MULTIMODAL AI DETECTION FOR MATH-HEAVY SCIENTIFIC TEXT

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language models become increasingly capable of generating, and assisting with, scientific writing, there is a need for greater transparency and disclosure around the use of AI in research. Consequently, a veritable subfield of research and a number of commercial services aim to flag AI-generated content automatically. However, existing detectors which rely only on text extracted from PDFs fail to benefit from the rich visual information contained in the formatting of *math-heavy* scientific content. To study this, we select human-written proof and prose passages from pre-2022 arXiv LaTeX sources and re-generate them using 15 modern LLMs within the context of the same paper. This leaves us with coupled human and AI outputs in three different representations: the LaTeX source, a rendered image of the PDF, and the text extracted from the PDF. We find that the accuracy of the state-of-the-art Pangram 4 drops significantly as the mathematical content in a passage increases, especially when evaluated on text as opposed to LaTeX. However, this is not an inherent limitation of detecting AI-generated math: finetuning Qwen-3.5-27B based detectors on this domain recovers the gap, with LaTeX inputs performing significantly better than text since they preserve more formatting information. Since LaTeX sources are rarely available in practice, we also train SNAPJUDGE: an image-based detector operating on screenshots of both prose and proofs from papers. At 0.1% false positive rate, SNAPJUDGE achieves a detection rate of 92% on proofs and 93% on prose generated by frontier models, substantially outperforming both Pangram 4 and our text-based detector. We will release our data, code and models, as well as a publicly available demo.

1 INTRODUCTION

In a short span of time, LLM mathematical abilities have risen from tackling middle-school math problems (Cobbe et al., 2021) to winning math olympiads (Luong & Lockhart, 2025), and now solving longstanding open problems, such as the Jacobian conjecture (Tao, 2026) and the Navier–Stokes Millennium Prize problem (OpenAI, 2026b). These advances highlight the urgent need for improving transparency about the use of AI when publishing scientific papers, including the technical content such as proofs and derivations that support their claims. Even when the claims in a paper are correct, understanding how they were produced still matters for evaluating researchers' contributions and assigning credit. This need is reflected in publication policies: many conferences and publishers now require authors to disclose their use of AI (NeurIPS, 2025; ICLR, 2025; NeurIPS, 2026; Nature Portfolio, Springer Nature, 2026; Mohapatra & Pruthi, 2026). Such policies rest on honest disclosure, which can only be verified, and therefore enforced, when accurate detection is available. Although the state-of-the-art commercial detector, Pangram 4 (Glickenhaus et al., 2026), is highly accurate at flagging general-purpose AI content, we find that its effectiveness degrades significantly on scientific text, especially math-heavy text, which comes with rich visual formatting.[1]

For ordinary prose, we can expect that any given idea can be expressed in many different ways, and the choices an LLM makes among them leave traces that can be detected (Mitchell et al., 2023; Hans et al., 2024; Liang et al., 2025). However, it is unclear whether such detectable signs remain in the much more constrained and formal setting of math-heavy scientific writing. In this work,

[1]The Pangram 4 report explicitly states that mathematical prompts are excluded from its evaluation.

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

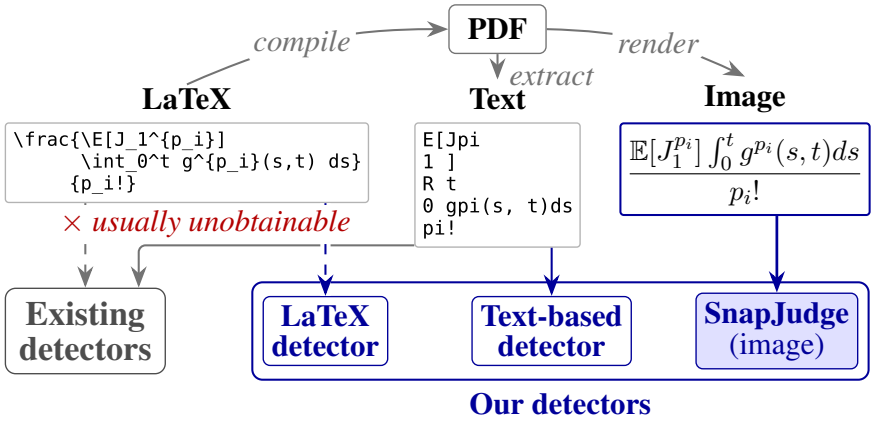


Figure 1: **The same mathematical expression shown as LaTeX source, PDF-extracted text, and a rendered image.** We study AI detection across three representations and find that rendered images generally yield higher detection rates than extracted text and approach the performance obtained with the original LaTeX source.

we present the first systematic study of AI text detection on math-heavy scientific writing under 3 different representations: LaTeX, text and rendered images of PDF pages (Figure 1). We find that **AI-generated math remains highly detectable under all three representations**.

LaTeX is the dominant language for typesetting math in scientific papers (Ginev et al., 2026), however, in practice readers seldom have access to the underlying LaTeX source of a paper they are reading. Prior detectors deal with PDFs by either: (i) extracting text from the PDF, using tools such as PyMuPDF (Artifex Software, 2026), which severely damages formatting of the mathematical content and, as we show, reduces detection accuracy (Section 4); or (ii) running OCR on the page images using specialized tools such as Marker and Tesseract (Paruchuri, 2023; Smith, 2007), which preserve math but introduce additional formatting signals that impede detection (Emi, 2025).

Hence, we develop SNAPJUDGE, an image-based detector that identifies AI-generated writing directly from rendered scientific passages. This requires minimal preprocessing and preserves the stylistic choices behind math writing which can aid with detection. Specifically, we construct our data by extracting the proof and prose passages from LaTeX sources of pre-2022 arXiv papers, and re-generating them using modern LLMs (conditioned on the rest of the paper). Using the resulting data, we finetune three Qwen3.5-based detectors (Qwen Team, 2026), one per representation, and show that while the LaTeX detector performs best, the image-based detector has comparable accuracy and also outperforms the text-based detector (Section 4). On the same test data, we find that Pangram 4 only detects 36% of the AI-generated proofs when tested on the LaTeX source, with even worse performance upon using the extracted text. Overall, our detectors lead to a $>90\%$ detection rate at a relatively strict threshold of 0.1% false positive rate, and also hold up outside mathematics. Despite concerns that supervised detectors collapse on unfamiliar domains (Shen et al., 2026), SNAPJUDGE performs excellently on MIRAGE (Fu et al., 2025), RAID (Dugan et al., 2024), AI-written peer reviews (Yu et al., 2026), and the EditLens dataset (Thai et al., 2026) (Section 4.3).

When we further reduce the false-positive rate to 0.01%, SNAPJUDGE still detects more than 82% of the AI-written passages (Figure 2), highlighting the viability of our tool for practical use. We further analyze what makes AI-generated math detectable, and rule out the explanation that it is qualitatively different or may contain more errors, making it easier to detect (Section 4.2). We also investigate if prose around the mathematical formulas gives away signs of AI writing, and find that passages do become harder to detect as the percentage of formulas in them increases, but we are still able to detect over 80% proofs which are composed primarily of formulas and equations.

To summarize, our contributions are the following:

1. We contribute a dataset containing math-heavy text, consisting of 47K human-AI paired proofs and 45K paired prose passages, where each pair includes three representations, namely the LaTeX source, the text extracted from the PDF, and the rendered image.

2. We show that existing zero-shot and open source detectors fail on proofs generated by frontier models, and even the state-of-the-art Pangram 4 detects only 32.4% of them using the extracted text, or 65.6% upon recalibration.

3. We systematically study the detection performance under three different representations, and find that detectors operating on the LaTeX source perform best, whereas text-based detectors struggle with math-heavy text. Image-based detectors are comparable to the performance of detectors that use LaTeX source.

4. We propose SNAPJUDGE, an image-based text detector, which also supports estimating the proportion of the text in an image that was AI-generated. It is the most effective practical detector for math-heavy text, outperforming the text-based Pangram 4 by 26.2 percentage points on frontier-model proofs.

5. We analyze AI usage trends in arXiv papers across fields and find that AI entered mathematics relatively late, but its use in mathematics is now growing faster and faster, with the share of mathematics passages detected as AI-generated rising from essentially zero before 2023 to 23.1% of prose and 17.6% of proofs in August 2026. Proofs still lag behind prose but are catching up quickly.

We will release our data, code and models, as well as a publicly available demo.

2 RELATED WORK

AI-Generated Text Detection. Post-hoc detectors of AI-generated text either score a text using statistics from a a language model, such as its likelihood or perplexity (Mitchell et al., 2023; Bao et al., 2024; Hans et al., 2024), or train a classifier using labeled human-written and AI-generated examples (Thai et al., 2026; Glickenhaus et al., 2026). The commercial Pangram 4 is the strongest and most widely used detector of the second kind, with a detection rate above 99% at a false-positive rate of 0.004% on its evaluation (Glickenhaus et al., 2026). Pangram, along with EditLens, go beyond a binary label by locating AI-written words or estimating the extent of AI editing (Glickenhaus et al., 2026; Thai et al., 2026). These detectors take text as input and are primarily developed and evaluated on natural-language prose. In math-heavy scientific writing, however, much of the information lies in the formatting of formulas, which may pose a challenge to such detectors. We evaluate the off-the-shelf detectors and find their performance to be limited, most markedly on proofs (Section 4.1).

Documents as Images. Our work builds on the recent trend of using vision-language models to process rendered images of text, a capability which has been steadily improving (Bai et al., 2025; Liu et al., 2024). PIXEL showed that language modeling can be built directly on the pixels of rendered text (Rust et al., 2022), ColPali retrieves documents from their page images without any text extraction (Faysse et al., 2025), and Glyph and LensVLM render long text into images to compress the context of a model (Cheng et al., 2026; Xie et al., 2026). For scientific document retrieval, Khalighinejad et al. (2026) find that whole-page image representations underperform the LaTeX-source form of the text and degrade more sharply as documents grow longer, which is consistent with our conclusion that the LaTeX source is the easiest representation to detect.

AI Detection from Images. Existing image-based detectors target whether the image itself was synthesized using AI, rather than whether the text shown in an image was written by AI (Wang et al., 2020; Gushchin et al., 2026; Stajduhar & Emi, 2026; Zhang et al., 2026; Wang et al., 2026). To our knowledge, the only other approach studying the latter setup is the concurrent work AIDoc (DeAlcala et al., 2026). However, its evaluation is limited to earlier open-weight generators, with reported AUROC scores of only 0.81-0.87, and does not establish effective detection at the low false-positive rates needed for practical deployment. Our work focuses on math-heavy scientific content, comparisons across input modalities, and detecting text generated by frontier models at extremely low false-positive rates. Another existing practice involves using OCR to recover text from images and then running detection on the result. Our experiments show that such OCR-recovered text embeds its own signatures and is difficult to detect for math-heavy content (Section 4.1).

3 SETUP & METHODS

3.1 DATASET

Source material. We extract texts from the LaTeX source of arXiv papers submitted between 2019 and 2021 (before ChatGPT was released) and never revised afterwards. This pool serves as our "Human" reference set and contains two kinds of texts: proofs and prose. A proof passage is the body of the proof of a theorem, lemma, proposition or corollary, taken from the LaTeX source as the text between `\begin{proof}` and `\end{proof}`. Prose is a paragraph that may contain inline mathematics and other LaTeX content such as references, but no theorem, proof, figure or table.

Paired regeneration. We flatten all LaTeX files of each paper into a single file that contains the complete LaTeX source. For each proof or prose passage, we prompt an LLM with the entire source of the paper except the target passage, and ask it to produce the LaTeX source of the missing passage in the style of the original paper. Each passage is regenerated by one of 15 LLMs: ten open-source models run locally with vLLM (Kwon et al., 2023) and five frontier models accessed through APIs (Appendix I). This results in a paired dataset of passages, where each passage has three modalities or representations: (1) the LaTeX source; (2) the PDF-extracted text, where each passage is compiled into its own PDF with the preamble and the macro definitions coming from the source paper, and the text is then extracted with PyMuPDF; and (3) the rendered image of the same PDF. We discard 2.4% of AI-generated proofs that are shorter than 50 words. Compiling each passage on its own with the preamble of its source paper succeeds for 94% of the pairs, which constitute the text and image versions of the dataset.

Real pages. We construct a variant that, we believe, is closer to real-world use. For each target passage, we place the model-generated content back at its original position in the flattened source and recompile the whole paper. This yields a real page on which the AI-generated passage is surrounded by human-written ones (an example is shown in Figure 7 in Appendix L). On these pages we crop windows that are as wide as the column and fully cover the target passage together with portions of human-written context. The AI fraction p of a window is defined as the ratio of the height of the target-passage text to the total height of the text within the window. A crop that covers exactly the target passage has $p = 1$ on the page with the AI-generated version and $p = 0$ on the page with the human-written original, and for the remaining windows p is continuously distributed in $(0, 1)$. We use this data to further train a *fraction estimator*, i.e., a model that predicts how much text in a window is AI (Appendix L). Those that contain only AI-generated text ($p = 1$) or only human-written text ($p = 0$) are also added to the training data of the text-based detector and SNAPJUDGE, so that the detectors become more robust to real-page artifacts such as line numbers and watermarks.

Multi-level AI-editing test set. To evaluate detection under varying degrees of AI editing, we construct a multi-level AI-editing test set from 1,500 human-written proofs and prose passages each, divided equally among the ten open-source generators. Each passage undergoes three editing exercises: sentence-level editing, whole-passage rewriting, and regeneration from an outline of its logical steps. Prompts are provided in Appendix I.

Splits and calibration pool. We split training, validation and test data by paper, and Table 1 lists their sizes. All validation pairs and 98.6% of the training pairs come from papers of 2019 and 2020, while all test pairs and the remaining 1.4% of the training pairs come from papers of 2021. For most passages, the training sets of the text-based detector and SNAPJUDGE additionally include the real-page versions described above. To set thresholds at low false-positive rates, we build calibration pools of about 350K human-written passages from papers of 2018 to 2020 (Table 1), one pool per representation. Each pool consists mostly of passages drawn from the body of papers, avoiding references, figures, tables, headers and footers, together with around 30K proofs.

	Train	Val	Test	Calib.
LaTeX	34.1k/30.2k	1.3k/2.2k	11.5k/12.2k	357.4k
Text	32.3k/30.1k	1.1k/1.9k	9.9k/11.9k	354.0k
Image	31.6k/30.0k	1.1k/1.7k	9.9k/11.8k	351.6k

Table 1: Dataset size: AI–human pairs (proofs / prose) per split, and human-written passages in the calibration pools.

Note on contamination. We acknowledge that source papers are likely included in LLM pretraining corpora, creating a risk of data contamination during both generation and subsequent detection (Magar & Schwartz, 2022). Specifically, there are two concerns: first, it is possible that AI mod-

els generate proofs that are verbatim reproductions of human-written ones, or are highly similar to them. To avoid such concerns during generation, we remove the 0.3% of AI-generated passages whose LaTeX source has more than 10% word-level 8-gram overlap with that of the original passage. Second, it is possible that any detector finetuned to flag AI-generated content may partly rely on whether its backbone encountered the text during pretraining. Unfortunately, this contamination risk is shared by all supervised detectors (including the baselines we compare against). Further, unlike other applications, contamination concerns are harder to alleviate for AI text detection, as there is simply no guarantee that any online text produced since 2022 is completely human written, so most detectors rely on pre-2022 content.

3.2 DETECTORS

We train a separate detector for each of the three representations, referred to as the LaTeX detector, the text-based detector and the image-based detector, which we call SNAPJUDGE, as well as an AI-fraction estimator that complements SNAPJUDGE on page images by predicting how much of an image is AI-generated (Appendix L). Each detector treats detection as binary classification at the passage level. Given one passage, it outputs the probability that the passage was generated by an LLM, and the passage is flagged as AI-generated when this score exceeds a threshold.

Backbone and training. The detectors use Qwen3.5 (Qwen Team, 2026) as the backbone. We fine-tune the attention and feed-forward projection matrices with LoRA (Hu et al., 2021) (the hyperparameters are given in Appendix A), and attach a linear classification head to the hidden state of the last token of the sequence. Training is done with the cross-entropy loss. The LaTeX detector and the text-based detector cut the input into overlapping windows of 512 tokens with a stride of 256 tokens, each window is one training example and inherits the label of the whole passage. SNAPJUDGE takes the whole rendered image as input. An image is encoded into visual tokens that each cover a 32×32 pixel patch, with a budget of about 1.25 megapixels in total. Around 6% of the images exceed this budget and are down-scaled proportionally before being fed to the model. Each detector is trained twice with different random seeds but otherwise identical settings. Reported detection rates are averaged across the two independently calibrated runs.

Scoring. At detection time, the LaTeX detector and the text-based detector divide the input into windows in the same way as in training. The passage score is the mean AI-class probability over its text windows or page images. All detection rates are obtained by thresholding the scores of the detector at a common false-positive rate on the human calibration pool described in Section 3.1. We assess score calibration in Appendix M.

3.3 BASELINES

For comparison, we select several off-the-shelf detectors and train our own shallow baselines. The shallow baselines are logistic-regression classifiers trained on the same examples as our other detectors, separately for each representation and for proofs and prose. We train three of them, using stylometric features (including sentence length, the frequency of punctuation and symbols, and vocabulary richness), TF-IDF features of character 3- to 5-grams, and TF-IDF features of word unigrams and bigrams, respectively. Among the off-the-shelf detectors, we select Binoculars (Hans et al., 2024) and Fast-DetectGPT (Bao et al., 2024) to represent zero-shot methods, which score a text by the difference in its likelihood under different language models. As supervised detectors, we choose the open-source EditLens (Thai et al., 2026) and the state-of-the-art Pangram 4 (Glickenhaus et al., 2026). Pangram is regarded as the most accurate detector available and is widely used to measure AI use in the wild, for example in newspaper articles (Russell et al., 2026), arXiv papers (Elazar & Antoniak, 2026) and self-published books (Chakrabarty et al., 2026).

4 RESULTS

4.1 DETECTION RATES UNDER THE THREE REPRESENTATIONS

In Table 2, we report the detection rates at a low false positive rate of 0.1% computed on a test set. For every detector, except Pangram, we set the threshold using the calibration pool of Section 3.1. For Pangram 4 we report both its own returned labels (AI and mixed are both counted as AI) and

	LaTeX				Text			
	Open avg.		Frontier avg.		Open avg.		Frontier avg.	
	Prose	Proofs	Prose	Proofs	Prose	Proofs	Prose	Proofs
Shallow baselines (features + logistic regression)								
stylometric	0.030	0.003	0.014	0.000	0.014	0.002	0.006	0.000
char TF-IDF	0.558	0.394	0.288	0.100	0.480	0.317	0.238	0.145
word TF-IDF	0.664	0.584	0.359	0.308	0.559	0.485	0.247	0.238
Off-the-shelf detectors								
Binoculars	0.101	0.093	0.002	0.006	0.002	0.037	0.001	0.001
Fast-DetectGPT	0.154	0.122	0.011	0.013	0.094	0.008	0.006	0.001
EditLens-Llama	0.657	0.086	0.340	0.007	0.525	0.014	0.245	0.000
EditLens-RoBERTa	0.268	0.093	0.116	0.022	0.191	0.021	0.082	0.008
Pangram v3.3.2	–	–	0.581	0.134	–	–	–	–
recalibrated	–	–	0.742	0.368	–	–	–	–
Pangram 4	–	–	0.797	0.363	–	–	0.694	0.324
recalibrated	–	–	0.908	0.875	–	–	0.871	0.656
OCR	–	–	–	–	–	–	0.660	0.250
OCR, recalibrated	–	–	–	–	–	–	0.860	0.500
Ours, 27B								
LaTeX / text-based	**0.943**	**0.920**	**0.957**	**0.938**	0.921	0.889	0.920	0.877
OCR(Tesseract)	–	–	–	–	0.892	0.867	0.885	0.861
SNAPJUDGE	–	–	–	–	**0.932**	**0.933**	**0.928**	**0.918**

Table 2: Detection rate at 0.1% FPR on the LaTeX source (left) and on inputs derived from the PDF (right), i.e, extracted text, OCR text (*OCR*) or the rendered page (SNAPJUDGE). *Open avg.* and *Frontier avg.* average over the ten open-source and the five frontier models. When the LaTeX source is not available, SNAPJUDGE is the most accurate detector, especially on frontier proofs. Owing to API cost, we evaluate Pangram on frontier models and its v3.3.2 only on LaTeX source.

the results obtained by re-thresholding its returned AI assistance score at a 0.1% false-positive rate. Owing to the Pangram API budget, we score only 2,500 proof pairs and 500 prose pairs, and its threshold for the 0.1% false-positive rate is determined on these 3,000 human-written passages.

The shallow baselines perform poorly, especially on proofs, where they still rely mostly on the text of a proof rather than mathematical symbols. Removing all mathematical symbols (inline and otherwise) from the texts and retraining the shallow baselines lowers their detection rate by only about 2% on proofs and 1% on prose, whereas keeping only these mathematical symbols makes them fail almost completely, suggesting that these baselines do not rely on the mathematical content. These shallow classifiers are not viable for detection, as the best of them detect only 0.308 of frontier proofs at a 0.1% false-positive rate.

Among the off-the-shelf detectors, only Pangram 4 clearly outperforms the shallow baselines. The two zero-shot detectors, Binoculars and Fast-DetectGPT, fail almost completely. EditLens detects part of the prose (0.245 on the text) but fails to detect proofs accurately. The state of the art, Pangram 4, detects only 0.324 of the proofs when given the text extracted from the PDF according to its own labels returned from the API, and even classifies 1 human-written passage as AI-assisted. After threshold recalibration, it's performance is 0.656, far below the >0.99 accuracy that it reports in other domains (Glickenhaus et al., 2026).

Of our own detectors, the LaTeX detector performs best, SNAPJUDGE comes close to it, and the text-based detector performs worst. This is in line with our expectation that LaTeX preserves the most formatting and text the least. We also experiment with an OCR-based approach, where we take the image representations of the passages and extract the text with two OCR tools, Tesseract (Smith, 2007) and Marker (Paruchuri, 2023). Neither tool works well, because OCR can add symbols that AI models commonly use to human-written text (Emi, 2025), which raises the detection threshold and thus lowers the detection rate. Table 2 reports the results of Tesseract, which performs somewhat better. On the representations available in practice, namely the text and the image, the frontier-family

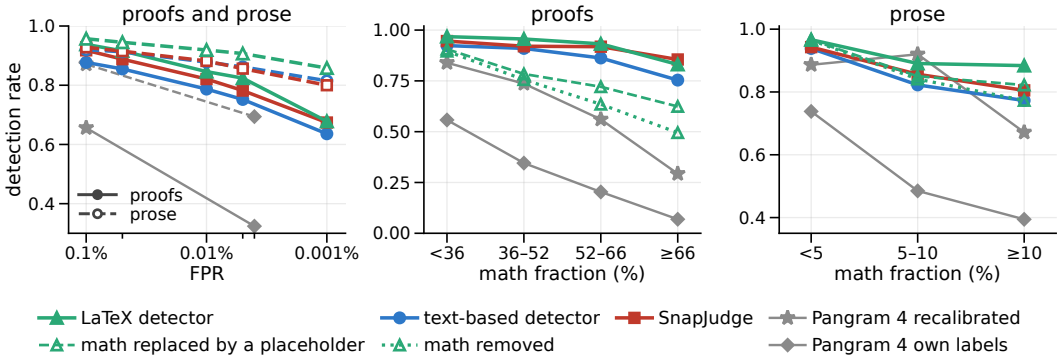


Figure 2: Detection rate of our 27B detectors on frontier passages. The left panel tightens the FPR from 0.1% to 0.001% for proofs (solid) and prose (dashed), and Table 8 lists the values. Pangram 4 is shown after recalibration at 0.1% and with its own labels at 0.004%, the FPR reported in its technical report. The right two panels split the passages by math fraction at 0.1% FPR, where the dashed and dotted green lines show the LaTeX detector on the same passages with every math segment replaced by a placeholder or with the math removed. Pangram 4 is scored on the text as written.

average of SNAPJUDGE (0.918 and 0.928) is far above the 0.656 and 0.871 that Pangram 4 reaches after calibration. Pangram 4 is especially weak on short passages, whereas our detectors remain accurate even below 100 words (Appendix C).

In Figure 2, we additionally report the results of our detectors at FPRs going as low as 0.001%. Pangram 4 reports a false-positive rate of 0.004% in its technical report (Glickenhaus et al., 2026), and we accordingly mark the values of Pangram in the figure as well. Even at these stricter operating points, which are often needed for large-scale commercial deployment, we find that SNAPJUDGE still detects above 0.67 of proofs and 0.80 of prose passages, far above Pangram. In addition, SNAPJUDGE still performs well at low resolution. Appendix H shows that after the images are down-scaled threefold, SNAPJUDGE still reaches a detection rate above 80%. Overall these results establish that: (i) AI-generated math-heavy writing, even from frontier models, can be detected with a high degree of accuracy; and (ii) image-based detection of text is a viable alternative to pure text-based detection when formatting may carry authorship signals.

4.2 DETECTION ACROSS MATHEMATICAL DENSITY AND PROOF QUALITY

Despite the high accuracy of our detectors, a natural question to ask is: what causes AI-generated math to be detectable? Our hypothesis is that, just like AI-generated prose carries signals based on word and phrase choices, mathematical writing also involves stylistic choices which do not impact the validity or correctness of the final output. In this section we rule out two alternative explanations.

First, we measure the amount of mathematics in a passage, which we define as the share of characters of the passage's LaTeX source that lie inside math mode. Figure 2 gives the detection rates across math-fraction bins, with proofs split at the quartiles of the test set and prose split at 5% and 10% (most prose passages have $<20\%$ math fraction). We find that, while the detectors all degrade as the math fraction rises, they remain above a 75% detection rate across the entire range. Moreover, removing the math content outright or replacing it with a placeholder for the LaTeX representation, leads to sharp drop in detection rate. Together these results suggest that, in addition to text, math equations and formulas also carry valuable signal which can help detect AI authorship.

Next, we examine whether our detectors rely on semantic signals which may be affected by the quality of the models involved, such as correctness or equivalence to human writing. We sample 1,100 proof pairs and 1,000 prose pairs generated by the five frontier models and present each pair to GPT-5.6 Sol (OpenAI, 2026a) to rate their similarity on a scale from 1 to 10. A pair comprises one human and one AI-generated proof (or prose, depending on the condition). We classify pairs with scores of 1–5 as non-equivalent and those with scores of 6–10 as equivalent. For proof pairs, the judge is additionally asked to judge whether each proof is correct or not. We show in Appendix K that GPT-5.6 Sol can distinguish correct proofs from incorrect ones (Table 16).

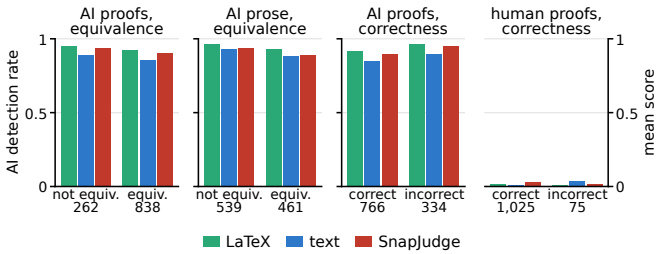


Figure 3: Detection rate at 0.1% FPR on subsets defined by GPT-5.6 (left three panels) and mean score of human-written proofs (right). Numbers below bars count the passages.

As Figure 3 shows, all three detectors still detect most AI proofs judged correct and most AI passages judged equivalent to the original, with rates only a few points below those on incorrect or non-equivalent passages. This small gap may arise because LLMs, when uncertain how to proceed, become less constrained by the content and fall back on characteristic stylistic patterns. On the human side, errors alone do not make a proof look AI-generated: human-written proofs that are judged to be incorrect are not flagged as AI by any of the three detectors at 0.1% FPR. Taken together, these results show that the detectors respond to how a passage is written rather than to its correctness or content.

4.3 Generalization to unseen generators and domains

We hold out each of GPT-5.1, o4-mini, Claude Sonnet 5 and Gemini 3.1 Pro in turn, train a detector on the remaining fourteen models and test it on the held-out model, using the Qwen3.5-9B backbone to keep the training cost manageable. Table 3 shows that our detectors still perform well on these held-out models. The LaTeX detector loses the most, 11.0 points on proofs and 6.9 on prose against 6.8 and 4.1 for SnapJudge. We conjecture that the LaTeX detector partly relies on markup patterns that differ among models, such as the choice between \(and $ for inline mathematics (Section 4.1), and such patterns do not generalize to an unseen model. Compilation renders these variants identically, so SnapJudge relies on what remains visible, such as the wording and the typeset formulas, which varies less across models and therefore, we speculate, it generalizes better.

	all 15 models in training		leave-one-model-out	
9B detector	Proofs	Prose	Proofs	Prose
LaTeX	0.855	**0.889**	0.745	**0.820**
text	0.754	0.849	0.669	0.808
SnapJudge	**0.880**	0.847	**0.812**	0.806

Table 3: Generalization of the 9B detectors at 0.1% FPR.

To test robustness to distribution shift, where human text can be confidently misclassified (Shen et al., 2026), we test our detectors on other public datasets to show that they perform equally well on unfamiliar domains. Table 4 reports the results on MIRAGE (Fu et al., 2025), the peer-review data of ICLR and NeurIPS (Yu et al., 2026), the subset of RAID generated by six chat models (Dugan et al., 2024), and the AI-generated part of the EditLens dataset (Thai et al., 2026). Under our own default thresholds, the false-positive rate is very low on every test set, no higher than 0.02%, and the detection rates are generally on-par with the best methods from the respective papers.

TPR at 0.1% FPR	MIRAGE	Peer review	RAID	EditLens
Binoculars	0.242	0.540	0.771	0.194
Fast-DetectGPT	0.033	0.459	0.479	0.332
EditLens-Llama	0.995	0.964	**0.849**	**1.000**
EditLens-RoBERTa	0.851	0.684	0.537	0.995
Benchmark paper	0.777‡	0.765	0.927‡	1.000
Ours (text)	**0.997**	**1.000**	0.823	0.998
SnapJudge	0.995	**1.000**	0.842	**1.000**

Table 4: Cross-domain evaluation on four public benchmarks. ‡: at 5% FPR.

We also check whether the scores of our detectors are well calibrated and find that SnapJudge performs best, with the lowest expected calibration error (Guo et al., 2017) on both our own test sets and the four external benchmarks of Table 4 (Appendix M).

4.4 Detection of AI-edited passages

We test our detectors on human-written texts with varying degrees of AI editing, using three public general-domain datasets, EditLens (Thai et al., 2026), MIRAGE (Fu et al., 2025) and APT-Eval (Saha & Feizi, 2025), together with our own multi-level AI-editing test set (Section 3.1).

Table 5 reports the Spearman correlation between the detector scores and the degree of AI editing, computed on each of the datasets above separately and then averaged. We use the same procedure as EditLens (Thai et al., 2026) for measuring editing degree, based on the cosine distance between the Linq-Embed-Mistral (Choi et al., 2024) embeddings of the human-written original and the edited passage. A cosine distance below 0.03 is recorded as 0, one above 0.15 as 1, and values between 0.03 and 0.15 are interpolated linearly, which gives the editing degree. The detection rate on each dataset and editing condition is listed in Table 9 (Appendix D).

Detector	Spearman ρ
Binoculars	0.420
Fast-DetectGPT	0.345
EditLens-Llama	**0.651**
EditLens-RoBERTa	0.561
Ours (text)	0.624
SnapJudge	0.649

Table 5: Spearman ρ with the degree of AI editing.

On these mixed datasets there is no fixed boundary between positives and negatives, and where to draw it depends on how much AI editing one is willing to accept. Although SNAPJUDGE is trained only on binary labels, the Spearman correlation between its scores and the degree of AI editing is about as high as that of EditLens, which is trained to predict this degree, and far higher than those of the zero-shot detectors. A user can therefore set their threshold accordingly based on how much AI editing they are willing to accept.

4.5 AI-GENERATED TEXT ON ARXIV OVER TIME

As a concrete application, we use the LaTeX detector to study how the share of AI-generated text in arXiv papers has changed over time. We sample arXiv papers with theorem proofs from every month between 2018 and 2026 and take one randomly sampled proof and prose passage from each, grouping the papers into mathematics, computer science and other fields(Appendix E). The detector uses the threshold for a false-positive rate of 0.01% on the calibration pool (Section 3.1). Detection rates on text before ChatGPT release confirms that FPR is extremely low in practice. Among the proofs and prose passages of 2018, 2019 and 2021, only 0.008% of the proofs and 0.003% of the prose are falsely flagged.

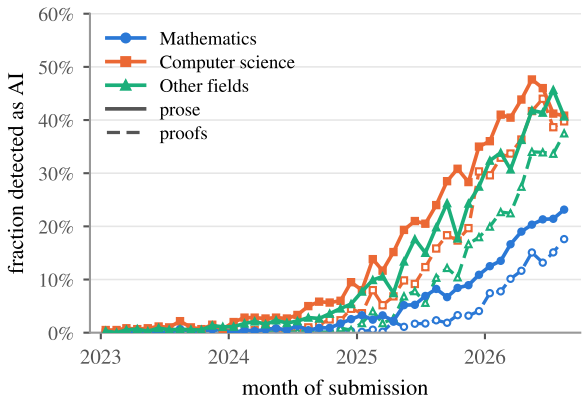


Figure 4: Share of arXiv passages flagged as AI by the LaTeX detector (tuned to have a 0.01% FPR).

Figure 4 shows that AI entered mathematics relatively late. Its use in mathematics is nevertheless growing rapidly, and since the beginning of 2026 the curves for both prose and proofs have become markedly steeper. By August 2026, 17.6% of proofs are detected as AI-generated. AI-generated math-heavy text is therefore no longer a rare case, which underscores the need for detectors that work reliably on math-heavy content.

5 CONCLUSION

In this work, we curated a dataset comprising coupled human-written and AI-generated proofs and prose. We explored training detectors using three different input representations: LaTeX, text extracted from PDF, and rendered images. Although it was unclear a priori if AI-generated math-heavy text would be detectable, our experiments consistently show that it is. In fact, our detectors are viable for practical use and outperform the commercial state-of-the-art detector, Pangram 4, on scientific text. Further, we demonstrated that our detectors generalize to unseen models and domains, and its scores are correlated with the extent of AI editing. Lastly, we analyzed recent trends related to AI-use in arXiv papers, and find AI use in mathematical content to be rapidly rising. Our work lays the foundation for interesting future work around building detectors that use multiple representations, and devising mechanisms to quantify AI contributions at a finer-level of detail.

6 AI USE STATEMENT

We used AI agents to assist in finding related work, brainstorming research ideas, writing code, scheduling training jobs, and computing experimental results. We also used LLMs to polish and revise parts of the text and to help produce the figures. We have manually reviewed all the outputs generated by AI models, and take full responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Anthropic. Introducing Claude Sonnet 5. `https://www.anthropic.com/news/claude-sonnet-5`, 2026.

Artifex Software. PyMuPDF: Python bindings for MuPDF. `https://github.com/pymupdf/PyMuPDF`, 2026. Version 1.28.0, accessed 2026-09-06.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

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 *International Conference on Learning Representations*, volume 2024, pp. 24814–24836, 2024.

Tuhin Chakrabarty, Xinyue Liu, Jane C Ginsburg, and Paramveer Dhillon. Generative ai floods and dilutes the market for books. *arXiv preprint arXiv:2607.20349*, 2026.

Jiale Cheng, Yusen Liu, Xinyu Zhang, Yulin Fei, Wenyi Hong, Ruiliang Lyu, Weihan Wang, Zhe Su, Xiaotao Gu, Xiao Liu, et al. Glyph: Scaling context windows via visual-text compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37145–37158, 2026.

Chanyeol Choi, Junseong Kim, Seolhwa Lee, Jihoon Kwon, Sangmo Gu, Yejin Kim, Minkyung Cho, and Jy-yong Sohn. Linq-embed-mistral technical report. *arXiv preprint arXiv:2412.03223*, 2024.

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

Daniel DeAlcala, Ching-Yun Ko, Aythami Morales, Julian Fierrez, Ruben Tolosana, and Pin-Yu Chen. AIDoc: A framework for generating and identifying AI-edited documents. `https://danieldealcala.github.io/assets/pdf/DocAI.pdf`, 2026. Manuscript.

Liam Dugan, Alyssa Hwang, Filip Trhlík, Andrew Zhu, Josh Magnus Ludan, Hainiu Xu, Daphne Ippolito, and Chris Callison-Burch. Raid: A shared benchmark for robust evaluation of machine-generated text detectors. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12463–12492, 2024.

Yanai Elazar and Maria Antoniak. Llm-generated or human-written? comparing review and non-review papers on arxiv. *arXiv preprint arXiv:2601.17036*, 2026.

Bradley Emi. Pangram predicts 21% of ICLR reviews are AI-generated. `https://www.pangram.com/blog/pangram-predicts-21-of-iclr-reviews-are-ai-generated`, 2025. Pangram Labs blog, November 18, 2025.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. Colpali: Efficient document retrieval with vision language models. In *International Conference on Learning Representations*, volume 2025, pp. 61424–61449, 2025.

Jiachen Fu, Chun-Le Guo, and Chongyi Li. Detectanyllm: towards generalizable and robust detection of machine-generated text across domains and models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 11229–11238, 2025.

Deyan Ginev, Brian Caruso, Bruce Miller, Jeff Sank, and Jacob Weiskoff. Scaling accessible mathematics on arxiv: Html conversion and mathml 4. 2026.

Ben Glickenhaus, Katherine Thai, Jenna Russell, Elyas Masrour, Yue Han, Max Spero, and Bradley Emi. Pangram 4 technical report. *arXiv preprint arXiv:2607.27183*, 2026.

Google DeepMind. Gemini 3.1 Pro model card. `https://deepmind.google/models/model-cards/gemini-3-1-pro/`, 2026.

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

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Aleksandr Gushchin, Khaled Abud, Ekaterina Shumitskaya, Artem Filippov, Georgii Bychkov, Sergey Lavrushkin, Mikhail Erofeev, Anastasia Antsiferova, Changsheng Chen, Shunquan Tan, et al. Ntire 2026 challenge on robust ai-generated image detection in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1895–1913, 2026.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Spotting llms with binoculars: Zero-shot detection of machine-generated text. 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. 2021.

ICLR. Policies on large language model usage at ICLR 2026. `https://iclr.cc/FAQ/LLM`, 2025.

Ghazal Khalighinejad, Raghuveer Thirukovalluru, Alexander H Oh, and Bhuwan Dhingra. Document-as-image representations fall short for scientific retrieval. *arXiv preprint arXiv:2604.18508*, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Ouxiang Li, Jiayin Cai, Yanbin Hao, Xiaolong Jiang, Yao Hu, and Fuli Feng. Improving synthetic image detection towards generalization: An image transformation perspective. In *Proceedings of the 31st ACM SIGKDD conference on knowledge discovery and data mining v. 1*, pp. 2405–2414, 2025.

Weixin Liang, Yaohui Zhang, Zhengxuan Wu, Haley Lepp, Wenlong Ji, Xuandong Zhao, Hancheng Cao, Sheng Liu, Siyu He, Zhi Huang, et al. Quantifying large language model usage in scientific papers. *Nature Human Behaviour*, 9(12):2599–2609, 2025.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024.

Thang Luong and Edward Lockhart. Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad. https://deepmind.google/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/, 2025. Google DeepMind blog, July 21, 2025.

Inbal Magar and Roy Schwartz. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 157–165, 2022.

Meta. Llama 4 Scout model card (Llama-4-Scout-17B-16E). https://huggingface.co/meta-llama/Llama-4-Scout-17B-16E, 2025. Model card, released April 5, 2025.

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

Arati Mohapatra and Danish Pruthi. Expectations and practices around ai disclosure in cs research. *arXiv preprint arXiv:2608.23271*, 2026.

Nature Portfolio, Springer Nature. Artificial intelligence (AI): Editorial policies. https://www.nature.com/nature-portfolio/editorial-policies/ai, 2026. Accessed 2026-09-06.

NeurIPS. NeurIPS 2025 policy on the use of large language models. https://neurips.cc/Conferences/2025/LLM, 2025.

NeurIPS. AI-generated papers in the NeurIPS 2026 position paper track. https://blog.neurips.cc/2026/06/02/ai-generated-papers-in-the-neurips-2026-position-paper-track/, 2026.

Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. Towards universal fake image detectors that generalize across generative models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24480–24489. IEEE, 2023.

OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, 2025a. Blog post, April 14, 2025.

OpenAI. GPT-5.1 Instant and GPT-5.1 Thinking system card addendum. https://openai.com/index/gpt-5-system-card-addendum-gpt-5-1/, 2025b. System card addendum, November 13, 2025.

OpenAI. OpenAI o3 and o4-mini system card. https://openai.com/index/o3-o4-mini-system-card/, 2025c. System card, April 16, 2025.

OpenAI. GPT-5.6 system card. https://deploymentsafety.openai.com/gpt-5-6/gpt-5-6.pdf, 2026a.

OpenAI. On the Navier–Stokes Millennium Prize Problem. https://openai.com/index/navier-stokes-solution/, 2026b. Announcement, September 8, 2026; 166-page manuscript with accompanying Lean proof files, independent verification pending.

Jeongsoo Park and Andrew Owens. Community forensics: Using thousands of generators to train fake image detectors. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8245–8257. IEEE, 2025.

Vik Paruchuri. Marker. https://github.com/datalab-to/marker, 2023. Software repository. Accessed 2026-09-15.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,

Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. 2025. URL https://arxiv.org/abs/2412.15115.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, February 2026.

Jenna Russell, Marzena Karpinska, Destiny Akinode, James Zhou, Katherine Thai, Bradley Emi, Max Spero, and Mohit Iyyer. Ai use in american newspapers is widespread, uneven, and rarely disclosed. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14554–14580, 2026.

Phillip Rust, Jonas F Lotz, Emanuele Bugliarello, Elizabeth Salesky, Miryam de Lhoneux, and Desmond Elliott. Language modelling with pixels. 2022.

Shoumik Saha and Soheil Feizi. Almost ai, almost human: The challenge of detecting ai-polished writing. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25414–25431, 2025.

Zhuoer Shen, Mingyi Wang, Shaofeng Zou, and Yuheng Bu. Rethinking ai-generated text detection: A strong baseline and the distribution-shift problem that remains. *arXiv preprint arXiv:2607.03680*, 2026.

Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pp. 629–633. IEEE, 2007.

Rachel Stajduhar and Bradley Emi. Introducing Pangram Image detection (research preview). https://www.pangram.com/blog/introducing-pangram-image-detection, 2026. Pangram Labs blog, July 29, 2026.

Chuangchuang Tan, Yao Zhao, Shikui Wei, Guanghua Gu, Ping Liu, and Yunchao Wei. Rethinking the up-sampling operations in cnn-based generative network for generalizable deepfake detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 28130–28139, 2024.

Terence Tao. A digestion of the Jacobian conjecture counterexample. https://terrytao.wordpress.com/2026/07/21/a-digestion-of-the-jacobian-conjecture-counterexample/, 2026. Blog post, What's new, July 21, 2026. Discusses the three-variable counterexample presented by Levent Alpöge on July 19, 2026 and credited to Claude Fable 5.

Katherine Thai, Bradley Emi, Elyas Masrour, and Mohit Iyyer. Editlens: Quantifying the extent of ai editing in text. In *International Conference on Learning Representations*, volume 2026, pp. 122997–123024, 2026.

Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8692–8701. IEEE, 2020.

Yijin Wang, Shuyi Wang, Wenhan Zhang, and Yuqi Ouyang. A multi-domain benchmark for detecting ai-generated text-rich images from gpt-image-2. *arXiv preprint arXiv:2606.19259*, 2026.

Roy Xie, Dan Friedman, Donghan Yu, Bowen Pan, Christopher Fifty, Jang-Hyun Kim, Xianzhi Du, Zhe Gan, Vivek Rathod, and Bhuwan Dhingra. Lensvlm: Selective context expansion for compressed visual representation of text. *arXiv preprint arXiv:2605.07019*, 2026.

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

Sungduk Yu, Man Luo, Avinash Madasu, Vasudev Lal, and Phillip Howard. Is your paper being reviewed by an llm? benchmarking ai text detection in peer review. In *International Conference on Learning Representations*, volume 2026, pp. 61517–61553, 2026.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

Yuning Zhang, Changtao Miao, Mingyu Liao, Tingyu Liu, Xinghao Wang, Tao Gong, Qi Chu, and Nenghai Yu. Textfake: Benchmarking ai-generated image detection on text-rich images. *arXiv preprint arXiv:2606.01050*, 2026.

A Training Details and Hyperparameters

27B models	LaTeX	Text	SnapJudge	OCR	AI-fraction estimator
backbone	Qwen3.5-27B	Qwen3.5-27B	Qwen3.5-27B	Qwen3.5-27B	Qwen3.5-27B
input	LaTeX source, 512-token windows, stride 256	PyMuPDF text, 512-token windows, stride 256	rendered image, visual tokens of 32×32 pixels	OCR text of the rendered image, 512-token windows, stride 256	rendered image
instruction	none	none	"Determine whether this mathematical proof was written by a human or generated by AI."	none	"Estimate how much of this scientific page excerpt was written by AI."
head output	2 classes	2 classes	2 classes	2 classes	5 buckets
LoRA rank r	8	8	16	8	16
learning rate	10^{-4}	10^{-4}	2×10^{-4}	10^{-4}	2×10^{-4}
effective batch size	16	16	64	16	64
epochs	up to 3, best on validation	up to 3, best on validation	up to 3, best on validation	up to 3, best on validation	up to 3, best on validation
seeds	42, 7	42, 7	42, 7	42	42
hardware, wall-clock	4 RTX Pro 6000, 18 h	4 H200, 12 h	4 H200, 10 h	4 H200, 5 h	8 H200, 4.3 h

9B models	LaTeX	Text	Image
backbone	Qwen3.5-9B	Qwen3.5-9B	Qwen3.5-9B
input	LaTeX source, 512-token windows, stride 256	PyMuPDF text, 512-token windows, stride 256	rendered image, visual tokens of 32×32 pixels
instruction	none	none	as SnapJudge 27B
head output	2 classes	2 classes	2 classes
LoRA rank r	8	8	16
learning rate	10^{-4}	10^{-4}	2×10^{-4}
effective batch size	16	16	64
epochs	up to 3, best on validation	up to 3, best on validation	up to 3, best on validation
seeds	7 (all 15 models), 42 and 7 (held out)	7 (all 15 models), 42 and 7 (held out)	7
hardware, wall-clock	2 H200, ≈5h	2 H200, ≈5h	2 H200, ≈5h

Table 6: Training settings of the 27B models behind our main results (top) and of the 9B detectors of Section 4.3 (bottom).

B Detection Rates by Generator Family

Table 7 expands Table 2: it lists, for every detector, the detection rate on each of the five frontier families that the main table averages into its last column.

Table 8 lists the values plotted in Figure 2.

		Proofs					Prose				
	rep.	GPT-4.1	GPT-5.1	o4-mini	Claude	Gemini	GPT-4.1	GPT-5.1	o4-mini	Claude	Gemini
LaTeX											
Shallow baselines (features + logistic regression)											
stylometric	LaTeX	0.000	0.000	0.000	0.000	0.000	0.026	0.002	0.001	0.008	0.030
char TF-IDF	LaTeX	0.227	0.030	0.093	0.065	0.088	0.661	0.119	0.145	0.098	0.418
word TF-IDF	LaTeX	**0.404**	**0.268**	**0.333**	**0.292**	**0.243**	**0.729**	**0.229**	**0.211**	**0.164**	**0.460**
Off-the-shelf detectors											
Binoculars	LaTeX	0.014	0.000	0.000	0.000	0.014	0.004	0.000	0.000	0.000	0.006
Fast-DetectGPT	LaTeX	0.035	0.012	0.000	0.000	0.016	0.020	0.002	0.000	0.008	0.026
EditLens-Llama	LaTeX	0.004	0.004	0.000	0.004	0.024	0.814	0.119	0.105	0.194	0.471
EditLens-RoBERTa	LaTeX	0.039	0.004	0.011	0.019	0.039	0.285	0.035	0.019	0.041	0.199
Pangram v3.3.2	LaTeX	0.098	0.043	0.389	0.095	0.045	0.860	0.546	0.716	0.333	0.450
calibrated	LaTeX	0.339	0.226	0.719	0.339	0.220	0.946	0.695	0.936	0.485	0.649
Pangram 4	LaTeX	0.355	0.307	0.527	0.307	0.319	0.950	0.814	0.929	0.546	0.747
calibrated	LaTeX	0.876	0.890	0.957	0.898	0.756	0.990	0.938	0.990	0.773	0.848
Ours, 9B											
leave-one-model-out	LaTeX	–	0.736	0.944	0.763	0.536	–	0.908	0.976	0.639	0.755
all 15 models	LaTeX	0.920	0.907	0.959	0.851	0.702	0.990	0.936	0.985	0.792	0.845
all 15 models, 0.01% FPR	LaTeX	0.790	0.730	0.842	0.691	0.545	0.982	0.884	0.962	0.702	0.744
Ours, 27B											
all 15 models	LaTeX	0.962	0.966	0.989	0.935	0.835	0.995	0.978	0.997	0.885	0.930
all 15 models, 0.01% FPR	LaTeX	0.845	0.912	0.952	0.856	0.666	0.991	0.951	0.990	0.819	0.844
all 15 models, 0.001% FPR	LaTeX	0.639	0.745	0.835	0.689	0.485	0.977	0.876	0.966	0.713	0.757
Text and image											
Shallow baselines (features + logistic regression)											
stylometric	text	0.000	0.000	0.000	0.000	0.000	0.010	0.002	0.002	0.006	0.011
char TF-IDF	text	0.202	0.087	0.032	0.183	0.220	**0.596**	0.094	0.107	0.088	**0.305**
word TF-IDF	text	**0.343**	**0.240**	**0.081**	**0.252**	**0.272**	0.578	**0.121**	**0.130**	**0.107**	0.297
Off-the-shelf detectors											
Binoculars	text	0.000	0.000	0.000	0.000	0.005	0.002	0.002	0.001	0.002	0.000
Fast-DetectGPT	text	0.002	0.002	0.000	0.000	0.000	0.016	0.000	0.001	0.004	0.011
EditLens-Llama	text	0.000	0.000	0.000	0.002	0.000	0.611	0.077	0.079	0.120	0.341
EditLens-RoBERTa	text	0.009	0.002	0.005	0.017	0.007	0.217	0.028	0.024	0.034	0.106
Pangram 4	text	0.278	0.347	0.387	0.351	0.259	0.870	0.763	0.745	0.443	0.646
calibrated	text	0.628	0.628	0.663	0.757	0.604	0.950	0.907	0.949	0.711	0.838
Ours, 9B, text											
leave-one-model-out	text	–	0.775	0.658	0.679	0.564	–	0.893	0.898	0.676	0.765
all 15 models	text	0.877	0.780	0.742	0.774	0.718	0.992	0.896	0.898	0.766	0.835
all 15 models, 0.01% FPR	text	0.693	0.561	0.456	0.568	0.557	0.964	0.787	0.806	0.675	0.693
SnapJudge, 9B											
leave-one-model-out	SnapJudge	–	0.944	0.775	0.859	0.668	–	0.893	0.920	0.660	0.752
all 15 models	SnapJudge	0.933	0.943	0.866	0.926	0.784	0.980	0.893	0.950	0.726	0.818
all 15 models, 0.01% FPR	SnapJudge	0.861	0.869	0.729	0.858	0.633	0.969	0.834	0.891	0.599	0.719
Ours, 27B, text											
all 15 models	text	0.913	0.918	0.848	0.928	0.780	0.982	0.938	0.953	0.832	0.895
all 15 models, 0.01% FPR	text	0.827	0.825	0.752	0.866	0.665	0.969	0.906	0.923	0.757	0.835
all 15 models, 0.001% FPR	text	0.662	0.659	0.601	0.748	0.507	0.959	0.852	0.859	0.680	0.725
SnapJudge, 27B											
all 15 models	SnapJudge	0.958	0.961	0.900	0.951	0.818	0.989	0.954	0.971	0.838	0.890
all 15 models, 0.01% FPR	SnapJudge	0.886	0.886	0.764	0.880	0.690	0.982	0.914	0.936	0.755	0.825
all 15 models, 0.001% FPR	SnapJudge	0.754	0.771	0.562	0.751	0.532	0.966	0.836	0.872	0.627	0.699

Table 7: Detection rate for each frontier family at 0.1% FPR on proofs and narrative prose; rows marked 0.01% and 0.001% FPR repeat our detectors at those FPRs.

		Proofs		Prose	
Detector	FPR	Frontier avg.	Open avg.	Frontier avg.	Open avg.
LaTeX	0.1%	0.938	0.920	0.957	0.943
	0.05%	0.916	0.900	0.945	0.938
	0.01%	0.846	0.849	0.919	0.924
	0.005%	0.824	0.832	0.907	0.918
	0.001%	0.678	0.745	0.858	0.893
text	0.1%	0.877	0.889	0.920	0.921
	0.05%	0.855	0.873	0.913	0.916
	0.01%	0.787	0.828	0.878	0.901
	0.005%	0.752	0.811	0.862	0.894
	0.001%	0.636	0.747	0.815	0.871
SnapJudge	0.1%	0.918	0.933	0.928	0.932
	0.05%	0.888	0.918	0.915	0.926
	0.01%	0.821	0.876	0.882	0.910
	0.005%	0.782	0.852	0.856	0.903
	0.001%	0.674	0.800	0.800	0.883
Pangram 4, LaTeX	0.1% (recalibrated)	0.875	–	0.908	–
	0.004% (own labels)	0.363	–	0.797	–
Pangram 4, text	0.1% (recalibrated)	0.656	–	0.871	–
	0.004% (own labels)	0.324	–	0.694	–

Table 8: Values behind Figure 2: detection rate of the 27B detectors at five false-positive rates, as the frontier models average plotted in the figure and as the average over the ten open-source models, together with the Pangram 4 values on the text layer plotted in the figure and the corresponding values on the LaTeX source; 0.004% is the false-positive rate that Pangram 4 reports for its own labels.

C Detection Rates by Passage Length

Figure 5 splits the frontier test passages of Table 2 by the length of the AI-generated passage, and keeps the thresholds of Table 2. Our detectors lose some accuracy on short passages but still detect more than 80% of proofs and 88% of prose below 100 words. Pangram 4 depends much more on length.

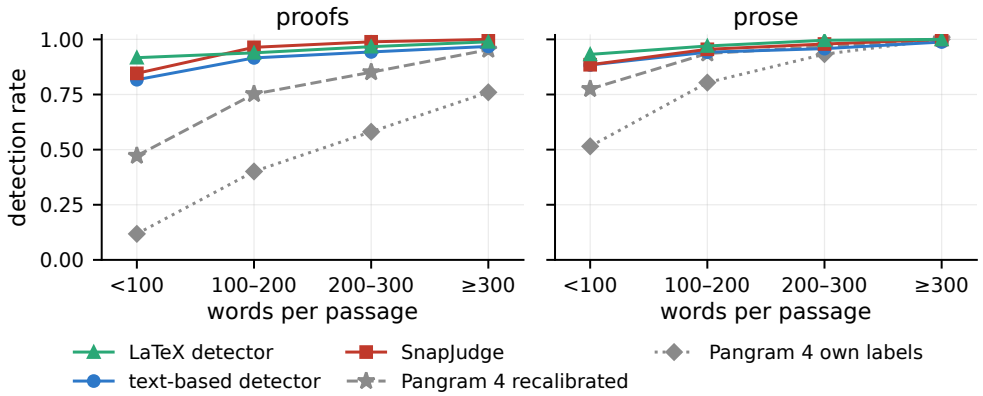


Figure 5: Detection rate at 0.1% FPR by passage length in words, frontier models average; Pangram 4 on the text input.

D Detection Rates on AI-Edited Datasets

Table 9 lists the detection rates of all detectors for every dataset and editing condition.

	EditLens Llama	EditLens RoBERTa	Fast-DetectGPT	Binoculars	Ours, text	SnapJudge
EditLens test set						
AI-edited	0.593	0.382	0.025	0.009	0.204	0.249
MIRAGE						
polish	0.706	0.254	0.002	0.015	0.439	0.393
rewrite	0.757	0.248	0.002	0.013	0.398	0.354
APT-Eval						
polish, extreme minor	0.078	0.026	0.015	0.046	0.030	0.029
10% of sentences polished	0.061	0.026	0.006	0.034	0.020	0.019
polish, minor	0.129	0.030	0.013	0.064	0.044	0.032
50% of sentences polished	0.260	0.066	0.003	0.043	0.074	0.052
polish, slight major	0.456	0.104	0.011	0.091	0.162	0.142
polish, major	0.651	0.260	0.014	0.182	0.440	0.408
Our multi-level AI-editing test set, proofs						
sentence-level rewrite	0.011	0.020	0.000	0.002	0.005	0.007
paragraph rewrite	0.015	0.018	0.008	0.010	0.076	0.077
two-stage generation	0.192	0.222	0.232	0.145	0.891	0.937
Our multi-level AI-editing test set, prose						
sentence-level rewrite	0.051	0.015	0.004	0.001	0.036	0.017
paragraph rewrite	0.071	0.014	0.008	0.005	0.131	0.067
two-stage generation	0.659	0.209	0.048	0.027	0.956	0.946

Table 9: Detection rate on AI-edited passages. Off-the-shelf detectors are held at 0.1% FPR; our detectors are at 0.01% FPR.

E MONTHLY DATA OF THE ARXIV TREND

For each month from 2018 to 2026 we sample arXiv papers that contain a theorem proof and take one proof and one prose passage from each paper. We skip 2020, because most of our training data come from papers of that year, and use the months of 2018, 2019 and 2021 to check how often the detector flags text written before the release of ChatGPT. Papers are assigned to mathematics, computer science (including eess and stat.ML) or other fields by their primary arXiv category. Up to 2022, each month contains about 1,000 papers, most of them from mathematics. From 2023 on, we add papers from computer science and other fields, so that each field contributes at least about 300 papers per month, and the months of 2026 contain more papers from every field. Table 10 lists, for each month from 2023 to 2026, the number of papers and the shares of prose and proofs detected as AI in each field, and Table 11 summarizes all months by year.

	Mathematics			Computer science			Other fields		
month	n	prose	proofs	n	prose	proofs	n	prose	proofs
2023-01	626	0.0	0.2	600	0.5	0.0	322	0.3	0.0
2023-02	583	0.0	0.2	600	0.5	0.0	304	0.0	0.0
2023-03	638	0.0	0.0	600	0.8	0.0	381	0.5	0.0
2023-04	645	0.2	0.2	600	0.7	0.2	299	0.7	0.3
2023-05	566	0.2	0.2	600	0.8	0.0	367	0.3	0.0
2023-06	613	0.2	0.0	600	1.2	0.2	349	0.9	0.0
2023-07	611	0.0	0.0	600	1.0	0.2	370	0.5	0.0
2023-08	692	0.7	0.0	600	2.2	0.2	308	0.6	0.0
2023-09	653	0.3	0.2	600	1.0	0.3	353	0.6	0.0
2023-10	616	0.3	0.0	600	0.7	0.2	397	0.5	0.0
2023-11	652	0.6	0.0	600	1.5	0.0	373	1.3	0.0
2023-12	648	0.5	0.0	600	0.7	0.5	366	1.1	0.0
2024-01	676	0.0	0.0	600	2.0	0.5	353	1.1	0.0
2024-02	583	0.5	0.0	600	2.8	0.0	405	1.7	0.0
2024-03	642	0.3	0.2	600	2.8	0.5	378	2.1	0.3
2024-04	674	0.6	0.0	600	2.7	1.0	381	1.6	0.3
2024-05	594	0.8	0.0	600	2.8	0.5	432	2.3	0.0
2024-06	595	0.5	0.0	600	2.7	0.3	382	1.8	0.0
2024-07	622	1.3	0.0	600	3.3	0.8	460	2.2	0.2
2024-08	654	0.5	0.2	600	5.0	1.0	387	2.8	0.3
2024-09	662	0.9	0.2	600	5.8	1.5	374	2.7	0.0
2024-10	550	0.9	0.0	600	5.7	2.5	560	3.6	0.2
2024-11	636	1.7	0.6	600	6.0	2.3	461	4.6	0.9
2024-12	666	2.6	0.3	600	9.5	4.5	409	5.4	0.5
2025-01	639	3.3	0.2	600	8.0	3.7	388	7.7	1.8
2025-02	531	2.4	0.6	600	13.8	8.0	323	9.9	4.0
2025-03	583	3.3	0.2	600	11.7	5.2	388	10.6	1.8
2025-04	628	2.2	2.1	600	15.2	6.8	336	7.4	2.7
2025-05	637	5.2	1.1	600	19.3	9.8	351	13.4	6.8
2025-06	534	5.2	1.7	600	21.0	9.2	358	17.6	7.8
2025-07	637	6.9	1.7	600	20.5	12.3	426	15.0	5.6
2025-08	644	8.2	2.3	600	24.0	15.8	348	19.8	10.3
2025-09	643	6.7	1.9	600	28.5	18.3	435	24.4	12.2
2025-10	546	8.4	3.3	600	30.8	17.3	547	17.7	10.4
2025-11	593	8.9	3.2	600	28.3	19.7	420	24.3	16.7
2025-12	615	10.9	4.1	600	35.0	30.3	484	27.5	18.0
2026-01	2,462	12.5	7.4	1,091	36.0	29.6	516	32.4	20.0
2026-02	2,264	13.5	7.8	1,741	41.0	32.9	520	33.8	22.7
2026-03	2,200	16.6	10.1	1,184	40.5	33.7	534	30.7	22.5
2026-04	2,404	19.0	11.6	1,343	43.9	36.3	568	36.3	27.5
2026-05	2,293	20.3	15.1	1,816	47.6	41.7	579	41.8	34.0
2026-06	2,444	21.3	13.2	1,302	46.0	44.0	563	41.4	33.9
2026-07	2,913	21.4	15.1	1,291	41.2	38.7	600	45.7	33.7
2026-08	3,119	23.1	17.6	1,303	40.8	39.8	600	40.7	37.5

Table 10: Monthly data behind Figure 4. For each field, n is the number of papers, and the prose and proof columns give the share (%) of passages detected as AI by the LaTeX detector at 0.01% FPR.

	Prose			Proofs		
year	math	CS	other	math	CS	other
2018	0.0	0.0	0.0	0.0	0.0	0.0
2019	0.0	0.0	0.0	0.0	0.0	0.0
2021	0.0	0.0	0.0	0.0	0.0	0.0
2022	0.0	0.1	0.1	0.0	0.0	0.0
2023	0.3	1.0	0.6	0.1	0.1	0.0
2024	0.9	4.3	2.7	0.1	1.3	0.2
2025	6.0	21.3	16.8	1.9	13.0	8.6
2026	18.8	42.5	38.1	12.5	37.3	29.3

Table 11: Share (%) of passages detected as AI by the LaTeX detector at 0.01% FPR, by year and field.

F Proof-Carrying Submissions at ICLR

The statistics quoted in the introduction come from the public OpenReview records of the seven ICLR editions from 2020 to 2026. A submission counts as proof-carrying if its PDF contains at least one passage that opens with a proof marker ("Proof." or "Proof of ...") and is closed by an end-of-proof mark. Table 12 gives, per year, the mean review score and the acceptance rate of such submissions against the other submissions of the same year. Proof-carrying submissions receive higher review scores in every year, with all seven differences significant (z from 3.8 to 16.7), and higher acceptance rates in every year, significant in six of the seven years (2021: $z = 1.8$).

Moreover, the share of papers that carry proofs keeps rising, especially among theory-oriented papers (Table 13). The acceptance premium of carrying proofs is likewise concentrated in the nine areas, at 11 to 15 percentage points in every year ($p < 2 \times 10^{-13}$).

		mean review score			acceptance rate (%)		
Year	submissions	with proofs	without	Δ (z)	with proofs	without	Δ (z)
2020	2,213	4.64	4.34	+0.30 (3.8)	36.5	29.1	+7.4 (3.3)
2021	2,594	5.65	5.46	+0.19 (4.4)	35.9	32.1	+3.8 (1.8)
2022	2,617	5.77	5.41	+0.36 (7.0)	47.2	39.3	+7.9 (3.8)
2023	3,792	5.68	5.42	+0.26 (5.4)	46.7	40.1	+6.6 (3.4)
2024	7,404	5.44	5.05	+0.39 (11.8)	46.0	36.2	+9.8 (6.7)
2025	11,672	5.45	5.06	+0.39 (14.8)	47.2	40.4	+6.8 (5.6)
2026	19,814	4.46	4.13	+0.33 (16.7)	39.6	34.2	+5.4 (6.0)

Table 12: Review scores and acceptance rates of proof-carrying versus other ICLR submissions, 2020–2026. Δ is the difference between proof-carrying and other submissions, with the two-sample z statistic in parentheses; $|z| > 1.96$ corresponds to significance at the 5% level (two-sided).

Primary area	share of submissions with proofs 2024	2025	2026
The nine high-proof areas (share $\geq 25\%$ in 2026)			
learning theory	69.0%	71.5%	78.4%
causal reasoning	49.5%	56.3%	58.5%
optimization	50.6%	57.8%	57.2%
probabilistic methods (Bayesian methods, sampling, UQ)	40.6%	42.4%	51.6%
learning on graphs and other geometries	45.6%	45.9%	44.0%
reinforcement learning	36.6%	41.6%	39.9%
other topics in machine learning	–	29.0%	29.8%
transfer, meta and lifelong learning	18.4%	17.4%	25.6%
unsupervised, self-supervised, semi-supervised and supervised representation learning	23.6%	25.2%	25.5%
the nine areas together	33.1%	39.9%	41.5%
All other areas	13.3%	13.6%	14.2%
Acceptance gap of carrying proofs (percentage points)			
the nine areas together	+15.4	+13.0	+11.1
all other areas	+4.2	+4.4	+4.9

Table 13: Proof-carrying submissions by research area, 2024–2026.

G AI-Generated Image Detectors on Rendered Pages

Existing image detectors ask whether an image was synthesized by an image generation model. Our rendered pages are all real compiler output, and the two sides differ only in who wrote the text, so these detectors should carry no signal on our task. Table 14 confirms this. We score every page of both batteries with five open-source detectors, averaging over the crops of each document. All of them sit at chance level.

Detector	Proof	Prose
CNNDetection (Wang et al., 2020)	0.55	0.54
UnivFD (Ojha et al., 2023)	0.49	0.54
NPR (Tan et al., 2024)	0.42	0.47
SAFE (Li et al., 2025)	0.53	0.48
Community Forensics (Park & Owens, 2025)	0.51	0.49
SnapJudge (ours)	0.99	0.99

Table 14: AUROC of AI-generated image detectors and of our detector on the rendered images.

H SnapJudge at Lower Resolution

To test how SnapJudge copes with low-quality screenshots, we render the frontier test passages of Table 2 at resolutions from 150 dpi, the resolution used in training, down to 20 dpi, and score them with the threshold of Table 2 unchanged. Figure 6 shows that detection on prose barely changes down to 40 dpi, while detection on proofs declines gradually, from 91.8% at 150 dpi to 81.8% at 50 dpi. Proofs are presumably more sensitive because subscripts and small symbols in the formulas become illegible first. Below 40 dpi, detection drops sharply for both registers, as the text itself becomes hard to read.

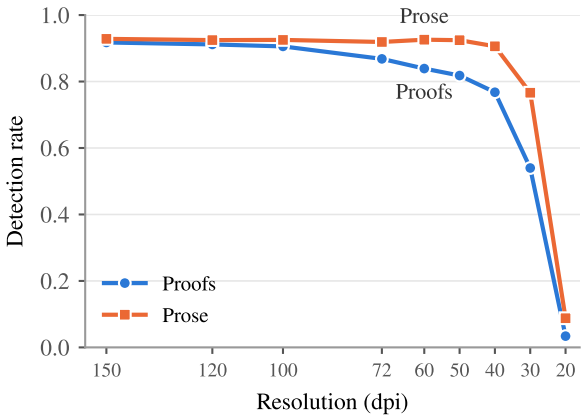


Figure 6: Detection rate of SNAPJUDGE on frontier-model test passages rendered at decreasing resolution.

I GENERATOR MODELS AND GENERATION PROMPTS

Generator models. The ten open-source generators are Qwen3-32B and Qwen3-235B (Yang et al., 2025), Qwen2.5-72B (Qwen et al., 2025), Llama-3.3-70B (Grattafiori et al., 2024), Llama-4-Scout (Meta, 2025), DeepSeek-R1-Distill-Qwen-32B and DeepSeek-R1-Distill-Llama-70B (Guo et al., 2025), GLM-4.5-Air (Zeng et al., 2025), Phi-4 (Abdin et al., 2024) and gpt-oss-120B (Agarwal et al., 2025), all run locally with vLLM (Kwon et al., 2023). The five frontier generators are GPT-4.1 (OpenAI, 2025a), GPT-5.1 (OpenAI, 2025b), o4-mini (OpenAI, 2025c), Claude Sonnet 5 (Anthropic, 2026) and Gemini 3.1 Pro (Google DeepMind, 2026), accessed through their APIs. Each target passage is generated by exactly one model.

Generation cost. Table 15 lists the number of passages from each frontier API that are used in the training, validation and test splits, and the corresponding cost. Every request carries the full LaTeX source of the paper as context, so the cost is dominated by input tokens and comes to about $0.03 per passage; the whole frontier half of the dataset cost about $390. The ten open-source generators ran on our own GPUs and incurred no API cost.

Judging cost. The LLM judge of Section 4.2 (GPT-5.6 Sol (OpenAI, 2026a), reasoning effort set to high) read 1,100 proof pairs and 1,000 prose pairs, about 3.3M input tokens and 0.7M output tokens including reasoning tokens, for about $40.

Generator	Proof passages	Prose passages	Cost (USD)
GPT-4.1	1,477	1,244	≈90
GPT-5.1	1,487	1,600	≈85
o4-mini	801	1,600	≈60
Claude Sonnet 5	1,053	1,147	≈70
Gemini 3.1 Pro	1,040	1,045	≈95
Total	5,858	6,636	≈390
GPT-5.6 Sol (judge, Sec. 4.2)	1,100 pairs	1,000 pairs	≈40

Table 15: API cost of the frontier-model passages used in the dataset, and of the LLM judge.

System prompt for proofs.

```
You are an expert mathematician completing the LaTeX source
of a research paper.  You will be given the paper's title,
abstract, macro definitions, and the full LaTeX source of
the paper with one proof removed and its position marked;
```

```
the theorem statement remains in place immediately before
the marker.  Your task is to write the missing proof.
Rules:
1.  Write a complete, rigorous proof of the stated theorem.
2.  Write the proof so that it reads as a seamless
continuation of the paper's own source.  Follow the
excerpt's writing style, notation, phrasing, and LaTeX
conventions as closely as you can:  use the same
inline/display math delimiters, the same environment names,
and the paper's macros where their meaning is clear from the
excerpt.
3.  Output ONLY the proof body in raw LaTeX. Do NOT include
\begin{proof} or \end{proof}, do not restate the theorem, and
do not add headings, greetings, or commentary of any kind.
4.  Aim for approximately N words.
```

System prompt for prose.

```
You are an expert academic writer completing the LaTeX
source of a research paper.  ONE paragraph of running text
has been removed from the body and replaced with a marker.
You are given the paper's title, abstract, macro
definitions, optionally its references, and the FULL LaTeX
source with that one paragraph missing.  Write the missing
paragraph.
Rules:
1.  Write a single paragraph that fits seamlessly at the
marker:  it must follow from the text before it and lead
into the text after it.
2.  Match the paper's own writing style, notation,
terminology, phrasing, and LaTeX conventions:  the same
inline math delimiters, the same custom macros, the same
citation commands (cite using keys that appear elsewhere in
the paper), and the same register.
3.  Output ONLY the missing paragraph as raw LaTeX running
text.  Do NOT add a section heading, list, theorem/proof
environment, figure, table, or display-math block; do not
restate the surrounding text; and do not add any greeting,
label, or commentary.
4.  Aim for approximately N words.
```

Prompts for the multi-level AI-editing test set.

Sentence-level rewrite.

```
Rewrite this sentence from an academic paper to improve
fluency and clarity.  Keep the meaning of the sentence
unchanged.  Output only the rewritten sentence in Latex,
nothing else.
Sentence:  {s}
```

Paragraph rewrite.

```
Rewrite the following passage from a mathematical paper to
improve its fluency and clarity.  Keep the meaning and all
mathematical content unchanged.  Output only the rewritten
passage in LaTeX, nothing else.
Passage:  {s}
```

Two-stage free generation, stage 1 (outline).

```
List every logical step and key point of the following
passage as a concise bullet outline, in your own words.  You
may use mathematical symbols.  Do not copy phrases from the
original text.  Output only the outline.
Passage:  {s}
```

Two-stage free generation, stage 2 (writing; proofs).

```
Write a complete, rigorous mathematical proof in standard
LaTeX that follows this outline.  Output only the proof
body, no preamble.
Outline:  {s}
```

Two-stage free generation, stage 2 (writing; prose).

```
Write a well-structured paragraph of an academic mathematics
paper in LaTeX that covers exactly the following points.
Output only the paragraph.
Points:  {s}
```

J PROMPTS OF THE EQUIVALENCE JUDGE

System prompt for proofs.

```
You are an expert research mathematician acting as a
referee.  You will be given a theorem statement taken from a
mathematics paper and two proofs of it, labelled Proof A and
Proof B. The material is an excerpt:  definitions, notation
and results cited from elsewhere in the paper are not shown,
and you should assume that such cited facts exist and are as
the authors say.
Your task has two parts.
1.  Validity.  For each proof, judge whether it is a correct
and complete proof of the statement as written.  Answer
"yes" if the argument is correct and no essential step is
missing, "partial" if the strategy is sound but a genuine
gap or error remains, and "no" if the argument does not
establish the statement.  Do not penalise brevity,
informality, omitted routine verifications or reliance on
cited results.
2.  Similarity of mathematical content.  Compare what the
two proofs actually do, not how they are written.  Ignore
wording, sentence order, verbosity, the names of variables
and the LaTeX formatting.  Focus on the overall strategy,
the key steps, the intermediate claims and the results
invoked.  Rate the similarity on a scale from 1 to 10.  Be
generous:  two proofs that reach the conclusion by the same
overall route deserve a high score even if they differ in
details, in the amount of detail shown, or in the order of
the steps.  Reserve low scores for proofs whose mathematical
strategies genuinely differ.  Use the following anchors:
10 = the same proof, the same argument with the same key
steps, differing only in presentation;
8-9 = the same argument with small differences of content,
such as an extra or omitted routine step, a different order
of the same steps, a different but equivalent computation,
or a different choice of which routine facts to spell out;
6-7 = the same overall strategy and mostly the same key
steps, but one key step is carried out differently or a
different intermediate result is invoked;
4-5 = the strategies overlap only in part, such as sharing
the main idea but reaching it through substantially
different intermediate steps;
2-3 = different strategies or different key steps, although
both arguments are plausible;
1 = fundamentally different arguments, or one of the proofs
is not a proof of the statement at all.
Do not try to guess who wrote either proof and do not let
style, polish or length influence the similarity rating.
Reply with one JSON object and nothing else, in a fenced
```json block, with exactly these fields:
```
```

```
{"A_valid": "yes|partial|no", "B_valid": "yes|partial|no",
"similarity": <integer 1-10>, "same_strategy":
<true|false>, "key_difference": "<one sentence naming the
most important difference in content, or 'none' if the
proofs are essentially the same>"}
```

User message for proofs.

```
THEOREM ({thm_kind}):
{statement}
PROOF A:
{proof_a}
PROOF B:
{proof_b}
```

System prompt for prose.

```
You are an expert academic reviewer. You will be given two
versions of one paragraph from a mathematics paper, labelled
Paragraph A and Paragraph B. Both are meant to occupy the
same position in the paper, and the material is an excerpt:
the rest of the paper, its definitions and its cited results
are not shown, and you should assume that such cited facts
exist and are as the authors say.
Your task is to judge how similar the two paragraphs are in
content. Compare what the paragraphs say, not how they say
it. Ignore wording, sentence order, verbosity, the choice
of words, citation commands and LaTeX formatting. Focus on
the claims made, the facts and results stated, the objects
and quantities discussed, and the arguments or motivations
given. Rate the similarity on a scale from 1 to 10. Be
generous: two paragraphs that make the same points deserve
a high score even if they differ in detail, in emphasis, in
the amount of detail given, or in the order in which the
points are made. Reserve low scores for paragraphs whose
content genuinely differs. Use the following anchors:
10 = the same content, the same points made in the same way,
differing only in presentation;
8-9 = the same points with small differences of content,
such as an extra or omitted minor detail, a different order
of the same points, or a different but equivalent way of
stating a fact;
6-7 = mostly the same points, but one substantive point is
added, dropped or stated differently;
4-5 = the paragraphs overlap only in part, such as sharing
the main point but otherwise discussing different things;
2-3 = largely different content, although both paragraphs
are plausible at this position in the paper;
1 = essentially unrelated content, or one of the paragraphs
does not fit the position at all.
Do not try to guess who wrote either paragraph and do not
let style, polish or length influence the rating. Reply
with one JSON object and nothing else, in a fenced ```json
block, with exactly these fields:
{"similarity": <integer 1-10>, "key_difference": "<one
sentence naming the most important difference in content, or
'none' if the paragraphs are essentially the same>"}
```

User message for prose.

```
PARAGRAPH A:
{para_a}
PARAGRAPH B:
{para_b}
```

K VALIDATION OF THE PROOF JUDGE

	Flawed proofs (17)	Control proofs (13)
Mean severity	7.08	4.23
Standard deviation	2.46	2.20
Median	8.0	4.0
Severity ≥ 5	12/17 (71%)	2/13 (15%)

Table 16: Severity scores from 0 to 10 that GPT-5.6 Sol assigns to proofs from withdrawn ICLR 2026 submissions whose reviewers pointed out a flaw, and to control proofs that reviewers described as correct, averaged over the judge's samples for each proof. For this check the judge uses the scoring prompt given below. A flawed proof receives a higher score than a control proof with probability 0.81 (one-sided Mann-Whitney test, $p = 0.0016$).

System prompt.

```
You are verifying a single mathematical result taken from a
machine learning paper.  You will be shown the problem
setting, the statements of results the proof relies on, and
one statement together with its proof.
Your task is NOT to decide whether the proof "has a
problem." It does.  They all do.  Every published proof
contains steps that are compressed, hypotheses left
implicit, and constants left unnamed.  Asking whether a
defect exists is a question whose answer is always yes, and
it carries no information.  What we need from you is a
measurement of CONSEQUENCE: how much work would it take to
make this proof correct?
WHAT YOU MUST DO, IN THIS ORDER
Step 1 -- Re-derive.  Work through the proof line by line.
Check each inference rather than reading for plausibility.
Where the algebra is checkable numerically, check it
numerically.  For every inequality, limit, expectation, or
asymptotic claim, ask what happens at the boundary of the
stated parameter range.
Step 2 -- Enumerate, including trivia.  List every
imperfection you can find, from notational slips upward.
Finding things is free here and proves nothing; we want the
list so that we can see you looked.  Three findings is a
reasonable number.  Do not stop at the first one and do not
pad the list with restatements of the same issue.
Step 3 -- Rank, then score ONLY the worst.  Severity does
not accumulate.  Three cosmetic slips are still cosmetic.
The count of findings is never the score.  If your top three
findings all look serious, you have almost certainly split
one issue into three; merge them.
Step 4 -- Attempt the repair.  This is the step that
determines your score, and it is the reason this task is not
a matter of opinion.
THE PRIMARY SCORE: repair
  none_needed  -- You searched and found only cosmetic or
notational issues.  A referee would not raise them.
  one_line  -- There is a real gap, and you can close it.
You MUST actually write the missing step or the missing
hypothesis in the repair_attempt field.  If you cannot write
it out, this is not the right level.
  nontrivial  -- You tried the one-line repair and it does
not work.  You MUST state in repair_attempt what new
argument is required AND why the short route fails.  "It
needs more detail" is not an answer.
```

```
  unfixable  -- The statement itself is false, or the
proof's central construction cannot exist.  You MUST exhibit
or sketch a counterexample to the STATEMENT in
repair_attempt -- concrete values, or a family of instances,
and what goes wrong.  A counterexample to one STEP is not
enough; a step can be replaced.  If you cannot produce a
counterexample to the statement, you are at nontrivial, not
here.
Each level costs you something we can check.  Do not claim a
level whose price you have not paid.
THE SECONDARY SCORE: severity, 0 to 10
  0  no defect found after genuine effort
  1-2  cosmetic, notational, or pedantic; a referee would
not mention it
  3-4  a real gap that a careful reader repairs in one line
while reading
  5-6  a genuine hole needing a nontrivial new argument; the
claim is still likely true
  7-8  the proof as written does not establish the claim,
and the repair route is unclear
  9-10  the claim as stated is false, and you can exhibit or
sketch a counterexample
Scoring 1 to 4 is the normal, expected, respectable outcome
for a competently written proof.  Do not inflate.  But do
not reflexively assign a low score either -- a low score
must be earned by a genuine search that turned up only minor
things, and your enumeration in step 2 must show that
search.
WHAT YOU ARE NOT ASKED FOR
Do not report how confident you feel.  We have measured
self-reported confidence on this task and it carries no
information; it will be discarded.  Ground your answer in
the mathematics and in the repair you did or could not do.
ON MATERIAL YOU WERE NOT GIVEN
You are shown an excerpt, not the whole paper.  If a
quantity looks undefined or a hypothesis looks missing,
consider first that it is probably established elsewhere in
the paper.  Treat that as the default.  Only count it as a
defect when the proof's own logic requires something that
could not plausibly be supplied elsewhere -- for example
when the step is false under any reading, or when the proof
itself asserts the missing fact rather than citing it.  Do
not build an objection on a hostile choice of parameters
that the paper's setting would rule out.
EVIDENCE RULES
Every claimed defect must be anchored in text you COPY
VERBATIM from the proof you were shown.  Copy the substring
exactly, including LaTeX. Do not paraphrase, do not
reconstruct, do not normalise the notation.  If you cannot
copy a real substring, you do not have a defect to report.
OUTPUT
Return one JSON object and nothing else, inside a fenced
code block marked json.  Escape backslashes correctly so the
object parses; LaTeX in your quotes must survive.
{
  "findings":  [
    {"quote":  "<verbatim substring from the proof>",
"what_is_wrong":  "<one sentence>"}
  ],
  "worst_finding_index":  <integer index into findings, or -1
if none>,
  "repair":  "none_needed" | "one_line" | "nontrivial" |
"unfixable",
```

```
  "repair_attempt":  "<the missing step written out; or why
the short route fails and what is needed; or the
counterexample.  Required unless repair is none_needed.>",
  "severity":  <integer 0-10>,
  "conclusion_status":  "true_as_stated" |
"true_after_restatement" | "probably_true_unproven" | "false",
  "reasoning":  "<why the worst finding sits at that repair
level, in a few sentences>"
}
```

User message.

```
Problem setting.
{setup}
Statements this proof relies on.
- {dependency 1}
- {dependency 2}
...
Statement and proof to evaluate.
{statement and proof}
```

L AI-FRACTION ESTIMATOR

To make SNAPJUDGE more practical, we separately train a Qwen3.5-27B estimator to predict the fraction of the text in an image, measured by height, that is AI-generated, using the same fine-tuning procedure as for the detectors and a cross-entropy loss over five buckets. Its bucket accuracy on the test set is 0.885. The fraction estimator is trained on the real-page windows of Section 3.1, grouped by their AI fraction p into the five buckets $(0, 0.25)$, $[0.25, 0.5)$, $[0.5, 0.75)$, $[0.75, 1)$ and $\{1\}$. It outputs bucket probabilities $p_1, \ldots, p_5$, the reported bucket is the most probable one, and the fraction estimate is $\hat{\rho} = \sum_j p_j m_j$ with $m = (0.125, 0.375, 0.625, 0.875, 1)$. It is only ever applied to passages that SnapJudge has already judged AI-generated.

Table 17 evaluates it on 5,000 framed windows from the 2021 test papers, 1,000 per true bucket. Part (a) shows the estimator alone. Part (b) shows the deployed cascade, in which SnapJudge first decides at its 0.1% false-positive threshold whether a window contains AI text, and only the windows it passes on are assigned a bucket. Besides the bucket decision, we measure the error of the fraction estimate $\hat{\rho}$ as the mean absolute error $|\hat{\rho} - \rho|$ against the AI fraction ρ of the window, reported per true bucket in the last column of each part. The estimator alone is off by 0.055 on average and the cascade by 0.051 on the windows it passes.

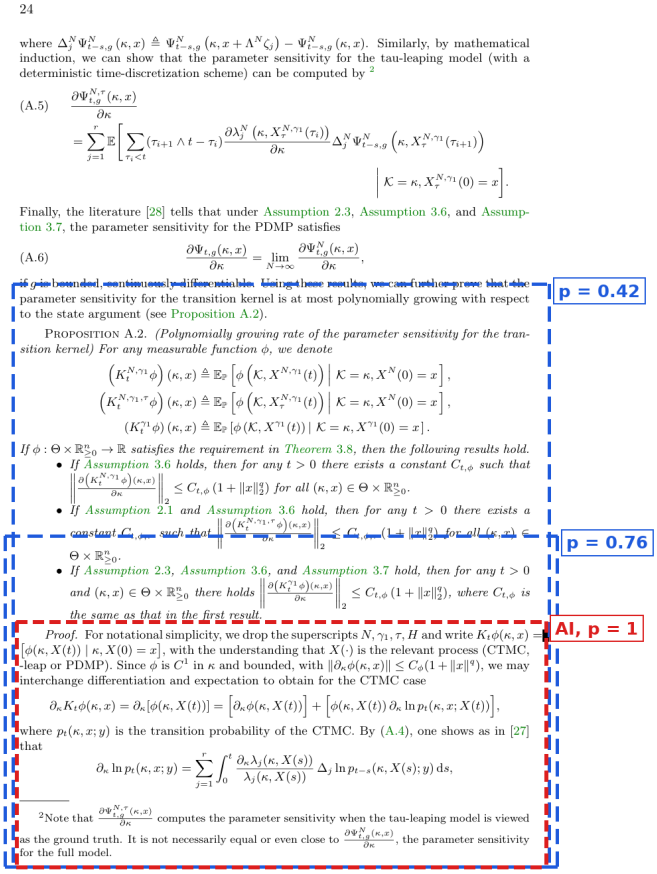
(a) estimator alone, all AI-containing windows

true \ pred.	1	2	3	4	5	MAE
1 $(0, .25)$	**91.4**	6.5	1.1	0.2	0.8	0.077
2 $[.25, .5)$	2.8	**91.6**	4.8	0.2	0.6	0.058
3 $[.5, .75)$	0.5	5.1	**89.0**	3.8	1.6	0.062
4 $[.75, 1)$	0.5	0.5	4.8	**72.4**	21.8	0.055
5 (AI)	0.1	0.2	1.0	0.6	**98.1**	0.023
all						0.055

(b) cascade: image-detector gate at 0.1% FPR, then the estimator

true \ pred.	passed	1	2	3	4	5	MAE
1 $(0, .25)$	66.1	**93.4**	5.3	0.9	0.0	0.4	0.072
2 $[.25, .5)$	81.2	2.2	**93.1**	4.2	0.1	0.4	0.057
3 $[.5, .75)$	86.9	0.1	5.2	**90.3**	3.6	0.7	0.059
4 $[.75, 1)$	88.5	0.2	0.5	4.8	**73.8**	20.7	0.053
5 (AI)	88.4	0.1	0.2	0.9	0.6	**98.2**	0.021
all	82.2						0.051

Table 17: The AI-fraction estimator on 5,000 framed test windows.



24

where $\Delta_j^N \Psi_{t-s,g}^N(\kappa,x) \triangleq \Psi_{t-s,g}^N(\kappa, x+\Lambda^N\zeta_j) - \Psi_{t-s,g}^N(\kappa,x)$. Similarly, by mathematical induction, we can show that the parameter sensitivity for the tau-leaping model (with a deterministic time-discretization scheme) can be computed by [2]

$$\frac{\partial \Psi_{t,g}^{N,\tau}(\kappa,x)}{\partial \kappa} = \sum_{j=1}^{r} \mathbb{E}\left[\sum_{\tau_i<t} (\tau_{i+1}\wedge t - \tau_i) \frac{\partial \lambda_j^N\left(\kappa, X_\tau^{N,\gamma_1}(\tau_i)\right)}{\partial \kappa} \Delta_j^N \Psi_{t-s,g}^N\left(\kappa, X_\tau^{N,\gamma_1}(\tau_{i+1})\right) \,\middle|\, \mathcal{K}=\kappa, X_\tau^{N,\gamma_1}(0)=x\right]. \tag{A.5}$$

Finally, the literature [28] tells that under Assumption 2.3, Assumption 3.6, and Assumption 3.7, the parameter sensitivity for the PDMP satisfies

$$\frac{\partial \Psi_{t,g}(\kappa,x)}{\partial \kappa} = \lim_{N\to\infty} \frac{\partial \Psi_{t,g}^N(\kappa,x)}{\partial \kappa}, \tag{A.6}$$

If ρ is bounded, continuously differentiable, and g and ν are bounded, we can further prove that the parameter sensitivity for the transition kernel is at most polynomially growing with respect to the state argument (see Proposition A.2).

PROPOSITION A.2. *(Polynomially growing rate of the parameter sensitivity for the transition kernel) For any measurable function ϕ, we denote*

$$\left(K_t^{N,\gamma_1}\phi\right)(\kappa,x) \triangleq \mathbb{E}_{\mathbb{P}}\left[\phi\left(\mathcal{K}, X^{N,\gamma_1}(t)\right) \,\middle|\, \mathcal{K}=\kappa, X^N(0)=x\right],$$
$$\left(K_t^{N,\gamma_1,\tau}\phi\right)(\kappa,x) \triangleq \mathbb{E}_{\mathbb{P}}\left[\phi\left(\mathcal{K}, X_\tau^{N,\gamma_1}(t)\right) \,\middle|\, \mathcal{K}=\kappa, X^N(0)=x\right],$$
$$\left(K_t^{\gamma_1}\phi\right)(\kappa,x) \triangleq \mathbb{E}_{\mathbb{P}}\left[\phi\left(\mathcal{K}, X^{\gamma_1}(t)\right) \mid \mathcal{K}=\kappa, X^{\gamma_1}(0)=x\right].$$

If $\phi: \Theta\times\mathbb{R}_{\geq 0}^n \to \mathbb{R}$ satisfies the requirement in Theorem 3.8, then the following results hold.

- *If Assumption 3.6 holds, then for any $t>0$ there exists a constant $C_{t,\phi}$ such that $\left\|\frac{\partial\left(K_t^{N,\gamma_1}\phi\right)(\kappa,x)}{\partial\kappa}\right\|_2 \leq C_{t,\phi}\left(1+\|x\|_2^q\right)$ for all $(\kappa,x)\in\Theta\times\mathbb{R}_{\geq 0}^n$.*
- *If Assumption 2.1 and Assumption 3.6 hold, then for any $t>0$ there exists a constant $C_{t,\phi}$ such that $\left\|\frac{\partial\left(K_t^{N,\gamma_1,\tau}\phi\right)(\kappa,x)}{\partial\kappa}\right\|_2 \leq C_{t,\phi}\left(1+\|x\|_2^q\right)$ for all $(\kappa,x)\in\Theta\times\mathbb{R}_{\geq 0}^n$.*
- *If Assumption 2.3, Assumption 3.6, and Assumption 3.7 hold, then for any $t>0$ and $(\kappa,x)\in\Theta\times\mathbb{R}_{\geq 0}^n$ there holds $\left\|\frac{\partial\left(K_t^{\gamma_1}\phi\right)(\kappa,x)}{\partial\kappa}\right\|_2 \leq C_{t,\phi}\left(1+\|x\|_2^q\right)$, where $C_{t,\phi}$ is the same as that in the first result.*

Proof. For notational simplicity, we drop the superscripts N,γ_1,τ,H and write $K_t\phi(\kappa,x) = \mathbb{E}[\phi(\kappa,X(t)) \mid \kappa, X(0)=x]$, with the understanding that $X(\cdot)$ is the relevant process (CTMC, tau-leap or PDMP). Since ϕ is C^1 in κ and bounded, with $\|\partial_\kappa\phi(\kappa,x)\| \leq C_\phi(1+\|x\|^q)$, we may interchange differentiation and expectation to obtain for the CTMC case

$$\partial_\kappa K_t\phi(\kappa,x) = \partial_\kappa[\phi(\kappa,X(t))] = \left[\partial_\kappa\phi(\kappa,X(t))\right] + \left[\phi(\kappa,X(t))\,\partial_\kappa \ln p_t(\kappa,x;X(t))\right],$$

where $p_t(\kappa,x;y)$ is the transition probability of the CTMC. By (A.4), one shows as in [27] that

$$\partial_\kappa \ln p_t(\kappa,x;y) = \sum_{j=1}^{r}\int_0^t \frac{\partial_\kappa\lambda_j(\kappa,X(s))}{\lambda_j(\kappa,X(s))}\,\Delta_j \ln p_{t-s}(\kappa,X(s);y)\,\mathrm{d}s,$$

[2] Note that $\frac{\partial\Psi_{t,g}^{N,\tau}(\kappa,x)}{\partial\kappa}$ computes the parameter sensitivity when the tau-leaping model is viewed as the ground truth. It is not necessarily equal or even close to $\frac{\partial\Psi_{t,g}^{N}(\kappa,x)}{\partial\kappa}$, the parameter sensitivity for the full model.

Figure 7: A recompiled page of a 2021 test paper in which an AI-generated proof (dashed red box, $p = 1$) is surrounded by the human-written text of the paper. The dashed blue boxes are two windows opened by the sampler on this page, each fully covering the AI passage together with part of the human-written context; their AI fraction p is the height of the AI text divided by the height of all text inside the window.

M CALIBRATION OF THE DETECTOR SCORES

Figure 8 shows the reliability diagrams of the three 27B detectors. We divide the scores reported by a detector into eight ranges (the detailed values are given in Table 18). The horizontal coordinate of each point is the mean score of the texts in that range and the vertical coordinate is the share of AI samples among them. Within every dataset the AI samples are first weighted to count as much as the human-written samples, so as to avoid the effect of unequal sample sizes. The figure shows that SnapJudge is less miscalibrated than the text-based detector on both the in-domain and the external test sets, and that it alleviates the problem of the model being over-confident on unfamiliar human-written text.

We use the expected calibration error (ECE) (Guo et al., 2017) to express the degree of miscalibration of a detector. It is computed as

$$\mathrm{ECE} = \sum_{b=1}^{8} \frac{n_b}{N} \, |y_b - x_b|, \tag{1}$$

where x_b and y_b are the horizontal and vertical coordinates of range b, n_b is the weighted number of texts in it and N is the total number of texts.

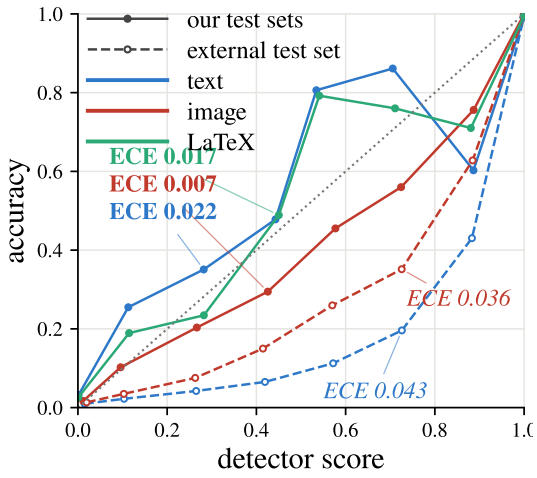


Figure 8: Reliability of the 27B detectors.

	Text		SnapJudge		LaTeX	
Score range	score	AI share	score	AI share	score	AI share
Our test sets						
0–0.05	0.001	0.032	0.012	0.017	0.001	0.026
0.05–0.2	0.113	0.255	0.096	0.102	0.115	0.189
0.2–0.35	0.282	0.351	0.267	0.203	0.282	0.234
0.35–0.5	0.443	0.478	0.426	0.295	0.450	0.489
0.5–0.65	0.534	0.806	0.577	0.455	0.541	0.792
0.65–0.8	0.706	0.861	0.724	0.560	0.711	0.760
0.8–0.95	0.887	0.603	0.887	0.756	0.881	0.710
0.95–1	1.000	0.992	0.999	0.998	1.000	0.995
ECE	0.022		0.007		0.017	
External test set						
0–0.05	0.015	0.010	0.019	0.013	–	–
0.05–0.2	0.103	0.022	0.103	0.035	–	–
0.2–0.35	0.265	0.042	0.263	0.075	–	–
0.35–0.5	0.419	0.065	0.415	0.150	–	–
0.5–0.65	0.572	0.113	0.570	0.260	–	–
0.65–0.8	0.726	0.196	0.726	0.351	–	–
0.8–0.95	0.883	0.430	0.885	0.628	–	–
0.95–1	0.999	0.991	0.999	0.995	–	–
ECE	0.043		0.036		–	

Table 18: Values behind Figure 8.