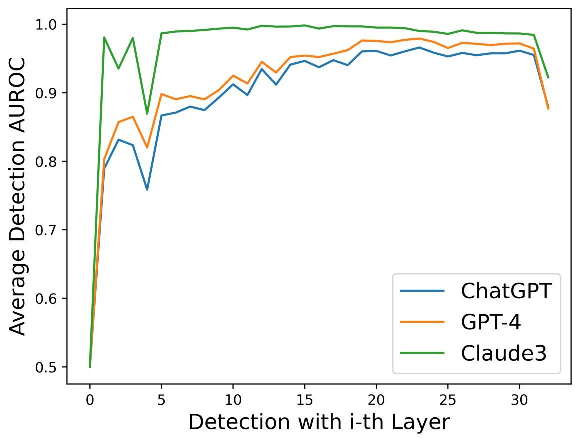


Figure 3: The average detection AUROC of three datasets generated by ChatGPT, GPT-4, and Cluade3 with the different layers.

To overcome this limitation, we hope to find a fixed layer with intrinsic features to reduce the cost of time while maintaining accuracy. Therefore, we tested the average detection AUROC of three datasets generated by ChatGPT, GPT-4, and Cluade3 with the different layers, as shown in Figure 3. We found that the average detection AUROC generally increases as the layers deepen but decreases after the 30-th layer. This observation also supports the effectiveness of using middle layers for detection. When using a fixed layer, the overall detection AUROC peaks at around the 30-th layer. Therefore, we use the detection with the 30-th layer to reduce time cost. The time cost for detecting a piece of text with the 30-th layer is shown in Table 4.

We also tested the AUROC of detection with the 30-th layer, shown in Table 5. The detection with the 30-th layer achieves an average AUROC of 96.19%, 97.23%, and 98.70% in detecting three datasets generated by ChatGPT, GPT-4, and Claude3, respectively. Text Fluoroscopy has higher average AUROC of 96.73%, 97.94%, and 98.71%, respectively. Using the fixed 30-th layer, the detection speed can be increased by approximately 5 times with an accuracy decrease of less than 0.7% compared to Text Fluoroscopy.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant U2336206, Grant 62472398, Grant 62102386, Grant 62121002, Grant U20B2047 and by Open Foundation of Key Laboratory of Cyberspace Security, Ministry of Education (No.KLCS20240207).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *ICLR 2020-Eighth International Conference on Learning Representations*, pages 1–14.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. In *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL).

Josh A Goldstein, Girish Sastry, Micah Musser, Renee DiResta, Matthew Gentzel, and Katerina Sedova. 2023. Generative language models and automated influence operations: Emerging threats and potential mitigations. *arXiv preprint arXiv:2301.04246*.

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. *arXiv preprint arXiv:2301.07597*.

Zhen Guo and Shangdi Yu. 2023. Authentigpt: Detecting machine-generated text via black-box language models denoising. *arXiv preprint arXiv:2311.07700*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577.

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 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 27469–27500.

Xiaoming Liu, Zhaohan Zhang, Yichen Wang, Hang Pu, Yu Lan, and Chao Shen. 2023. Coco: Coherence-enhanced machine-generated text detection under low resource with contrastive learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16167–16188.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Hope McGovern, Rickard Stureborg, Yoshi Suhara, and Dimitris Alikaniotis. 2024. Your large language models are leaving fingerprints. *arXiv preprint arXiv:2405.14057*.

Alex Mitchell. 2022. Professor catches student cheating with chatgptl: 'i feel abject terror'.

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

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

OpenAI. 2022. Introducing chatgpt.

Mary Louise Kelly Patrick Wood. 2023. 'everybody is cheating': Why this teacher has adopted an open chatgpt policy.

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*.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472.

Alejo Jose G Sison, Marco Tulio Daza, Roberto Gozalo-Brizuela, and Eduardo C Garrido-Merchán. 2024. Chatgpt: More than a "weapon of mass deception" ethical challenges and responses from the human-centered artificial intelligence (hcai) perspective. *International Journal of Human–Computer Interaction*, 40(17):4853–4872.

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

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

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Derek F Wong, and Lidia S Chao. 2023. A survey on llm-gernerated text detection: Necessity, methods, and future directions. *arXiv preprint arXiv:2310.14724*.

Junchao Wu, Runzhe Zhan, Derek F Wong, Shu Yang, Xuebo Liu, Lidia S Chao, and Min Zhang. 2024. Who wrote this? the key to zero-shot llm-generated text detection is gecscore. *arXiv preprint arXiv:2405.04286*.

Xianjun Yang, Wei Cheng, Yue Wu, Linda Ruth Petzold, William Yang Wang, and Haifeng Chen. 2023. Dna-gpt: Divergent n-gram analysis for training-free detection of gpt-generated text. In *The Twelfth International Conference on Learning Representations*.

A The KL Divergence and Selection of Layers.

To fully illustrate the validity of using KL Divergence for layer selection, we tested the KL Divergence between the distributions of the first layer and the i-th layer, and the AUROC of detection with the i-th layer. The results are shown in Figure 4. The figure shows that the KL Divergence and AUROC exhibit similar trends. They both gradually increase over the first 30-th layers but show a decreasing trend after the 30-th layer.

B Additional Experimental Results

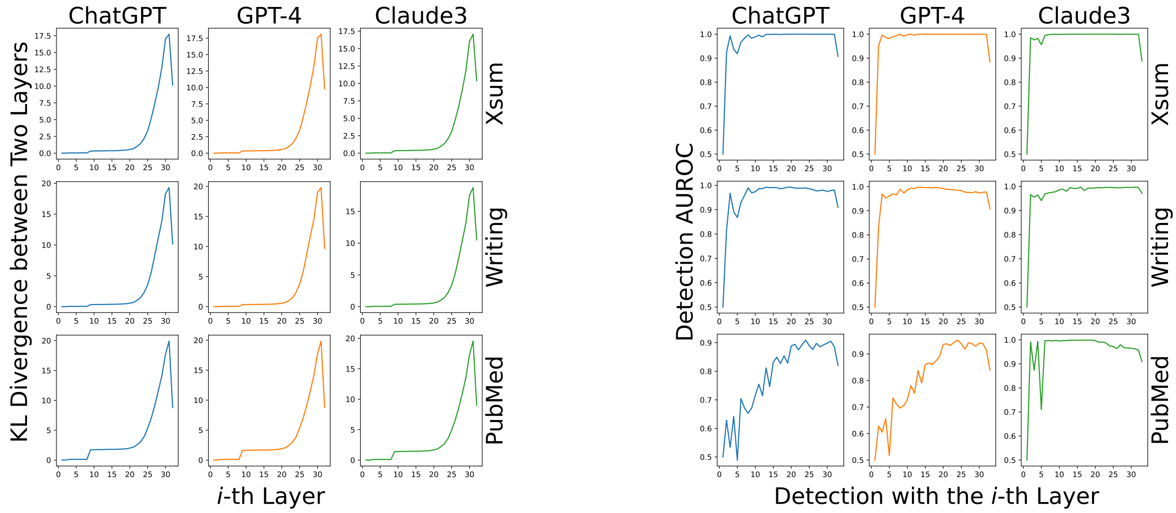
Existing research (Krishna et al., 2023; Sadasivan et al., 2023) has pointed out that previous methods exhibit performance degradation in complex scenarios where the text to be detected is subjected to perturbations. To better understand the performance of Text Fluoroscopy in real-world scenarios, we evaluate our detection method under two different modification methods.

The first one is the proposed paraphrasing attack called DIPPER (Krishna et al., 2023) (or Discourse Paraphrase). DIPPER is an 11B-parameter paraphrase generation model built by fine-tuning `T5-XXL`. It can paraphrase paragraph-length texts, re-order content, and optionally leverage context, such as input prompts.

The second perturbation method we used, the so-called back-translation attack, is more accessible to a broader audience and does not require specialized knowledge. Back-translation refers to the action of *translating a work that has previously been translated into the same language*. We employed DeepL Translator [9] to translate the given English text into Chinese, followed by a subsequent translation back into English.

We present the detection performance of our method and baselines in detecting the Xsum dataset generated by ChatGPT, GPT-4, and Claude3 with interference in Table 6. RADAR shows the smallest decrease among baselines against DIPPER attacks, especially for text generated by GPT-4, with a decrease of 00.07%, illustrating the robustness of RADAR in incorporating adversarial networks into detection. However, Our method maintains optimal detection performance after both DIPPER and back-translation attacks. The detection AUROC of our method is 99.96% and 99.80% for detecting the Xsum dataset generated by ChatGPT under DIPPER and back-translation attacks, respectively, indicating that our method is more robust in real-world scenarios.

[9] `https://www.deepl.com/en/docs-api/`



(a) The KL Divergence between the distributions of the first layer and the i-th layer.

(b) The AUROC of detection with the i-th layer.

Figure 4: The KL Divergence between the distributions of the first layer and the i-th layer, and the AUROC of detection with the i-th layer.

Methods	ChatGPT				GPT-4				Claude3			
	XSum	Writing	PubMed	Avg.	XSum	Writing	PubMed	Avg.	XSum	Writing	PubMed	Avg.
RoBERTa-base	0.9150	0.7084	0.6188	0.7474	0.6778	0.5068	0.5309	0.5718	0.8944	0.8036	0.3647	0.6876
RoBERTa-large	0.8507	0.5480	0.6731	0.6906	0.6879	0.3822	0.6067	0.5589	0.9027	0.7128	0.3579	0.6578
RADAR	0.9972	0.9593	0.7372	0.8979	0.9931	0.8593	0.8029	0.8851	0.9952	0.9438	0.8029	0.9139
CoCo	0.5392	0.7741	0.5847	0.6327	0.5495	0.7473	0.5197	0.6055	0.4808	0.7633	0.7388	0.6610
Likelihood	0.9577	0.9739	0.8776	0.9364	0.7982	0.8553	0.8100	0.8212	0.9760	0.9744	0.9240	0.9581
Entropy	0.3305	0.1901	0.2766	0.2657	0.4364	0.3703	0.3296	0.3788	0.4109	0.0836	0.1686	0.2210
LogRank	0.9584	0.9656	0.8680	0.9307	0.7980	0.8289	0.7997	0.8089	0.9783	0.9732	0.9260	0.9592
LRR	0.9164	0.8962	0.7421	0.8516	0.7453	0.7040	0.6810	0.7101	0.9609	0.9598	0.8334	0.9180
DNA-GPT	0.9040	0.9449	0.7598	0.8696	0.7267	0.8164	0.7163	0.7531	0.9071	0.9655	0.5911	0.8212
NPR	0.7845	0.9697	0.5483	0.7675	0.5211	0.8276	0.4976	0.6154	0.9232	0.9696	0.7746	0.8891
DetectGPT	0.4594	0.8008	0.3804	0.5469	0.3408	0.6542	0.3675	0.4542	0.4323	0.6800	0.7559	0.6227
Fast-DetectGPT	0.9907	**0.9916**	0.9021	0.9615	0.9064	0.9611	0.8498	0.9058	0.9942	0.9783	0.9035	0.9587
Text Fluoroscopy	**0.9996**	0.9856	**0.9167**	**0.9673**	**0.9998**	**0.9835**	**0.9548**	**0.9794**	0.9998	**0.9979**	0.9636	**0.9871**
Text Fluoroscopy (30-th Layer)	0.9991	0.9833	0.9032	0.9619	0.9994	0.9803	0.9373	0.9723	**0.9999**	0.9969	**0.9641**	0.9870

Table 5: The detection performance (AUROC) of baselines and Text Fluoroscopy on three datasets generated by ChatGPT, GPT-4, and Claude3.

Methods	ChatGPT			GPT-4			Claude3		
	Ori.	DIPPER	Back-translate	Ori.	DIPPER	Back-translate	Ori.	DIPPER	Back-translate
RoBERTa-base	0.9150	0.8148	0.8379	0.6778	0.6469	0.7536	0.8944	0.8120	0.8052
RoBERTa-large	0.8507	0.7884	0.6853	0.6879	0.6833	0.6660	0.9027	0.8153	0.7583
RADAR	0.9972	0.9964	0.9801	0.9931	0.9924	0.9608	0.9952	0.9940	0.9701
CoCo	0.5392	0.5374	0.5525	0.5495	0.5627	0.5510	0.4808	0.4886	0.5075
Likelihood	0.9577	0.8438	0.9306	0.7982	0.6296	0.8449	0.9760	0.9080	0.9446
Entropy	0.3305	0.4514	0.3008	0.4364	0.5552	0.3705	0.4109	0.4978	0.3639
LogRank	0.9584	0.8596	0.9260	0.7980	0.6432	0.8436	0.9783	0.9256	0.9488
LRR	0.9164	0.8448	0.8621	0.7453	0.6607	0.8003	0.9609	0.9240	0.9243
DNA-GPT	0.9040	0.7733	0.8624	0.7267	0.5595	0.7776	0.9071	0.7876	0.8399
NPR	0.7845	0.5648	0.8050	0.5211	0.3006	0.6820	0.9232	0.7860	0.9042
DetectGPT	0.4594	0.3074	0.5417	0.3408	0.1823	0.4530	0.4323	0.3283	0.5273
Fast-DetectGPT	0.9907	0.9536	0.9711	0.9064	0.8057	0.9137	0.9942	0.9720	0.9860
Text Fluoroscopy	**0.9996**	**0.9996**	**0.9980**	**0.9998**	**0.9994**	**0.9961**	0.9998	0.9996	**0.9995**
Text Fluoroscopy(30-th Layer)	0.9991	0.9990	0.9952	0.9994	0.9991	0.9941	**0.9999**	**0.9999**	0.9990

Table 6: Detection performance of Text Fluoroscopy and the baselines in detecting Xsum dataset generated by ChatGPT, GPT-4, and Claude3 with interference.