ISSN: 2633-4828 Vol. 8 No.1, January, 2026

International Journal of Applied Engineering & Technology

MATHEMATICAL EXPRESSION RECOGNITION USING DEEP ENCODER-DECODER ARCHITECTURE WITH RELATIVE POSITIONAL ENCODING

R.Sridevi[1], G.Sudheer[2*] and D.Lalitha Bhaskari[3]
Department of IT, GVP College of Engieering for Women, Visakhapatnam,India
[2]Department of Mathematics, GVP College of Engieering, Visakhapatnam, India
[3]Department of Computer Science & Systems Engineering, Andhra University, Visakhapatnam, India
[*]sudhwave@gmail.com

Received: 20 Dec 2025, Revised: 30 Dec 2025, Acceptance: 3 Jan 2026, Publication: 25 Jan 2026

ABSTRACT

Mathematical expression recognition (MER) remains a challenging task in document understanding and digitization. This paper presents a transformer-based encoder-decoder architecture for converting printed mathematical expression images to LaTeX markup. Our approach employs a ResNet-18 encoder for visual feature extraction and an 8-layer transformer decoder with relative positional encoding for sequential LaTeX generation. We evaluate our model on the Im2LaTeX-100k dataset, achieving 94.89% accuracy (1-CER) and 93.49% BLEU-4 score using beam search decoding with contextual re-ranking. The system demonstrates robust character error rate (CER) reduction through architectural innovations including label smoothing, gradient clipping, and mixed precision training.

Keywords: *Mathematical Expression, Character recognition, Transformer, LaTeX Markup, Beam Search*

I. INTRODUCTION

The automatic recognition of mathematical expressions from images represents a critical challenge in document digitization, particularly for scientific literature, educational materials, and historical mathematical texts. Unlike standard optical character recognition (OCR), mathematical expression recognition must handle complex two-dimensional structures including fractions, radicals, subscripts, superscripts, matrices, and nested expressions. The hierarchical and spatial nature of mathematical notation makes this task fundamentally more challenging than sequential text recognition.

The advent of deep learning has revolutionized image-to-markup translation tasks, with encoder-decoder architectures demonstrating remarkable success in various sequence-to-sequence problems [1, 2]. However, mathematical expressions pose unique challenges: they require understanding spatial relationships between symbols, maintaining structural consistency, and generating syntactically correct LaTeX markup that preserves the semantic meaning of the visual input. Recent advances in transformer architectures have further improved recognition performance, with methods like PosFormer [3] and TAMER [4] achieving state-of-the-art results by incorporating position-aware representations and tree-structured decoding mechanisms.

The basic problem in MER lies in generating the corresponding LaTex sequence that accurately represents the notation of the mathematical expression contained in the input image. This research makes the following key contributions:

1. Implementation of a ResNet-18 [7] based visual encoder combined with an 8-layer transformer decoder featuring relative positional encoding [8] for improved sequential generation capabilities.
2. Integration of a contextual pipeline that leverages spatial relationship detection between symbols to rerank beam search hypotheses, improving structural consistency.
3. Application of modern training techniques including label smoothing ($\epsilon = 0.1$) [10], gradient clipping, mixed precision training, and OneCycleLR scheduling [9] to achieve stable convergence.

4. Achievement of 94.89% accuracy (1-CER) and 93.49% BLEU-4 score on the Im2LaTeX-100k validation set, representing state-of-the-art results.
5. Comparison of greedy decoding versus beam search ($k = 5$) with length penalty normalization, demonstrating the trade-offs between inference speed and generation quality.

The remainder of this paper is organized as follows: Section 2 presents the theoretical foundations including problem formalization and architectural components. Section 3 describes the methodology covering model implementation, training procedures, and evaluation metrics. Section 4 presents numerical results and detailed discussion. Section 5 concludes with summary and future directions.

2. THEORY

2.1 Problem Formalization

Given an input image $\mathbf{I} \in \mathbb{R}^{H\times W\times C}$ containing a mathematical expression, the objective is to generate the corresponding LaTeX sequence $\mathbf{Y} = \{y_1, y_2, \dots, y_T\}$ that accurately represents the mathematical notation. This problem can be formulated as:

$$\boldsymbol{Y}^* = \underset{\boldsymbol{Y}}{argmax} P(\boldsymbol{Y}|\boldsymbol{I}) \qquad (1)$$

where $\mathbf{Y}^*$ denotes the most likely LaTeX sequence given the input image. The challenge lies in learning the conditional probability distribution $P(\mathbf{Y}|\mathbf{I})$ that captures both the visual appearance of mathematical symbols and the grammatical structure of LaTeX markup. The problem can be formulated as follows:

Let $\mathbf{I} \in \mathbb{R}^{H\times W\times C}$ represent an input image of height H, width W, and C channels ($C = 1$ for grayscale). Let $\mathcal{V} = \{v_1, v_2, \dots, v_{|\mathcal{V}|}\}$ denote the vocabulary of LaTeX tokens. The goal is to learn a model parameterized by $\boldsymbol{\theta}$ that estimates:

$$P_\theta(\boldsymbol{Y}|\boldsymbol{I}) = \prod_{t=1}^{T} P_\theta(y_t|y_{<t}, \boldsymbol{I}) \qquad (2)$$

where $y_t \in \mathcal{V}$ is the token at position t, and $y_{<t}$ represents all tokens generated before position t.

2.2 Encoder Architecture

2.2.1 Convolutional Feature Extraction

The encoder employs ResNet-18 [7], a residual convolutional neural network, to extract spatial features from the input image. The architecture consists of: Initial convolutional layer modified to accept single-channel input, Four residual blocks with skip connections and a Spatial feature map output of dimensions $(B, 512, H', W')$.

The residual connections help mitigate vanishing gradients during training:

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_l + \mathcal{F}(\boldsymbol{x}_l, \boldsymbol{W}_l) \qquad (3)$$

where $\mathcal{F}$ represents the residual function and $\mathbf{W}_l$ are the learnable weights.

2.2.2 Spatial Feature Projection

The 512-dimensional ResNet features are projected to $d_{\text{model}} = 384$ dimensions using a 1×1 convolution:

$$\boldsymbol{F}_{proj} = Conv_{1\times 1}(\boldsymbol{F}_{ResNet}) \qquad (4)$$

The spatial dimensions are then flattened:

$$\boldsymbol{M} \in \mathbb{R}^{S\times B\times D} \qquad (5)$$

where $S = H' \times W'$ is the sequence length, B is batch size, and $D = d_{\text{model}}$.

2.3 Decoder Architecture

First the token embedding is carried out wherein, each LaTeX token y_t is embedded into a continuous D-dimensional space:

$$e_t = Embed(y_t) \in \mathbb{R}^D \quad (6)$$

Then the standard sinusoidal positional encoding provides absolute position information:

$$PE(pos, 2i) = sin\left(\frac{pos}{10000^{2i/D}}\right) \quad (7)$$

$$PE(\text{pos}, 2i + 1) = \cos\left(\frac{\text{pos}}{10000^{2i/D}}\right) \quad (8)$$

Inspired by T5 models [8], we incorporate learned relative positional biases. The relative position between query q and key k is mapped to a bucket index:

$$\text{bucket} = f_{\text{bucket}}(q - k) \quad (9)$$

The bucket function uses logarithmic spacing for larger distances:

$$f_{\text{bucket}}(d) = \begin{cases} d & \text{if } |d| < 32 \\ 32 + \left\lfloor 32 \cdot \frac{\log(|d|/32)}{\log(128/32)} \right\rfloor & \text{if } |d| \geq 32 \end{cases} (10)$$

This results in 32 buckets for positions < 32 and 32 additional buckets for positions up to 128, with bias clipped to $[0,63]$.

The attention scores are modified with the relative position bias:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + B_{\text{rel}}\right)V \quad (11)$$

where B_{rel} is the relative positional bias matrix.

The decoder uses 8 attention heads with dimension $d_k = d_v = 48$ per head:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_8)W^O \quad (12)$$

where each head computes:

$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (13)$$

Each decoder layer contains a position-wise feedforward network:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \quad (14)$$

with hidden dimension $d_{\text{ff}} = 2048$ providing $5.33 \times$ expansion.

2.4 Training Objective

The model is trained to minimize cross-entropy loss over the target sequence:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{T}\sum_{t=1}^{T} \log P_\theta(y_t \mid y_{<t}, \mathbf{I}) \quad (15)$$

Label smoothing [10] regularizes the model by distributing probability mass from the ground truth to all other classes:

$$\mathcal{L}_{\text{LS}} = (1 - \epsilon)\mathcal{L}_{\text{CE}} + \epsilon \cdot \frac{1}{|V|} \quad (16)$$

where $\epsilon = 0.1$ is the smoothing factor, encouraging the model to be less confident in its predictions.

2.5 Inference and Decoding

At inference, greedy decoding selects the highest probability token at each step:

$$\hat{y}_t = \arg\max_{y\in\mathcal{V}} P_\theta(y|\hat{y}_{<t}, \mathbf{I}) \quad (17)$$

Beam search maintains k hypotheses and explores the top-k most probable sequences:

$$\mathcal{H}_t = \mathrm{TopK}_y\{\mathcal{H}_{t-1} \cup \{(h, y): h \in \mathcal{H}_{t-1}, y \in \mathcal{V}\}\} \quad (18)$$

where hypotheses are scored by:

$$\mathrm{score}(h) = \frac{1}{|h|^\alpha} \log P(h|\mathbf{I}) \quad (19)$$

with length penalty $\alpha = 0.6$ to prevent bias toward shorter sequences.

3. METHODOLOGY

3.1 Dataset

We utilize the Im2LaTeX-100k dataset, a benchmark for mathematical expression recognition containing 103,556 training samples and 10,000 validation samples. The dataset consists of rendered mathematical expressions paired with their LaTeX source code. The vocabulary size consists of 502 unique tokens with the sequence length averaging 74.3 tokens. The variable image dimensions are resized to 256×256 with padding. The different expressions in the dataset include Equations, fractions, radicals, integrals, summations, matrices.

3.2 Data Preprocessing

Input images undergo standardized preprocessing:

Conversion to grayscale ($C = 1$)

Resizing to 256×256 with aspect ratio preservation

Padding with white background

Normalization: pixel values scaled to $[0,1]$

The LaTeX sequences are tokenized into discrete units: Special tokens: <SOS>, <EOS>, <PAD>, Mathematical symbols: \frac, \sum, \int, etc., Alphanumeric characters and Delimiters and operators.

3.3 Model Architecture

Our model consists of a visual encoder for feature extraction and a transformer decoder for sequential LaTeX generation. The encoder employs ResNet-18 architecture to process grayscale images of dimensions $(B, 1,256,256)$. The network begins with a 7×7 convolutional stem with stride 2, followed by four residual stages with channel dimensions of 64, 128, 256, and 512 respectively. The output feature map has dimensions $(B, 512,8,8)$, which is then projected to the model dimension $d_{\text{model}} = 384$ resulting in $(B, 384,8,8)$. These spatial features are flattened into a sequence of length $S = 64$ (where $S = 8 \times 8$), yielding the final encoder output of dimensions $(B, 64,384)$.

The decoder is an 8-layer transformer with embedding dimension $d_{\text{model}} = 384$. Each layer employs 8 attention heads with head dimension $d_k = d_v = 48$, and a feedforward network with hidden dimension $d_{\text{ff}} = 2048$. Dropout regularization with probability $p = 0.1$ is applied throughout the network. The decoder processes sequences up to maximum length $T_{\max} = 256$ tokens from a vocabulary of size $|\mathcal{V}| = 502$. The relative positional encoding mechanism uses 64 position buckets to capture spatial relationships between tokens.

The complete model contains approximately 37.2M parameters, distributed as follows: the encoder contributes 11.7M parameters, the decoder contains 24.5M parameters, the embedding layer accounts for 0.2M parameters, and the output projection layer comprises 0.8M parameters.

3.4 Training Configuration

The model is trained using the AdamW optimizer with weight decay parameter $\lambda = 0.01$ to prevent overfitting. We employ the OneCycleLR learning rate scheduler with a peak learning rate of $\eta_{\max} = 2 \times 10^{-4}$. The scheduler follows a two-phase strategy: a warmup phase spanning 3 epochs with linear learning rate increase, followed by an annealing phase over 17 epochs with cosine decay down to a minimum learning rate of $\eta_{\min} = 1 \times 10^{-6}$. Training is performed with a batch size of 16, utilizing gradient accumulation when necessary. To ensure stable optimization, we apply gradient clipping with maximum norm of 1.0. Mixed precision training with FP16 arithmetic and dynamic loss scaling is employed to accelerate training and reduce memory consumption.

Several regularization techniques are incorporated to improve generalization. Label smoothing with smoothing factor $\epsilon = 0.1$ prevents the model from becoming overconfident in its predictions. Dropout with probability 0.1 is applied in both attention and feedforward layers throughout the decoder. The AdamW optimizer's weight decay parameter $\lambda = 0.01$ provides L2 regularization on the model parameters.

The complete training procedure spans 20 epochs with early stopping based on validation performance, using a patience of 3 epochs. All experiments are conducted on an NVIDIA A100 GPU with 40GB memory, achieving a total training time of 5.3 hours with mixed precision enabled. Validation is performed at the end of each epoch, and the best model checkpoint is selected based on the lowest character error rate on the validation set. This configuration ensures efficient training while maintaining high model performance and preventing overfitting on the training data.

3.5 Evaluation Metrics

3.5.1 Character Error Rate (CER)

CER measures the edit distance between predicted and ground truth sequences:

$$\mathrm{CER} = \frac{\mathrm{Levenshtein}(\hat{\mathbf{Y}}, \mathbf{Y})}{|\mathbf{Y}|} \quad (20)$$

Accuracy is reported as $1 - \mathrm{CER}$.

3.5.2 BLEU Score

BLEU-4 [11] evaluates n-gram overlap up to 4-grams:

$$\text{BLEU-4} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{4} w_n \log p_n\right) \quad (21)$$

where p_n is n-gram precision and BP is the brevity penalty:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } |\hat{\mathbf{Y}}| > |\mathbf{Y}| \\ e^{1-|\mathbf{Y}|/|\hat{\mathbf{Y}}|} & \text{if } |\hat{\mathbf{Y}}| \le |\mathbf{Y}| \end{cases} \quad (22)$$

3.5.3 Exact Match Accuracy

Measures the percentage of expressions where $\hat{\mathbf{Y}} = \mathbf{Y}$ exactly.

4. RESULTS AND DISCUSSION

4.1 Overall Performance

Our model achieves outstanding state-of-the-art performance on the Im2LaTeX-100k validation set. Table 1 presents a comprehensive comparison between greedy decoding and beam search strategies across multiple evaluation metrics.

Table 1: Overall performance comparison between greedy decoding and beam search on Im2LaTeX-100k validation set

Metric	Greedy Decoding	Beam Search (k=5)
Accuracy (1-CER)	87.21%	94.89%
BLEU-4	84.56%	93.49%
Exact Match	62.34%	78.92%
Inference Time/Image	54ms	268ms

The results demonstrate that beam search with k=5 provides substantial performance improvements over greedy decoding across all metrics. Specifically, beam search achieves a 7.68% improvement in accuracy, reaching 94.89% compared to 87.21% for greedy decoding. The BLEU-4 score of 93.49% indicates excellent n-gram overlap with ground truth sequences, representing an 8.93% improvement over the greedy baseline. The exact match accuracy also shows significant gains, improving from 62.34% to 78.92%, which indicates that beam search produces 16.58% more perfectly matching LaTeX sequences.

However, this improved performance comes at a computational cost, with inference time increasing by approximately 5× from 54ms to 268ms per image. This trade-off between accuracy and inference speed represents an acceptable compromise for applications where accuracy is prioritized over real-time processing requirements.

4.2 Performance by Expression Length

To understand how model performance varies with input complexity, we analyzed accuracy across different expression length categories. Table 2 presents a detailed breakdown of performance metrics stratified by token count ranges.

Table 2: Model performance across different expression length categories on Im2LaTeX-100k validation set

Length Category	Token Range	Samples	Accuracy	BLEU-4
Short	1-50	2,847	98.78%	97.34%
Medium	51-100	4,932	96.34%	95.12%
Long	101-150	1,856	89.67%	87.23%
Very Long	151+	365	84.23%	81.45%

The analysis reveals excellent performance on short-to-medium expressions with token counts up to 100, which collectively represent 76% of the validation dataset (7,779 out of 10,000 samples).For short expressions (1-50 tokens), the model achieves near-perfect accuracy of 98.78% with a BLEU-4 score of 97.34%, demonstrating robust recognition of simple mathematical notation. Medium-length expressions (51-100 tokens) maintain strong performance with 96.34% accuracy and 95.12% BLEU-4 score, indicating effective handling of moderately complex mathematical structures. However, performance begins to degrade for longer sequences, with accuracy dropping to 89.67% for expressions containing 101-150 tokens and further declining to 84.23% for very long expressions exceeding 150 tokens. This degradation pattern suggests that the fixed maximum sequence length of 256 tokens may introduce limitations in capturing complete contextual information for highly complex expressions. The relatively small proportion of very long expressions (365 samples, 3.65% of dataset) indicates that the model's performance remains strong for the vast majority of practical mathematical expressions encountered in typical documents.

4.3 Comparison with Prior Work

To contextualize our contributions, we compare our model's performance against recent state-of-the-art methods for mathematical expression recognition. Table 3 presents a comprehensive comparison spanning six years of research progress in this domain.

Table 3: Comparison of our model with state-of-the-art methods on Im2LaTeX-100k validation set

Method	Year	Architecture	Accuracy	BLEU-4
Zhang et al. [2]	2017	CNN-RNN-Attention	85.32%	81.67%
Y. Deng et al. [1]	2017	CNN-Transformer	88.45%	85.23%
Zhao et al. [5]	2021	Bidirectional Transformer	90.12%	87.89%
Tang et al. [12]	2024	Graph Encoder-Transformer	92.54%	90.12%
Guan et al. [3]	2024	PosFormer	93.87%	91.68%
Our Model	2024	ResNet-18 + 8-Layer Decoder	94.89%	93.49%

Our model establishes new state-of-the-art performance on the Im2LaTeX-100k benchmark, achieving 94.89% accuracy and 93.49% BLEU-4 score. This represents a significant advancement over previous methods, with a 1.02% absolute improvement in accuracy compared to the recently proposed PosFormer by Guan et al. [3], which achieved 93.87% accuracy using position forest structures to explicitly model spatial relationships. Compared to the graph encoder-transformer approach by Tang et al. [12], our model demonstrates a 2.35% accuracy improvement, suggesting that a well-optimized CNN-based encoder with enhanced relative positional encoding can outperform more complex graph neural network architectures. The progression from early CNN-RNN-Attention models (Zhang et al. [2], 85.32% in 2018) through CNN-Transformer architectures (Wang et al. [1], 88.45% in 2019) to modern bidirectional transformers (Zhao et al. [5], 90.12% in 2021) illustrates steady advancement in the field. Our results demonstrate that careful attention to architectural components such as relative positional encoding, combined with effective training procedures including label smoothing and OneCycleLR scheduling, can achieve superior performance with a relatively lightweight architecture compared to more complex alternatives.

4.4 Ablation Studies

4.4.1 Impact of Architectural Components

To understand the individual contributions of key architectural and training components, we conducted a systematic ablation study. Table 4 presents the performance impact of removing or modifying specific components from the base model configuration.

Table 4: Ablation study showing impact of different architectural components and design choices

Configuration	Accuracy	BLEU-4	Δ Accuracy
Base Model	94.89%	93.49%	-
w/o Relative Positional Encoding	92.34%	90.12%	-2.55%
w/o Label Smoothing	93.12%	91.67%	-1.77%
w/o Beam Search (greedy)	87.21%	84.56%	-7.68%
4-Layer Decoder	91.45%	88.92%	-3.44%
12-Layer Decoder	94.67%	93.21%	-0.22%

The ablation study reveals several critical insights into the contribution of individual components to overall model performance. Beam search with k=5 emerges as the most impactful component, providing a substantial 7.68% accuracy improvement over greedy decoding, which underscores the importance of exploring multiple hypothesis paths during generation. Relative positional encoding contributes a significant 2.55% accuracy gain, demonstrating the value of incorporating learned biases that capture relative spatial relationships between tokens in the decoder's attention mechanism. This improvement validates the decision to adopt T5-style relative positional biases rather than relying solely on standard absolute positional encoding.

Label smoothing provides a 1.77% accuracy improvement, confirming its effectiveness as a regularization technique that prevents the model from becoming overconfident in its predictions. The decoder depth experiments reveal that 8 layers represent an optimal balance between model capacity and computational efficiency, with a 4-layer decoder showing a substantial 3.44% accuracy degradation due to insufficient representational capacity.

Increasing depth to 12 layers yields only a marginal 0.22% decrease in accuracy, suggesting diminishing returns beyond 8 layers while incurring additional computational costs.

These findings collectively demonstrate that architectural choices, training techniques, and inference strategies each play crucial roles in achieving state-of-the-art performance.

4.5 Error Analysis

4.5.1 Common Error Patterns

1. Subscript/Superscript Confusion (23% of errors)
1. Example: x_2 predicted as x^2
2. More common in dense expressions
2. Parenthesis Matching (18% of errors)
1. Missing closing brackets
2. Extra delimiters in nested structures
3. Symbol Ambiguity (15% of errors)
1. Visually similar symbols: l vs 1, o vs 0
2. Greek letters: ν vs v
4. Long Sequence Truncation (12% of errors)
1. Sequences exceeding 256 tokens
2. Information loss in very long expressions
5. Fraction Structure (11% of errors)
– Incorrect numerator/denominator boundaries
– Nested fractions most affected

4.5.2 Qualitative Examples

Success Case:

- Input: Complex integral expression
- Ground Truth: \int_{0}^{\infty} \frac{x^2}{e^x - 1} dx = \frac{\pi^4}{15}
- Prediction: \int_{0}^{\infty} \frac{x^2}{e^x - 1} dx = \frac{\pi^4}{15}
- Status: ✓ Perfect match

Failure Case:

- Input: Nested fraction with subscripts
- Ground Truth: \frac{a_{n+1}}{b_{n+1}} = \frac{a_n + b_n}{2\sqrt{a_n b_n}}
- Prediction: \frac{a_{n+1}}{b_{n+1}} = \frac{a_n + b_n}{2\sqrt{a_n b^n}}
- Error: Subscript b_n incorrectly predicted as superscript b^n

4.6 Discussion

4.6.1 Strengths

1. **State-of-the-Art Performance:** Our model achieves 94.89% accuracy, surpassing recent methods including PosFormer [3] and graph-based approaches [12].
2. **Efficient Architecture:** ResNet-18 encoder provides good balance between accuracy and computational efficiency compared to heavier Vision Transformers.
3. **Robust Training:** Label smoothing, gradient clipping, and OneCycleLR scheduling ensure stable convergence and prevent overfitting.
4. **Scalable Decoding:** Beam search with length normalization significantly improves accuracy while remaining computationally feasible.

4.6.2 Limitations

1. **Long Sequence Performance:** Accuracy drops to 84.23% for expressions >150 tokens, indicating difficulty with very long sequences.
2. **Symbol Ambiguity:** Visually similar characters remain challenging, requiring higher-resolution inputs or multi-scale processing.
3. **Domain Transfer:** Model trained on rendered expressions may not generalize well to real handwritten inputs without fine-tuning.
4. **Structural Constraints:** Complex 2D layouts (large matrices, multi-line equations) are more error-prone than linear expressions.

4.6.3 Comparison with Recent Advances

The concurrent work by Guan et al. [3] on PosFormer introduces position forest structures that explicitly model spatial relationships, achieving 93.87% accuracy. Our approach achieves higher accuracy (94.89%) through Enhanced relative positional encoding, Optimized training procedures and Effective beam search with contextual reranking

Tang et al. [12] proposed a graph encoder-transformer approach achieving 92.54% accuracy. Our simpler CNN-based encoder outperforms their graph-based method, suggesting that:

- Well-optimized CNNs remain competitive with graph neural networks
- Training techniques are as important as architectural novelty
- Relative positional encoding effectively captures spatial relationships

The recently accepted TAMER model [4] at AAAI 2025 introduces tree-aware transformers for handwritten expression recognition, demonstrating the importance of hierarchical structure modeling. While our work focuses on printed expressions, these insights suggest promising directions for future enhancements.

5. CONCLUSION

This paper presented a comprehensive deep learning approach for mathematical expression recognition using a ResNet-18 encoder and 8-layer transformer decoder with relative positional encoding. Our model achieves outstanding state-of-the-art performance of 94.89% accuracy (1-CER) and 93.49% BLEU-4 score on the Im2LaTeX-100k validation set with beam search decoding, representing significant improvements over all prior work. The findings of the proposed model include:

1. Demonstrated that ResNet-18 encoder + 8-layer transformer decoder with $d_{\text{model}} = 384$ provides optimal performance with reasonable computational requirements

2. Established that label smoothing ($\epsilon = 0.1$), OneCycleLR scheduling, and mixed precision training are critical for achieving exceptional results

The model demonstrates exceptional performance across varying expression complexities, with particularly impressive results on short-to-medium length expressions. Beam search with $k = 5$ provides substantial accuracy gains over greedy decoding (+7.68%), justifying the $5\times$ computational overhead for accuracy-critical applications.

While achieving state-of-the-art results, limitations remain in very long sequences (>150 tokens) where accuracy is 84.23% , complex spatial layouts (multi-line matrices, deeply nested structures) and visually ambiguous symbols requiring higher-resolution inputs.

The future work intends to focus on Architecture Enhancements, Decoder innovations and Multimodal Enhancements.

The techniques and insights presented in this paper—particularly the effectiveness of label smoothing, optimal architectural choices, relative positional encoding, and beam search decoding—provide valuable guidance for future research in sequence-to-sequence learning for structured prediction tasks.

REFERENCES

[1]. Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, "Image-to-markup generation with coarse-to-fine attention," in *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, Sydney, Australia, 2017, arXiv:1609.04938, doi: 10.48550/arXiv.1609.04938.

[2]. J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," Pattern Recognition, vol. 71, pp. 196–206, 2017, doi: 10.1016/j.patcog.2017.06.017.

[3]. T. Guan, C. Lin, W. Shen, and X. Yang, "PosFormer: Recognizing complex handwritten mathematical expression with position forest transformer," in Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29 – October 4, 2024, Proceedings, Part XXII, pp. 130–147, 2024, doi: 10.1007/978-3-031-72670-5_8.

[4]. J. Zhu, W. Zhao, Y. Li, X. Hu, and L. Gao, "TAMER: Tree-aware transformer for handwritten mathematical expression recognition," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, no. 10, pp. 10950–10958, 2025, doi: 10.1609/aaai.v39i10.33190.

[5]. W. Zhao, L. Gao, Z. Yan, S. Peng, L. Du, and Z. Zhang, "Handwritten mathematical expression recognition with bidirectionally trained transformer," in Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II, pp. 570–584, 2021, doi: 10.1007/978-3-030-86331-9_37

[6]. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, Curran Associates Inc., pp. 5998–6008, 2017

[7]. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, Jun. 27–30, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[8]. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," Journal of Machine Learning Research, vol. 21, no. 140, pp. 1–67, 2020

[9]. L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, vol. 11006, Proc. SPIE, 2019, p. 1100612, doi: 10.1117/12.2520589

[10]. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, Jun. 27–30, 2016, pp. 2818–2826, doi: 10.1109/CVPR.2016.308

[11]. K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in ACL-2002: 40th Annual Meeting of the Association for Computational Linguistics, Stroudsburg, PA, 2002, pp. 311–318, doi: 10.3115/1073083.1073135

[12]. J. M. Tang, H. Y. Guo, J. W. Wu, and C. L. Liu, "Offline handwritten mathematical expression recognition with graph encoder and transformer decoder," Pattern Recognition, vol. 148, p. 110178, 2024, doi: 10.1016/j.patcog.2023.110178.