

PP-FormulaNet: Bridging Accuracy and Efficiency in Advanced Formula Recognition

Hongen Liu^{1,2}, Cheng Cui¹, Yuning Du¹, Yi Liu¹, Gang Pan²

¹PaddlePaddle Team, Baidu Inc.

²College of Intelligence and Computing, Tianjin University

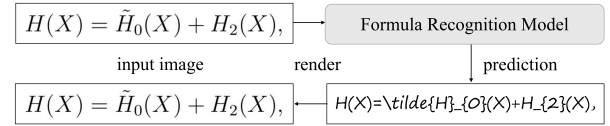
{liuhongen, cuicheng01}@baidu.com, pangang@tju.edu.cn

Abstract

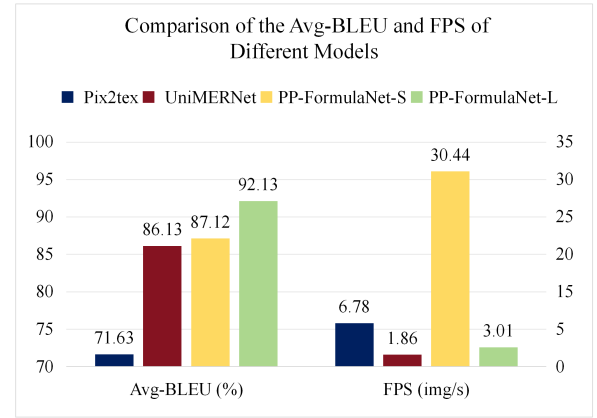
Formula recognition is an important task in document intelligence. It involves converting mathematical expressions from document images into structured symbolic formats that computers can easily work with. LaTeX is the most common format used for this purpose. In this work, we present **PP-FormulaNet**, a state-of-the-art formula recognition model that excels in both accuracy and efficiency. To meet the diverse needs of applications, we have developed two specialized models: **PP-FormulaNet-L**, tailored for high-accuracy scenarios, and **PP-FormulaNet-S**, optimized for high-efficiency contexts. Our extensive evaluations reveal that **PP-FormulaNet-L** attains accuracy levels that surpass those of prominent models such as **UniMERNet** by a significant 6%. Conversely, **PP-FormulaNet-S** operates at speeds that are over 16 times faster. These advancements facilitate seamless integration of **PP-FormulaNet** into a broad spectrum of document processing environments that involve intricate mathematical formulas. Furthermore, we introduce a **Formula Mining System**, which is capable of extracting a vast amount of high-quality formula data. This system further enhances the robustness and applicability of our formula recognition model. Code and models are publicly available at [PaddleOCR](#)¹ and [PaddleX](#)².

1. Introduction

Formulas, as the core knowledge carriers in scientific literature, technical documents, and educational materials, embody the abstract logic and mathematical expressions of human civilization. With the development of large language models, multimodal AI, and intelligent scientific computation, the structured parsing of formula semantics has become a critical breakthrough for constructing scientific knowledge graphs and enhancing AI’s mathemati-



(a) Definition of Formula Recognition



(b) Comparison of the Avg-BLEU and FPS of Different Models

Figure 1. Definition of Formula Recognition, and Comparison of the Avg-BLEU and FPS of Different Models. GPU Inference Time tested on 32G Tesla V100 with batch size of 15.

cal reasoning capabilities. As shown in Figure 1, utilizing formula recognition technology to structurally parse inter-line formulas, inline formulas, and handwritten formulas in contexts such as academic papers and engineering drawings—combined with symbolic semantic understanding and LaTeX generation capabilities—can infuse large model training with precise mathematical prior knowledge. This significantly enhances AI’s problem-solving, theorem derivation, and scientific document generation abilities in the research field. Additionally, formula recognition can be integrated with large models to facilitate the construction of knowledge bases in the scientific domain.

¹<https://github.com/PaddlePaddle/PaddleOCR>

²<https://github.com/PaddlePaddle/PaddleX>

In recent years, thanks to the powerful sequence modeling capabilities of transformers, formula recognition algorithms have developed rapidly. However, most algorithms have a limited vocabulary size of only 300 classes and are focused on the singular scenario of handwritten formula recognition, making them difficult to apply to real-world document parsing scenarios. Additionally, these algorithms tend to prioritize improvements in formula accuracy, often at the expense of inference speed and model size. Therefore, enhancing the generalization ability of formula algorithms in complex scenarios, while optimizing inference speed and model size, holds significant research value.

To address the aforementioned issues, UniMERNet [1] uses large-scale pre-training datasets from arXiv to enhance model generalization for complex formulas. Strategies like reducing model dimensions, replacing attention modules with large-kernel convolutions, and using multi-line bidirectional decoding [2] improve inference speed. However, these methods have limitations. Crawling programs that rely on `\begin{equation}` tags often discard formulas if rendering fails, missing complex structures defined with commands like `\def` and `\newcommand`. Moreover, reducing model size doesn't always enhance inference speed due to the autoregressive nature of models, where multiple iterative steps are required to predict the formula. However, during model lightwighting, there is an inevitable loss of accuracy, making it challenging for the model to accurately predict the sequence termination point. Consequently, although the number of steps for single inference has decreased, the total number of inference steps has actually increased. Balancing both, there is little change in the overall inference speed of the model.

In this paper, we adopt the method of crawling arXiv papers to obtain a large dataset of formula training data. Unlike previous approaches, we develop a more comprehensive formula extraction script that handles user-defined commands and ensured compatibility with various formula identifiers. Based on this program, a dataset of 4 million arXiv formulas is constructed. Additionally, we significantly enhance the CNN network's capability to model formula structures by distilling the weights of the multi-modal large model GOT2.0-OCR [3] onto PP-HGNetV2-B4[4]. Meanwhile, considering the issue that the original pre-trained weights cannot be loaded after model dimensionality reduction, we propose a dimensional interpolation technique to adjust the pre-trained weights. Finally, we introduce a multi-token prediction method that significantly reduces the number of steps in formula sequence inference, thereby improving inference efficiency.

In summary, this work developed an accurate and comprehensive formula mining system and proposed PP-FormulaNet based on techniques such as weight interpolation, knowledge distillation, and multi-token prediction, to

meet the performance requirements of practical document processing scenarios. The main contributions are as follows:

- We develop an innovative formula mining system that considers various potential formula identifiers and restores complex formulas containing user-defined commands. This significantly enhances the completeness and accuracy of formula dataset construction.
- We introduce PP-FormulaNet, a formula recognition model specifically designed for real-world document analysis scenarios. As depicted in Figure 1, **PP-FormulaNet-L** delivers precision levels that are **6%** higher than those of leading models such as UnimerNet, while **PP-FormulaNet-S** functions at speeds that are **16** times faster.
- We design methods such as weight interpolation and model distillation to effectively utilize the formula representations in pre-trained weights during the model lightwighting process.
- We offer a new perspective on accelerating formula inference by employing multi-token prediction techniques, which significantly reduce the number of inference steps for formula sequences.

2. Related Works

Existing formula recognition algorithms can mainly be classified into three categories: traditional machine learning methods, deep learning methods, and large model-driven methods.

2.1. Traditional Machine Learning Methods

Traditional machine learning methods [5–7] primarily rely on handcrafted feature extraction and classifiers. These methods typically consist of three main steps: character grouping, character recognition and relationship analysis. [7] was the first to use a set of replacement rules to model the two-dimensional structure of formula images. [6] improved the accuracy of formula character recognition by using segmental K-means to initialize the Gaussian Mixture Model parameters and introduced features such as the normalized distance of stroke edges to model formula characters. [5] used Hidden Markov Models (HMM) to recognize formula characters and employed stochastic context-free grammar to model the structural relationships between characters. While these methods achieved some success in early formula recognition tasks, their generalization ability and robustness are limited due to the reliance on handcrafted features, especially when dealing with complex formulas or images with significant noise.

2.2. Deep Learning Methods

In recent years, with the rapid development and widespread application of deep learning technology, the field of formula

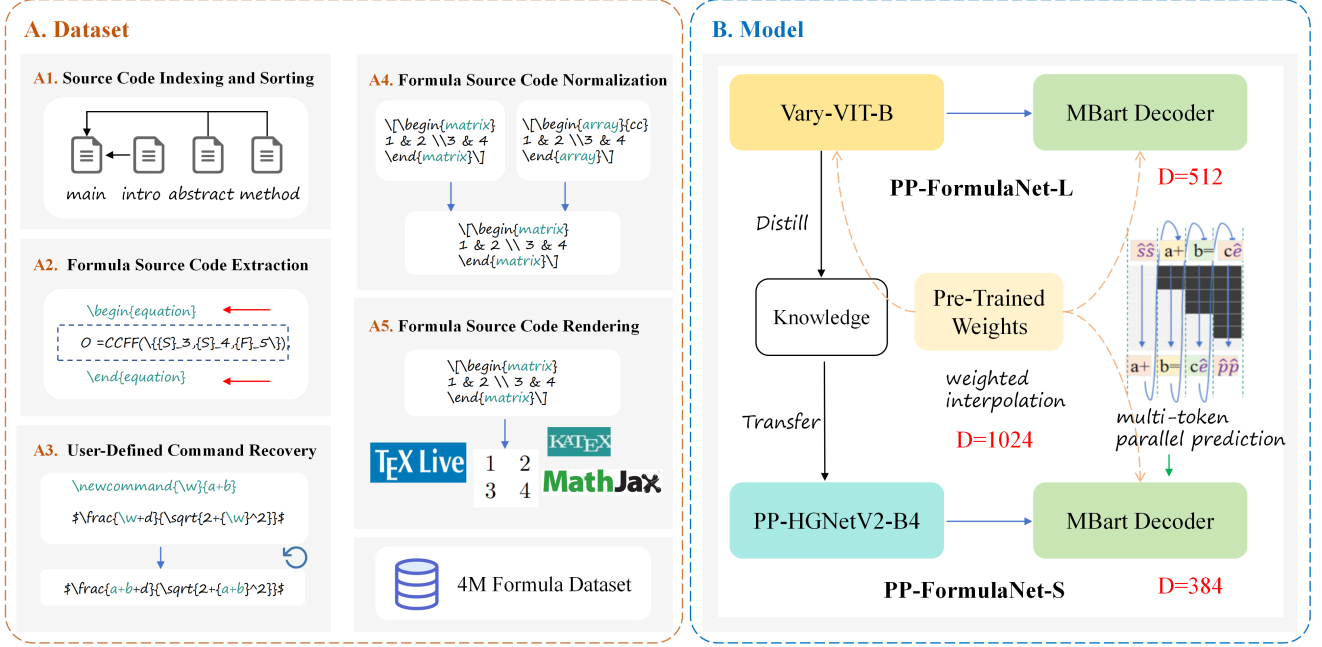


Figure 2. The overall architecture of PP-FormulaNet. Initially, formula-image pairs are extracted from arXiv paper sources through a formula mining system, resulting in a dataset comprising 4 million formulas. PP-FormulaNet consists of two variants: (1) PP-FormulaNet-L: A vision encoder based on the Vary-VIT-B backbone from GOT-2.0, combined with a 512-dimensional MBart Decoder. (2) PP-FormulaNet-S: A vision encoder based on the distilled PP-HGNetV2-B4, combined with a 384-dimensional MBart Decoder. To maintain the formula representation capability of pre-trained weights, weight interpolation is applied to adjust the Vary-VIT-B resolution from 1024×1024 to 768×768 , and the decoder dimensions from 1024 to 512/384. Additionally, multi-token prediction is implemented in PP-FormulaNet-S to enhance inference speed by predicting multiple consecutive tokens in a single forward pass. $\hat{s}$, $\hat{e}$ and $\hat{p}$ represent the start token, end token, and padding token, respectively.

recognition has achieved breakthrough progress, with significant improvements in both accuracy and recognition efficiency. Current deep learning-based methods [2, 8–15] typically employ an encoder-decoder architecture. The encoder encodes the features of the formula image into visual tokens, and subsequently, the decoder applies an attention mechanism to these visual tokens to obtain the decoded formula sequence. The WAP [8] uses convolutional networks to encode formula images and recurrent neural networks to decode the encoded features. It introduces a coverage attention mechanism to address the issues of over-parsing and under-parsing during the formula decoding process. Building on this foundation, the DWAP[9] and WS-WAP[10] enhance visual features by introducing multi-scale attention and a symbol classifier into the encoder, respectively. The CAN [11] addresses the issue of inaccurate attention when decoding complex formulas by introducing a character counting task during formula recognition training. BTTR [12] and ABM [13] utilize the Transformer architecture to replace traditional RNN decoders. By incorporating a bidirectional language modeling mechanism, they significantly enhance the model’s capability to handle long

LaTeX expression sequences. LAST [2] innovatively employs a line-level bidirectional decoder strategy combined with a line position encoding module to achieve parallel bidirectional decoding of multi-line mathematical formulas. CoMER [14] introduces an attention refinement module and divides coverage attention into self-coverage and cross-coverage to address the issue of lacking coverage information in transformer-based formula recognition models. SAN [15] defines a systematic set of syntactic rules to convert LaTeX sequences into structured parse tree representations. By reformulating the sequence prediction task as a traversal process of the parse tree, the model significantly reduces the error rate in formula prediction. Existing methods have primarily been optimized for handwritten formula recognition tasks, often underperforming when dealing with printed formulas. To address the challenges of formula recognition in real document parsing scenarios, researchers have introduced innovative methods such as pix2tex [16], texify [17], and UniMERNet [1]. Notably, UniMERNet [1] has been trained on a large-scale dataset consisting of millions of samples, significantly enhancing the model’s recognition performance across various scenarios.

2.3. Large Model-Driven Methods

Amidst the rapid advancements in large model technologies, researchers are focusing on transferring the pre-training paradigm of large-scale vision-language models to the domain of formula recognition. Donut [18] and Nougat [19] demonstrate significant advantages in understanding documents, and excel in capturing the overall structure and content of entire documents. On the other hand, Vary [20] and GOT-OCR2.0 [3] focus on more fine-grained document analysis and perception tasks, allowing for detailed examination and processing of specific document components. At the same time, general-purpose multimodal large models, exemplified by Qwen2.5-VL [21] and InternVL2.5 [22], show exceptional performance in formula recognition. These models are capable of effectively handling the parsing requirements of most complex formulas. However, due to the large scale of model parameters, the inference efficiency of multimodal models is significantly lower than that of traditional deep learning inference methods. This lack of computational efficiency severely limits their practical applicability in formula-intensive document parsing scenarios.

3. Approach

The overall architecture of PP-FormulaNet is illustrated in Figure 2. A formula extraction system is employed to gather a substantial dataset of 4 million formula source code and image pairs from the source code of arXiv papers. The backbone network Vary-VIT-B from GOT2.0-OCR [3] is utilized to develop PP-FormulaNet-L, serving as the visual encoder, along with an MBart Decoder with a dimension of 512 as the decoder. For PP-FormulaNet-S, the PP-HGNetv2-B4[4], distilled from Vary-VIT-B, is used as the visual encoder, paired with an MBart Decoder with a dimension of 384. Weight interpolation techniques are applied to adjust the Vary-VIT-B weights, initially designed for a resolution of 1024×1024 , down to 768×768 . Additionally, the MBart decoder dimensions are interpolated from the original 1024 to 512 and 384, ensuring maximal retention of pre-trained weights for formula representation. Lastly, a multi-token prediction technique is implemented to optimize the speed of PP-FormulaNet-S, allowing it to predict multiple consecutive tokens simultaneously during inference.

3.1. Formula Mining System

Existing formula extraction methods typically rely on parsing specific markup (such as `\begin{equation}` and `\end{equation}`) within the source code of papers to extract formulas. They then use rendering engines like KaTeX or pdflatex to convert the formula source code into images. However, the source code for formulas in real documents often has complex structures and diverse formats, such as

custom macros, nested commands, or dynamically generated content. This complexity limits traditional methods to extracting formulas in simpler forms, making it difficult to comprehensively cover the complex formula expressions found in real-world scenarios.

To address the aforementioned issues, we design an innovative formula extraction system. This system takes into account various formula identifiers and user-defined commands that may appear in real documents. It primarily consists of the following five steps:

1) Source Code Indexing and Sorting

The order of the source code is crucial for the proper parsing of formulas, particularly because user-defined commands are typically defined at the beginning of the source code. Ensuring the correct order of the source code is essential; otherwise, these custom commands cannot be accurately restored using regular expressions. However, the actual arXiv source code often includes multiple source files with complex inter-file referencing relationships. For instance, a file like `main.tex` might need to reference `abstract.tex` and `Intro.tex`. To address this issue, the main tex file is first identified using identifiers such as `\usepackage`, and then the order of the source code is restored based on the main tex file and reference identifiers.

2) Formula Source Code Extraction

Previous methods typically extract formulas using simple identifiers such as `$$` or `\begin{equation}`, resulting in a limited variety of extracted formula types. To extract a richer and more diverse range of formulas, an in-depth analysis of formula identifiers in arXiv papers is conducted. First, figures and tables are filtered out from the source code since arXiv papers often contain numerous formulas coupled with figures and tables (such as inter-figure and inter-table formulas). These formulas are intermingled with the figure and table source code, making them difficult to separate and potentially interfering with the algorithm’s ability to learn formula features. Next, the list of formula identifiers is expanded to include additional identifiers such as `\begin{align}`, `\begin{align*}`, `\begin{multline}`, `\begin{gather}`, and `\begin{eqnarray}`. Finally, using this expanded list of identifiers, a comprehensive extraction of formula source code is conducted, enabling a more accurate capture of a diverse range of formula structures.

3) User-Defined Command Recovery

In actual arXiv papers, authors often use commands like `\DeclarePairedDelimiter`, `\newcommand`, and `\DeclareMathOperator` to customize formula source code, such as substituting `\eps` for `\varepsilon`. These custom commands do not conform to standard LaTeX syntax, rendering formulas that include them unable to be directly processed. To address this issue, a series of regular expression rules were designed to parse and re-

store these custom commands. For definitions such as `\newcommandx\mathbbfx`, the mapping between custom commands and actual LaTeX code is first stored in a hash table. Regular expressions are then used to match all instances containing the command and identify their starting positions. A stack algorithm is employed to determine their ending positions. Finally, based on the starting and ending positions and the regular expression rules, custom commands are replaced with standard LaTeX code.

4) Formula Source Code Normalization When processing mathematical formula source code scraped from arXiv, a significant amount of ambiguity is encountered, posing notable challenges for subsequent machine learning tasks. A typical example is the different LaTeX representations for visually identical matrix structures. For instance, a matrix can be defined using either the `\begin{array}` or the `\begin{matrix}`. Although they produce the same visual result when rendered, these different representations introduce unnecessary noise at the data level, potentially affecting the model’s training performance. To address this issue, a systematic solution is implemented by employing KaTeX to normalize the source code. KaTeX, a fast and reliable library for rendering LaTeX in web environments, provides a consistent way to parse and render mathematical expressions. By using KaTeX, different but equivalent LaTeX syntax is converted into a standardized form, thus reducing variability and ambiguity in the dataset.

5) Formula Source Code Rendering

After cleaning the formula source code, the local rendering engine pdfflatex is used to render the formulas, converting the formula source code into formula images and thus constructing source code-image pairs. This process involves three steps: first, embedding the formula into a LaTeX template to generate a .tex file. Next, pdfflatex renders the .tex file into a PDF containing the formula. Finally, the fitz tool is utilized to convert the PDF into page images, and OpenCV algorithms are employed to crop out the formula regions.

3.2. Weighted Interpolation

Pre-trained models are essential for enhancing the generalization capabilities of formula recognition models. Experiments demonstrate that models loaded with pre-trained weights achieve significantly higher recognition accuracy than those trained from scratch. However, compressing the formula recognition model necessitates trimming model dimensions (e.g., reducing from 1024 dimensions to 512 or 384), which prevents directly loading the original pre-trained weights. Notably, existing pre-trained models for formula recognition, such as Donut [18] and GOT2.0-OCR [3], typically require training on datasets with millions of samples using dozens of GPUs for hundreds of hours, making it unfeasible to re-pretrain the trimmed models in prac-

tical scenarios. To overcome this challenge, an innovative weight interpolation method is proposed, which adaptively adjusts the original pre-trained weights, enabling the trimmed model to effectively load them. This approach maintains the model’s lightweight nature while fully leveraging the benefits of pre-trained models.

To address the challenge of adapting pre-trained models to reduced dimensions, an attention module typically composed of linear layers and normalization layers can be optimized. The key assumption here is that adjacent dimensions of attention weights in the decoder encode similar semantic information. This assumption permits the mixing of dimensional features through interpolation, allowing the reduced-dimension model to still effectively load pre-trained weights. For the linear layers and normalization layers, the calculation method for the weights after dimension reduction is as follows:

$$\begin{cases} F_{\text{Linear}}^D = \varphi(F_{\text{Linear}}) \\ F_{\text{norm}}^D = \tau_2(\varphi(\tau_1(F_{\text{norm}}))) \end{cases} \quad (1)$$

Here, F_{Linear}^D and F_{norm}^D respectively represent the original pre-trained weights of the linear layer and the normalization layer. F_{Linear} and F_{norm} denotes the interpolated pre-trained weights of the linear layer and the normalization layer. φ denotes the nearest neighbor interpolation, τ_1 represents the unsqueeze operation used to reshape the dimensions of the normalization layer from $\mathbb{R}^C$ to $\mathbb{R}^{C \times 1}$, and τ_2 represents the squeeze operation used to reshape the dimensions of the normalization layer from $\mathbb{R}^{C \times 1}$ to $\mathbb{R}^C$.

3.3. Knowledge Distillation for PP-FormulaNet-S Backbone

For PP-FormulaNet-S, its backbone network faces challenges in adapting to formula recognition tasks due to limited parameter capacity and absence of domain-specific knowledge in mathematical notation. To address this, we employ knowledge distillation to transfer capabilities from Vary-VIT-B. The teacher network parameters remain frozen while training PP-HGNetV2-B4, leveraging feature-level supervision through a fully-connected layer.

Let $\mathbf{F}_{\text{Vary-VIT-B}} \in \mathbb{R}^{B \times D}$ and $\mathbf{F}_{\text{PP-HGNetV2-B4}} \in \mathbb{R}^{B \times P}$ denote the feature tensors from teacher and student networks respectively, where D and P represent their corresponding feature dimensions, B denotes the Batch size. To bridge the dimensional discrepancy ($D \neq P$), we introduce a learnable linear projection $\varphi: \mathbb{R}^P \rightarrow \mathbb{R}^D$. The distillation loss is formulated as:

$$\mathcal{L}_{\text{Distill}} = \frac{1}{B} \sum_{i=1}^B \left\| \mathbf{F}_{\text{Vary-VIT-B}}^{(i)} - \varphi(\mathbf{F}_{\text{PP-HGNetV2-B4}}^{(i)}) \right\|_2^2 \quad (2)$$

The distillation framework was trained on a diverse cor-

Table 1. The Recognition Performance of Different Methods on the UniMERNet-1M and arXiv-4M Test Sets.

Model	UniMERNet-1M		arXiv-4M			Avg- BLEU↑	GPU inference time (ms) (batch=1)	GPU inference time (ms) (batch=15)
	SPE- BLEU↑	CPE- BLEU↑	Easy- BLEU↑	Middle- BLEU↑	Hard- BLEU↑			
Pix2tex[16]	0.8780	0.5670	0.8151	0.7546	0.5746	0.7163	<u>1244.61</u>	<u>147.38</u>
UniMERNet [1]	0.9187	0.9251	0.8659	0.8229	0.7741	0.8613	2266.96	536.75
PP-FormulaNet-S	0.8694	0.8071	<u>0.9295</u>	<u>0.9113</u>	<u>0.8392</u>	<u>0.8712</u>	202.25	32.85
PP-FormulaNet-L	<u>0.9055</u>	<u>0.9207</u>	0.9392	0.9273	0.9142	0.9213	1976.52	332.12

pus containing 500000 document samples spanning five domains:

- Mathematical formulas (including equation derivations and symbolic notations)
- Financial documents (reports and balance sheets)
- Scientific literature (arxiv papers in STEM fields)
- Academic dissertations (with complex layout structures)
- Tabular data (statistical reports and spreadsheets)

Training was conducted at 768×768 resolution for 50 epochs using AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$). The distilled PP-HGNetV2-B4 achieves effective feature extraction capabilities with only 15.6 M parameters.

This compact model learns transferable representations for OCR tasks, elevating the formula recognition BLEU score from 80.87 to 84.32 (+3.43pp) compared to ImageNet-pretrained baselines.

3.4. Multi-Token Parallel Prediction

Current mainstream formula recognition algorithms typically employ an autoregressive generation architecture, which involves predicting the formula sequence token by token. This approach can significantly reduce inference efficiency, particularly when dealing with complex formulas that exceed 1024 tokens in length. While existing studies, such as the LAST method [2], introduced multi-line parallel inference strategies to enhance efficiency, these strategies are generally confined to scenarios involving multi-line formulas and still encounter challenges with complex structures like matrices and braces. To address these limitations, an innovative multi-token parallel prediction method is proposed, designed to improve the inference efficiency of complex formulas.

As shown in the Fig. 2, the parallel causal mask is a key component of the multi-token prediction technique, and its mathematical expression is as follows:

$$\mathbf{M}_{ij} = \begin{cases} 0, & \text{if } \lfloor \frac{i}{step} \rfloor \geq \lfloor \frac{j}{step} \rfloor \\ -\infty, & \text{if } \lfloor \frac{i}{step} \rfloor < \lfloor \frac{j}{step} \rfloor \end{cases} \quad (3)$$

Here, $\mathbf{M}_{ij}$ represents the element in the i -th row and j -th column of the parallel causal mask, and $step$ indicates the

number of tokens predicted at each step. $\lfloor \cdot \rfloor$ represents the floor operation.

By introducing the parallel causal mask mechanism, the decoder is equipped to perform multi-step predictions during training: it begins by predicting characters from position $step + 1$ to $2step$, using the initial step characters as a basis. Next, it predicts characters from position $2step + 1$ to $3step$, relying on the first $2step$ characters, and continues this process until the maximum length of the current data batch is reached. During the inference phase, the initial characters are replicated $step$ times as input. The model then predicts the next sequence, also of length $step$, based on this initial sequence, thus achieving multi-token parallel prediction.

4. Experiment

4.1. Datasets and Evaluation Metrics

In this paper, the model is trained on the UniMERNet-1M dataset [1] and a self-constructed arXiv-4M dataset, totaling 5 million entries. The proposed method is evaluated through comparative experiments across multiple test sets. Specifically, the test sets include the simple and complex formula test sets from UniMERNet [1], as well as the simple, medium, and complex formula test sets from arXiv-4M. In line with most formula recognition methods, the BLEU-Score [23] on the validation set is used as the evaluation metric.

4.2. Implementation Details

During the training phase, a data augmentation strategy consistent with UniMERNet [1] is employed, which specifically includes operations such as dilation, erosion, weather noise, random affine transformations, Gaussian noise, and random brightness and contrast adjustments. For optimization, the AdamW optimizer is utilized with parameters set as $\beta_1=0.9$, $\beta_2=0.999$, and a weight decay of 0.05. The batch size per GPU is set to 6, and the total number of training epochs is 10. To ensure that the network can progressively learn the features of formulas, we employ a WarmUp strategy with a learning rate of $1e-5$ for the first 5,000 steps to preheat the training process, followed by a cosine annealing strategy with a learning rate of $1e-4$ to dynamically adjust the learning rate.

Table 2. The Impact of Weight Interpolation on Recognition Performance.

Backbone	Dim	Weight Interpolation	CPE-BLEU $\uparrow$
Donut-Swin[1]	1024		0.8968
	512		0.7970
		✓	0.8445
Vary-VIT-B [3]	512	✓	0.9148

4.3. Comparison with State-of-the-Arts

Comparative experiments were conducted with state-of-the-art (SOTA) methods on the test sets of the arXiv-4M dataset and UniMERNet-1M, as detailed in Table 1. In terms of accuracy, PP-FormulaNet-S and PP-FormulaNet-L achieve an average BLEU score (Avg-BLEU) improvement of 0.99% and 6%, respectively, over the current open-source SOTA method, UniMERNet³, across five test datasets, demonstrating significant performance gains. Regarding inference speed, under batch=15, PP-FormulaNet-S achieves a 16x faster GPU inference speed compared to UniMERNet, while PP-FormulaNet-L reduces the inference time by 61%. These results indicate that PP-FormulaNet not only surpasses existing methods in accuracy but also achieves substantial improvements in inference efficiency. This is particularly evident in batch processing scenarios, where its performance advantages are even more pronounced.

4.4. Ablation Study

Ablation experiments were conducted on the UniMERNet-1M dataset to verify the impact of knowledge distillation, weight interpolation, and multi-token parallel prediction on formula recognition performance.

Impact of weight interpolation To enable the dimension-pruned model to continue utilizing pre-trained weights, a weight interpolation technique is proposed. Experimental results in Table 2 demonstrate that this method effectively preserves the performance advantages of the pre-trained model. Specifically, when the recognition network is UniMERNet, using pre-trained weights results in a 4.75% performance improvement on the complex formula dataset, fully validating the effectiveness of the weight interpolation technique. However, despite the substantial performance improvement achieved by the weight interpolation technique, a 5.23% performance gap remains compared to the original UniMERNet model with a dimension of 1024. To further optimize model performance, replacement experiments on the backbone network were conducted. The results indicate that when Vary-VIT-B is used as the backbone network, the model’s recognition performance reaches

³We are comparing the original version of UniMERNet. We will provide a more detailed comparison with the new version in the future.

Table 3. The Impact of Knowledge Distillation on Recognition Performance Across Different Backbone Networks.

Backbone	Knowledge Distillation	SPE-BLEU $\uparrow$	CPE-BLEU $\uparrow$	Params
ResNet50[24]	✓	0.7871 0.8107	0.5737 0.6216	23.6 M
PP-HGNetV2-B1[4]	✓	0.5963 0.7506	0.2703 0.4851	2.2M
PP-HGNetV2-B4[4]	✓	0.8087 0.8432	0.6357 0.6874	15.6M

0.9148, representing a 1.8% improvement compared to the original UniMERNet with a dimension of 1024. Based on this outcome, Vary-VIT-B is ultimately selected as the backbone network for PP-FormulaNet-L.

Impact of knowledge distillation

Despite the Vary-VIT-B backbone network achieving outstanding recognition performance, its large number of parameters and the inherent $O(n^2)$ computational complexity of Transformers lead to longer inference times. To address this issue, knowledge distillation experiments were conducted on Vary-VIT-B using various CNN-based backbone networks. The experimental results in Table 3 demonstrate that knowledge distillation significantly enhances model performance across different backbone networks. Specifically, when the backbone network is ResNet50, knowledge distillation results in BLEU-Score improvements of 2.36% and 4.79% on simple and complex formula datasets, respectively. When the backbone network is PP-HGNetV2-B1, the effect of knowledge distillation is even more remarkable, achieving BLEU-Score improvements of 15.43% and 21.48% on simple and complex formula datasets, respectively. Similarly, when the backbone network is PP-HGNetV2-B4, knowledge distillation also performs excellently, with BLEU-Score improvements of 3.45% and 5.17% on simple and complex formula datasets, respectively. These results fully demonstrate the effectiveness and versatility of knowledge distillation across different backbone networks. Additionally, given that PP-HGNetV2-B4 achieved the optimal BLEU scores in the recognition task, it is selected as the backbone network for PP-FormulaNet-S.

Impact of multi-token parallel prediction To accelerate the inference of autoregressive models, a multi-token parallel prediction technique is designed, enabling the decoder to predict multiple tokens in a single step. As shown in Table 4, when the parallel steps are set to 2, 3, 4, and 5, the model’s inference speed increases by 2.05 times, 2.86 times, 3.77 times, and 4.63 times, respectively. However, as the number of parallel steps increases, the model’s accuracy declines more rapidly: when the parallel steps increase from 1 to 2, accuracy decreases by 1.26%, when the paral-

Table 4. The Impact of Multi-token Parallel Prediction on Recognition Performance.

Backbone	Parallel Step	CPE-BLEU $\uparrow$	GPU inference time (ms)(CPE) (batch=1)
Vary-VIT-B [3]	1	0.9092	2779.76
	2	0.8876	1353.93
	3	0.8689	969.30
	4	0.8403	735.58
	5	0.8048	600.08

l steps increase from 3 to 4, accuracy decreases by 2.86%, and when it increases from 4 to 5, the decline in accuracy reaches 3.55%. Considering the trade-off between accuracy and speed, a parallel step count of 3 is ultimately chosen for the PP-FormulaNet-S model.

5. Conclusion and Discussion

In this paper, we propose an advanced formula recognition model, PP-FormulaNet, aimed at addressing the inefficiencies and inaccuracies of existing methods in formula recognition within real-world document parsing scenarios. Firstly, a more comprehensive formula mining system is designed, significantly enhancing the coverage and diversity of formula data by expanding the list of formula identifiers and supporting the recovery of user-defined commands. Secondly, formula interpolation and model distillation techniques are innovatively combined to efficiently transfer formula representations from pre-trained weights to PP-FormulaNet, thereby enhancing the model’s generalization ability. Finally, a multi-token prediction technique is introduced to effectively accelerate the inference process of autoregressive models. Experimental results on the UniMERNet-1M and the self-constructed arXiv-4M datasets demonstrate that PP-FormulaNet achieves significant improvements in both recognition accuracy and inference efficiency. However, while the proposed multi-token prediction technique improves inference speed, it also results in some accuracy loss. Reducing this accuracy loss in multi-token prediction is a key focus for future research.

References

- [1] Bin Wang, Zhuangcheng Gu, Guang Liang, Bo Zhang Chao Xu, Botian Shi, and Conghui He. Unimernet: A universal network for real-world mathematical expression recognition. *arXiv preprint arXiv:2404.15254v1*, 2024. 2, 3, 6, 7, 1, 4
- [2] Wentao Yang, Zhe Li, Dezhi Peng, Lianwen Jin, Mengchao He, and Cong Yao. Read ten lines at one glance: Line-aware semi-autoregressive transformer for multi-line handwritten mathematical expression recognition. *ACM MM*, 2023. 2, 3, 6
- [3] Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, Chunrui Han, and Xiangyu Zhang. General ocr theory: Towards ocr-2.0 via a unified end-to-end model. *arXiv preprint arXiv:2409.01704*, 2024. 2, 4, 5, 7, 8
- [4] Baidu PaddlePaddle Vision Team. Pp-hgnetv2. https://github.com/PaddlePaddle/PaddleClas/blob/release/2.6/docs/zh_CN/models/ImageNet1k/PP-HGNetV2.md, 2023. 2, 4, 7
- [5] Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models. *Pattern Recognition Letters*, 35:58–67, 2014. 2
- [6] Lei Hu and Richard Zanibbi. Hmm-based recognition of on-line handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature. In *ICDAR*, pages 457–462, 2011. 2
- [7] Robert H. Anderson. 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*, page 436–459, 1967. 2
- [8] Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jinshui Hu, Si Wei, and Lirong Dai. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71:196–206, 2017. 3
- [9] Jianshu Zhang, Jun Du, and Lirong Dai. Multi-scale attention with dense encoder for handwritten mathematical expression recognition. *ICPR*, pages 2245–2250, 2018. 3
- [10] Thanh-Nghia Truong, Cuong Tuan Nguyen, Khanh Minh Phan, and M. Nakagawa. Improvement of end-to-end offline handwritten mathematical expression recognition by weakly supervised learning. *ICFHR*, pages 181–186, 2020. 3
- [11] Bohan Li, Ye Yuan, Dingkan Liang, Xiao Liu, Zhilong Ji, Jinfeng Bai, Wenyu Liu, and Xiang Bai. When counting meets hmer: Counting-aware network for handwritten mathematical expression recognition. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *ECCV*, pages 197–214, 2022. 3
- [12] Wenqi Zhao, Liangcai Gao, Zuoyu Yan, Shuai Peng, Lin Du, and Ziyin Zhang. Handwritten mathematical expression recognition with bidirectionally trained transformer. In *ICDAR*, page 570–584, 2021. 3
- [13] Xiaohang Bian, Bo Qin, Xiaozhe Xin, Jianwu Li, Xuefeng Su, and Yanfeng Wang. Handwritten mathematical expression recognition via attention aggregation based bidirectional mutual learning. In *AAAI*, 2021. 3
- [14] Wenqi Zhao and Liangcai Gao. Comer: Modeling coverage for transformer-based handwritten mathematical expression recognition. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *ECCV*, pages 392–408, 2022. 3
- [15] Ye Yuan, Xiao Liu, Wondimu Dikubab, Hui Liu, Zhilong Ji, Zhongqin Wu, and Xiang Bai. Syntax-aware network for handwritten mathematical expression recognition. In *CVPR*, pages 4543–4552, 2022. 3

- [16] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. Image-to-markup generation with coarse-to-fine attention. In *ICML, ICML'17*, page 980–989, 2017. 3, 6, 1, 2, 4
- [17] V. Paruchuri. Texify. <https://github.com/VikParuchuri/texify>, 2023. 3
- [18] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer. In *ECCV*, page 498–517, 2022. 4, 5
- [19] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*, 2023. 4
- [20] Haoran Wei, Lingyu Kong, Jinyue Chen, Liang Zhao, Zheng Ge, Jinrong Yang, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. Vary: Scaling up the vision vocabulary for large vision-language model. In *ECCV*, pages 408–424, 2025. 4
- [21] Qwen Team. Qwen2.5-vl. <https://qwenlm.github.io/blog/qwen2.5-vl/>, January 2025. 4, 1, 2, 3
- [22] Zhe Chen, Weiyun Wang, Yue Cao, and et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024. 4
- [23] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL, ACL '02*, page 311–318, 2002. 6
- [24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 7

Appendix of “PP-FormulaNet: Bridging Accuracy and Efficiency in Advanced Formula Recognition”

1. The Visualization Results of Different Formula Recognition Methods on the UniMERNet-1M and arXiv-4M Test Sets

1.1. Simple Formula

$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$	$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$
--	--

(a) Pix2tex [16] (25 M)

$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$	$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$
--	--

(b) UniMERNet [1] (392 M)

$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$	$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$
--	--

(c) GPT-4o (Unknown)

$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$	$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$
--	--

(d) Qwen2.5-VL[21] (72B)

$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$	$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$
--	--

(e) PP-FormulaNet-S (58 M)

$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$	$f(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x)}{h}.$
--	--

(f) PP-FormulaNet-L (179 M)

Figure A. Visualization of recognition results from different methods on simple formulas. The left image is the test image, and the right image is the prediction result. The blank on the right indicates that there is a syntax error in the generated formula, preventing it from rendering.

1.2. Middle Formula

$$(\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases}$$

(a) Pix2tex [16] (25 M)

$$(\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases} \quad (\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} \mathcal{N}(\beta_{i,1} | 0, 1)\mathcal{N}(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ \mathcal{N}(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases}$$

(b) UniMERNet [1] (392 M)

$$(\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases} \quad (\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases}$$

(c) GPT-4o (Unknown)

$$(\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases} \quad (\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases}$$

(d) Qwen2.5-VL-72B[21] (72B)

$$(\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases} \quad (\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases}$$

(e) PP-FormulaNet-S (58 M)

$$(\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases} \quad (\beta_{i,1}, \beta_{i,2})' \sim \begin{cases} N(\beta_{i,1} | 0, 1)N(\beta_{i,2} | 0, 1) & \xi_i = 0, \\ N(\beta_{i,1} | 0, 1)\delta_{\beta_{i,1}}(\beta_{i,2}) & \xi_i = 1, \end{cases}$$

(f) PP-FormulaNet-L (179 M)

Figure B. Visualization of recognition results from different methods on middle formulas. The left image is the test image, and the right image is the prediction result. The blank on the right indicates that there is a syntax error in the generated formula, preventing it from rendering.

1.3. Hard Formula

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q-1)} & -\frac{1}{(4q-1)} & \frac{8q}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{4}{(4q-1)} & \frac{4(18q-13)}{(4q-3)(4q-1)} \\ & & \frac{4(2q-1)}{(4q-3)(4q-1)} & -\frac{4(2q-1)(9q-1)}{q(4q-3)(4q-1)} \\ & & & \frac{12(q-1)(2q-1)}{q(4q-5)(4q-3)} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

(a) Pix2tex [16] (25 M)

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q+1)} & -\frac{1}{(4q+1)} & \frac{8q}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{4}{(4q-1)} & \frac{4(18q-13)}{(4q-3)(4q-1)} \\ \frac{4(2q-1)}{(4q-3)(4q-1)} & \frac{4(2q-1)}{(4q-3)(4q-1)} & \frac{q(4q-3)(4q-1)}{12(q-1)(2q-1)} & \frac{q(4q-5)(4q-3)}{q(4q-5)(4q-3)} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

$$\begin{pmatrix} V_0 \\ V_1 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q+1)} & -\frac{1}{(4q+1)} & \frac{8q}{16q^2-1} & -\frac{4(q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{4}{(4q-1)} & \frac{4(18q-13)}{(4q-3)(4q-1)} \\ \frac{4(2q-1)}{(4q-3)(4q-1)} & -\frac{4(2q-1)(9q-1)}{q(4q-3)(4q-1)} & \frac{4(q-3)(4q-1)}{q(4q-5)(4q-3)} & \frac{12(q-1)(2q-1)}{q(4q-5)(4q-3)} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \end{pmatrix}$$

(b) UniMERNet [1] (392 M)

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q+1)} & -\frac{1}{(4q+1)} & \frac{8q}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{4}{(4q-1)} & \frac{4(18q-13)}{(4q-3)(4q-1)} \\ \frac{4(2q-1)}{(4q-3)(4q-1)} & \frac{4(2q-1)(9q-1)}{(4q-3)(4q-1)} & -\frac{q(4q-3)(4q-1)}{12q(q-1)(2q-1)} & -\frac{q(4q-5)(4q-3)}{12q(q-1)(2q-1)} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q+1)} & -\frac{1}{(4q+1)} & \frac{8q}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{4}{4(18q-13)} & \frac{4(18q-13)}{4(18q-13)} \\ & & \frac{1}{(4q-3)(4q-1)} & \frac{q(4q-3)(4q-1)}{4(2q-1)(4q-1)} \\ & & & \frac{q(q-5)(4q-3)}{12(4q-1)(2q-1)} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

(c) GPT-4o (Unknown)

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(q-1)} & -\frac{1}{(q-1)} & \frac{8q}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(q-1)} & -\frac{4}{(4q-1)} & \frac{4(18q-13)}{(16q^2-1)} \\ & & \frac{4(2q-1)}{(4q-3)(4q-1)} & \frac{4q-3}{4(2q-1)} \frac{(4q-1)}{(9q-1)} \\ & & & \frac{12(q-1)(2q-1)}{q(4q-3)(4q-1)} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q+1)} & -\frac{4(1)}{(4q+1)} & \frac{8q^2-1}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{4}{4(2q-3)(4q-1)} & \frac{4(18q^2-13)}{12(q-1)(2q-1)} \\ -\frac{1}{(4q+1)} & -\frac{4}{4(2q-1)} & \frac{4(2q-1)(9q-1)}{4(4q-3)(4q-1)} & \frac{q(q-5)(4q-3)}{12(q-1)(2q-1)} \\ \frac{8q^2-1}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} & -\frac{4(4q-3)(4q-1)}{4(18q^2-13)} & \frac{q(q-5)(4q-3)}{12(q-1)(2q-1)} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

(d) Qwen2.5-VL[21] (72B)

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q+1)} & -\frac{1}{(4q+1)} & \frac{8q}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{4}{(4q-1)} & \frac{4(18q-13)}{(4q-3)(4q-1)} \\ & & \frac{4(2q-1)}{(4q-3)(4q-1)} & \frac{4(2q-1)(9q-1)}{(4q-3)(4q-1)} \\ & & & \frac{12(q-1)(2q-1)}{q(4q-5)(4q-3)} \end{pmatrix}$$

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q+1)} & -\frac{1}{(4q+1)} & \frac{8q}{16q^2-1} & -\frac{4(q+1)}{(4q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{1}{(4q-1)} & \frac{4(q-1)}{(4q^2-1)} \\ 0 & \frac{(2q-1)}{(4q-1)} & \frac{(4q-1)}{(4q-1)} & -\frac{1}{(4q-1)(4q-1)} \\ \frac{(4q-1)(4q-1)}{4(q-1)(4q-3)} & \frac{q}{(4q-1)(4q-3)} & & \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

(e) PP-FormulaNet-S (58 M)

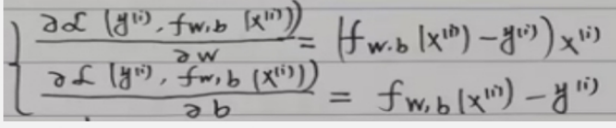
$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q+1)} & -\frac{1}{(4q+1)} & \frac{8q}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{4}{(4q-1)} & \frac{4(18q-13)}{(4q-3)(4q-1)} \\ & & \frac{4(2q-1)}{(4q-3)(4q-1)} & \frac{4(2q-1)(9q-1)}{q(4q-3)(4q-1)} \\ & & & \frac{12(q-1)(2q-1)}{q(4q-5)(4q-3)} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{(2q+1)^2}{4\pi q} \begin{pmatrix} \frac{1}{(4q+1)} & -\frac{1}{(4q+1)} & \frac{8q}{16q^2-1} & -\frac{4(6q+1)}{(16q^2-1)} \\ 0 & \frac{1}{(4q-1)} & -\frac{4}{(4q-1)} & \frac{4(18q-13)}{(4q-3)(4q-1)} \\ & & \frac{4(2q-1)}{(4q-3)(4q-1)} & -\frac{4(2q-1)(9q-1)}{q(4q-3)(4q-1)} \\ & & & \frac{12(q-1)(2q-1)}{q(4q-5)(4q-3)} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

(f) PP-FormulaNet-L (179 M)

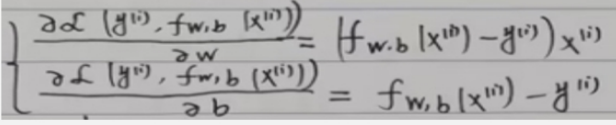
Figure C. Visualization of recognition results from different methods on hard formulas. The left image is the test image, and the right image is the prediction result. The blank on the right indicates that there is a syntax error in the generated formula, preventing it from rendering.

1.4. Handwritten Formula



$$\begin{cases} \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial w} = (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial b} = f_{w,b}(x^{(i)}) - y^{(i)} \end{cases}$$

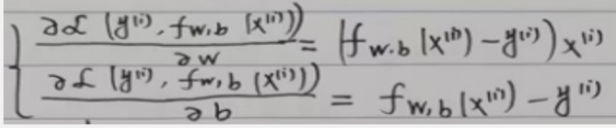
(a) Pix2tex [16] (25 M)



$$\begin{cases} \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial w} = (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial b} = f_{w,b}(x^{(i)}) - y^{(i)} \end{cases}$$

$$\frac{3d((y^{(i)} + w, b(x^{(i)})))}{3w} = (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

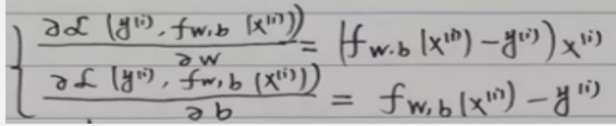
(b) UniMERNet [1] (392 M)



$$\begin{cases} \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial w} = (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial b} = f_{w,b}(x^{(i)}) - y^{(i)} \end{cases}$$

$$\begin{cases} \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial w} = (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial b} = f_{w,b}(x^{(i)}) - y^{(i)} \end{cases}$$

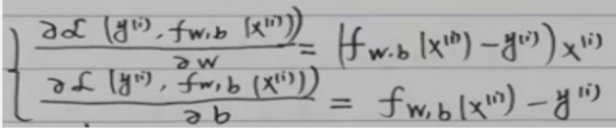
(c) GPT-4o (Unknown)



$$\begin{cases} \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial w} = (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial b} = f_{w,b}(x^{(i)}) - y^{(i)} \end{cases}$$

$$\begin{aligned} \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial w} &= (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial b} &= f_{w,b}(x^{(i)}) - y^{(i)} \end{aligned}$$

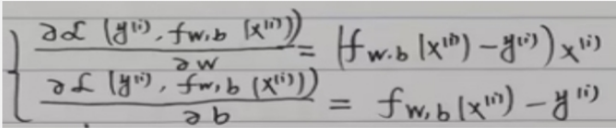
(d) Qwen2.5-VL[21] (72B)



$$\begin{cases} \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial w} = (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial b} = f_{w,b}(x^{(i)}) - y^{(i)} \end{cases}$$

$$\frac{\frac{2}{2}((y^{(11)} - w^{(11)}(x^{(11)})))}{2\psi} = (\frac{frac{2}{2}y \cdot dot{frac{2}{2}y \cdot dot{(11)}}}{frac{2051}{(11)} - y \frac{frac{2020}{psi}}{psi})$$

(e) PP-FormulaNet-S (58 M)



$$\begin{cases} \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial w} = (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial b} = f_{w,b}(x^{(i)}) - y^{(i)} \end{cases}$$

$$\begin{aligned} \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial w} &= (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \frac{\partial \mathcal{L}(y^{(i)}, f_{w,b}(x^{(i)}))}{\partial b} &= f_{w,b}(x^{(i)}) - y^{(i)} \end{aligned}$$

(f) PP-FormulaNet-L (179 M)

Figure D. Visualization of recognition results from different methods on handwritten formulas. The left image is the test image, and the right image is the prediction result. The blank on the right indicates that there is a syntax error in the generated formula, preventing it from rendering.