LATTE: Improving LaTeX Recognition for Tables and Formulae With Iterative Refinement

Nan Jiang[1], Shanchao Liang[1], Chengxiao Wang[1, 2], Jiannan Wang[1], Lin Tan[1]

[1] Purdue University, USA
[2] University of Illinois Urbana-Champaign, USA
jiang719@purdue.edu, liang422@purdue.edu, cw124@illinois.edu, wang4524@purdue.edu, lintan@purdue.edu

Abstract

Portable Document Format (PDF) files are dominantly used for storing and disseminating scientific research, legal documents, and tax information. LaTeX is a popular application for creating PDF documents. Despite its advantages, LaTeX is not WYSWYG—what you see is what you get, i.e., the LaTeX source and rendered PDF images look drastically different, especially for formulae and tables. This gap makes it hard to modify or export LaTeX sources for formulae and tables from PDF images, and Fault Location

existing work is still limited. First, prior work generates LaTeX sources in a single iteration and struggles with complex LaTeX formulae. Second, existing work mainly recognizes and extracts LaTeX sources for formulae; and is incapable or ineffective for tables. This paper proposes LATTE, the first *iterative refinement* framework for LaTeX recognition. Specifically, we propose `delta-view` as feedback, which compares and pinpoints the differences between a pair of rendered images of the extracted LaTeX source and the expected correct image. Such `delta-view` feedback enables our fault localization model to localize the faulty parts of the incorrect recognition more accurately and enables our LaTeX refinement model to repair the incorrect extraction more accurately. LATTE improves the LaTeX source extraction accuracy of both LaTeX formulae and tables, outperforming existing techniques as well as GPT-4V by at least 7.07% of exact match, with a success refinement rate of 46.08% (formula) and 25.51% (table).

Introduction

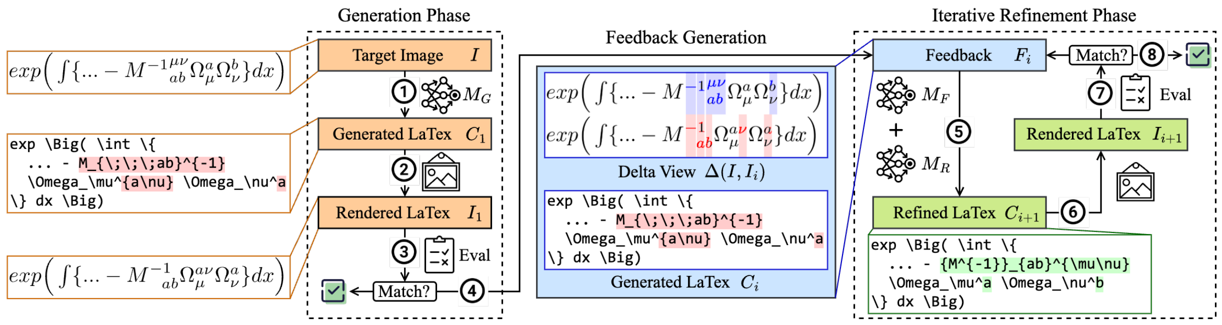
Portable Document Format (PDF) files are dominantly used for storing and disseminating academic research, legal documents, and tax information (Blecher et al. 2023; Kuchta et al. 2018a). While documents in such format provide exceptional cross-platform consistency and readability and are flexible in display resolutions, the source code of the PDF files is typically unavailable to the readers. Thus, it is hard to modify, extract, and export PDF documents to other target formats, especially those containing mathematical formulae and tables with complex structures and styles.

Since LaTeX is one widely used system to produce PDF documents (Kuchta et al. 2018b), researchers have explored LaTeX recognition (LR) to extract mathematical expressions from images using either rule-based or learning-based approaches (Wang and Liu 2021; Peng et al. 2021; Deng et al. 2017; Pang et al. 2021; Yan et al. 2021; Long, Hong, and Yang 2023; Anderson 1967). Other work focuses on extracting table structures or detecting the content in each cell (Hashmi et al. 2021; Kayal et al. 2022), which helps understand and analyze tables.

These existing techniques (Wang and Liu 2021; Peng et al. 2021; Deng et al. 2017; Pang et al. 2021; Yan et al. 2021; Long, Hong, and Yang 2023; Anderson 1967) is limited in recognizing LaTeX images. First, they produce LaTeX sources in a single round of generation and often fail to recognize complex formulae. Our insight is that *humans often write complex formulae and tables in multiple iterations*. For example, if the first version of the LaTeX source is incorrect, they fix the mistakes, re-render the modified LaTeX source, and keep this iterative process. Second, existing techniques focus on LaTeX formulae. The few table recognition techniques do not extract ready-to-use LaTeX source for tables (Hashmi et al. 2021; Kayal et al. 2022), but only extract table structures or textual content. Simply combining the table structures and content does not produce LaTeX sources that can be rendered (Kayal et al. 2022), because the structures and content may mismatch with each other.

In this paper, we propose **LATTE**, a LaTeX recognition framework with **iterative refinement**. We also create a new LaTeX table dataset, TAB2LATEX, by collecting LaTeX tables source code from arXiv preprints and the corresponding LaTeX sources. TAB2LATEX is a dataset for end-to-end LaTeX tables recognition, aiding the development of techniques to produce renderable LaTeX sources for tables. We demonstrate the effectiveness of LATTE on recognizing **both LaTeX tables and formulae**.

The concept of iterative refinement has been applied in various fields, including code generation and code refinement (Madaan et al. 2023; Scheurer et al. 2023; Chen et al. 2023a,c; Olausson et al. 2023). The process typically consists of two parts: generating an initial draft and then iteratively refining it using collected feedback until it meets the requirements. Yet, applying the refinement framework in the field of LR is challenging, as it is hard to generate feedback that effectively connects the expected ground-truth image and the generated textual draft. The difference between the


Generation Phase
Feedback Generation
Iterative Refinement Phase
$exp\Big(\int\{\ldots - M^{-1}{}^{\mu\nu}_{ab}\Omega^a_\mu\Omega^b_\nu\}dx\Big)$
Target Image I
① M_G
Generated LaTex C_1
②
Rendered LaTex I_1
③ Eval
Match? ④
$exp\Big(\int\{\ldots - M^{-1}_{ab}\Omega^{a\nu}_\mu\Omega^a_\nu\}dx\Big)$
$exp\Big(\int\{\ldots - M^{-1}{}^{\mu\nu}_{ab}\Omega^a_\mu\Omega^b_\nu\}dx\Big)$
$exp\Big(\int\{\ldots - M^{-1}_{ab}\Omega^{a\nu}_\mu\Omega^a_\nu\}dx\Big)$
Delta View $\Delta(I, I_i)$
Generated LaTex C_i
Feedback F_i
M_F
+
M_R
⑤
Match? ⑧
⑦ Eval
Rendered LaTex I_{i+1}
Refined LaTex C_{i+1} ⑥

```
exp \Big( \int \{
  ... - M_{\;\;\;ab}^{-1}
  \Omega_\mu^{a\nu} \Omega_\nu^a
\} dx \Big)
```

```
exp \Big( \int \{
  ... - M_{\;\;\;ab}^{-1}
  \Omega_\mu^{a\nu} \Omega_\nu^a
\} dx \Big)
```

```
exp \Big( \int \{
  ... - {M^{-1}}_{ab}^{\mu\nu}
  \Omega_\mu^a \Omega_\nu^b
\} dx \Big)
```

Figure 1: Overview of LATTE. M_G, M_F, and M_R are the initial generation, fault localization, and the refinement models.

expected image and the image rendered from the draft needs to be identified automatically. And the model is also required to learn the portions of the generated text that cause such differences in images, i.e., building the connection between textual scripts and rendered images. To overcome this challenge, we propose an `ImageEdit` algorithm to pinpoint the differences between the ground truth and rendered images, referred to as `delta-view`. **We use delta-view as feedback** to aid LATTE to localize and refine the error.

Another challenge of applying the refinement framework in LR is identifying the faulty location of the generated LaTeX script. The faulty LaTeX scripts typically are only incorrect in a small portion, such as a few incorrect characters for mathematical formulae, or a few cells for tables. Instead of re-generating the whole LaTeX script, one can localize the faulty parts and re-generate those parts only. Thus, we implement a fault localization model trained along with the refinement model to predict the faulty location. Once we surmount this challenge and successfully identify the faulty location of the LaTeX script, LATTE only need to re-generate the incorrect portion of it, which minimizes the learning challenges of the refinement model.

To sum up, this paper makes the following contributions:

- We create the first iterative-refinement approach for LaTeX recognition, which fine-tunes a localization model to identify the faulty part in LaTeX sources and use a refinement model to regenerate the faulty part of the LaTeX sources iteratively.
- We propose a novel algorithm, `ImageEdit`, which produces effective feedback, `delta-view`, to enhance the refinement accuracy.
- We collect and open-source a new dataset for LaTeX tables recognition, TAB2LATEX, filling the blank of no published dataset for end-to-end LaTeX table recognition.
- By combining iterative-refinement and `ImageEdit`, we develop **LATTE** to produce renderable LaTeX code for both formulae and tables, outperforming existing techniques on formulae by 7.07% of exact match, and commercial tools on tables by 56.00%, with an overall fault localization accuracy of 56.90–60.53%, and refinement rate of 25.51–46.08%.
- Availability: https://github.com/lt-asset/Latte.git

Approach

Figure 1 provides an overview of LATTE, which consists of two phases — the Generation Phase and the Iterative Refinement Phase. Given the target document image I to recognize, the generation model M_G generates a LaTeX output C_1 as the initial draft (step ①). LATTE then uses `pdflatex` to render the LaTeX source draft into an image I_1 (step ②) and compares it with the ground-truth image I (step ③). If they match at the pixel level, signaling that the LaTeX source C_1 is correct, the process ends and LATTE outputs C_1.

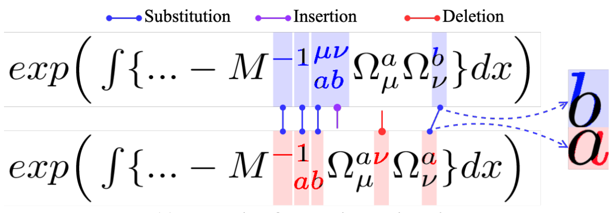
Otherwise, LATTE enters the Refinement Phase (step ④). During the i^{th} iteration of the refinement phase, LATTE automatically generates feedback F_i consisting of C_i and `delta-view` $\Delta(I, I_i)$, highlighting the difference between the ground truth and the rendered image. Then, LATTE uses the fault localization model, M_F, to predict the faulty location in the LaTeX script. The predicted location is used to construct the input for the refinement model M_R. M_R generates the refined LaTeX script starting from the predicted faulty location (step ⑤), which replaces the faulty parts in C_i to form the fully refined script C_{i+1}. The refined script C_{i+1} is rendered into a new image I_{i+1} (step ⑥), and is compared to the ground-truth image for evaluation (step ⑦). Such a refinement phase continues until the evaluation passes (step ⑧) or reaches the iteration limit.

Generation Phase

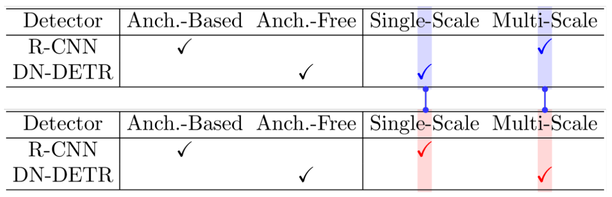
LATTE's generation model, M_G, is fine-tuned on top of the Nougat-base (Blecher et al. 2023), a multi-modal vision-encoder-decoder (Li et al. 2023b) LLM pre-trained on RGB images of academic documents and their markdown sources. The input for the M_G is an image $I \in \mathbb{N}^{H\times W\times 3}$ of a rendered formula or a table, where H and W represent the height and width of the image respectively, and 3 refers to the color channels in RGB images. The vision-encoder of M_G encodes the image, and the text decoder generates the corresponding LaTeX source code of the input image.

Evaluation and Feedback Generation

After the generation model M_G produces the initial LaTeX draft, LATTE evaluates its correctness by rendering it using renderer `pdflatex`, and comparing it with the ground-truth image. If the rendered image matches the ground-truth im-


Substitution
Insertion
Deletion
$exp\left(\int\{\ldots - M^{-1\,\mu\nu}_{\quad ab}\Omega^{a}_{\mu}\Omega^{b}_{\nu}\}dx\right)$
$exp\left(\int\{\ldots - M^{-1}_{ab}\Omega^{a\nu}_{\mu}\Omega^{a}_{\nu}\}dx\right)$
b
a

(a) Example of Formula's Delta-View



Detector	Anch.-Based	Anch.-Free	Single-Scale	Multi-Scale
R-CNN	✓			✓
DN-DETR		✓	✓	

Detector	Anch.-Based	Anch.-Free	Single-Scale	Multi-Scale
R-CNN	✓		✓	
DN-DETR		✓		✓

(b) Example of Table's Delta-View

Figure 2: Formula and table examples of `delta-view` generated by the `ImageEdit` algorithm.

age, the generated LaTeX script will be returned without refinement. Otherwise, it needs to be refined.

The feedback, which is the input to the refinement model, contains two elements: `delta-view` $\Delta(I, I_i)$ and the generated LaTeX script of the current refinement iteration C_i. To facilitate the localization of the fault in the incorrect script C_i, this work proposes the `ImageEdit` algorithm, which highlights the differences between the rendered image I_i and the ground-truth image I, and generates $\Delta(I, I_i)$. The `ImageEdit` algorithm is based on the Wagner–Fischer algorithm (Wagner and Fischer 1974) used for computing the Levenshtein-Distance. `ImageEdit` treats LaTeX images as lists of columns of pixels. It calculates the least number of insertions, deletions, and substitutions of columns needed to transform the rendered image to the ground truth image, which is marked by light blue or light red backgrounds.

Figure 2 (a) provides an example of computing the `delta-view` for LaTeX formula. `ImageEdit` uses four blocks of substitutions, one block of deletion, and one block of insertion to show the difference. For example, the $^{a}_{\nu}$ in the rendered image is incorrect and should be replaced by the $^{b}_{\nu}$ in the ground truth. In addition, `ImageEdit` provides finer-grained differences to help M_F generate more accurate refined LaTeX sources. For the substitution from $^{a}_{\nu}$ to $^{b}_{\nu}$, `ImageEdit` identifies that only a portion of a and b is different, which is highlighted in blue and red. The identical parts, i.e., ν and a portion of a and b, are shown in black.

For the table example shown in Figure 2 (b), the colored backgrounds mark the mismatched columns, while the blue and red edits show that the real fault is the positions of the checks in each column. In addition, to handle the more complex 2-D structure of table images, both the column-wised Levenshtein-Distance and the row-wised Levenshtein-Distance are calculated and the `delta-view` is generated using the solution with fewer edit percentages.

Iterative-Refinement Phase

The refinement phase of LATTE consists of two steps: fault localization and refinement. The fault localization model

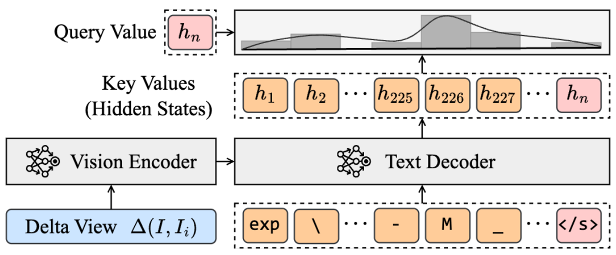


Figure 3: Fault localization model architecture.

pinpoints the faulty portion in the incorrect LaTeX script, which enables the refinement model to focus on modifying the wrong portion. The refinement model then generates the refined LaTeX script to replace the faulty portion suggested by the localization model.

Fault Localization Model LATTE's fault localization model predicts the location of the first erroneous token in the corresponding LaTeX script. Figure 3 shows the fault localization model's architecture.

The fault localization model includes a vision-encoder-decoder (VED) model and an attention layer. Given the incorrect LaTeX script $C_i = \{c_1, \ldots, c_n\}$ and the `delta-view` $\Delta(I, I_i)$ as input, the vision-encoder-decoder model calculates the hidden states of c_i to c_n as shown in Equation (1) (notated by H).

The following attention layer calculates the attention scores on each token c_i, using its hidden states H as keys and h_n as the query. The reason for using h_n, the hidden states of the last token `</s>` at the end of the incorrect LaTeX script, as the query for calculating attention score is that: h_n is the only hidden states in H that incorporates the features of the entire C_i. As in the text decoder, every other token only incorporates the features of tokens before them, missing the global view of the whole incorrect LaTeX script.

In the attention layer, W_q, W_k are trainable weights to encode the query and keys to compute the attention score distribution P. Once P is obtained, the index with the highest attention score will be selected as the faulty location l. The full formulation of fault localization is as follow:

$$\begin{aligned} &H = \text{VED}(C_i, \Delta(I, I_i)) \\ &Q = \text{ReLU}(W_q \cdot h_n) \quad , \quad K = \text{ReLU}(W_k \cdot H) \\ &P = \text{Softmax}\left(QK^{\top}\right), \quad l = \underset{1 \le i \le n}{\text{argmax}}(P) \end{aligned} \tag{1}$$

The training objective for the fault localization model is to minimize the Negative Log-Likelihood (NLL) loss on the probability of predicting the ground-truth faulty location l_i for the given LaTeX script C_i to refine by updating the fault localization model's weights θ_{M_F}.

Refinement Model with Fault Location As Figure 4 shows, given a wrong LaTeX C_i to refine and the faulty location l_i of it, the textual input for refinement model is structured as follows: "$C_i[l_i :]$ `<s>` $C_i[: l_i]$".

Different from inputting the whole incorrect script C_i as input and training the refinement model to generate the

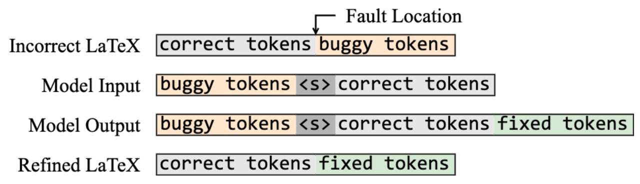

Fault Location
Incorrect LaTeX
correct tokens buggy tokens
Model Input
buggy tokens <s> correct tokens
Model Output
buggy tokens <s> correct tokens fixed tokens
Refined LaTeX
correct tokens fixed tokens

Figure 4: Workflow of the refinement model.

whole refined script, this template utilizes the faulty location by putting the faulty part of C_i at the beginning of the prompt. `<s>` is used as a separator and tokens after it are correct parts (tokens with light grey background). Such an input format design is more effective than inputting the whole incorrect script as is (Hossain et al. 2024). The refinement model is fine-tuned to generate the refined LaTeX tokens replacing the faulty parts (e.g., generating the LaTeX script with green background in Figure 4). The final refined LaTeX script can be easily reconstructed from the prompt and refinement model's generation, which is the non-faulty parts (LaTeX script before the faulty location) followed by the refinement model's generation.

Formally, given the incorrect LaTeX script C_i and faulty location l_i, we notate the ground-truth of the refined part to be $R_i = \{r_1, r_2, \ldots, r_m\}$, then the training objective of the refinement model is minimizing the negative log-likelihood of generating the R_i based on the prompt by updating the model's weights θ_{M_R}:

$$L_R(\theta_{M_R}) = -\log(P(R_i | \{c_{l_i}, c_{l_i+1}, \ldots, c_n, \texttt{<\textbackslash s>}, c_1, \ldots, c_{l_i-1}\}, \Delta(I, I_i))) \quad (2)$$

During inference, the predicted faulty location l'_i generated by the fault localization model is used to build the prompt for the refinement model. The refinement model will generate the refined part $R'_i = \{r'_1, r'_2, \ldots, r'_{m'}\}$. The final refined LaTeX script is constructed as $C_{i+1} = \{c_1, \ldots, c_{l'_i-1}, r'_1, \ldots, r'_{m'}\}$.

Experimental Setup

Datasets

For formulae recognition, we use **Img2Latex-100k**, which consists of 103,556 rendered images of mathematical formulae and the corresponding LaTeX scripts, collected from over 60,000 published academic documents (Deng et al. 2017). During preprocessing, the source LaTeX scripts are first rendered to PDF format and then converted to PNG format with 240 dpi. Then, the PNG images are either resized or padded to the resolution of 1344×224 pixels.

For table recognition, as there are no open-sourced datasets for end-to-end LaTeX table recognition yet, this work constructs a new dataset, **Tab2Latex**. Tab2Latex consists of 97,532 rendered images of tables (resolution of 1344×672 pixels) and their LaTeX sources.

Formula Models Training

We use the default split of the Img2Latex-100k dataset, which has 73,812 training, 18,672 validation, and 10,072 test instances, to train the generation model. We fine-tune the pre-trained Nougat-base model (Blecher et al. 2023), using a batch size of 16. The model weights are optimized using the AdamW (Loshchilov and Hutter 2019) optimizer, with the learning rate set to $3e^{-5}$, using 1,000 steps of warm-up and adjusted by a cosine decay scheduler.

For fault localization and refinement models, we collect incorrect LaTeX sources by sampling 20 LaTeX sources per image in the training set of Img2Latex-100k (sampling temperature set to 0.8). The sampled LaTeX sources are rendered and compared with the ground-truth images to judge their correctness, among which we collect 569,499 incorrect LaTeX sources and their corresponding ground-truth refinement. Fault localization and refinement models are fine-tuned independently, but both from the Nougat-base checkpoint and for one epoch, using a batch size of 32. The optimizer, and learning rate are the same as above.

Table Models Training

For table recognition, the generation model is fine-tuned on Tab2Latex (87,513 training, 5,000 validation, and 5,000 test instances). The other hyper-parameters are set in the same way as the fine-tuning of the generation model for formulae. The training data for fault localization and refinement models are collected in the same way as formulae, which contain 326,185 incorrect LaTeX sources and their ground-truth refinements.

Results

To evaluate LATTE on LaTeX recognition, we study the following research questions:

- **RQ1: What is the recognition accuracy of LATTE?**
- **RQ2: How is LATTE's iterative refinement ability?**
- **RQ3: What is the impact of each design of LATTE?**

Since LATTE refines incorrect LaTeX sources iteratively before generating the correct ones or reaching the budget, we use LATTE$_k$ to present our approach with at most k rounds of generation (one round of generation and $k-1$ round of refinements). LATTE$_1$ refers to the result of only generating the initial draft using the LATTE's generation model M_G. LATTE$_2$, LATTE$_3$, and LATTE$_4$ refer to the result of letting LATTE refine the incorrect LaTeX sources for one, two, and three rounds. We let LATTE refine at most three rounds.

RQ1: LATTE Recognition Accuracy

We used the five metrics for evaluation. **Match** (exact match accuracy) requires the rendered generated LaTeX source to have the same pixel values as the ground-truth image. **CW-SSIM** (Sampat et al. 2009) (complex-wavelet structural similarity index) measures the structural similarity between rendered and ground-truth images (we use MATLAB's implementation (Mehul 2024) with `level=4, or=8, K=0.01`). **BLEU** (Papineni et al. 2002) measures the textual similarity between the generated LaTeX source and the ground-truth LaTeX source (we report BLEU-4). **Edit** measures the column-wised edit distance between the rendered image and the ground-truth image, calculated by

Method	Match ↑	CW-SSIM ↑	BLEU ↑	Edit ↑	Time ↓
WYGIWYS	77.46	-	87.73	87.60	-
DA	79.81	-	88.42	88.57	-
EDPA	82.07	-	92.31	91.39	-
WAP	82.08	-	88.21	89.58	-
MI2LaTeX	82.33	-	90.28	91.90	-
ConvMath	83.41	-	88.33	90.80	-
Vary-1.8B	11.91	0.7895	69.46	63.47	2.27s
Llava-v1.5-7B	13.54	0.7548	75.40	64.61	2.29s
LATTE$_1$	82.27	0.9462	92.91	93.11	0.87s
LATTE$_2$	**90.44**	**0.9844**	**93.25**	**97.69**	1.53s

Table 1: Evaluation on IMG2LATEX-100K.

Method	Match ↑	CW-SSIM ↑	BLEU ↑	Edit ↑	Time ↓
Vary-1.8B	6.92	0.6253	62.89	30.50	7.13s
Llava-v1.5-7B	13.90	0.7278	64.19	39.84	6.13s
LATTE$_1$	45.20	0.8128	79.06	73.82	2.24s
LATTE$_2$	**59.18**	**0.8221**	**83.81**	**77.51**	5.34s

Table 2: Evaluation on TAB2LATEX.

$1 - \frac{\text{column-wised edit distance}}{\text{number of pixel columns}}$. Lastly, we report the used time per sample for available techniques.

We compare LATTE with a wide range of previous SOTAs (Long, Hong, and Yang 2023; Wang and Liu 2021; Yan et al. 2021; Zhang, Bai, and Zhu 2019; Deng et al. 2017), and also other MLLMs including a Vary-1.8B (Wei et al. 2023) model fully fine-tuned using the training data, and a Llava-v1.5-7B (Liu et al. 2023a) model fine-tuned using LoRA (Hu et al. 2022). Lastly, we also report the performance of commercial tools such as GPT-4V, Gemini-1.5-Pro, and Mathpix (commercial software for LaTeX recognition).

Formulae Table 1 shows the evaluation of LATTE$_1$ and LATTE$_2$ (the study of LATTE$_3$ and LATTE$_4$ are in RQ2). On IMG2LATEX-100K benchmark, with one round of refinement, LATTE$_2$ successfully refines 823 incorrect LaTeX sources from LATTE$_1$ and achieves 90.44% Match, significantly outperforming all the existing state-of-the-art techniques. The CW-SSIM, BLEU, and Edit scores are also improved with the refinement by 0.0382, 0.34%, and 4.58%.

Tables Table 2 shows the evaluation results on TAB2LATEX. As there are no open-sourced tools we can directly run on table recognition, we compare LATTE$_1$ and LATTE$_2$ with the fine-tuned Vary-1.8B and Llava-v1.5-7B. LATTE$_1$'s fine-tuned generation model generates LaTeX sources for tables matching 2,260 samples out of 5,000 (45.20% Match). The lower scores of the evaluation metrics suggest the challenge and potential for improvement of LaTeX table recognition. With one round of refinement, LATTE$_2$ correctly refines 699 incorrect sources and boosts the Match to 59.18%. The CW-SSIM, BLEU, and Edit scores are also improved by 0.0093, 4.75%, and 3.68%. Both LATTE$_1$ and LATTE$_2$ significantly outperform the other MLLMs we fine-tuned.

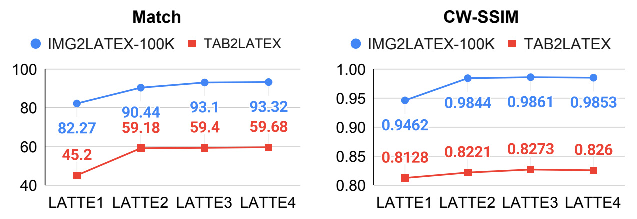


Figure 5: Evaluation of LATTE$_1$ to LATTE$_4$.

Comparing with Commercial Tools Table 3 shows the comparison with commercial tools on a subset of 100 samples from IMG2LATEX-100K and TAB2LATEX. Similarly, we use GPT-4V$_1$ to refer to prompting GPT-4V to generate the initial draft of LaTeX source, and GPT-4V$_2$ to refer to prompting it for one round of refinement of the incorrect sources (same for Gemini-1.5-Pro, and Mathpix is not applicable for refinement). We use few-shot learning (Brown et al. 2020) with three shots provided when prompting GPT-4V and Gemini-1.5-Pro for generation and refinement.

Table 3 shows that GPT-4V, Gemeni-1.5-Pro and Mathpix fail to generate the correct LaTeX source code most of the time. They also do not show the ability to effectively refine the incorrect LaTeX source with rendering feedback. Both LATTE$_1$ and LATTE$_2$ generate significantly more correct LaTeX sources than commercial MLLMs and software.

RQ2: LATTE's Iterative Refinement Ability

Figure 5 shows the results when LATTE refines multiple rounds. On IMG2LATEX-100K, LATTE's Match keeps increasing, from 82.27% to 93.32% after three rounds of refinement, with the most significant improvement during the first refinement iteration. Similarly, for TAB2LATEX, LATTE's Match increases from 45.20% to 59.68% in three rounds of refinement, and the biggest gain also happens at the first refinement round, by 13.99%. For CW-SSIM, LATTE's performance on IMG2LATEX-100K first increases from 0.9462 to 0.9844, which is a big gain of 0.0382, then fluctuates around it. The improvements on TAB2LATEX are more moderate compared to that of IMG2LATEX-100K, by 0.0930 in the first round, with the remaining rounds staying around the same values.

Overall, LATTE shows the ability to consistently improve the Match result by conducting iterative refinements, while the first round of refinement brings the most improvements.

RQ3: Impact of Each Component of LATTE

LATTE contains two designs: the `delta-view` feedback, and the fault localization model. To illustrate the effectiveness of each component, we design an ablation study by comparing LATTE with the following variants (only one round of refinement is conducted using each method):

- LATTE$_{-fl-dv}$ is LATTE without fault localization and `delta-view`. The refinement model generates a new LaTeX source (instead of starting from the fault location), given the original ground truth image.
- LATTE$_{-fl}$ is LATTE without fault localization. The refinement model generates a new LaTeX source, with `delta-view` as the feedback.

Method	Img2Latex-100k				Tab2Latex			
	Match ↑	CW-SSIM ↑	BLEU ↑	Edit ↑	Match ↑	CW-SSIM ↑	BLEU ↑	Edit ↑
GPT-4V$_1$	3.00	0.7480	52.77	61.25	2.00	0.5189	49.56	8.98
GPT-4V$_2$	7.00	0.7212	50.87	59.46	2.00	0.5059	44.22	5.64
Gemini$_1$	19.00	0.6485	21.47	63.60	0.00	0.3482	35.19	0.94
Gemini$_2$	19.00	0.6191	25.78	61.58	0.00	0.3911	37.58	1.27
Mathpix	20.00	0.8684	20.71	84.44	11.00	0.6749	49.45	28.31
Latte$_1$	77.00	**0.9878**	92.45	**97.68**	40.00	0.8659	77.53	67.49
Latte$_2$	**87.00**	0.9778	**93.72**	96.92	**67.00**	**0.8723**	**83.82**	**77.36**

Table 3: Comparison with commercial tools on 100 samples from Img2Latex-100k and Tab2Latex.

Method	Img2Latex-100k		Tab2Latex	
	Match ↑	Ref. Rate ↑	Match ↑	Ref. Rate ↑
Latte$_{-fl-dv}$	86.42	23.40	49.86	8.50
Latte$_{-fl}$	88.55	35.44	**59.52**	**26.13**
Latte	**90.44**	**46.08**	59.18	25.51
Latte*	90.45	46.14	59.72	26.50

Table 4: Impact of Fault Localization and `Delta-View`.

Table 4 shows the comparison between Latte$_{-fl-dv}$, Latte$_{-fl}$ and Latte. Match and refinement rate (the number of correct refinements divided by the total number of incorrect sources that need refinement) are reported. By using `delta-view` as feedback, Latte$_{-fl}$ outperforms Latte$_{-fl-dv}$ on formulae by 2.13% more Match (88.55% vs. 86.42%), and 12.04% higher refinement rate (35.44% vs. 23.40%). On tables dataset, Latte$_{-fl}$ benefits from `delta-view` by improving the Match from 49.86% to 59.52% and refinement rate from 8.50% to 26.13%. Results show that `delta-view` is much more effective than just providing the ground-truth image.

As for the fault localization model, when comparing Latte and Latte$_{-fl}$, the fault localization model helps the refinement model refine more incorrect formulae (46.08% vs. 35.44%), further increasing the Match from 88.55% to 90.44%. On the tables dataset, a slight decrease is observed, that using the fault localization model decreases the Match by 0.34%. Such a decrease may be due to the lower fault localization accuracy on tables than on formulae.

To better understand the impact of fault localization, Figure 6 (a) shows the fault localization accuracy on formulae and tables scripts with different lengths. On formulae, the fault localization accuracy (with `delta-view` or not) drops dramatically when the incorrect LaTeX scripts get longer. Yet the accuracy when using `delta-view` as feedback still consistently outperforms the accuracy when using the ground-truth image as feedback and the overall average is 60.53% versus 53.36%. Surprisingly, we find the fault localization accuracy does not drop much with longer incorrect tables, although the overall average accuracy of using `delta-view` is much higher (56.90% versus 45.22%).

Figure 6 (b) shows the refinement model's successful refinement rate on incorrect formulae and tables with different lengths. On formulae, the refinement rate also drops significantly when the incorrect LaTeX script gets longer. Additionally, when the fault localization model predicts the faulty location correctly, the refinement model has a higher success rate. On the tables dataset, the trend is slightly different, as the refinement rate is always low if the fault localization model predicts incorrect faulty locations.

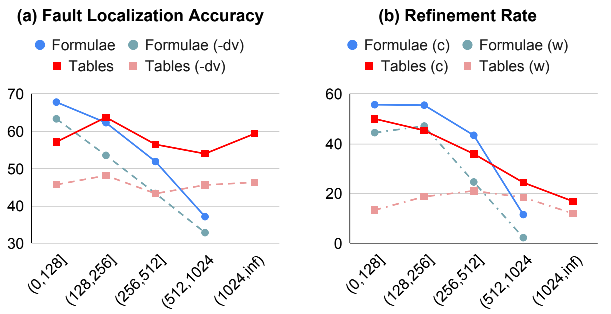


Figure 6: (a) Fault localization accuracy with `delta-view` and with ground-truth image (-dv). (b) Success refinement rate under correct (c) or wrong (w) fault localization. The x-axis is the length, in the number of characters, of the incorrect LaTeX to be refined.

Potential Improvement By digging into the fault localization accuracy, we see potential space for improvement by combining Latte with Latte$_{-fl}$, referred to as Latte* in Table 4. Latte* uses Latte$_{-fl}$ to refine incorrect LaTeX scripts with more than 512 characters, and uses Latte to refine those shorter than 512 characters. Such a simple ensemble reaches a higher Match and refinement rate than each single approach. Despite such potential, Latte is still the most effective single approach overall, and advanced ensemble approaches could be promising future work.

Related Work

LaTeX Recognition

Existing work on LR includes rule-based, grammar-based, and deep learning-based methods (Yan et al. 2021). Learning-based solutions utilize the encoder-decoder architecture to tackle LR. The encoders often consist of convolution neural networks to extract image features with the decoders being recurrent neural networks to generate the output sequence in an end-to-end manner (Zhang et al. 2017; Peng et al. 2021; Zhang, Bai, and Zhu 2019; Wang and Liu 2021; Long, Hong, and Yang 2023; Mirkazemy et al.

2023). On table recognition, due to the difficulty, earlier work mainly focuses on table detection and table structure recognition (Hashmi et al. 2021). Recently, IBM researchers proposed an encoder-dual-decoder model to separately collect the structural information and contents within table cells (Zhong, ShafieiBavani, and Jimeno Yepes 2020). However, even with all such information, users still struggle with reproducing the renderable source of LaTeX tables, as the structure and content extracted cannot be combined. Conversely, another work proposes a dataset (Deng, Rosenberg, and Mann 2019) containing pairs of table images and the corresponding LaTeX source code. This is the only work we have found that works on the end-to-end LaTeX table recognition, yet their dataset is not accessible anymore.

This work not only adds the iterative-refinement pipeline to the generation process but also includes a fault localization model to predict the faulty location for those incorrect sources to improve the recognition accuracy. The proposed TAB2LATEX dataset contains 97K well-filtered table images and source code pairs and fills the blank of the end-to-end LaTeX tables recognition dataset.

Iterative Refinement Framework

Researchers have identified the process of refining one's answer as a typical part of the problem-solving process (Amabile 1983; Simon 1962; Jiang et al. 2024), which has been added to numerous fields, e.g., program repair, code, and text generation. Existing work on the program repair applies automatic refinement for repair and fault localization on imperative programs based on symbolic execution (Könighofer and Bloem 2011). For code and text generation, some work prompts the LLM to provide feedback by itself, then refine its answer without additional training (Madaan et al. 2023; Chen et al. 2023c), while others fine-tune the LLMs to enhance their ability to adopt the feedback (Scheurer et al. 2023; Chen et al. 2023a) for better refinement. This work is the first to apply the iterative refinement framework within the field of LR. Applying iterative-refinement in LR is new and has unique challenges regarding providing effective image feedback. LATTE introduces `delta-view` as novel feedback to address such challenges in multi-modal generation and refinement, which is shown to be helpful in the ablation study.

Multi-Modal Large Language Models

Many MLLMs have shown great ability in vision and text tasks, such as image captioning (Radford et al. 2021; Li et al. 2022, 2023a), image understanding (Lee et al. 2023), visual question answering (Li et al. 2022, 2023a; Liu et al. 2023b,a; Dai et al. 2023), etc. The early paradigm for building MLLMs involves jointly training vision and text models, e.g., CLIP and BLIP (Radford et al. 2021; Li et al. 2022). Later work train adapters to connect the pre-trained vision encoder and text decoder, to borrow the strong text generation ability of textual LLMs without retraining from scratch (Liu et al. 2023b; Dai et al. 2023; Liu et al. 2023a; Li et al. 2023a; Bai et al. 2023; Chen et al. 2023b). However, most existing MLLMs (including GPT-4V) are optimized for understanding pictures and natural language, not document images and LaTeX. Nougat (Blecher et al. 2023) is the only existing MLLM pre-trained on documents, on which we build LATTE's models.

Limitation

One limitation of our work is that we only explore Nougat as LATTE's backbone model. Many other MLLMs, such as Llava and Vary, can be used as our backbone model. However, they are mostly pre-trained on natural language and pictures and show very poor performance on LaTeX recognition. Nougat is the best MLLM we can find, and it is dominantly pre-trained on documents.

Another possible limitation of LATTE is that existing metrics evaluate LATTE and other related LaTeX recognition work cannot reflect human's preference on rendered LaTeX formulae or tables results. These metrics are either very harsh, i.e., pixel level matching, or only consider one-dimensional column-wise matching. To make our evaluation more robust, we add CW-SSIM to investigate the structural similarity of the rendered formulae or tables, but there could be potentially better metrics.

Conclusion

This work proposes LATTE, the first iterative-refinement approach for Latex recognition for both formulae and tables. LATTE uses a generation model to produce LaTeX sources from images; and builds a fault localization model and a refinement model to refine the generated LaTeX source iteratively. To provide effective feedback to the iterative process, this work proposes the `ImageEdit` algorithm, which generates `delta-view` that pinpoints the difference between the ground truth and rendered images. This work also constructs a new LaTeX table recognition dataset TAB2LATEX. With one round of refinement, LATTE outperforms existing techniques by 7.03% of the exact match on LaTeX formulae recognition. Besides, LATTE's formulae and table recognition ability exceed commercial tools by a significant margin, showing great generalizability and effectiveness. In the future, it would be promising to develop better algorithms for pinpointing image differences for tables and formulae to boost the performance of our iterative refinement approach.

Acknowledgments

This research was supported in part by NSF 1901242 and 2006688 and a CFI fund.

References

Amabile, T. M. 1983. *A Theoretical Framework*, 65–96. New York, NY: Springer New York. ISBN 978-1-4612-5533-8.

Anderson, R. H. 1967. Syntax-directed recognition of hand-printed two-dimensional mathematics. In *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, 436–459. New York, NY, USA: Association for Computing Machinery. ISBN 9781450373098.

Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S.; Wang, P.; Lin, J.; Zhou, C.; and Zhou, J. 2023. Qwen-VL: A Frontier Large Vision-Language Model with Versatile Abilities. *arXiv preprint arXiv:2308.12966*.

Blecher, L.; Cucurull, G.; Scialom, T.; and Stojnic, R. 2023. Nougat: Neural Optical Understanding for Academic Documents. arXiv:2308.13418.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. *CoRR*, abs/2005.14165.

Chen, A.; Scheurer, J.; Korbak, T.; Campos, J. A.; Chan, J. S.; Bowman, S. R.; Cho, K.; and Perez, E. 2023a. Improving Code Generation by Training with Natural Language Feedback. arXiv:2303.16749.

Chen, L.; Li, J.; Dong, X.; Zhang, P.; He, C.; Wang, J.; Zhao, F.; and Lin, D. 2023b. ShareGPT4V: Improving Large Multi-Modal Models with Better Captions. *arXiv preprint arXiv:2311.12793*.

Chen, X.; Lin, M.; Schärli, N.; and Zhou, D. 2023c. Teaching Large Language Models to Self-Debug. arXiv:2304.05128.

Dai, W.; Li, J.; Li, D.; Tiong, A. M. H.; Zhao, J.; Wang, W.; Li, B.; Fung, P.; and Hoi, S. 2023. InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning. arXiv:2305.06500.

Deng, Y.; Kanervisto, A.; Ling, J.; and Rush, A. M. 2017. Image-to-markup generation with coarse-to-fine attention. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 980–989. JMLR.org.

Deng, Y.; Rosenberg, D.; and Mann, G. 2019. Challenges in End-to-End Neural Scientific Table Recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 894–901.

Hashmi, K. A.; Liwicki, M.; Stricker, D.; Afzal, M. A.; Afzal, M. A.; and Afzal, M. Z. 2021. Current Status and Performance Analysis of Table Recognition in Document Images with Deep Neural Networks. *CoRR*, abs/2104.14272.

Hossain, S. B.; Jiang, N.; Zhou, Q.; Li, X.; Chiang, W.-H.; Lyu, Y.; Nguyen, H.; and Tripp, O. 2024. A Deep Dive into Large Language Models for Automated Bug Localization and Repair. *Proc. ACM Softw. Eng.*, 1(FSE).

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Jiang, N.; Li, X.; Wang, S.; Zhou, Q.; Hossain, S. B.; Ray, B.; Kumar, V.; Ma, X.; and Deoras, A. 2024. LeDex: Training LLMs to Better Self-Debug and Explain Code. arXiv:2405.18649.

Kayal, P.; Anand, M.; Desai, H.; and Singh, M. 2022. Tables to LaTeX: structure and content extraction from scientific tables. *International Journal on Document Analysis and Recognition (IJDAR)*, 26(2): 121–130.

Könighofer, R.; and Bloem, R. 2011. Automated error localization and correction for imperative programs. In *Proceedings of the International Conference on Formal Methods in Computer-Aided Design*, FMCAD '11, 91–100. Austin, Texas: FMCAD Inc. ISBN 9780983567813.

Kuchta, T.; Lutellier, T.; Wong, E.; Tan, L.; and Cadar, C. 2018a. On the correctness of electronic documents: studying, finding, and localizing inconsistency bugs in PDF readers and files. *Empirical Software Engineering*, 23(6): 3187–3220.

Kuchta, T.; Lutellier, T.; Wong, E.; Tan, L.; and Cadar, C. 2018b. On the correctness of electronic documents: studying, finding, and localizing inconsistency bugs in PDF readers and files. *FSE Journal First, Empirical Software Engineering*, 23(6): 3187–3220.

Lee, K.; Joshi, M.; Turc, I.; Hu, H.; Liu, F.; Eisenschlos, J.; Khandelwal, U.; Shaw, P.; Chang, M.-W.; and Toutanova, K. 2023. Pix2Struct: Screenshot Parsing as Pretraining for Visual Language Understanding. arXiv:2210.03347.

Li, J.; Li, D.; Savarese, S.; and Hoi, S. 2023a. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Li, J.; Li, D.; Xiong, C.; and Hoi, S. 2022. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 12888–12900. PMLR.

Li, M.; Lv, T.; Chen, J.; Cui, L.; Lu, Y.; Florencio, D.; Zhang, C.; Li, Z.; and Wei, F. 2023b. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. In *AAAI 2023*.

Liu, H.; Li, C.; Li, Y.; and Lee, Y. J. 2023a. Improved Baselines with Visual Instruction Tuning.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023b. Visual Instruction Tuning. arXiv:2304.08485.

Long, J.; Hong, Q.; and Yang, L. 2023. An Encoder-Decoder Method with Position-Aware for Printed Mathematical Expression Recognition. In Fink, G. A.; Jain, R.; Kise, K.; and Zanibbi, R., eds., *Document Analysis and Recognition - ICDAR 2023*, 167–181. Cham: Springer Nature Switzerland. ISBN 978-3-031-41676-7.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; Gupta, S.; Majumder, B. P.; Hermann, K.; Welleck, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback. arXiv:2303.17651.

Mehul. 2024. Complex-Wavelet Structural Similarity Index (CW-SSIM). MATLAB Central File Exchange. Retrieved March 7, 2024.

Mirkazemy, A.; Adibi, P.; Ehsani, S. M. S.; Darvishy, A.; and Hutter, H.-P. 2023. Mathematical expression recognition using a new deep neural model. *Neural Networks*, 167: 865–874.

Olausson, T. X.; Inala, J. P.; Wang, C.; Gao, J.; and Solar-Lezama, A. 2023. Demystifying GPT Self-Repair for Code Generation. arXiv:2306.09896.

Pang, N.; Yang, C.; Zhu, X.; Li, J.; and Yin, X.-C. 2021. Global Context-Based Network with Transformer for Image2latex. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 4650–4656.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, 311–318. USA: Association for Computational Linguistics.

Peng, S.; Gao, L.; Yuan, K.; and Tang, Z. 2021. Image to LaTeX with Graph Neural Network for Mathematical Formula Recognition. In *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II*, 648–663. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-030-86330-2.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 8748–8763. PMLR.

Sampat, M. P.; Wang, Z.; Gupta, S.; Bovik, A. C.; and Markey, M. K. 2009. Complex Wavelet Structural Similarity: A New Image Similarity Index. *IEEE Transactions on Image Processing*, 18(11): 2385–2401.

Scheurer, J.; Campos, J. A.; Korbak, T.; Chan, J. S.; Chen, A.; Cho, K.; and Perez, E. 2023. Training Language Models with Language Feedback at Scale. arXiv:2303.16755.

Simon, H. A. 1962. The Architecture of Complexity. *Proceedings of the American Philosophical Society*, 106(6): 467–482.

Wagner, R. A.; and Fischer, M. J. 1974. The String-to-String Correction Problem. *J. ACM*, 21(1): 168–173.

Wang, Z.; and Liu, J.-C. 2021. Translating math formula images to LaTeX sequences using deep neural networks with sequence-level training. *International Journal on Document Analysis and Recognition (IJDAR)*, 24(1): 63–75.

Wei, H.; Kong, L.; Chen, J.; Zhao, L.; Ge, Z.; Yang, J.; Sun, J.; Han, C.; and Zhang, X. 2023. Vary: Scaling up the Vision Vocabulary for Large Vision-Language Models. *arXiv preprint arXiv:2312.06109*.

Yan, Z.; Zhang, X.; Gao, L.; Yuan, K.; and Tang, Z. 2021. ConvMath: A Convolutional Sequence Network for Mathematical Expression Recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 4566–4572. Los Alamitos, CA, USA: IEEE Computer Society.

Zhang, J.; Du, J.; Zhang, S.; Liu, D.; Hu, Y.; Hu, J.; Wei, S.; and Dai, L. 2017. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71: 196–206.

Zhang, W.; Bai, Z.; and Zhu, Y. 2019. An Improved Approach Based on CNN-RNNs for Mathematical Expression Recognition. In *Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing*, ICMSSP '19, 57–61. New York, NY, USA: Association for Computing Machinery. ISBN 9781450371711.

Zhong, X.; ShafieiBavani, E.; and Jimeno Yepes, A. 2020. Image-Based Table Recognition: Data, Model, and Evaluation. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J.-M., eds., *Computer Vision – ECCV 2020*, 564–580. Cham: Springer International Publishing. ISBN 978-3-030-58589-1.