TEXFORM: AUDITING AND IMPROVING LATEX FORMULA NORMALIZATION IN DATA AND EVALUATION PIPELINES

Anonymous authors
Paper under double-blind review

ABSTRACT

LaTeX normalization maps the many equivalent forms of a formula to one canonical string. Formula-recognition data pipelines and the formula tracks that rank today's vision-language models all silently rely on normalizers, yet not one of these normalizers states what it preserves. Our experiments reveal failures accumulated invisibly for years: normalization in dataset construction changes what its inputs render to, with case studies exposing symbol loss and structural truncation; inside three widely used benchmarks (UniMER-Test, OCRBench v2, and OmniDocBench), the official scoring pipelines falsely reject 19–26% of render-equivalent prediction–label pairs, with subset rates spanning 6–55%, while accepting some pairs that render differently as false-accept candidates. Beyond traditional text metrics, CDM, the recently proposed render-aware metric, also embeds a normalizer that alters what 14% of inputs render to. To address this systematically, we introduce LaTeX normalization as a formal task with four measurable properties (safety, canonicity, idempotence, and coverage), and characterize when normalize-then-match is a valid equivalence test. Guided by this formalization, we build **TeXForm**, a LaTeX normalizer driven by an explicit knowledge base of command argument structure and rewrite rules. Among all normalizers we evaluate, TeXForm is the only one to combine substantial canonicity (37–47%) with render-safety above 97%, the property that normalized training labels require. On UniMER-Test and OmniDocBench, replacing built-in normalization with TeXForm recovers 38.05% and 89.83% of previously rejected render-equivalent pairs, respectively. The audited normalization configurations change model rankings, including a leader reversal on UniMER-Test. We release TeXForm, including its knowledge base, as open source.

1 INTRODUCTION

As large language models and vision-language models are increasingly deployed on scientific documents (Zhang et al., 2024; Blecher et al., 2024), mathematical formulas have become central to both training and evaluation. Formula recognizers produce LaTeX strings from images of expressions (Deng et al., 2017; Gu et al., 2026; Mahdavi et al., 2019; Xie et al., 2023; Gervais et al., 2025; Schmitt-Koopmann et al., 2024), and the formula tracks of OCR and document-parsing benchmarks now rank general-purpose VLMs by comparing such strings (Fu et al., 2025; Ouyang et al., 2025). Yet LaTeX is highly flexible: the same formula admits many valid representations, so both ends of the pipeline must decide when two of them denote the same formula. The data side makes this decision while producing training corpora and benchmark labels from raw LaTeX, and the evaluation side while comparing each prediction against a reference. Both make that decision with the same step, which the literature calls *LaTeX normalization*: folding the equivalent representations of one formula into a single canonical string before the string is stored or compared.

Prior work has reported that normalization shifts scores (Li et al., 2025a), yet we find no audit of whether the normalizers themselves preserve rendered content and fold equivalent spellings. We audit existing normalizers and find failures in two directions. **(1) They fold too little.** These normalizers fold only some render-equivalent LaTeX spellings into one canonical string. A prediction that writes `\rightarrow` where the reference writes `\to` therefore reaches the metric as a dif-

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.

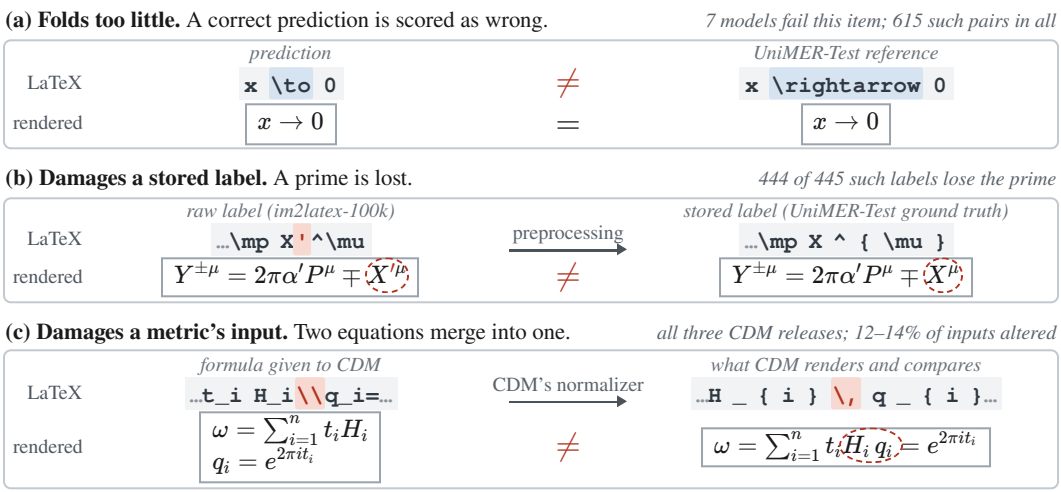

Figure 1: LaTeX normalization fails silently in two directions. Some render-equivalent spellings stay distinct, so a correct prediction is scored as wrong **(a)**; rewrites change what a formula renders to, corrupting a benchmark label **(b)** and the input of a render-aware metric **(c)**. Each panel sets two versions of a formula side by side, their LaTeX above and their renderings below, with their relation or the rewriting tool between them. Blue shading marks a harmless spelling difference, red shading a damaging token, and a dashed ring the site where the rendering changed; the note beside each title gives how often the pattern occurs.

ferent string, and the benchmark scores a correct answer as wrong (Figure 1a). **(2) They damage what they rewrite.** These normalizers rewrite formulas into strings that render differently from their inputs, and storing such an output as a label corrupts the reference when rendering preservation is required (Figure 1b). The same rewrites merge formulas that render differently, so evaluation also accepts pairs it should reject. Even CDM (Wang et al., 2025), the render-aware metric built to escape string comparison, embeds such a normalizer and alters what its inputs render to before the formulas are rendered (Figure 1c).

Failures in both directions have persisted undetected for years, and for a common reason: *LaTeX normalization has not been defined as a task with stated requirements.* The term is widely used: it appears as "enhanced LaTeX normalization" (Schmitt-Koopmann et al., 2024), "LaTeX Source Normalization" (Wang et al., 2025), and simply "normalized" labels (Gervais et al., 2025), but none of these works defines what the operation must preserve or when it can fail. Without such a declaration, there is nothing against which a normalizer can be checked, and its failures produce no signal. Damage at the data end is then hidden a second time at the evaluation end: a corrupted label becomes the reference, so a prediction that reads the image correctly is scored as wrong, and one that repeats the corruption is scored as correct.

To address this, we formalize LaTeX normalization as a task with stated requirements, use those requirements to audit the ecosystem, and then present TeXForm, a normalizer built to close the gap. Our primary contributions are:

- **Formalizing LaTeX normalization and defining four measurable properties.** We pair a normalizer with a declared equivalence and require four properties (safety, canonicity, idempotence, and coverage), each grounded in a class of observed failure. We then state when comparing normalized strings is a valid equivalence test, which makes both failure directions measurable for any normalizer and separates what the data end and the evaluation end each require.
- **A systematic audit of normalization as practiced.** We measure the safety and canonicity of widely adopted normalizer implementations under a unified rendering-equivalence criterion, audit the data end (for render-faithfulness loss) and the evaluation end (for false rejects and false-accept candidates), and trace changed labels into downstream corpora and benchmark references.
- **TeXForm, a knowledge-driven normalizer for LaTeX formulas.** We build TeXForm to provide the operation end to end, from parsing through transformation to serialization, driven by a knowl-

edge base of 1,213 command and character records and by 151 syntax-tree rewrite rules. Under the same audit protocol, it is the only normalizer we examine that combines substantial canonicity with render-safety above 97%. Replacing the built-in normalization of UniMER-Test and OmniDocBench with TeXForm recovers a substantial share of previously rejected render-equivalent pairs, and on UniMER-Test it reverses the leading pair of models. We open-source TeXForm together with its knowledge base and its Python and JavaScript bindings.

2 Related Work

Formula recognition. Formula recognition takes an image of a mathematical expression and produces a LaTeX string that renders to it. Printed-input datasets pair rendered crops with the sources they were rendered from, from im2latex-100k (Deng et al., 2017) through the re-rendered collection released with pix2tex (Blecher, 2022) to the million-sample corpus behind UniMERNet (Gu et al., 2026). Recognizers trained on them share a similar end-to-end image-to-markup formulation (Schmitt-Koopmann et al., 2024; Liu et al., 2025), document parsing absorbs formula recognition as a subtask (Blecher et al., 2024; Cui et al., 2025; Niu et al., 2026), and multimodal large models perform it out of their general visual understanding (Bai et al., 2025a; Zhu et al., 2025).

Evaluating formula recognition. Early formula recognition work adopted text metrics to score a predicted formula against a reference. Dedicated benchmarks and the handwriting competitions report exact match alongside BLEU and edit distance (Gu et al., 2026; Mahdavi et al., 2019; Xie et al., 2023), borrowed from machine translation and text recognition where surface similarity moves with correctness; TeXBLEU (Jung et al., 2025) adapts n-gram scoring to LaTeX tokens but remains at the string level. CDM (Wang et al., 2025) leaves string comparison behind, rendering prediction and reference into individually colorable tokens and computing an F1 score by matching characters at bounding-box level. It attributes the bias of text metrics to training-label conventions: a recognizer reproduces the spelling habits of its own labels, so the score measures how closely those habits match the reference's, which CDM argues makes text metrics neither reliable nor fair here. Document-parsing and OCR benchmarks absorbed formula recognition as a subtrack and draw on both families: OmniDocBench scores display formulas with CDM while the formula task of OCRBench v2 stays with string comparison (Ouyang et al., 2025; Fu et al., 2025). PiE-MER (Wieckowiak et al., 2026) converts predicted LaTeX into MathML, label graphs, and rendered images, and finds that BLEU and Levenshtein distance alone are insufficient for evaluation. Uni-MuMER (Li et al., 2025a) reports that normalizing the Im2LaTeXv2 test labels raises ExpRate by 8–12 points while ExpRate@CDM stays unchanged, and shows a subscript error that CDM misses. Beyond formula recognition, label errors in test sets reorder model rankings (Northcutt et al., 2021), and undisclosed preprocessing inside BLEU shifts reported scores enough to call for a standardized, versioned implementation (Post, 2018); neither line examines how structured markup is normalized. Where these studies treat normalization, they measure its effect on scores; none audits the normalizer itself.

Parsers, canonical forms, and equivalence checkers. Tools from adjacent problems supply parts of what is missing, under different assumptions. LaTeX parsers and converters (Miller, 2024; Faist, 2023; Khan Academy, 2024; The MathJax Consortium, 2025; Siefken, 2024; MacFarlane, 2025) provide syntax-aware representations; unified-latex also records argument signatures for selected packages and supports AST manipulation and serialization back to LaTeX. TeXForm combines these capabilities with an explicit knowledge base and rewrite policies for formula normalization, evaluated for render-safety, canonicity, idempotence, and coverage. Equivalence checkers written to grade mathematical answers, Math-Verify (Kydlíček, 2025) and the computer-algebra backends it wraps, return a verdict rather than a representative string, and they decide a coarser relation: they discard the presentation that a recognition label exists to record. Mathematical information retrieval canonicalizes expressions for indexing (Altamimi & Youssef, 2007; Zanibbi et al., 2016; Davila & Zanibbi, 2017), targeting retrieval matches rather than fidelity to the rendered formula.

3 CURRENT NORMALIZERS AND THEIR FAILURE MODES

We examine the implementations used in dataset construction, recognizer post-processing and evaluation scoring (Deng et al., 2017; Gu et al., 2026; Schmitt-Koopmann et al., 2024; Fu et al., 2025; Wang et al., 2025), and show why their failures are structural rather than incidental.

The most consequential problem is that one flawed script has been widely reused: most of the normalization implementations we audit descend from the preprocessing script released with im2latex-100k (Deng et al., 2017). We summarize its defects in four layers; further defects we identify are in Appendices C.3 and C.5. **(1) Parsing layer.** It modifies the internal parsing logic of the bundled KaTeX 0.6.0 in multiple places, commenting out checks and making destructive changes: an apostrophe is silently dropped under certain conditions, so a derivative loses its mark (Figure 1b). **(2) String substitution layer.** Its substitutions around the parse-and-serialize round trip are blind to structure: a line break (\\) is replaced by a thin space outside a few environments, collapsing a two-line formula to one line (Figure 1c). **(3) Serialization layer.** Serialization loses much of the original syntax structure: it turns text-mode content into math mode and emits multi-letter operator names character by character. **(4) Exception-handling layer.** Any exception during parsing or serialization falls into a single catch that returns the empty string, so an out-of-domain input is not rejected but returned as a syntactically valid empty formula.

Later work carried these defects forward to today: MathNet (Schmitt-Koopmann et al., 2024), the CDM released with UniMERNet (Wang et al., 2025), and the formula CDM in OmniDocBench v1.5 (Ouyang et al., 2025) each vendor a copy of its modified KaTeX parser (lineage analysis in Appendices C.3 and C.4). One exception falls outside the lineage: the post-processing on the recognizer's inference side, which only cleans whitespace and repairs Unicode (Gu et al., 2026).

CDM (Wang et al., 2025) replaces text metrics with a render-aware metric but does not escape the problem: the tokenization it decomposes each formula with descends from the same script and carries the same defects. Its later update in OmniDocBench v1.6 stops reusing the script, yet rebuilds the original serialization behavior and inherits the defects with it. Components written a decade later reproduce the same gap, so the problem lies not in any single inherited implementation but in an operation whose requirements and guarantees have not been specified.

Repairing these defects would reduce damage, yet a normalizer that damages nothing can still fold too little. Existing implementations fold only the variants their authors enumerated, and folding more requires per-command knowledge that is hard to obtain. A parser that does not know how many arguments a command takes builds the wrong tree, yet these argument definitions exist only in package documentation written in prose and in the code of renderers and converters, each in its own format. None of the implementations we survey systematically folds the interchangeable spellings LaTeX provides for one mathematical symbol. MathNet, which comes closest, maps only a hand-picked list of a few dozen tokens, such as `\leq` to `\le`, so `\to` and `\rightarrow` remain distinct strings after normalization (Figure 1a).

Recognizing these problems requires defining the LaTeX normalization task (Section 4), and resolving them requires new infrastructure (Section 5).

4 LATEX NORMALIZATION AS A FORMAL TASK

To make normalizers auditable, we define LaTeX normalization as a formal task and state the properties a normalizer should have.

Definition 1 (Normalizer)**.** A normalizer is a pair $(N, \equiv_N)$: $N : \Sigma^* \rightharpoonup \Sigma^*$ is a partial function on the set Σ^* of strings, and $\equiv_N$ is its *declared equivalence*.

Here N is partial because not every string is one a normalizer can read; $\equiv_N$ is the relation it claims to preserve. Testing such a claim requires a relation given independently of it. We therefore fix $\equiv$ as rendering equivalence: let R be the function computed by a designated renderer from strings to images, and let $x \equiv y$, read "x and y are render-equivalent", hold if and only if $R(x)$ and $R(y)$ are both defined and equal. A string outside $\mathrm{dom}(R)$ is therefore render-equivalent to no string. Below, $=$ denotes string equality, and free variables in the properties range over the domain of N, $\mathrm{dom}(N)$.

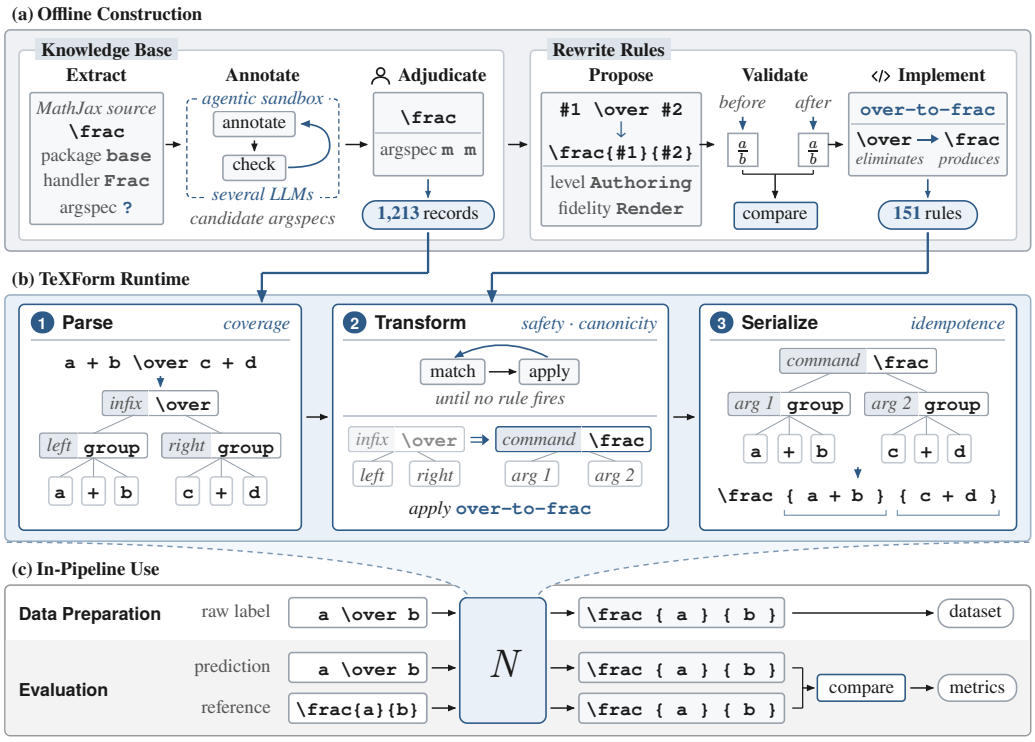

Figure 2: Construction, runtime, and in-pipeline use of TeXForm. **(a)** The knowledge base and the rewrite rules are built offline, with machines proposing or checking and humans deciding. **(b)** Three contract-carrying stages take a formula from string to tree, tree to tree, and tree to string. **(c)** The same N serves both ends of the pipeline. Data preparation keeps the normalized label, while evaluation keeps only the match decision.

Definition 2 (Properties of a normalizer)**.**

- **Safety.** $N(x) \equiv x$. The normalized output is render-equivalent to the input.
- **Canonicity.** $x \equiv y \Rightarrow N(x) = N(y)$. Render-equivalent representations converge to one string.
- **Idempotence.** $N(N(x)) = N(x)$. The output is unchanged by another normalization pass.
- **Coverage.** $\mathrm{dom}(N)$ is declared and can be checked, and N fails explicitly outside its domain.

Each of the four is violated by implementations in use: the truncation of Section 3 breaks safety, the empty string returned on a parse failure breaks both safety and coverage, the unfolded command spellings break canonicity, and the metric-internal processor of CDM adds a brace layer on every pass, so that `\overline p` becomes `\overline{p}` and then `\overline{{p}}`, breaking idempotence. Practical implementations have only partial knowledge of command semantics and renderer behavior, so merging more equivalent representations risks altering the rendered output. We therefore treat these properties as goals and measure how far each normalizer attains them.

Evaluation in this field compares exactly these normalized strings, and the following proposition states when that comparison is a valid test of $\equiv$.

Proposition 1. If N satisfies both safety and canonicity, then for all $x, y \in \mathrm{dom}(N)$, $N(x) = N(y)$ if and only if $x \equiv y$.

The direction from $x \equiv y$ to $N(x) = N(y)$ is canonicity itself; the converse is given by safety, since $N(x) = N(y)$ implies $x \equiv N(x) = N(y) \equiv y$. Neither direction is dispensable: if canonicity fails, the comparison rejects an equivalent pair solely because its representations differ, a false reject; if N merges a pair that renders differently, the comparison accepts that pair, a false accept.

The two ends of the pipeline need different parts of this guarantee: data preparation keeps $N(x)$ as a training label and therefore needs safety at every point; evaluation keeps only the decision, and merging no pair that renders differently already rules out false accepts. The latter condition is strictly weaker than safety: even if a normalizer never merges a pair that renders differently, it can still ruin every string it emits.

5 THE TEXFORM NORMALIZER

We introduce TeXForm as an infrastructure tool for parsing, editing, and normalizing LaTeX formulas. TeXForm normalizes a formula in three stages: a parser turns it into a typed syntax tree using a knowledge base of command and environment specifications, a transform engine rewrites that tree under rules that each declare the equivalence they preserve, and a serializer writes the tree back to a canonical string. Its rewrite rules state what they preserve, and it refuses an input it cannot read instead of altering it. Figure 2 illustrates our approach.

Knowledge base and rewrite rules. For every command, the parser needs its argument shape, its mode (math or text), and its syntactic role (prefix, infix, or declarative). No standalone reference provides these for LaTeX commands (Section 3), so we take MathJax (The MathJax Consortium, 2025) as the authority on what each command accepts and derive the knowledge base from its source. After a static analyzer catalogs that source, several LLM agents independently propose an annotation for each entry, and we adjudicate every entry by hand against the MathJax source, accepting, combining, or rewriting the proposals (the final specification departs from every proposal for 17% of the annotated entries). We then render examples of each adjudicated argument branch in MathJax, KaTeX and XeTeX to record the entry's support across renderers. Each rewrite rule is proposed from adjudicated equivalences and LaTeX conventions, reviewed against rendered test cases that set its level and fidelity, and implemented as a Rust function whose metadata declares the node forms that trigger it and those it eliminates or produces. The knowledge base holds 1,213 command and character records and 35 environments, and the rule set holds 151 rules (construction details in Appendix B). Both the knowledge base and the rule set are built offline, so at run time TeXForm reads the knowledge base as a static table and calls neither an LLM nor a renderer; it is implemented as a Rust library with Python bindings and JavaScript bindings compiled to WebAssembly.

Parsing. The parser descends recursively through the math constructs and queries the knowledge base at each command and environment, so the knowledge base fixes $\mathrm{dom}(N)$, the declared domain that coverage requires: a command it does not describe is either rejected or kept as an explicitly unknown node.

Transforming. The transform engine rewrites the tree by matching the patterns that rules declare and replacing the matched subtrees. Alongside rewriting, rule-independent passes canonicalize style commands and remove redundant braces, and the engine repeats all of these steps until none of them changes the tree. The engine topologically sorts rules by the dependencies their declared node forms induce, and a contract check after the last transform confirms that each promised elimination holds. To serve uses needing different trade-offs between safety and canonicity, we group the rules into four cumulative profiles: Authoring, Faithful, Corpus and Equiv (details in Appendix B.2). The declared equivalence therefore differs from profile to profile, and TeXForm is four normalizers in the sense of Definition 1, one per profile. Our main experiments compare the three profiles intended for normalized outputs, and Appendix C.2 reports Equiv separately.

Serializing. The serializer emits a LaTeX string from the transformed tree. Each node records its content mode, the argument slot it fills, and its lexical boundaries, so the serializer translates node by node and one tree yields one string. To achieve idempotence, TeXForm combines this common fixed point with a serializer whose output reparses into an equivalent tree.

Table 1: Corpus-level rates (%) of safety, canonicity, idempotence, and coverage (Section 4); bold marks the best value in a block, † a tool that one TeXForm profile exceeds on safety, canonicity and idempotence at once. Above 90% safety no prior tool folds more than a fifth of the equivalent pairs, while TeXForm's three profiles fold 29–47%; coverage varies only from 98.32 to 99.40%. Each CDM row audits the normalizer embedded in that release, not CDM's matching mechanism.

Normalizer	Role	Safety	Pair canonicity	Idempotence	Coverage
Identity	Anchor	100.00	0.00	100.00	99.40
UniMERNet postprocess	Recognizer output	**99.83**	8.36	**100.00**	99.40
im2markup†	Data preparation	94.43	18.65	63.30	99.06
MathNet	Data preparation	59.65	**66.42**	98.48	98.32
CDM (in UniMERNet)†	Metric front-end	86.77	25.14	90.32	99.34
CDM (in OmniDocBench v1.5)†	Metric front-end	87.68	25.01	90.31	99.34
CDM (in OmniDocBench v1.6)†	Metric front-end	86.22	24.96	90.64	98.97
TeXForm (Authoring)	Proposed	**98.85**	28.54	**100.00**	99.16
TeXForm (Faithful)	Proposed	98.84	37.50	**100.00**	99.16
TeXForm (Corpus)	Proposed	97.20	**46.58**	**100.00**	99.16

Table 2: Rendering-equivalence audits (%) of three benchmark tracks under official full-score rules and with TeXForm (Faithful) (*). Recovered is the share of false rejects turned into accepts; false-accept-candidate rates are the rendering-inequivalent share of accepted pairs with a rendering verdict. Reference safety measures $x \equiv N(x)$. Paired cells read official → TeXForm*. ‡A further 28.00% of rendering-equivalent pairs are unavailable. UniMER-Test pools four subsets with false-reject rates of 6.00–54.54%.

Benchmark track	Full-score rule	False reject	Recovered	False-accept candidates	Reference safety
OCRBench v2	Substring	19.97 → 7.35‡	33.33	3.51 → 1.74	52.58 → 97.64
UniMER-Test	Edit distance 0	26.36 → 16.33	38.05	0.24 → 0.16	96.88 → 98.47
OmniDocBench v1.6	CDM $F_1 = 1$	18.98 → 3.53	89.83	41.99 → 36.91	71.69 → 97.60

6 Experiments

6.1 Setup and Measurement Criterion

We test normalization at three positions: as a tool in its own right (Section 6.2), as the step a dataset applies before publishing labels (Section 6.3), and as the step a benchmark applies before deciding whether two strings are the same (Section 6.4). TeXForm is audited alongside the other tools and also serves as the replacement normalizer in the last two. We audit six deployed normalization implementations spanning dataset preparation, recognizer post-processing, and metric front-ends, together with an identity anchor and the three TeXForm profiles aimed at normalized output (selection rationale in Appendix C.1). The models and evaluation sets are likewise case studies rather than estimates of a false-verdict rate across the ecosystem (Appendix E).

Rendering equivalence. We make every comparison against one criterion: each formula is rendered by a fixed MathJax renderer (The MathJax Consortium, 2025), rasterized at a fixed resolution, and compared under a frozen pixel-level test (protocol in Appendix A). This *rendering-equivalence criterion* inherits the convention of Section 4 that a string which fails to render is equivalent to nothing, so a silently emptied output violates safety outright. It is finer than mathematical equivalence, so the two error directions behave differently: a false reject stays a false reject under any coarser equivalence, while a false accept holds only relative to this criterion, and we call it a *false-accept candidate* throughout. To test whether our readings depend on the choice of renderer, pixel threshold, or canonicity accounting, Appendix F repeats the audits under KaTeX and XeTeX, under strict pixel identity and a grid of thresholds, and under cluster-level accounting.

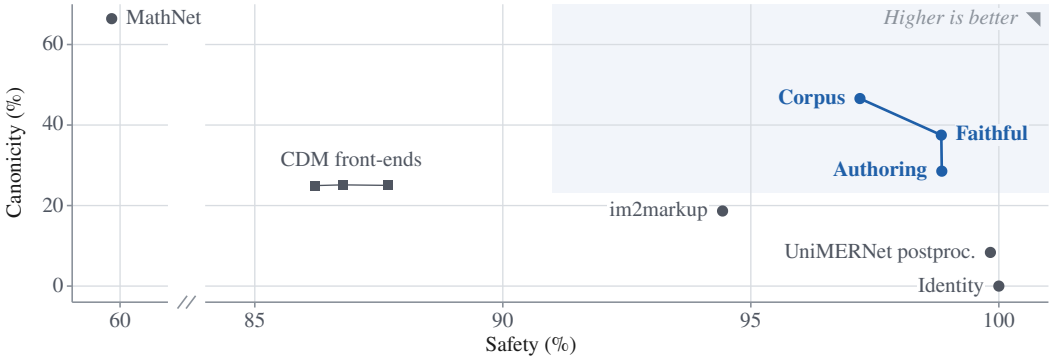

Figure 3: Safety and pair-level canonicity of the audited tools (gray) and TeXForm's three profiles (blue); canonicity counts equivalent pairs differing beyond whitespace and redundant braces. The three joined squares are the CDM front-ends shipped with UniMERNet and with OmniDocBench v1.5 and v1.6; the safety axis is broken between 62 and 84%.

6.2 Normalizer Properties

The six deployed implementations we audit are the dataset-construction script of im2latex-100k (Deng et al., 2017), MathNet's preprocessing normalizer (Schmitt-Koopmann et al., 2024), UniMERNet's output post-processor (Gu et al., 2026), and the processors embedded in three CDM releases (Wang et al., 2025; Ouyang et al., 2025). Canonicity is measured on equivalence pairs drawn from benchmark labels and from the model predictions collected in Section 6.4, independently of any audited tool, so the tool comparison and the evaluation-side audit read the same corpus.

Among the deployed tools, safety and canonicity trade against each other (Table 1, Figure 3): UniMERNet's post-processor stays safe by folding almost nothing, MathNet folds the most while altering the rendering of two in five inputs, and every tool marked † is exceeded by a single TeXForm profile on safety, canonicity, and idempotence at once. Moving from Authoring through Faithful to Corpus, TeXForm's profiles raise canonicity at each step while giving up little safety. What Faithful still changes is mostly layout: 95% of its render drifts leave the glyphs and structure of the MathML output unchanged, and content or structure changes in 0.02% of the audited inputs (Appendix B.8).

6.3 Normalization in Dataset Construction

At the data end normalization is usually impossible to audit: most datasets publish only post-normalization labels, and the raw string disappears at release. Two public lineages are exceptions: MathWriting-2024 (649,196 pairs) and im2latex-100k (94,000) publish raw and normalized columns with sample-level alignment, MathWriting's normalization being an intentional task abstraction its authors declare. Their own normalization preserves the rendering of 62.02% and 92.79% of labels, where TeXForm (Faithful) on the same raw strings preserves 99.70% and 97.01% (Table 12). The rewritten labels remain renderable while producing different images, and propagate verbatim into downstream corpora and benchmark ground truths (propagation chains in Appendix C.6). In a two-annotator review of sampled changes, spacing-only edits account for 43–47% of the inspected im2latex sample, and the two annotators place 89% and 90% of 100 divergent MathWriting pairs within its declared abstraction (categories and examples in Appendix D.3).

6.4 Normalization in Benchmark Scoring

Evaluation-side normalization is applied before the metric decides, so its defects become the defects of the decision. We replay the official scoring of three benchmark tracks on frozen predictions and compare each official full-score rule against rendering equivalence (Table 2): the text-metric tracks of UniMER-Test (23,757 formulas) and OCRBench v2 (508) over 12 models, specialist recognizers and general-purpose VLMs alike (Cui et al., 2025; Duan et al., 2026; Hunyuan Vision Team et al., 2025; Niu et al., 2026; Bai et al., 2025b; Zhu et al., 2025), and OmniDocBench v1.6's display-

Table 3: UniMER-Test exact match on reference-renderable samples ($n = 23{,}300$) under the official and TeXForm (Faithful) front-ends (*); bold is the highest rate in each column, underline the second, # the rank. Replacing the front-end raises every model's rate and reverses the official top pair.

Model	Bench.	#	TeXForm*	#	Δ
UniMERNet-base	**0.4630**	1	0.4962	3	+0.0331
PaddleOCR-VL-1.6	<u>0.4414</u>	2	**0.5189**	1	+0.0775
PP-FormulaNet-L	0.4203	3	<u>0.5040</u>	2	+0.0838
PaddleOCR-VL	0.3695	4	0.4261	4	+0.0566
PaddleOCR-VL-1.5	0.3489	5	0.4062	5	+0.0573
MinerU2.5	0.3337	6	0.3527	6	+0.0190
GLM-OCR	0.3152	7	0.3290	7	+0.0138
Qwen2.5-VL-7B-Instruct	0.2979	8	0.3136	9	+0.0157
HunyuanOCR v1.0	0.2933	9	0.3107	10	+0.0174
Qwen3-VL-8B-Instruct	0.2428	10	0.3180	8	+0.0752
InternVL3-8B	0.2275	11	0.2906	11	+0.0631
pix2tex	0.1255	12	0.1373	12	+0.0118

formula track over 11, scored at one third of the overall weight with CDM, a render-aware metric that still takes its input from a processor of the same lineage (Section 3).

All three tracks reject a fifth to a quarter of the pairs rendering equivalence deems equivalent, and all three admit false-accept candidates. We replace the normalization front-end with TeXForm (Faithful), retaining each track's input and full-score rule, including OCRBench's raw outputs and substring acceptance. On UniMER-Test and OmniDocBench the replacement both recovers false rejects and removes false-accept candidates on net; on OCRBench's raw outputs, true accepts fall from 1,046 to 845, with 280 previous accepts becoming unavailable because the raw outputs wrap the formula in prose or delimiters. Reference safety rises on all three tracks (Table 2). CDM scores through its own pdfLaTeX backend, and 30.8% of its scored rows lack a MathJax rendering verdict; conditional rates use the judged samples (missing verdicts in Appendix E.6).

Replacing the normalization front-end changes model rankings under a fixed scoring rule (Table 3). Every model's exact-match rate rises, and the relative order of the two leading models reverses: the exact-match gap of UniMERNet-base over PaddleOCR-VL-1.6 is $+0.0216$ (paired bootstrap 95% CI [0.0145, 0.0280]) under the official front-end and -0.0227 (95% CI $[-0.0298, -0.0158]$) under TeXForm. The other two tracks retain their leaders (Appendix E.7).

7 Limitations

No TeXForm profile is fully safe: each still changes the rendering of a small share of inputs, and that share grows with the canonicity the profile targets, so a pipeline should choose its profile by the end it serves. Nor does any profile approach full canonicity: even Corpus leaves more than half of the render-equivalent pairs in the audit corpus distinct, because of distinctions its policy retains, missing rewrites, and normalization failures, so matching normalized strings still rejects many equivalent pairs. A command outside TeXForm's finite knowledge base, such as a user-defined macro, is rejected or kept as an opaque node that no rule rewrites. Uncovered commands and missing rewrites can be added by extending the knowledge base and rule set, which will continue to evolve.

8 Conclusion

We define LaTeX normalization as a task with four measurable properties and state when matching normalized strings is a valid equivalence test. Our audits find failures in both directions: published labels are rewritten into formulas that render differently and reach benchmark references, and benchmark normalization falsely rejects enough equivalent predictions to reverse one leaderboard's top position. Of the audited implementations, TeXForm alone combines substantial canonicity with high render-safety; we open-source it. Because normalization decides what a dataset stores and what a benchmark accepts, it should be an explicit, versioned, and replaceable component of both.

REPRODUCIBILITY STATEMENT

We provide supplementary material containing the TeXForm source, knowledge base, rewrite rules, and code for the main dataset, normalizer, and benchmark audits, transform-fidelity evaluation, and pipeline ablation. The accompanying README files document environment setup, public-data preparation, model inference, and experiment execution. Appendix A specifies the rendering criterion, audit supports, rate denominators, and resampling settings; Appendix C.2 describes the frozen normalizer implementations; and Appendix E.1 documents the model checkpoints and inference setup. All benchmark comparisons use fixed model predictions. The knowledge-base annotation pipeline, human damage-type annotations, and review applications are not included.

AI USE STATEMENT

We used large language models to assist with LaTeX typesetting, polish the writing, and draft parts of the manuscript, including the related work section. We also used AI-assisted deep research tools to search for, identify, and review related literature. We used LLMs to assist with Rust development for TeXForm, implementation of the experimental code, and interpretation of the experimental results. LLM-assisted knowledge-base annotation and human adjudication are described in Section 5 and Appendix B.3. We take responsibility for the final content of this paper, including the text, claims, code, and experimental results produced with AI assistance.

REFERENCES

Moody E. Altamimi and Abdou Youssef. A More Canonical Form of Content MathML to Facilitate Math Search. In *Proceedings of Extreme Markup Languages*, 2007.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report, 2025a. URL https://arxiv.org/abs/2511.21631. Version Number: 2.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report, 2025b. URL https://arxiv.org/abs/2502.13923. Version Number: 1.

Lukas Blecher. pix2tex: LaTeX OCR, 2022. URL https://github.com/lukas-blecher/LaTeX-OCR.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural Optical Understanding for Academic Documents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fUtxNAKpdV.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Handong Zheng, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. PaddleOCR-VL: Boosting Multilingual Document Parsing via a 0.9B Ultra-Compact Vision-Language Model, 2025. URL https://arxiv.org/abs/2510.14528. Version Number: 4.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. PaddleOCR-VL-1.5: Towards a Multi-Task 0.9B VLM for Robust In-the-Wild Document Parsing, 2026. URL https://arxiv.org/abs/2601.21957. Version Number: 2.

Kenny Davila and Richard Zanibbi. Layout and Semantics: Combining Representations for Mathematical Formula Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1165–1168, Shinjuku Tokyo Japan, August 2017. ACM. ISBN 978-1-4503-5022-8. doi: 10.1145/3077136.3080748. URL https://dl.acm.org/doi/10.1145/3077136.3080748.

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. Image-to-Markup Generation with Coarse-to-Fine Attention. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 980–989. PMLR, August 2017. URL https://proceedings.mlr.press/v70/deng17a.html.

Shuaiqi Duan, Yadong Xue, Weihan Wang, Zhe Su, Huan Liu, Sheng Yang, Guobing Gan, Guo Wang, Zihan Wang, Shengdong Yan, Dexin Jin, Yuxuan Zhang, Guohong Wen, Yanfeng Wang, Yutao Zhang, Xiaohan Zhang, Wenyi Hong, Yukuo Cen, Da Yin, Bin Chen, Wenmeng Yu, Xiaotao Gu, and Jie Tang. GLM-OCR Technical Report, 2026. URL https://arxiv.org/abs/2603.10910. Version Number: 2.

Philippe Faist. pylatexenc: A LaTeX parser for Python, 2023. URL https://github.com/phfaist/pylatexenc.

Ling Fu, Zhebin Kuang, Jiajun Song, Mingxin Huang, Biao Yang, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, Zhang Li, Guozhi Tang, Bin Shan, Chunhui Lin, Qi Liu, Binghong Wu, Hao Feng, Hao Liu, Can Huang, Jingqun Tang, Wei Chen, Lianwen Jin, Yuliang Liu, and Xiang Bai. OCRBench v2: An Improved Benchmark for Evaluating Large Multimodal Models on Visual Text Localization and Reasoning. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025), Datasets and Benchmarks Track*. Curran Associates, Inc., 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/8c2e6bb15be1894b8fb4e0f9bcad1739-Abstract-Datasets_and_Benchmarks_Track.html.

Philippe Gervais, Anastasiia Fadeeva, and Andrii Maksai. MathWriting: A Dataset For Handwritten Mathematical Expression Recognition. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 5459–5469, Toronto ON Canada, August 2025. ACM. ISBN 979-8-4007-1454-2. doi: 10.1145/3711896.3737436. URL https://dl.acm.org/doi/10.1145/3711896.3737436.

Zhuangcheng Gu, Guang Liang, Bin Wang, Zhiyuan Zhao, Qintong Zhang, Weijia Li, Chao Xu, Bo Zhang, Botian Shi, Jiang Wu, Wentao Zhang, and Conghui He. UniMERNet: A Universal Network for Real-World Mathematical Expression Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 34106–34115, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Gu_UniMERNet_A_Universal_Network_for_Real-World_Mathematical_Expression_Recognition_CVPR_2026_paper.html.

Hunyuan Vision Team, Pengyuan Lyu, Xingyu Wan, Gengluo Li, Shangpin Peng, Weinong Wang, Liang Wu, Huawen Shen, Yu Zhou, Canhui Tang, Qi Yang, Qiming Peng, Bin Luo, Hower Yang, Xinsong Zhang, Jinnian Zhang, Houwen Peng, Hongming Yang, Senhao Xie, Longsha Zhou, Ge Pei, Binghong Wu, Rui Yan, Kan Wu, Jieneng Yang, Bochao Wang, Kai Liu, Jianchen Zhu, Jie Jiang, Linus, Han Hu, and Chengquan Zhang. HunyuanOCR Technical Report, 2025. URL https://arxiv.org/abs/2511.19575. Version Number: 2.

Kyudan Jung, Nam-Joon Kim, Hyun Gon Ryu, Sieun Hyeon, Seung-Jun Lee, and Hyuk-Jae Lee. TeXBLEU: Automatic Metric for Evaluate LaTeX Format. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, Hyderabad, India, April 2025. IEEE. ISBN 979-8-3503-6874-1. doi: 10.1109/ICASSP49660.2025.10888244. URL https://ieeexplore.ieee.org/document/10888244/.

Khan Academy. KaTeX: Fast math typesetting for the web, 2024. URL https://github.com/KaTeX/KaTeX.

Hynek Kydlíček. Math-Verify: A robust mathematical expression evaluation system, 2025. URL https://github.com/huggingface/Math-Verify.

Yu Li, Jin Jiang, Jianhua Zhu, Shuai Peng, Baole Wei, Yuxuan Zhou, and Liangcai Gao. UniMuMER: Unified Multi-Task Fine-Tuning of Vision-Language Model for Handwritten Mathematical Expression Recognition. In *Advances in Neural Information Processing Systems 38*, pp. 143134–143168, San Diego, California, USA and Mexico City, Mexico, 2025a. Neural Information Processing Systems Foundation, Inc. (NeurIPS). ISBN 979-8-3313-3827-5. doi: 10.52202/085713-4297. URL http://www.proceedings.com/085713-4297.html.

Yumeng Li, Guang Yang, Hao Liu, Bowen Wang, and Colin Zhang. dots.ocr: Multilingual Document Layout Parsing in a Single Vision-Language Model, 2025b. URL https://arxiv.org/abs/2512.02498. Version Number: 4.

Hongen Liu, Cheng Cui, Yuning Du, Yi Liu, and Gang Pan. PP-FormulaNet: Bridging Accuracy and Efficiency in Advanced Formula Recognition, 2025. URL https://arxiv.org/abs/2503.18382. Version Number: 1.

John MacFarlane. texmath: Conversion between math formats, 2025. URL https://github.com/jgm/texmath.

Mahshad Mahdavi, Richard Zanibbi, Harold Mouchere, Christian Viard-Gaudin, and Utpal Garain. ICDAR 2019 CROHME + TFD: Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1533–1538, Sydney, Australia, September 2019. IEEE. ISBN 978-1-7281-3014-9. doi: 10.1109/ICDAR.2019.00247. URL https://ieeexplore.ieee.org/document/8978036/.

Bruce R. Miller. LaTeXML: A LaTeX to XML/HTML/MathML Converter, 2024. URL https://github.com/brucemiller/LaTeXML.

Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, Fan Wu, Qintong Zhang, Zhenjiang Jin, Guang Liang, Rui Zhang, Wenzheng Zhang, Yuan Qu, Zhifei Ren, Yuefeng Sun, Zirui Tang, Boyu Niu, Yuanhong Zheng, Dongsheng Ma, Ziyang Miao, Hejun Dong, Siyi Qian, Junyuan Zhang, Fangdong Wang, Jingzhou Chen, Xiaomeng Zhao, Liqun Wei, Wei Li, Shasha Wang, RuiLiang Xu, Yuanyuan Cao, Lu Chen, Qianqian Wu, Huaiyu Gu, Lindong Lu, Dechen Lin, Shenguanlin, Xuanhe Zhou, Linfeng Zhang, Yuhang Zang, Xiaoyi Dong, Jiaqi Wang, Bo Zhang, Lei Bai, Pei Chu, Weijia Li, Jiang Wu, Lijun Wu, Zhenxiang Li, Guangyu Wang, Zhongying Tu, Chao Xu, Kai Chen, Bowen Zhou, Dahua Lin, Wentao Zhang, and Conghui He. MinerU2.5: A Decoupled Vision-Language Model for Efficient High-Resolution Document Parsing. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 13–42, San Diego, California, USA, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-industry.3. URL https://aclanthology.org/2026.acl-industry.3.

Curtis G. Northcutt, Anish Athalye, and Jonas Mueller. Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/f2217062e9a397a1dca429e7d70bc6ca-Abstract-round1.html.

OleehyO. latex-formulas-80M. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/OleehyO/latex-formulas-80M.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. OmniDocBench: Benchmarking Diverse PDF Document Parsing with Comprehensive Annotations. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24838–24848, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.02313. URL https://ieeexplore.ieee.org/document/11091844/. arXiv: 2412.07626.

Matt Post. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191. Association for Computational Linguistics, 2018. doi: 10.18653/v1/w18-6319. URL http://dx.doi.org/10.18653/v1/W18-6319.

Felix M. Schmitt-Koopmann, Elaine M. Huang, Hans-Peter Hutter, Thilo Stadelmann, and Alireza Darvishy. MathNet: A Data-Centric Approach for Printed Mathematical Expression Recognition. *IEEE Access*, 12:76963–76974, 2024. ISSN 2169-3536. doi: 10.1109/ACCESS.2024.3404834. URL https://ieeexplore.ieee.org/document/10538105/.

Jason Siefken. unified-latex: Utilities for parsing and manipulating LaTeX ASTs with the Unified.js framework, 2024. URL https://github.com/siefkenj/unified-latex.

The MathJax Consortium. MathJax: Beautiful and accessible math in all browsers, 2025. URL https://www.mathjax.org.

Bin Wang, Fan Wu, Linke Ouyang, Zhuangcheng Gu, Rui Zhang, Renqiu Xia, Botian Shi, Bo Zhang, and Conghui He. Image Over Text: Transforming Formula Recognition Evaluation with Character Detection Matching. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19681–19690, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.01833. URL https://ieeexplore.ieee.org/document/11094480/.

François Wieckowiak, Véronique Eglin, Tony Bonnet, Stéphane Bres, and Laëtitia Rousseau. A Multimodal Evaluation Pipeline for Mathematical Expression Recognition: Comparisons of Datasets, Metrics, and Models. In Xu-Cheng Yin, Dimosthenis Karatzas, and Daniel Lopresti (eds.), *Document Analysis and Recognition – ICDAR 2025*, volume 16026, pp. 120–136. Springer Nature Switzerland, Cham, 2026. ISBN 978-3-032-04626-0 978-3-032-04627-7. doi: 10.1007/978-3-032-04627-7_7. URL https://link.springer.com/10.1007/978-3-032-04627-7_7. Series Title: Lecture Notes in Computer Science.

Hao Wu, Haoran Lou, Xinyue Li, Zuodong Zhong, Zhaojun Sun, Phellon Chen, Xuanhe Zhou, Kai Zuo, Yibo Chen, Xu Tang, Yao Hu, Boxiang Zhou, Jian Wu, Yongji Wu, Wenxin Yu, Yingmiao Liu, Yuhao Huang, Manjie Xu, Gang Liu, Yidong Ma, Zhichao Sun, and Changhao Qiao. FireRed-OCR Technical Report, 2026. URL https://arxiv.org/abs/2603.01840. Version Number: 1.

Yejing Xie, Harold Mouchère, Foteini Simistira Liwicki, Sumit Rakesh, Rajkumar Saini, Masaki Nakagawa, Cuong Tuan Nguyen, and Thanh-Nghia Truong. ICDAR 2023 CROHME: Competition on Recognition of Handwritten Mathematical Expressions. In Gernot A. Fink, Rajiv Jain, Koichi Kise, and Richard Zanibbi (eds.), *Document Analysis and Recognition - ICDAR 2023*, volume 14188, pp. 553–565. Springer Nature Switzerland, Cham, 2023. ISBN 978-3-031-41678-1 978-3-031-41679-8. doi: 10.1007/978-3-031-41679-8_33. URL https://link.springer.com/10.1007/978-3-031-41679-8_33. Series Title: Lecture Notes in Computer Science.

Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. NTCIR-12 MathIR Task Overview. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*, 2016. URL https://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/Overview/01-NTCIR12-OV-MathIR-ZanibbiR.pdf.

Yu Zhang, Xiusi Chen, Bowen Jin, Sheng Wang, Shuiwang Ji, Wei Wang, and Jiawei Han. A Comprehensive Survey of Scientific Large Language Models and Their Applications in Scientific Discovery. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8783–8817, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.498. URL https://aclanthology.org/2024.emnlp-main.498.

Zelun Zhang, Hongen Liu, Suyin Liang, Yubo Zhang, Yiqing Xiang, Jiaxuan Liu, Ting Sun, Manhui Lin, Yue Zhang, Changda Zhou, Tingquan Gao, Cheng Cui, Yi Liu, Dianhai Yu, and Yanjun Ma. PaddleOCR-VL-1.6: Expanding the Frontier of Document Parsing with Under-Optimized Region

Refinement and Progressive Post-Training, 2026. URL https://arxiv.org/abs/2606.03264. Version Number: 1.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. InternVL3: Exploring Advanced Training and Test-Time Recipes for Open-Source Multimodal Models, 2025. URL https://arxiv.org/abs/2504.10479. Version Number: 3.

APPENDIX

The appendices give the measurement protocol, the construction of TeXForm, and the evidence behind each audit in the main text. Appendix A specifies the rendering criterion, the audit corpora, and the denominator of every rate. Appendix B describes how TeXForm, its knowledge base, and its rules are built and checked, and breaks down its residual failures (Sections 5 and 6.2). Appendix C documents the audited normalizers and their shared lineage (Section 3 and Table 1). Appendices D and E give the dataset-side and evaluation-side evidence behind Sections 6.3 and 6.4. Appendix F repeats the audits under other renderers, pixel thresholds, and canonicity accounting (Section 6.1).

A FORMAL TASK AND MEASUREMENT PROTOCOL

Section 4 defines the normalization task. Here we specify the rendering criterion, audit corpora, and aggregation rules used to measure its four properties and the resulting benchmark decisions.

A.1 RENDERING-EQUIVALENCE CRITERION

The common criterion renders each formula with MathJax 4.1.1 and rasterizes its SVG with resvg 2.6.2. The package set is fixed to **`base`**, **`ams`**, **`textmacros`**, **`boldsymbol`**, **`bboldx`**, and **`braket`**; each source record retains its display or inline mode. Section F repeats the audit with KaTeX and XeTeX.

A pixel comparator evaluates the full frames and an alternative obtained by cropping white margins and aligning the shorter image dimension, keeping the better similarity reading. A channel difference greater than 2 marks a pixel as different. Images match exactly when their dimensions and pixels agree, or approximately when SSIM is at least 0.99 and the differing-pixel ratio is at most 0.001. Each run records renderer versions, comparison settings, and source fingerprints; cached comparison statistics support threshold rejudgment.

Proposition 1 concerns exact rendering equality. The threshold-based criterion above is an empirical approximation and need not be transitive, so the proposition does not establish an exact equivalence test for this approximate relation.

Pairwise outcomes are **`pixel-fallback`** (accepted), **`pixel-reject`** (rejected), or **`render_error`** (unavailable). Rendering errors remain separate in pairwise audits. For input–output safety, an empty or unrenderable output from a renderable input counts as a failure. Section F.2 compares the frozen thresholds with strict pixel identity and a threshold grid.

A.2 AUDIT CORPORA AND EQUIVALENCE-PAIR CONSTRUCTION

The audit protocol assigns distinct fixed supports to single-input safety, pairwise canonicity, dataset fidelity, standalone renderability, and benchmark scoring because these quantities require different observational units. Table 4 records those supports before any property rate is computed.

The canonicity corpus is constructed independently of every normalizer under audit: its members are third-party benchmark labels and predictions that the fixed rendering criterion has already placed in the same benchmark-sample cluster. Identical strings are merged while their provenance is retained, so repeated production of one spelling cannot increase its weight. An unordered pair enters the primary denominator only when at least one side differs from the cluster reference by more than whitespace and redundant braces, preventing trivial formatting pairs from dominating the reading. The generic leave-one-out accounting removes members contributed solely by the audited tool while retaining the same string when raw data or another provenance also contributes it; normalization failures remain in the unconditional denominator as failures to collapse. Because the primary model-prediction corpus contains no audited-tool outputs, this step leaves its support unchanged and serves as an explicit independence check. Of the 19,787 source clusters, 18,899 contain at least two distinct strings before the nontrivial-variation filter; 5,441 contain at least one member differing from the reference beyond whitespace and redundant braces and enter the cluster-level denominator. These clusters supply the 29,528 eligible pairs. Cluster-level canonicity requires every retained member of each of these 5,441 clusters to converge to one string.

Table 4: Fixed audit supports and their construction. Counts describe the audited artifacts themselves and are not estimates of an ecosystem-wide population.

Audit role	Fixed support	Source	Construction
Paired-label fidelity	743,196 raw–published pairs	MathWriting-2024 and im2latex-100k	Full enumeration of the publisher-aligned pairs; the two sides are compared without consulting the formula image.
Standalone label renderability	2,764,004 label instances	im2latex-100k, LaTeX-OCR, MathWriting-2024, OmniDocBench v1.6, and UniMER	Full enumeration by label kind, with MathJax followed by XeTeX only when MathJax fails.
Normalizer safety	347,182 formulas	MathWriting human raw labels and im2latex raw labels	All rows selected by the frozen dataset and split filters; the audit applies no sampling limit.
Normalizer canonicity	66,047 members in 19,787 clusters; 5,441 clusters and 29,528 pairs enter canonicity rates	Frozen benchmark labels and model predictions	Exact-string de-duplication within benchmark-sample clusters followed by the nontrivial-pair and leave-one-out filters described below.
Benchmark scoring	12 models on 23,300 and 466 reference-renderable formulas; 11 models on page-track rows	UniMER-Test, OCRBench v2 formula, and OmniDocBench v1.6 display formulas	The retained text-track source scopes contain 23,757 and 508 formulas, while the page track uses model-specific fixed ground-truth-backed match rows drawn from 1,651 pages.

A.3 Denominators, Abstentions, and Statistical Uncertainty

For every reported rate, we define its numerator, denominator, and unavailable-state handling before aggregation. The rates use the following conventions. The official OmniDocBench false-reject rate is conditional on backend availability: seven of the 7,000 rendering-equivalent rows fail in the official backend, leaving a denominator of 6,993; Faithful scores all 7,000. These seven failures are unavailable outcomes, not rejects. Fixed-support model scores still retain them as zero.

- **Paired-label fidelity** divides accepted pixel verdicts by pairs with a final relation verdict; render errors are excluded from that rate and reported separately, while joint comparisons require both candidate relations to have verdicts.
- **Standalone renderability** divides labels rendered by either configured engine by all label instances in the stated scope; a label that fails both engines remains in the denominator as a standalone-unrenderable candidate.
- **Safety** divides rendering-equivalent input–output pairs by rows that the normalizer accepts and whose raw input renders; empty outputs, unrenderable outputs, comparison errors, and render drift remain failures, whereas refusals and raw-render failures leave this denominator.
- **Coverage** measures usable-output coverage on the audit corpus: nonempty, MathJax-renderable first outputs divided by all single-input audit records, exposing refusals that would otherwise raise conditional safety.
- **Idempotence** divides exact second-pass string repetitions by rows whose nonempty first output enters a second normalization pass; second-pass normalization errors and string drift count as failures, while second-output renderability is reported as a separate diagnostic.
- **Pair-level canonicity** divides eligible unordered pairs whose two normalized strings are identical by all eligible pairs after leave-one-out; normalization failures count as non-collapse rather than abstentions.
- **False rejects** for a normalization front-end divide its rejections by rows that the rendering criterion accepts, with an unavailable front-end verdict retained in the fixed relation denominator

and reported separately except for the official OmniDocBench backend failures specified above; **false-accept candidates** divide rendering-criterion rejections by that front-end's accepted rows with a final rendering verdict under the specified decision rule.

- **Ranking and normalized exact-match rates** use fixed reference-renderable support shared across models; prediction, extraction, or normalization failure remains in that support and scores zero, while covered-only rates are labeled as coverage sensitivity and are not used for ranking.

The full-enumeration and fixed-corpus audits are descriptive measurements of their named artifacts, so we report exact counts rather than attach a population confidence interval to an unspecified ecosystem. Benchmark uncertainty uses paired resampling: formula-string tracks resample sample identifiers with replacement, and the page benchmark resamples formula-bearing page identifiers as clusters. We draw $1{,}000$ resamples with seed 42, share each resample across models and normalization front-ends, and report a 95% percentile interval only when at least 80% of resamples are valid. Sharing resamples preserves the pairing needed for score differences, fixed adjacent-model gaps, and head-to-head normalization transitions.

B TeXForm Construction and Verification

TeXForm combines a knowledge-driven parser, a tree transform engine, and a serializer. Offline pipelines produce the command specifications and rewrite rules consumed by this runtime.

B.1 Runtime Architecture and Declared Domain

The parser uses command, environment, character, and delimiter specifications to recover typed argument slots, modes, scripts, infix structure, and environment bodies. The knowledge assets span seven packages: 584 command records, 629 character records, 36 environment records covering 35 unique names, and 48 delimiter records. Command and character records total $1{,}213$.

Parsing returns a complete document, a partial document retaining unrecovered source in error nodes, or a hard failure. Normalization accepts complete documents from the same parser context. Unknown commands can be rejected or retained as explicit zero-argument nodes; enabled packages and parsing modes determine the active domain. The serializer uses node modes, argument slots, and lexical boundaries to emit a stable string from the transformed tree.

B.2 Profile Policies and Fidelity Metadata

TeXForm profiles enable cumulative rule levels and select a group-flattening preset (Table 5). Four semantic guards protect script binding, environment cells, declarative scope, and infix scope; six spacing guards control atom spacing and unary/binary classification. Authoring and Faithful retain all ten guards. Corpus and Equiv retain the four semantic guards. Moving from Faithful to Corpus adds 21 rules and changes the flattening preset.

Table 5: Cumulative TeXForm profile policies, group-flattening presets, and registered rewrite counts.

Profile	Rules	Group flattening	Declared product policy
Authoring	70	All guards	Canonical author-facing syntax that folds clear aliases while retaining author-visible choices.
Faithful	119	All guards	A presentation-preserving normalized form that additionally expands convenience and legacy structures into more general syntax.
Corpus	140	Semantic only	A stable corpus label that additionally removes selected layout hints while retaining a complete label for the original formula.
Equiv	151	Semantic only	An aggressive intermediate for equivalence comparison that may fold visually salient presentation choices and is not recommended as a published corpus label.

A rule's level identifies the first profile that enables it; its fidelity states the weakest declared relation it preserves, from mathematical meaning through reading order and structure to rendering. Rule review checks the declared fidelity against the rule's rendered examples, and the transform runtime does not call a renderer when a rule fires. These two metadata fields guide rule selection and interpretation of the profile comparisons; the corpus-scale audit below reports how far each profile preserves rendering in practice.

B.3 Knowledge-Base Construction and Adjudication

The pipeline extracts a deterministic catalog from MathJax, obtains independent annotations, reviews every record, and generates runtime tables. The catalog records each entry's package, parsing method, aliases, modes, and implementation source. Tasks group commands and environments by their parsing method. GPT-5.4, Gemini 3.1 Pro, and Claude Sonnet 4.6 annotate each group independently at their highest reasoning settings.

All three annotators use the same tools: a read-only shell over the vendored MathJax source, a MathJax parser for checking what the reference implementation accepts, and a temporary TeXForm context that registers candidate argument specifications and parses snippets against them. Every submitted example must parse under MathJax and under the candidate specification, and the annotator inspects the resulting tree, because an argument that is silently left behind as a trailing sibling still parses. Each record stores its runtime signature, rewrite evidence, and validation examples. A human reviewer compares the source and all annotations, then selects, combines, or rewrites the specification with a recorded justification. MathJax behavior is the primary reference; LaTeX2e and package documentation provide supporting evidence.

Of 621 annotatable records across 140 parsing-method tasks, 612 have all three annotations. The models agree on command kind and argument specification for 487 records (79.6%). The final specification differs from all three proposals for 104 records (17.0%), including 81 on which the models had agreed. These counts concern command kind and argument specification. They show the contribution of source-based human adjudication even when model annotations agree.

The generator writes runtime YAML from the catalog and review decisions. It handles environment delimiters structurally, adds a manual specification for the separator in **`\root ... \of`**, and generates character and delimiter records directly from the catalog. Corpus parse rates do not enter adjudication, and none of the generator's inputs is derived from a corpus. Corpus-level regression tracks parser error-rate drift after a change but never acts as an objective that a specification is tuned toward.

B.4 Transform Scheduling and Fixed-Point Execution

The transform engine runs four phases. LowerAttributes unifies declarative and prefix styles while preserving scope; Rewrite applies the rule plan to its own fixed point, as described below; FinalizeAst merges adjacent primes and text runs and removes empty text; FlattenGroups removes redundant braces under the profile's preserve guards, except that a brace layer directly inside a command argument is always removed. Algorithm 1 repeats the phases in a fixed order and skips a phase that has already processed the current tree, so a change made by any phase, such as flattening that exposes adjacent style commands or a new rewrite match, makes the other phases run again. Each phase reaches its own fixed point in one invocation and reports every change it makes. When the loop exits, every enabled phase has processed the current tree, so running any of them again leaves the tree unchanged: the result is a common fixed point of all enabled phases, reached without enumerating how the phases interact. Rewrite, FinalizeAst, and FlattenGroups report no change at their own fixed points, while LowerAttributes may conservatively report a change when it regenerates an identical tree; this keeps an already stable tree from exhausting the round budget, and exceeding the budget returns an error rather than an unstable tree.

Each rule declares triggers, eliminated forms, touched forms, produced forms, packages, level, and fidelity. The planner filters rules by profile and packages, checks produced forms against the knowledge base, and orders dependencies from producers to rules that eliminate or touch those forms. It rejects dependency cycles and resolves ties by registry order. Context-sensitive conditions, such as an operator whitelist or neighboring atom class, are checked by the rule body.

Algorithm 1 Transform scheduling. L, R, F, and G denote LowerAttributes, Rewrite, FinalizeAst, and FlattenGroups.

Require: tree T, enabled phases $\mathcal{P} \subseteq \{L, R, F, G\}$, round budget $K = 8$
$v \leftarrow 0$; $s_p \leftarrow \perp$ for every phase p; $k \leftarrow 0$ ▷ v counts changes to T
while some $p \in \mathcal{P}$ has $s_p \neq v$ **do**
  if $k = K$ **then return** NOTCONVERGED
  $k \leftarrow k + 1$
  for p in (L, R, L, F, G) **do**
    if $p \in \mathcal{P}$ and $s_p \neq v$ **then**
      run p on T; **if** it changed T **then** $v \leftarrow v + 1$
      $s_p \leftarrow v$ ▷ p has processed version v
    end if
  end for
end while
check that no eliminated form remains in T (read-only)
return T

Table 6: Representative rewrite rules. Level is the first profile that enables the rule and fidelity the weakest relation it declares to preserve; each output is produced by that profile, where the rule is the only one to fire.

Rule	Level	Fidelity	Input	Output
over-to-frac	Authoring	Render	`a \over b`	`\frac { a } { b }`
le-to-leq	Authoring	Render	`x \le y`	`x \leq y`
operatorname-to-named-function	Authoring	Render	`\operatorname{sin} x`	`\sin x`
prime-to-prime-node	Authoring	Reading	`f^\prime(x)`	`f' ( x )`
rightarrow-to-to	Faithful	Render	`x \rightarrow 0`	`x \to 0`
implies-to-Longrightarrow	Faithful	Render	`p \implies q`	`p \; \Longrightarrow \; q`
dfrac-to-frac	Corpus	Reading	`\dfrac{a}{b}`	`\frac { a } { b }`
limits-drop	Corpus	Reading	`\sum\limits_k a_k`	`\sum _ { k } a _ { k }`
big-class-variants-to-big	Corpus	Reading	`\bigl( x \bigr)`	`\big ( x \big )`
big-delimiter-size-drop	Equiv	Reading	`\big( x \big)`	`( x )`
cfrac-to-frac	Equiv	Math	`\cfrac{1}{2}`	`\frac { 1 } { 2 }`
widehat-to-hat	Equiv	Math	`\widehat{AB}`	`\hat { A B }`

Rules are indexed by trigger name. Each rewrite iteration traverses a preorder snapshot; the first matching rule that rewrites a node wins, and newly created nodes enter the next iteration. Under Faithful, **`\derivative[2]{x}{t}`** first becomes **`\dv`** and then expands into a fraction of differentials, converging in three iterations. A no-change iteration ends rewriting; exceeding the iteration limit returns an error. Rules that repeatedly touch existing forms require a decreasing measure in their implementation to ensure local convergence.

B.5 REWRITE-RULE PROPOSALS AND DEVELOPMENT

Rule candidates come from adjudicated equivalences, source-backed desugarings, documented preferred forms, and LaTeX conventions. Of the 151 rules, 85 trigger on an entry whose adjudicated record gives an exact equivalence, a desugaring, or a discouraged status, and the remaining 66 come from documentation and convention. Each proposal specifies its tree edit, scheduling metadata, and examples covering the applicable branches.

For example, the **`\over`** rule rewrites the infix pattern **`#1 \over #2`** as **`\frac{#1}{#2}`**. Its metadata declares the trigger and eliminated form **`\over`**, the produced form **`\frac`**, the base package, Authoring level, and rendering fidelity. The knowledge base supplies the argument structure needed to recover the two operands. Table 6 lists further rules from each level, from alias folding in Authoring to presentation hints that only Corpus and Equiv drop.

Before implementation, checks validate the referenced forms, argument branches, fidelity level, elimination ownership, and acyclic dependencies. Input–output examples are rendered in display and inline modes and reviewed against the profile policy. Approved rules are implemented as Rust

Table 7: TeXForm pipeline ablation on the common normalizer audit supports.

Configuration	Safety (%)	Pair canonicity (%)	Idempotence (%)	Coverage (%)
TeXForm (w/o Transform)	99.72	12.54	100.00	99.17
TeXForm (w/o Rewrite)	99.68	16.47	100.00	99.17
TeXForm (Authoring)	98.85	28.54	100.00	99.16
TeXForm (Faithful)	98.84	37.50	100.00	99.16
TeXForm (Corpus)	97.20	46.58	100.00	99.16

functions and tested on their examples; changes to rules or shared runtime components also run the corpus-level contract checks below.

B.6 Contract and Fidelity Checks

The checks above act on one rule at a time. Three further checks act on the assembled system, and each answers a different question: whether the pipeline still closes its own eliminated-form contract, whether the emitted specifications describe inputs that renderers actually accept, and whether transformation preserves rendering at corpus scale.

The transform engine ends every run with a contract check that returns an error if any eliminated form remains, so the transform-fidelity run below also tests the contract: every formula that parsed completed the Faithful transform without an error. Its corpora overlap the audit corpora through UniMER and MathWriting. Safety and canonicity use the supports specified in Section A.

The argument-specification validation generated $1{,}307$ cases from 620 explicit command and environment specifications. Here, full, partial, and none indicate whether all, some, or no generated positive cases rendered successfully under a named renderer configuration. MathJax classified 619 specifications as full and one as none; KaTeX classified 316 as full, 7 as partial, and 297 as none; XeTeX classified 479 as full, 17 as partial, and 124 as none.

The full transform-fidelity run applied Faithful to $1{,}296{,}525$ formulas from three public corpora (UniMER (Gu et al., 2026), the latex-formulas-80M benchmark partitions (OleehyO, 2025), and MathWriting-2024 raw labels (Gervais et al., 2025)), of which $1{,}295{,}217$ transformed successfully. The remaining $1{,}308$ formulas failed before transformation because parsing produced an incomplete document (475) or no document (833). Among $1{,}276{,}638$ input–output pairs with a final MathJax pixel verdict, $1{,}266{,}345$ matched and $10{,}293$ were rejected, for an observed fidelity of 99.19%; the remaining $18{,}579$ successful transforms lacked a comparable rendering verdict.

B.7 Pipeline Ablation

A component ablation augments the main normalizer ledger with two diagnostic configurations that are not formal participants in the common normalizer comparison. TeXForm (w/o Transform) invokes only the Parser and canonical Serializer, while TeXForm (w/o Rewrite) uses the Authoring configuration with Rewrite disabled and retains LowerAttributes, FinalizeAst, and guarded FlattenGroups.

The w/o Transform configuration establishes measurable exact collapse before Transform runs. Adding the non-Rewrite phases increases pair collapse with a modest safety decrease. Compared with the step from w/o Transform to w/o Rewrite, the step to the full Authoring configuration yields a larger increase in pair collapse and a larger safety decrease. These comparisons evaluate cumulative configurations of the TeXForm pipeline on one fixed support; they do not estimate the marginal effect of any individual non-Rewrite phase or rewrite rule.

B.8 Residual Property Failures

Table 8 breaks down the residual outcomes of all four profiles on the common audit supports. Broader profiles collapse more equivalent pairs and produce more render drift, with the largest increase from Corpus to Equiv. Normalization errors affect coverage; empty or unrenderable outputs

Table 8: Observable residual statuses for the four TeXForm profiles. The final column reports string drift followed by second-pass normalization error.

		First-pass output			Second pass
Profile	Normalize error	Empty	Unrenderable	Render drift	Drift / error
Authoring	289	723	6	3,236	0/0
Faithful	277	723	6	3,264	0/0
Corpus	277	723	6	8,933	0/0
Equiv	277	724	6	21,933	0/0

affect coverage and safety; second-pass errors and string drift affect idempotence. No profile shows string drift or a second-pass error on the audit supports. Section A.3 gives the denominators.

Residual non-collapse includes distinctions retained by the profile policy, missing rewrites, and normalization failures. Section F examines how the aggregate patterns change across renderers.

To judge how severe the residual failures are, we examine every residual failure of the Faithful profile on the safety support. All 723 empty outputs come from im2latex records that consist entirely of a LaTeX comment; such an input renders blank, so no visible content is lost, although the protocol counts the empty output as a failure. All six unrenderable outputs arise when LowerAttributes extends a declarative style such as `\rm` past an alignment separator (`&` or `\cr`), which MathJax then rejects as misplaced. To classify the 3,264 render drifts, we convert each input and its output to MathML with MathJax and compare the two trees after removing spacing elements, grouping rows, and delimiter sizes; the comparison keeps each glyph, its font variant, and its place in scripts, fractions, radicals, and table cells (Table 9). Nearly all drifts leave the MathML unchanged or only move a script within a run such as `0.9` or `pK` that MathJax reads as one token, so the visible change is confined to spacing, sizes, and positions. Content or structure changes in 72 drifts, 0.02% of the Faithful safety support: those that change a font variant or a character, and three in which `\skew` or a picture-mode `\framebox` takes a different argument. The largest group restores content that MathJax hides after a comment character followed by a carriage return, which TeX treats as a line end; every other change traces to an identifiable parser or rule defect.

Normalization errors are returned explicitly; empty or unrenderable outputs can be detected by an additional output check. A pipeline that stores labels can also catch drift by comparing the renderings of each input and its output, as our audit does.

Table 9: Render drifts of TeXForm (Faithful), classified by comparing the MathML of each input and its output after removing spacing elements, grouping rows, and delimiter sizes. The Inputs column counts distinct raw formulas; the last column gives causes with their drift counts.

MathML comparison	Drifts	Share (%)	Inputs	Causes
Identical	3,090	94.7	2,901	Spacing, size, or position only, e.g., `\over` to `\frac` or merged spacing commands
Script attached differently	105	3.2	24	Run read by MathJax as one token (102); `\skew` or picture-mode `\framebox` argument (3)
Font variant changed	23	0.7	23	Unbraced text-command argument read as a word (13); plain `\cases` set as math (3); nested text styles reordered (3); style across alignment cells (2); other (2)
Characters changed	46	1.4	46	Comment ended by a carriage return (36); `\not` drawn as an overlay (7); unbraced text-accent argument read as a word (2); plain `\cases` set as math (1)
Total	3,264	100.0		

Table 10: Complete property pass rates (%) for the audited configurations; Section A.3 defines the tool-specific denominators.

		Canonicity			
Implementation	Safety	Pair	Cluster	Idempotence	Coverage
Identity	100.00	0.00	0.00	100.00	99.40
UniMERNet postprocess	99.83	8.36	0.04	100.00	99.40
im2markup	94.43	18.65	5.90	63.30	99.06
MathNet	59.65	66.42	58.96	98.48	98.32
CDM (in UniMERNet)	86.77	25.14	11.34	90.32	99.34
CDM (in OmniDocBench v1.5)	87.68	25.01	11.23	90.31	99.34
CDM (in OmniDocBench v1.6)	86.22	24.96	11.17	90.64	98.97
TeXForm (Authoring)	98.85	28.54	18.58	100.00	99.16
TeXForm (Faithful)	98.84	37.50	25.07	100.00	99.16
TeXForm (Corpus)	97.20	46.58	35.20	100.00	99.16
TeXForm (Equiv)	93.43	47.62	36.54	100.00	99.16

C Normalizer Ecosystem and Technical Lineage

We compare runnable normalization components and trace their reuse through source snapshots, file comparisons, and observed behavior.

C.1 Survey Scope and Selection Criteria

The inventory, last revised on July 9, 2026, includes upstream components deployed to normalize a LaTeX string into another LaTeX or LaTeX-like string. This interface supports the common four-property audit. We group implementations by role and retain the identity function as a no-rewrite anchor. Parser libraries, intermediate-representation converters, symbolic evaluators, and scorer-only helpers fall outside this deployed-normalizer comparison.

We collapse duplicate cleaners: pix2tex and PaddleOCR LaTeXOCR share a whitespace-cleaning core, so we retain the UniMERNet path, which also applies **`ftfy.fix_text`**. CDM preprocessing components retain their metric-internal role in the comparison.

C.2 Frozen Implementations and Interfaces

All implementations in the common normalizer audit are frozen in the accompanying experiment repository. The repository vendors the audited upstream source and TeXForm release together with the wrappers and callable configurations used to produce the reported results. Its per-tool documentation records upstream versions and commit hashes, while the repository revision identifies artifact-local snapshots.

Table 10 reports every audited configuration under the same definitions; pair and cluster canonicity use the model-prediction source, exact string collapse, unconditional failures, and leave-one-out membership.

Table 10 extends the main comparison with cluster-level canonicity and Equiv. The three CDM configurations represent separately versioned processors from one technical lineage.

C.3 The im2markup Preprocessing Lineage

The im2markup preprocessor is parse-based: it calls the private KaTeX parser, obtains an abstract syntax tree, and serializes that tree with a builder copied and adapted from KaTeX's MathML construction code. Its vendored parser identifies itself as KaTeX 0.6.0, but a file-level comparison against the pristine package shows local changes to parser validation, symbol typing, supported environments, and token accumulation. These changes disable checks for repeated scripts and other malformed structures, assign unknown control sequences the node type used for **`\sigma`**, and convert any thrown exception into an empty output string. The resulting failure mode is therefore silent:

Table 11: Behavioral signatures observed at multiple points in the im2markup processing lineage.

Signature	Published-label observation	Runnable-processor observation
Line break becomes thin space	2,045 affected im2latex published labels	567 im2markup drifts and 564, 564, and 565 drifts in the three CDM variants.
Percent handling truncates content	810 affected im2latex published labels	Among 127 inputs containing an escaped percent, im2markup drifts on 107 and returns an empty string on 10; each CDM variant drifts on 10.
Optional root index becomes `[object Object]`	21 published im2latex labels contain the literal artifact	The artifact appears in 1,390 im2markup outputs and 1,389 MathNet outputs.

the processor can discard a script, continue with an incorrectly typed tree, or return an empty string without a distinct refusal signal.

Source snapshots establish later code reuse more strongly than behavioral similarity does. MathNet, UniMERNet CDM, and OmniDocBench v1.5 carry a **`katex.js`** file copied from im2markup, while the UniMERNet and OmniDocBench v1.5 tokenizers also share the same preprocessing files. OmniDocBench v1.6 instead ports the tokenizer behavior to Python and uses pylatexenc, so it belongs to the same processing lineage without being a copy of the JavaScript implementation. Uni-MuMER (Li et al., 2025a) also adopts im2markup tokenization and extends MathNet normalization; its released pipeline invokes a separate Uni-MuMER-specific cleaner, which is not included in our six-implementation audit.

Of the 94,000 im2latex labels, 92,370 (98.27%) match the im2markup replay after whitespace normalization, including 71 rows for which both the published label and replay output are empty; 1,591 rows mismatch and 39 fail normalization. This high replay agreement supports attribution of the published label transformation to the legacy preprocessor while preserving the observed exceptions. The original processor was a reasonable research prototype for character-level recognition; the lineage concern is its later use as shared infrastructure without an explicit preservation contract.

C.4 CDM Processor Lineage and Metric Adoption

CDM composes two distinguishable components before rendered-character matching. Its **`tokenize_latex.py`** explicitly states that it was taken and modified from im2markup and invokes the inherited parser and serializer path. Its separate **`latex_processor.py`** is a custom processor that prepares token coloring, rendering, bounding-box matching, and scoring through additional regular-expression rewrites. Code lineage therefore applies to the tokenizer component, whereas processor composition describes the complete CDM front-end; treating the entire custom processor as an im2markup fork would conflate these relations.

The surveyed adoption path places this composed front-end inside the UniMERNet CDM metric and the Formula CDM implementations used by OmniDocBench, and it also records a benchmark-bound CDM preprocessing path in Tiny-Doc-Math, the evaluation set released with CDM (Wang et al., 2025). These edges establish metric or pipeline adoption, not direct source copying by every downstream benchmark or model. Version-specific testing remains necessary because UniMERNet CDM and OmniDocBench v1.5 share their tokenizer but use different custom processors, while OmniDocBench v1.6 replaces the tokenizer implementation.

C.5 Behavioral Correspondence Across Reused Components

The fixed experiments expose distinctive transformations in published labels and in runnable processors, providing a behavioral check on the source-level lineage. Table 11 reports signatures whose forms and mechanisms can be inspected directly; the counts use their native populations and are not estimates of a common prevalence.

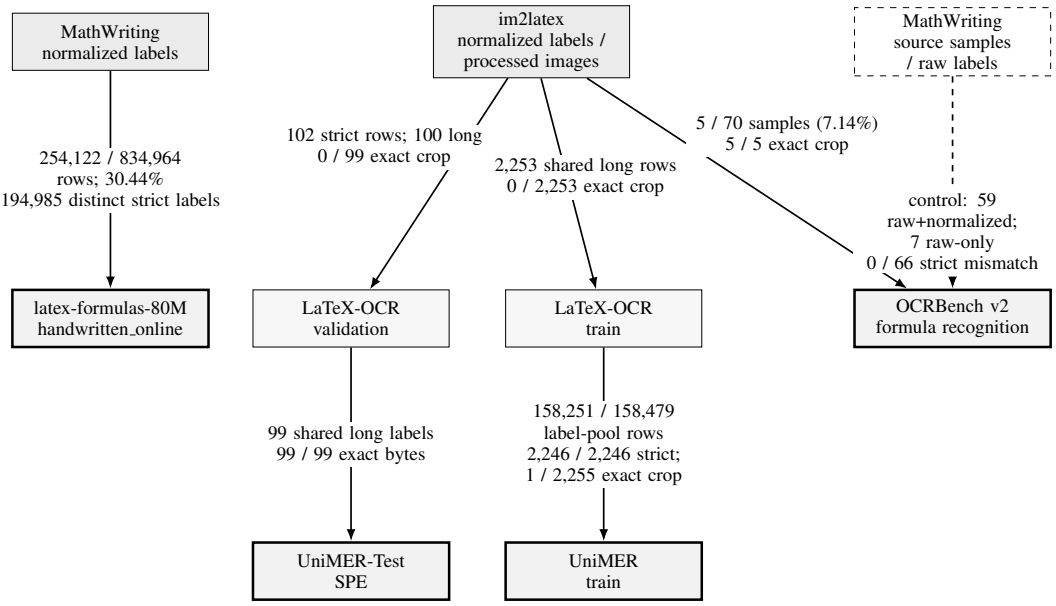

Figure 4: Observed propagation of render-different normalized labels into downstream datasets and benchmarks. Solid edges are confirmed propagation chains and the dashed edge is a source-related negative control; edge denominators are local and non-interchangeable.

C.6 Observed Downstream Propagation

Figure 4 records the propagation chains that the lineage audit confirms, together with one negative control. Here, "strict" denotes a paired label whose raw and normalized forms differ under the rendering-faithfulness criterion; it does not by itself assert mathematical inequivalence. Each edge uses its own row, label, image-pair, or source-conditioned denominator, so percentages must not be compared as if they estimated one population rate.

MathWriting normalized labels occur in 254,122 of 834,964 target rows in the **`latex-formulas-80M`** handwritten-online partition (OleehyO, 2025), covering all 194,985 distinct strict source labels. The im2latex chain provides a stronger direction check than exact labels alone: the LaTeX-OCR validation images and UniMER-Test SPE images match byte-for-byte for all 99 shared long labels, whereas none of the 99 source-to-LaTeX-OCR cropped-image pairs matches exactly. The corresponding train chain covers 2,246 of 2,246 shared long strict labels, but only one of 2,255 LaTeX-OCR-to-UniMER crops and none of the 2,253 im2latex-to-LaTeX-OCR crops is pixel-identical. These two chains therefore support label propagation through different image packaging rather than duplicate-image leakage at every edge.

OCRBench v2 contains five of 70 source-conditioned im2latex samples, and all five cropped images match exactly; by contrast, the MathWriting control retains a known source relationship but contains zero strict mismatches among 66 matched samples. The MathWriting control contains source overlap without strict mismatches, complementing the positive propagation chains.

The five OCRBench v2 images are the im2latex crops, which were rendered from the raw sources, so the propagated labels serve as references that no longer match their images. In sample en-4758, the image shows G'^{*} while the reference reads G^{*}: eight of the twelve models transcribe the prime and are rejected under the official substring rule, whereas InternVL3-8B omits it and receives full score. In en-5034, the reference likewise drops the prime of α'^{2}; all twelve models transcribe it, and all are rejected.

Table 12: Raw-label rendering fidelity under the fixed MathJax protocol. Rates are conditional on a valid verdict for the indicated pair.

Dataset	Paired	Labels	Equiv./comp.	Rate (%)	Render unavail.	Norm. unavail.
MathWriting-2024	649,196	Published	400,091/645,119	62.02	4,077	–
		Faithful	643,178/645,118	99.70	4,077	1
im2latex-100k	94,000	Published	84,325/90,877	92.79	3,123	–
		Faithful	89,335/92,091	97.01	1,633	276

Table 13: Joint rendering verdicts on records for which both raw comparisons are available.

		Preserves the raw rendering			
Dataset	Comparable	Both	Faithful only	Published only	Neither
MathWriting-2024	645,117	399,681 (61.95%)	243,496 (37.74%)	410 (0.06%)	1,530 (0.24%)
im2latex-100k	90,809	82,479 (90.83%)	5,616 (6.18%)	1,845 (2.03%)	869 (0.96%)

D Dataset-Side Evidence

Paired raw and published labels measure rendering changes introduced by dataset processing. A complementary corpus-wide audit measures standalone renderability.

D.1 Paired-Dataset Scope and Audit Protocol

Through July 20, 2026, we surveyed paper-backed or ecologically significant mathematical-expression-recognition and formula-OCR dataset lineages and found two independent lineages whose publishers provide sample-aligned labels before and after processing: MathWriting-2024 and im2latex-100k. Inclusion required two representations of the same target, a publisher-provided row mapping or stable sample identifier, the same underlying data population, and public auditability; lineages, rather than repackaged copies, were the unit of selection. The TexTeller **`latex-formulas`** corpus was excluded because its merged and rewritten releases do not provide a stable publisher-supplied row mapping between the candidate versions.

For each aligned record, we compare the raw label with the published label and, separately, with the output of TeXForm's Faithful profile on the same raw string. Both comparisons use the fixed MathJax–resvg rendering protocol and the same pixel comparator; a comparison is recorded as equivalent, non-equivalent, or a rendering error. Each fidelity rate therefore uses the records for which that particular pair has a valid rendering verdict, while the joint distribution uses only records for which both pairs have verdicts.

D.2 Paired-Label Fidelity and Joint Distributions

Table 12 reports the complete paired audit. The published MathWriting labels preserve substantially fewer raw renderings than the Faithful transformation, while the im2latex gap is smaller but remains visible under the same protocol. The nonzero rendering and normalization-unavailable columns require pair-specific comparable denominators because the two comparisons can fail on different records.

The joint distribution in Table 13 distinguishes rendering-fidelity recovery from cases in which the published transformation preserves a rendering that Faithful does not. For MathWriting, the Faithful-only cell dominates the published-only cell; im2latex shows the same direction with a less one-sided distribution.

D.3 Rendering Changes and Representative Cases

We randomly sampled 100 raw-to-published pairs with a non-equivalent rendering verdict from each lineage and asked two annotators to classify the fixed renderings independently. The main axis distinguishes renderings that are indistinguishable under normal side-by-side inspection, visi-

	Damage type			Declared scope	
	Indistinguishable	Spacing Only	Symbol Altering	Within declared	Beyond declared
Raw	$\frac{(\frac{9}{6}+309)}{\frac{10}{10}}$	$\int_{-\infty}^{\infty} \psi_n(t)\,dt = 0.$	$\varnothing \in D;$	$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$	$\sum_{i=1}^{m} k_i = n \quad \text{and} \quad \sum_{i=1}^{m} p_i = 1.$
Published	$\frac{(\frac{9}{6}+309)}{\frac{10}{10}}$	$\int_{-\infty}^{\infty} \psi_n(t)dt = 0.$	$\emptyset \in D;$	$F = q(E + v \times B)$	$\sum_{i=1}^{m} k_i = n and \sum_{i=1}^{m} p_i = 1.$

Figure 5: MathWriting examples from the fixed human-review stimuli. The columns show a near-identical rendering, spacing removal, a glyph change, a policy-permitted font change, and beyond-scope word-boundary loss, respectively. Each raw/published pair shares a scale and received unanimous assignments on the relevant axis.

	Indistinguishable	Spacing Only	Symbol Altering
Raw	$\Pi(s) = \frac{\pi^2 R^4}{2\kappa_6^2}[s \ln(-R^2 s)],$	$\mathbf{P}(V) \cong \bigwedge(\mathbf{L}(V))$	$R_{21} = \begin{pmatrix} q & & \\ 1 & q - q^{-1} & \\ 0 & 1 & \\ & & -q^{-1} \end{pmatrix}$
Published	$\Pi(s) = \frac{\pi^2 R^4}{2\kappa_6^2}[s \ln(-R^2 s)],$	$\mathbf{P}(V) \cong \quad \bigwedge \quad (\mathbf{L}(V))$	$R_{21} = (\ q \qquad)$

Figure 6: im2latex-100k examples from the fixed human-review stimuli. The first pair is a visibility-threshold disagreement (A: indistinguishable; B: Symbol Altering); the others are consensus Spacing Only (expanded space around a wedge) and Symbol Altering (matrix truncation) cases. Each raw/published pair shares a scale.

Table 14: Human review of 100 divergent pairs per dataset, as the percentage of pairs in each category with Wilson 95% intervals. MathWriting scope uses the same 100 pairs as its rendering categories. Neither annotator marked any pair Uncertain (0.0%, interval [0.0, 3.7]).

	Annotator A		Annotator B	
Category	%	95% CI	%	95% CI
MathWriting: rendering differences				
Indistinguishable	5.0	[2.2, 11.2]	2.0	[0.6, 7.0]
Spacing Only	20.0	[13.3, 28.9]	20.0	[13.3, 28.9]
Symbol Altering	75.0	[65.7, 82.5]	78.0	[68.9, 85.0]
MathWriting: declared scope				
Within declared	89.0	[81.4, 93.7]	90.0	[82.6, 94.5]
Beyond declared	11.0	[6.3, 18.6]	10.0	[5.5, 17.4]
im2latex-100k: rendering differences				
Indistinguishable	13.0	[7.8, 21.0]	2.0	[0.6, 7.0]
Spacing Only	43.0	[33.7, 52.8]	47.0	[37.5, 56.7]
Symbol Altering	44.0	[34.7, 53.8]	51.0	[41.3, 60.6]

ble changes confined to inter-symbol spacing, and changes to a visible symbol or its presentation; font, weight, italic or upright form, glyph shape, size, decorators, and material changes to symbol position or layout belong to the last category. For MathWriting, a separate axis records whether the change is consistent with the publisher's declared abstraction over handwriting-specific print variation, visual style, size, and non-unique representation. We report the two annotations separately, without adjudication or a pooled category proportion. Across all 200 pairs, the annotators agreed on 89.0% of the damage types (Cohen's $\kappa = 0.784$).

For MathWriting, damage-type agreement was 93.0% ($\kappa = 0.813$), and Symbol Altering dominated both annotations. Table 14 reports the two assignments separately.

Declared-scope agreement on MathWriting was 99.0% ($\kappa = 0.947$), with most pairs assigned within the publisher's target abstraction (Table 14).

Because the reported results remain unadjudicated, we retain the annotator-assigned labels even where their application is not mechanically identical to the written rubric: five MathWriting pairs involving \scriptstyle or \tfrac-to-\frac size changes were marked Spacing Only by both annotators, although the rubric places size changes under Symbol Altering.

The two axes capture different properties: a visible font or style change can alter the rendering while remaining consistent with the published target policy.

The ten MathWriting pairs that both annotators placed beyond the declared scope followed two recurring patterns. Seven lost the operator status of \lim, which changed its font and spacing and moved the condition from beneath the operator to a side subscript in display rendering. The remaining three lost word boundaries in embedded text, producing joined strings such as **nand**, **PRvotesratio**, and **0fork**.

For im2latex-100k, damage-type agreement was 85.0% ($\kappa = 0.737$), with comparable Spacing Only and Symbol Altering proportions (Table 14).

The two annotations place 87–98% of the im2latex mismatches in a visibly distinguishable category. Eleven of the 15 disagreements contrast an indistinguishable rating with a visible category, whereas only four cross between the two visible categories; the main disagreement therefore concerns the visibility threshold.

Case inspection also reveals clear processing failures within the heterogeneous Symbol Altering category. In im2latex sample **4de5b14cbf8d42d0366f75dc**, the published rendering retains only the first row of a four-row matrix. In sample **5d39f2caeb4a5467604cb6c0**, it drops the prime from φ'^2. The latter belongs to a broader source-level pattern: among 445 im2latex raw labels containing a prime immediately followed by a superscript marker, the published preprocessing loses a prime in 444, and 431 of those pairs receive a non-equivalent rendering verdict. The corresponding parser path expands the prime into a superscript and then silently overwrites it when it encounters the following superscript.

Across both sampled lineages, the two annotators place the majority of the reviewed divergences in a visibly distinguishable category, which is evidence against reading these divergences as visually negligible.

D.4 Dataset-Wide Renderability

Paired labels expose changes that remain renderable, but they cover only datasets that publish both sides of the transformation. We therefore apply a weaker, fully automatic audit to every available label instance in five datasets: MathJax is attempted first, and a nonempty label that fails is retried with XeTeX. A label is accepted if either renderer succeeds; failure under both renderers is recorded as a standalone-unrenderable candidate rather than a general label error. For im2latex, the raw, normalized, and tokenized fields are separate label instances, and for MathWriting the raw and normalized fields are likewise counted separately.

The failures do not share a single cause. At least 4,872 XeTeX failures report an undefined control sequence, while the remaining cases include empty labels, missing arguments or delimiters,

Table 15: Dataset-wide standalone renderability with MathJax followed by XeTeX fallback.

		Rendered				
Dataset	Label instances	MathJax	XeTeX rescue	Total	Failed	Success (%)
im2latex-100k	282,000	273,915	4,150	278,065	3,935	98.60
LaTeX-OCR	165,244	163,747	1,495	165,242	2	100.00
MathWriting-2024	1,304,815	1,300,435	139	1,300,574	4,241	99.67
OmniDocBench	2,066	2,010	50	2,060	6	99.71
UniMER	1,009,879	994,849	13,662	1,008,511	1,368	99.86
Overall	2,764,004	2,734,956	19,496	2,754,452	9,552	99.65

legacy matrix environments, and text–math-mode conflicts. MathWriting further illustrates why label fields must remain separate: 4,240 failures occur in the raw **`label`** field, compared with 1 in **`normalizedLabel`**.

Standalone renderability tests whether a label produces an image; paired-label fidelity tests whether processing preserves that image. Section C.6 traces the propagation of changed labels into downstream corpora and benchmark references.

E Evaluation-Side Evidence

We audit how normalization and scoring rules affect acceptance and model rankings on frozen predictions. The independent rendering criterion supplies the pairwise verdicts used to assess these decisions.

E.1 Audited Tracks and Model Cohorts

The audit covers two text-based formula-recognition tracks and one document-level track with a render-aware formula metric. OCRBench v2 uses substring acceptance, UniMER-Test reports corpus BLEU and admits an edit-distance-zero full-credit boundary, and OmniDocBench v1.6 embeds its LaTeX processor inside Formula CDM. Table 17 records the retained and reference-renderable supports. Table 16 lists the two model cohorts: 12 models on the text tracks and 11 on the OmniDocBench page track, nine of which appear in both. They comprise formula recognizers (Gu et al., 2026; Liu et al., 2025; Blecher, 2022), document parsers (Cui et al., 2025; 2026; Zhang et al., 2026; Niu et al., 2026; Duan et al., 2026; Hunyuan Vision Team et al., 2025; Li et al., 2025b; Wu et al., 2026), and general-purpose VLMs (Bai et al., 2025b;a; Zhu et al., 2025). Closed-source recognition systems are excluded because their normalization runs inside a proprietary pipeline that can be neither invoked separately nor replaced, so it admits neither of our audits. Every model runs on a single A100 40GB GPU with its official weights and inference engine. Specialist models use their official formula-recognition or page-parsing entry points; the three general-purpose VLMs receive a LaTeX-only instruction on the formula tracks and the prompt pinned by OmniDocBench on the page track. On OmniDocBench, each model parses all 1,651 pages, the pinned v1.6 evaluator matches predicted to ground-truth display formulas, and Formula CDM scores the 2,352 ground-truth formulas per model, with unmatched ground truth scored as zero.

UniMER-Test contains SPE, CPE, SCE, and HWE subsets with respectively 6,762, 5,921, 4,742, and 6,332 retained samples. The reference-renderable supports contain 6,692, 5,632, 4,679, and 6,297 samples in the same order. Pooled BLEU is recomputed on their union rather than averaged across subset scores.

All model predictions are fixed before the front-end and decision-rule comparisons. Prediction, extraction, normalization, and rendering failures remain associated with their original sample and score zero where the fixed-support rule requires a verdict. For the UniMERNet-base cohort member, the stored prediction is the official decoded **`pred_str`**; every record satisfies **`raw_output == output`**, ruling out an additional benchmark-normalization pass before this audit. Table 23 reports per-model results on the text tracks, and Table 25 summarizes leader and rank changes on all three tracks.

E.2 Official Replay and External Anchors

The official score paths are checked on their native supports before the fixed-support comparisons below. OCRBench replay applies the official substring rule to all 508 retained formula rows, whereas the common-support leaderboard later restricts every model and front-end to the same 466 reference-renderable rows. For UniMER-Test, the official UniMERNet-base checkpoint provides a stricter external anchor because its decoded predictions can be replayed on every row of each published subset.

The four strict-checkpoint differences are at most 0.0144 in absolute BLEU, establishing a bounded reproduction discrepancy rather than exact equality with the published BLEU scores. The replay also includes pix2tex as a non-strict checkpoint sanity check, but does not treat it as a reproduction

Table 16: Model cohorts. Checkpoints are Hugging Face repositories at the pinned revision unless a package release is named; parameter counts follow the model cards or papers. Tracks gives the UniMER-Test and OCRBench v2 formula tracks (Formula), the OmniDocBench v1.6 page track (Page), or both.

Model	Params	Checkpoint	Tracks
Formula recognizers			
UniMERNet-base	325M	`wanderkid/unimernet_base@af898d4`	Formula
PP-FormulaNet-L	179M	PaddleOCR 3.7.0 inference model	Formula
pix2tex	25M	pix2tex 0.1.4, release v0.0.1 weights	Formula
Document parsers			
PaddleOCR-VL	0.9B	`PaddlePaddle/PaddleOCR-VL@f54aa90`	Both
PaddleOCR-VL-1.5	0.9B	`PaddlePaddle/PaddleOCR-VL-1.5@426bf5b`	Both
PaddleOCR-VL-1.6	0.9B	`PaddlePaddle/PaddleOCR-VL-1.6@66317ac`	Both
MinerU2.5	1.2B	`opendatalab/MinerU2.5-2509-1.2B@1aa090b`	Both
GLM-OCR	0.9B	`zai-org/GLM-OCR@ca5d8b3`	Both
HunyuanOCR v1.0	1B	`tencent/HunyuanOCR@de8f10a` (`v1.0/`)	Both
dots.ocr	3B	`rednote-hilab/dots.ocr@c0111ce`	Page
FireRed-OCR	2B	`FireRedTeam/FireRed-OCR@7bf5e9d`	Page
General-purpose VLMs			
Qwen2.5-VL-7B-Instruct	7B	`Qwen/Qwen2.5-VL-7B-Instruct@cc59489`	Both
Qwen3-VL-8B-Instruct	8B	`Qwen/Qwen3-VL-8B-Instruct@0c351dd`	Both
InternVL3-8B	8B	`OpenGVLab/InternVL3-8B@853e3a7`	Both

Table 17: Audited benchmark tracks and fixed cohorts. "Reference-renderable" denotes a model-independent ground-truth support for the text-based tracks; OmniDocBench instead retains model-specific rows drawn from 1,651 pages.

Track	Retained inputs	Reference-renderable	Models	Official scoring scope
OCRBench v2 formula	508	466	12	Substring accuracy
UniMER-Test, pooled	23,757	23,300	12	Corpus BLEU; edit-distance-zero full credit
OmniDocBench v1.6	25,872	–	11	Formula CDM and derived Overall

Table 18: Full-support UniMERNet-base BLEU replay against the published UniMER reference values. The checkpoint match is strict; Δ is replayed minus reported.

Subset	Rows	Replayed	Reported	Δ
SPE	6,762	0.9006	0.9150	-0.0144
CPE	5,921	0.9132	0.9250	-0.0118
SCE	4,742	0.6267	0.6260	$+0.0007$
HWE	6,332	0.8901	0.8950	-0.0049

anchor. OmniDocBench uses a pinned replay of its official v1.6 processor and backend and requires an exact parity check before the formal run rather than matching an external paper-value anchor.

E.3 Normalization Front-Ends and Decision Rules

OCRBench uses substring acceptance officially and normalized equality diagnostically. Its normalization-only intervention retains raw model outputs; the formula-extracted configurations apply extraction before TeXForm. UniMER's normalized equality is the official edit-distance-zero boundary, while corpus BLEU supplies graded scores. OmniDocBench retains the Formula CDM backend and replaces its LaTeX processor, and its binary audit uses rounded Formula CDM $F_1 = 1$. Each track compares the benchmark processor with the three TeXForm profiles under one decision rule; on the text tracks, a whitespace-cleanup front-end serves as a baseline that folds no spelling variation.

False-reject rates use the rendering-equivalent support, with normalization failures recorded separately. False-accept-candidate rates use accepted pairs with a rendering verdict. OmniDocBench's official false-reject denominator excludes seven backend failures; Faithful returns scores for all 7,000 rendering-equivalent rows. Fixed-support model scores retain failures as zero. Section A.3 specifies the paired bootstrap.

E.4 Misverdicts and Transition Matrices

Table 19 reports false rejects and false-accept candidates for every front-end. Whitespace cleanup alone rejects 72.30% and 77.52% of the equivalent pairs on the two text tracks, so the benchmark processors already remove most of the rejections that spelling variation causes. Among the TeXForm profiles, Corpus rejects the fewest equivalent pairs on all three tracks but admits more false-accept candidates than Faithful on each, which is the trade-off between canonicity and safety of Section 4 at the evaluation end. On OCRBench, replacing normalization on raw outputs yields 96 explicit rejects and 366 unavailable decisions among 1,307 rendering-equivalent pairs. Formula extraction before Faithful yields 165 rejects and no unavailable decisions on these pairs; substring acceptance then gives 21 candidates among 1,163 accepted pairs.

Figure 7 shows one pair for each outcome. Faithful still rejects 19,987 equivalent UniMER-Test pairs and Corpus 11,715, so Corpus rejects 8,272 fewer. Figure 7b shows one distinction that Faithful keeps by policy: an explicit `\limits` changes the rendering in inline mode even when the display-mode renderings match. Some spellings no profile merges, because they render differently in other contexts, such as stretchy and fixed-size delimiters (Figure 7c). On OmniDocBench, a false-accept candidate persists under Faithful when the Formula CDM backend rather than the front-end accepts the pair: in Figure 7d, both front-ends keep the misplaced -1 in the normalized prediction, and Formula CDM still returns $F_1 = 1$.

Table 20 compares recovered and lost true accepts and introduced and removed candidates on jointly available decisions. On UniMER, candidate transitions remove 43 candidates on net on this support. Another 14 previous candidates become unavailable, so the candidate totals under the two front-ends fall from 220 to 163; the accepted denominator grows from 90,380 to 102,594, reducing the conditional rate from 0.24% to 0.16%.

Table 21 includes unavailable outcomes on fixed OCRBench and UniMER relation supports. For raw-output OCRBench, 280 previous true accepts become unavailable, and true accepts fall from 1,046 to 845. On UniMER, six previous true accepts and four previous false rejects become unavailable.

E.5 Per-Model and Per-Subset Results

UniMER's pooled false-reject rate hides substantial subset heterogeneity. Across the four writing domains, the benchmark false-reject rate varies more than ninefold, whereas the fraction recovered by Faithful varies more than twofold. Table 22 reports both quantities. This variation makes the pooled value a cohort summary rather than a property of a typical subset.

The same heterogeneity appears across models. Table 23 reports the net true-accept gain as a fraction of each model's criterion-accepted rows, holding the exact decision rule fixed. All model-level

Table 19: Front-end misverdicts for raw-output substring acceptance and formula-extracted normalized equality on OCRBench, edit-distance-zero acceptance on UniMER, and Formula CDM full credit on OmniDocBench; Authoring, Faithful, and Corpus are TeXForm profiles. Rates are percentages; for raw-output OCRBench, Faithful non-acceptance splits into explicit rejection (R) and unavailable outcomes (U). Dashes mark scopes without a paired 95% confidence interval.

	False reject			False-accept candidate		
Front-end	Rate	Count	95% CI	Rate	Count	95% CI
OCRBench v2, raw output / substring acceptance						
Benchmark	19.97	261/1,307	–	3.51	38/1,084	–
Faithful, R	7.35	96/1,307	–	1.74	15/860	–
Faithful, U	28.00	366/1,307	–			
OCRBench v2, extracted formula / normalized equality						
Whitespace cleanup	72.30	945/1,307	[69.59, 74.98]	0.00	0/362	[0.00, 0.00]
Benchmark	19.97	261/1,307	[15.36, 25.33]	2.15	23/1,069	[0.82, 3.85]
Authoring	12.70	166/1,307	[8.75, 17.14]	0.35	4/1,145	[0.00, 1.09]
Faithful	12.62	165/1,307	[8.62, 16.96]	0.44	5/1,147	[0.00, 1.20]
Corpus	9.72	127/1,307	[6.42, 13.21]	1.75	21/1,201	[0.16, 3.85]
UniMER-Test pooled, edit distance zero						
Whitespace cleanup	77.52	94,907/122,428	[77.26, 77.78]	0.00	0/27,521	[0.00, 0.00]
Benchmark	26.36	32,268/122,428	[25.83, 26.93]	0.24	220/90,380	[0.19, 0.30]
Authoring	17.32	21,209/122,428	[16.84, 17.82]	0.16	162/101,371	[0.12, 0.21]
Faithful	16.33	19,987/122,428	[15.85, 16.81]	0.16	163/102,594	[0.12, 0.21]
Corpus	9.57	11,715/122,428	[9.22, 9.91]	1.21	1,356/112,059	[1.06, 1.35]
OmniDocBench v1.6, Formula CDM full credit						
Benchmark	18.98	1,327/6,993	–	41.99	4,102/9,768	–
Authoring	5.41	379/7,000	[3.93, 7.31]	37.13	3,910/10,531	[31.09, 43.63]
Faithful	3.53	247/7,000	[2.39, 4.97]	36.91	3,951/10,704	[31.04, 43.36]
Corpus	3.10	217/7,000	[2.08, 4.44]	38.91	4,321/11,104	[32.74, 45.47]

Table 20: Paired transitions among jointly available decisions. Net percentages use rendering-equivalent supports for true accepts and rendering-inequivalent supports for candidates. Table 21 includes unavailable outcomes.

	True accepts		False-accept candidates	
Scope	Recovered/lost	Net gain	Introduced/removed	Net cost
OCRBench raw / substring	87/8	+79 (6.04%)	3/16	−13 (−0.34%)
OCRBench extracted / equality	116/20	+96 (7.35%)	1/19	−18 (−0.47%)
UniMER-Test pooled	12,278/1	+12,277 (10.03%)	93/136	−43 (−0.03%)
OmniDocBench Formula CDM	1,192/112	**+1,080 (15.43%)**	862/1,015	−153 (−1.44%)

Table 21: Official-rule transitions on fixed supports of 5,099 OCRBench and 265,736 UniMER pairs with rendering verdicts. Rows give the benchmark decision; columns give Faithful decisions. A: accept; R: reject; U: unavailable. No benchmark decision is unavailable on these supports. The 493 OCRBench and 13,864 UniMER pairs with renderable references but no pairwise rendering verdict are excluded.

	Rendering-equivalent			Rendering-inequivalent		
Benchmark / decision	A	R	U	A	R	U
OCRBench / A	758	8	280	12	16	10
OCRBench / R	87	88	86	3	2,593	1,158
UniMER / A	90,153	1	6	70	136	14
UniMER / R	12,278	19,986	4	93	142,142	853

Case	LaTeX source	Rendering	Off.	Faith.	Corp.
(a) UniMER-Test SCE Qwen3-VL-8B-Instruct	*ref* `f ^ { \prime } ( x ) = 4 x ^ { 3 } - 6 x ^ { 2 } , f ^ { \prime } ( 2 ) = 8` *pred* `f'(x) = 4x^3 - 6x^2, f'(2) = 8`	$f'(x) = 4x^3 - 6x^2, f'(2) = 8$	×	✓	✓
(b) UniMER-Test HWE InternVL3-8B	*ref* `\sum \limits _ { i = 1 } ^ { n } a _ { i }` *pred* `\sum_{i=1}^{n}a_{i}`	$\sum_{i=1}^{n} a_i$	×	×	✓
(c) UniMER-Test SPE PaddleOCR-VL	*ref* `\langle \mathbf { w } , \mathbf { v } \rangle` *pred* `\left\langle \mathbf{w},\mathbf{v} \right\rangle`	$\langle \mathbf{w}, \mathbf{v} \rangle$	×	×	×
(d) OmniDocBench InternVL3-8B	*ref* `C_{k}^{n}=C_{k}^{n-1}+ C_{k-1}^{n-1},` *pred* `\[ C_k^n = C_{k-1}^n + C_{k-1}^{n-1}, \]`	$C_k^n = C_k^{n-1} + C_{k-1}^{n-1},$ $C_k^n = C_{k-1}^n + C_{k-1}^{n-1},$	✓	✓	–

Figure 7: Evaluation-side examples under the rendering criterion, with the decision of the official (Off.), Faithful (Faith.), and Corpus (Corp.) front-ends (✓ accept, × reject). In (a)–(c), the reference and the prediction render to identical pixels, so one rendering is shown: Faithful recovers (a); Faithful keeps the **`\limits`** of (b), which changes the rendering in inline mode, whereas Corpus drops it; and no profile merges the stretchy and fixed-size delimiters of (c), which differ around taller content. In (d), whose reference rendering is shown above the prediction, Formula CDM gives full credit under both front-ends although the prediction moves the -1 of the second term from the superscript to the subscript.

Table 22: UniMER false rejects and recovery by subset.

Subset	Benchmark false reject (%)	Faithful false reject (%)	Recovered (%)
SPE	40.75	27.72	31.97
CPE	54.54	33.53	38.53
SCE	22.10	13.09	40.71
HWE	6.00	1.95	67.46

Table 23: Per-model net true-accept gain under normalized equality; OCRBench uses formula-extracted predictions. Rows follow the official UniMER-Test order of Table 3.

Model	OCRBench v2 (%)	UniMER-Test pooled (%)
UniMERNet-base	4.65	5.65
PaddleOCR-VL-1.6	7.69	13.34
PP-FormulaNet-L	6.03	13.94
PaddleOCR-VL	6.50	10.90
PaddleOCR-VL-1.5	7.20	11.62
MinerU2.5	6.59	4.26
GLM-OCR	3.39	3.58
Qwen2.5-VL-7B-Instruct	8.51	4.27
HunyuanOCR v1.0	0.00	4.95
Qwen3-VL-8B-Instruct	26.85	19.59
InternVL3-8B	5.08	18.30
pix2tex	8.11	6.56

Table 24: Front-end output rate and rendering safety (%). Safety is reported with its safe count over the eligible denominator; the output rate is the share of rows with an input that yield a nonempty output. OCRBench prediction rows use extracted formulas.

		Benchmark front-end			Faithful		
Benchmark	Side	Output	Safety	Safe / eligible	Output	Safety	Safe / eligible
OCRBench v2	Ground truth	100.00	52.58	245/466	99.02	97.64	455/466
	Prediction	100.00	65.70	3,605/5,487	95.21	98.70	5,394/5,465
UniMER-Test	Ground truth	100.00	96.88	22,572/23,300	99.87	98.47	22,927/23,284
	Prediction	100.00	98.04	264,678/269,974	97.61	98.69	265,673/269,209
OmniDocBench	Ground truth	99.96	71.69	1,461/2,038	99.78	97.60	1,989/2,038
	Prediction	99.96	56.94	476/836	98.73	98.30	635/646

changes are nonnegative in these two tracks, but their sizes differ sharply; the result requires model-specific reporting.

E.6 Front-End Safety, Unavailable Outcomes, and Missing Verdicts

Table 24 reports front-end safety together with the output rate, because a front-end can raise its conditional safety rate by refusing difficult inputs. Safety conditions on inputs that the front-end accepts and that render before normalization; an empty or unrenderable output remains in that denominator and is unsafe. The output rate divides nonempty outputs by rows with an input, while missing extraction or matching produces a separate no-input state; unlike coverage (Section A.3), it does not require the output to render, because safety already counts an unrenderable output as a failure.

Normalized exact acceptance keeps every reference-renderable sample in its denominator and assigns zero to prediction, extraction, or normalization outcomes that cannot produce a pair verdict. A covered-only rate is retained as a model-specific sensitivity diagnostic, but it is not used as an alternative ranking because its denominator changes with the model and front-end.

Across the two text tracks, the rendering oracle has no final relation for 14,357 of 285,192 fixed-support model–sample rows; these rows are neither forced into false reject nor into false-accept candidate. OmniDocBench has the larger measurement boundary: 7,826 of 25,424 input-bearing formula-scoring rows (30.8%) produce a rendering error under the independent MathJax protocol, while another 448 rows have no prediction input; 17,598 rows remain eligible for the conditional verdict audit. These missing rendering verdicts limit the oracle-based audit but do not remove rows from Formula CDM or Overall scoring, which continue to use their fixed benchmark denominators.

Table 25: Graded-score ranking sensitivity to replacing the benchmark front-end with TeXForm (Faithful) on fixed common supports. Under Ranks, "Changed" counts models whose displayed rank differs between the two front-ends, and "Max. shift" is the largest absolute displacement. Bold marks the two views whose leader changes.

			Ranks	
Graded-score view	Official leader	Faithful leader	Changed	Max. shift
OCRBench v2	UniMERNet-base	UniMERNet-base	8/12	2
UniMER-Test pooled	UniMERNet-base	**PaddleOCR-VL-1.6**	12/12	3
UniMER-Test SPE	UniMERNet-base	**PP-FormulaNet-L**	11/12	2
UniMER-Test CPE	PaddleOCR-VL-1.6	PaddleOCR-VL-1.6	11/12	2
UniMER-Test SCE	GLM-OCR	GLM-OCR	10/12	3
UniMER-Test HWE	HunyuanOCR v1.0	HunyuanOCR v1.0	8/12	2
OmniDocBench Formula CDM	PaddleOCR-VL-1.6	PaddleOCR-VL-1.6	3/11	2
OmniDocBench Overall	PaddleOCR-VL-1.6	PaddleOCR-VL-1.6	3/11	2

Table 26: UniMER-Test pooled corpus-BLEU on the 23,300-row fixed support, ordered by official rank. Bold marks the leader under each front-end.

Model	Official BLEU (95% CI)	Rank	Faithful BLEU (95% CI)	Rank
UniMERNet-base	**0.9054** [0.9002, 0.9102]	1	0.8673 [0.8575, 0.8770]	4
PaddleOCR-VL-1.6	0.9034 [0.8992, 0.9070]	2	**0.9158** [0.9107, 0.9201]	1
PaddleOCR-VL	0.8795 [0.8753, 0.8833]	3	0.8849 [0.8788, 0.8903]	2
PaddleOCR-VL-1.5	0.8740 [0.8698, 0.8776]	4	0.8757 [0.8696, 0.8812]	3
PP-FormulaNet-L	0.8718 [0.8661, 0.8773]	5	0.8093 [0.7955, 0.8216]	6
MinerU2.5	0.8052 [0.7997, 0.8097]	6	0.8117 [0.8048, 0.8181]	5
GLM-OCR	0.7524 [0.7477, 0.7563]	7	0.7400 [0.7330, 0.7468]	8
HunyuanOCR v1.0	0.7399 [0.7307, 0.7443]	8	0.7151 [0.7063, 0.7228]	9
Qwen2.5-VL-7B-Instruct	0.7100 [0.6956, 0.7236]	9	0.7527 [0.7440, 0.7600]	7
Qwen3-VL-8B-Instruct	0.7047 [0.6982, 0.7104]	10	0.6389 [0.6288, 0.6493]	11
pix2tex	0.6057 [0.6002, 0.6111]	11	0.3724 [0.3592, 0.3846]	12
InternVL3-8B	0.5923 [0.5854, 0.5989]	12	0.6883 [0.6812, 0.6954]	10

E.7 Score and Ranking Sensitivity

The most direct normalization consequence uses a fixed exact rule and changes only the front-end. Table 3 reports the complete cohort under that comparison. TeXForm (Faithful) raises exact acceptance for every model, but the gain is representation-dependent and therefore changes the ordering.

For the official top pair, the UniMERNet-base minus PaddleOCR-VL-1.6 exact gap is $+0.0216$ (95% CI [0.0145, 0.0280]) with the benchmark front-end and -0.0227 (95% CI [-0.0298, -0.0158]) with Faithful. The paired intervals exclude zero on opposite sides, localizing the leader reversal to the normalization front-end under a fixed exact rule. The exact leader on OCRBench does not change under formula-extracted replacement.

Table 25 summarizes the graded-score ranking changes across OCRBench v2, pooled and subset-level UniMER-Test, and OmniDocBench. Every view changes at least one model's rank, and only UniMER-Test pooled and SPE change the leader. Table 26 reports the complete pooled leaderboard because it is the principal graded-score leader reversal; Table 25 summarizes the leaders and rank changes for the other views.

The complete pooled comparison shows UniMERNet-base moving from first to fourth and PaddleOCR-VL-1.6 becoming the leader. Under the independent rendering-criterion acceptance rule, PP-FormulaNet-L instead leads UniMER pooled at 0.6080, but that ordering is a sensitivity diagnostic rather than a uniquely true ranking. A comparison between that criterion-based ordering and either front-end in Table 26 would change both the front-end and the decision rule and is therefore not assigned a single cause.

E.8 CDM Front-End Replacement in OmniDocBench

The replacement retains Formula CDM's token coloring, pdfLaTeX/CJK rendering, bounding-box extraction, Hungarian matching, and aggregation. Formula scores average fixed ground-truth-backed match rows, with processing failures scored as zero. Overall keeps Text and Table scores fixed and changes only the Formula component. Bootstrap resampling uses page identifiers.

The replacement improves reference rendering safety and recovers criterion-equivalent accepts, while the unchanged CDM backend measures rendered-character correspondence. Both Formula CDM and Overall retain PaddleOCR-VL-1.6 as leader and change middle ranks (Table 25).

F Cross-Renderer Replication and Robustness Checks

We test whether the main findings survive changes to the rendering instrument, pixel threshold, canonicity accounting, and missing verdicts.

F.1 Cross-Renderer Replication

We repeat the normalizer and benchmark audits with MathJax 4.1.1, KaTeX 0.16.45, and XeTeX from TeX Live 2026, using the same white-margin removal, scale alignment, and pixel thresholds. The benchmark replication uses the formula-extracted Faithful configuration for OCRBench; its raw-output replacement is reported in Section E.

Common-support protocol. Cross-renderer comparisons use three matched supports. Normalizer safety aligns the same normalizer-accepted, MathJax-raw-renderable tool–formula rows in every branch and retains failures under the evaluated renderer as failures; this ledger contains 3,793,349 rows. Before comparing renderers, we exclude 1,041 keys (less than 0.03%) whose non-renderer outputs differ among the three executions; this exclusion is independent of the renderer verdict. Canonicity intersects the fixed corpus with members having a final relation under all three renderers, excluding 7 label identities and 44 model–prediction identities absent from the fixed corpus; it retains 63,301 of 66,047 members (95.84%), 18,424 of 18,899 clusters with at least two distinct strings before the nontrivial-variation filter (97.49%), and 23,913 of 29,528 eligible pairs (80.98%).

Benchmark misverdict rates likewise require a final relation under all three renderers, yielding 4,560 OCRBench rows, 262,148 UniMER rows, and 17,082 OmniDocBench rows, respectively 74.8%, 91.9%, and 67.2% of their audit rows. Ranking uses ground-truth renderability under all three renderers and retains prediction-side rendering failures as zero; the common reference set contains 418 of 508 OCRBench v2 formulas (82.3%) and 23,179 of 23,757 UniMER-Test formulas (97.6%).

Normalizer properties. Table 27 combines one common canonicity denominator with renderer-specific preservation rates. The absolute preservation rate includes renderer failures, so the identity row isolates the acceptance domain of each engine: it is 100.00% under MathJax, 84.82% under KaTeX, and 98.95% under XeTeX. For this reason, the three safety columns should be compared within a renderer rather than averaged across renderers.

The profile sequence is stable under all three engines. Faithful adds 10.73 percentage points of canonicity over Authoring at a 0.01-point safety cost; Corpus adds 10.67 points at a 1.5–1.6-point cost. Equiv adds only 0.45 points while losing another 1.9–4.0 safety points. MathNet combines the strongest collapse with 56.17–60.39% safety. The safety–canonicity trade-off therefore persists under KaTeX and XeTeX.

Benchmark misverdicts. Table 28 reports official-rule errors and Faithful recovery on common rows. The official false-reject rate varies by at most 1.2 percentage points across renderers, and Faithful recovers 36.9–48.1% of those rejects on the two text tracks and 90.6–93.7% on OmniDocBench.

The OCRBench and UniMER false-accept-candidate rates remain respectively below 4.8% and 0.25% under every engine. OmniDocBench remains qualitatively different: all three renderers place its candidate rate between 41.3% and 44.4%. This agreement rules out a MathJax-specific acceptance domain as the sole explanation for that audit signal. The cohort result is also consistent

Table 27: Pair-level canonicity on the 23,913-pair common corpus and renderer-specific safety on common tool–formula rows. Bold marks the highest canonicity among all audited configurations and the highest safety within the TeXForm block; identity provides the renderer-coverage anchor.

		Safety (%)		
Implementation	Pair canonicity (%)	MathJax	KaTeX	XeTeX
Identity	0.00	100.00	84.82	98.95
UniMERNet postprocess	8.94	99.83	84.81	98.77
im2markup	19.09	94.43	82.37	95.43
MathNet	**66.26**	59.65	56.17	60.39
CDM (in UniMERNet)	26.23	86.77	75.73	87.54
CDM (in OmniDocBench v1.5)	26.09	87.68	76.39	88.48
CDM (in OmniDocBench v1.6)	25.94	86.22	75.14	86.96
TeXForm (Authoring)	32.32	**98.85**	**84.14**	**98.82**
TeXForm (Faithful)	43.06	98.84	84.13	98.81
TeXForm (Corpus)	53.73	97.20	82.66	97.19
TeXForm (Equiv)	54.18	93.43	80.74	93.23

Table 28: Official full-score audits and Faithful recovery on renderer-common relation supports. OCRBench recovery uses normalized equality after formula extraction and Faithful; false-accept-candidate columns report the official rule.

		Official rule (%)		
Benchmark	Renderer	False reject	False-accept candidate	Faithful recovery (%)
OCRBench v2	MathJax	19.94	3.53	44.79
	KaTeX	19.01	3.15	47.35
	XeTeX	19.13	4.73	48.15
UniMER-Test pooled	MathJax	26.29	0.24	38.15
	KaTeX	25.80	0.16	38.65
	XeTeX	27.01	0.16	36.90
OmniDocBench v1.6	MathJax	18.86	41.26	90.65
	KaTeX	18.15	44.38	93.69
	XeTeX	18.28	41.37	92.87

Table 29: Cross-renderer stability of the rendering-equivalence rankings on common reference sets. Spearman ρ compares each complete 12-model order with the MathJax order; on OCRBench v2 the top-two margin is 0.000 under MathJax and XeTeX and 0.007 under KaTeX.

		ρ vs. MathJax	
Ranking scope	Leading rendering-equivalence result	KaTeX	XeTeX
OCRBench v2	MathJax/XeTeX: PaddleOCR-VL-1.6; KaTeX: UniMERNet-base	0.937	0.944
UniMER pooled	PP-FormulaNet-L > PaddleOCR-VL-1.6 > UniMERNet-base under all three renderers	1.000	1.000
UniMER CPE	UniMERNet-base leads; complete order unchanged	1.000	1.000
UniMER HWE	HunyuanOCR v1.0 leads; complete order unchanged	1.000	1.000
UniMER SCE	PaddleOCR-VL-1.6 leads; only tail positions change	0.993	0.986
UniMER SPE	PP-FormulaNet-L leads; complete order unchanged	1.000	1.000

Table 30: Renderer-specific failures before the common-support intersection; the text tracks pool OCRBench v2 and UniMER-Test.

Failure type	MathJax	KaTeX	XeTeX
Text tracks			
Reference render failure	5,988	5,148	3,360
Relation abstention	20,345	20,298	17,455
OmniDocBench v1.6			
Reference render failure	3,133	2,604	2,977
Relation abstention	7,826	7,404	7,694

across individual models: Faithful recovers a positive share of official false rejects in all 105 model–renderer–benchmark cells, ranging from 15.4–80.4% on OCRBench, 17.1–56.6% on UniMER, and 86.7–97.5% on OmniDocBench.

Ranking stability. On the common UniMER reference set, normalized exact match with the benchmark front-end ranks UniMERNet-base ahead of PaddleOCR-VL-1.6 by 2.20 points. Authoring, Faithful, and Corpus all reverse that relative order by 2.26, 2.24, and 1.66 points and yield the same top three: PaddleOCR-VL-1.6, PP-FormulaNet-L, and UniMERNet-base. The leader change therefore persists across all three output profiles rather than depending on a selected TeXForm configuration.

Table 29 compares each rendering-equivalence ordering with the MathJax ordering on the same reference set. The UniMER pooled order is identical across all 12 models: PP-FormulaNet-L ranks first, PaddleOCR-VL-1.6 second, and UniMERNet-base third under every renderer. The complete CPE, HWE, and SPE orders are also identical; SCE differs only through tail exchanges.

UniMER remains sensitive to the normalization front-end, while the identical three-renderer pooled order rules out a MathJax-specific rendering ranking as the explanation.

OCRBench warrants a narrower conclusion. Its cross-renderer rank correlations remain high, but the common reference set contains only 418 formulas and the top two are tied at the reported precision under MathJax and XeTeX. The replication therefore supports broad ordering stability on OCRBench, not a renderer-invariant winner.

Renderer failures and scope. The engines do not succeed on the same inputs at the same rate, as Table 30 shows. XeTeX yields the fewest relation abstentions on the two text tracks, whereas KaTeX yields the fewest on OmniDocBench.

F.2 Pixel-Threshold and Rendering-Failure Sensitivity

We rejudge the stored continuous comparisons without rerendering any formula, holding the cross-renderer common-relation and common-GT supports fixed. The sensitivity protocol includes strict

Table 31: Pixel-threshold sensitivity on fixed cross-renderer common supports (%). Exact columns require strict pixel identity, and grid columns give the range over nine fixed SSIM/differing-pixel-ratio settings; FAC denotes official false-accept candidates, and recovery is the share of official false rejects accepted by the Faithful front-end.

		False reject		FAC		Recovery	
Benchmark	Renderer	Exact	Grid	Exact	Grid	Exact	Grid
OCRBench v2	MathJax	19.81	19.81–20.61	3.53	3.53	45.14	42.96–45.14
	KaTeX	18.94	18.94–20.18	3.15	3.15	47.54	43.94–47.54
	XeTeX	19.07	19.07–19.77	4.73	4.73	48.35	46.25–48.35
UniMER pooled	MathJax	25.52	26.03–26.69	0.37	0.24–0.25	39.68	37.38–38.68
	KaTeX	25.33	25.54–26.20	0.16	0.16	39.60	37.88–39.16
	XeTeX	26.51	26.73–27.40	0.16	0.16	37.84	36.18–37.44
OmniDocBench v1.6	MathJax	18.54	18.59–19.15	42.00	41.19–41.53	92.53	89.64–91.99
	KaTeX	17.79	17.87–18.40	44.45	44.25–44.43	95.68	92.57–95.19
	XeTeX	17.88	17.98–18.54	41.51	41.03–41.48	95.03	91.18–94.34

pixel identity and the complete Cartesian grid $\mathrm{SSIM} \in \{0.980, 0.990, 0.995\}$ and differing-pixel ratio $\in \{0.0005, 0.0010, 0.0020\}$; the frozen criterion $(0.990, 0.0010)$ is one grid point. Strict identity requires zero differing pixels and equal compared-image dimensions; the stored comparisons contain 13 MathJax rows on the text tracks with zero differing pixels but unequal dimensions, and none under the other renderers or on OmniDocBench.

Table 31 leaves the substantive conclusions unchanged. Across the threshold grid, official false rejects remain 18.9–20.6% on OCRBench, 25.5–27.4% on UniMER, and 17.9–19.2% on OmniDocBench; Faithful continues to recover 43.0–48.3%, 36.2–39.2%, and 89.6–95.2%, respectively. Strict pixel identity preserves the same qualitative result and places OmniDocBench recovery at 92.5–95.7%. OmniDocBench's false-accept-candidate signal also remains large at 41.0–44.5% across strict identity and every threshold setting, whereas the UniMER candidate rate remains at or below 0.37%.

At the frozen threshold, tolerance-only matches constitute 0.08–0.15% of accepted OCRBench relations, 0.63–1.17% on UniMER, and 0.57–1.63% on OmniDocBench, so 98.4–99.9% of accepted relations are already strict pixel identities. Strict pixel identity excludes every nonzero-difference acceptance, including visually similar but distinct glyphs. Changing the differing-pixel-ratio bound accounts for most observed movement; varying SSIM at a fixed ratio changes few rows.

The UniMER pooled order is identical for all 12 models under all ten settings and all three renderers ($\rho = 1.000$ relative to the frozen-threshold order): PP-FormulaNet-L remains first, PaddleOCR-VL-1.6 second, and UniMERNet-base third. OCRBench again supports only a broad-ordering conclusion: correlations with the frozen order remain at least 0.979, but the winner or runner-up changes under some settings whose differing-pixel-ratio bound is 0.0020. Rendering failures are unaffected by threshold rejudgment and retain the counts in Table 30.

F.3 Canonicity Accounting Choices

Pair-level canonicity counts an eligible unordered pair when both strings collapse to one output, whereas cluster-level canonicity requires every remaining member of an eligible cluster to collapse to one output. Both measures are exact and unconditional: normalization failures remain unsuccessful outcomes. Table 32 stratifies the same model-prediction clusters by their distinct member count after leave-one-out removal.

The size strata contain 850 eligible pairs at $n = 2$, 3,397 at $n = 3$, and 25,281 at $n \geq 4$; the corresponding cluster-level denominators are 850, 1,434, and 3,157. Requiring whole-cluster collapse lowers the absolute pass rates because every retained member must converge to one output. Excluding all two-member clusters preserves the TeXForm profile ordering and the contrast between strong legacy collapse and its safety cost, so the conclusion is not driven by their inclusion, although the rates remain conditional on this fixed model-prediction corpus.

Table 32: Canonicity pass rates (%) by post-leave-one-out cluster size; the $n \geq 3$ columns exclude all two-member clusters.

	Pair				Cluster
Implementation	$n = 2$	$n = 3$	$n \geq 4$	$n \geq 3$	$n \geq 3$
Identity	0.00	0.00	0.00	0.00	0.00
UniMERNet postprocess	0.00	5.89	8.98	8.61	0.04
im2markup	2.82	11.19	20.18	19.12	6.47
MathNet	52.35	56.40	68.24	66.84	60.18
CDM (in UniMERNet)	5.18	15.34	27.13	25.73	12.48
CDM (in OmniDocBench v1.5)	5.06	15.28	26.99	25.60	12.37
CDM (in OmniDocBench v1.6)	5.41	15.25	26.92	25.54	12.24
TeXForm (Authoring)	25.18	33.97	27.93	28.64	17.36
TeXForm (Faithful)	29.18	38.83	37.60	37.75	24.31
TeXForm (Corpus)	35.29	45.33	47.12	46.91	35.18
TeXForm (Equiv)	38.00	47.07	48.01	47.90	36.27

The primary canonicity corpus contains third-party benchmark labels and model predictions. Full and leave-one-out membership coincide for every audited tool; pair-versus-cluster accounting and cluster-size stratification supply the substantive perturbations.

F.4 Missing-Verdict Sensitivity

The fixed-support exact-match analysis keeps every unavailable verdict in the denominator as zero, while its covered sensitivity divides accepted matches by each model–front-end pair's known verdicts. The aggregation code distinguishes extraction failure, prediction failure, ground-truth refusal, prediction refusal, and pair-level refusal, so a missing verdict is not silently merged with an incorrect one. Per model, an exact-match verdict is available for 97.85–99.14% of the reference-renderable OCRBench v2 samples and 97.18–99.28% of the UniMER-Test samples under the benchmark front-end, and for 68.24–98.93% and 83.62–98.91% under TeXForm (Faithful).

For the UniMER top pair, UniMERNet-base has 182 more pair-normalization refusals than PaddleOCR-VL-1.6; even assigning every extra row as an accept leaves the Faithful exact-match gap at -0.0149, so this targeted availability bound preserves the observed leader reversal. OmniDocBench has a distinct missingness boundary because the independent renderer cannot assign a relation verdict to a substantial share of its scored rows; Section E.6 reports the missing-verdict counts and denominators.