Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Inception-Transformer: End-To-End Printed Mathematical Expression Recognition Using Convolutional and Transformer Networks

Everistus Zeluwa Orji, Ali Haydar, İbrahim Erşan

Department of Computer Engineering, Girne American University, Mersin-10, Karaman, Turkey
Email: orjizeluwa@gmail.com

The task of extracting mathematical expressions (ME) from images and converting them into LaTeX code has been a longstanding challenge in research since the 1960s. Traditional Optical Character Recognition (OCR) methods struggle with this because of the intricate two-dimensional layout of mathematical symbols. Recent advancements in encoder-decoder models have significantly enhanced recognition accuracy for both handwritten and printed MEs. However, RNN-based models face limitations like handling long-range dependencies and slower inference speeds. While transformer-based models offer improved performance, they demand substantial training datasets and considerable computational power. In this study, we introduce a hybrid model specifically designed for printed MEs conversion to LaTeX, combining an Inception-based CNN for feature extraction with an encoder-decoder modeled after transformer architectures. We evaluated the model on the IM2LATEX-100K dataset, showing significant improvements in performance, accuracy, and computational efficiency as opposed to the most advanced models. Our approach achieved a BLEU score of 92.76% and an image edit distance accuracy of 95.09%. Additionally, the model efficiently handled complex mathematical structures, utilizing only 8.58 million trainable parameters.
Keywords: Mathematical Expression Recognition, LaTeX Conversion, Inception-based CNN, Transformer Model, Image-to-Text Conversion, Encoder-Decoder Architecture, Computational Efficienc.

1. Introduction

The study of mathematical expression (ME) recognition has been a key research interest since the 1960s (Anderson, 1967). The goal of ME recognition is to convert images of mathematical content into sequential formats, such as LaTeX. This process involves three main tasks: (i) identifying mathematical symbols within the image, (ii) recognizing them, and (iii) analyzing the structural relationships between them. Unlike linear text, mathematical expressions often exhibit a two-dimensional structure made up of symbols like superscripts, subscripts, and fraction lines, making the recognition process more complex (Yan et al., 2020).

Traditional methods like Optical Character Recognition (OCR) tackle this problem in two distinct phases: segmenting the formula into individual characters and classifying them based on a predefined vocabulary (Orji et al., 2023). However, ME recognition poses greater challenges, as it requires not only symbol identification but also an understanding of the spatial relationships among these symbols (Oral et al., 2020). For decades, ME recognition has attracted significant interest within the pattern recognition community, and various international competitions have encouraged advancements in the recognition of handwritten MEs (Mouchère et al., 2014, 2016). Printed mathematical expression (PME) recognition, in contrast to handwritten MEs, has received less attention. This task is valuable for applications such as developing systems for mathematical content retrieval or for generating and editing LaTeX representations of complex mathematical expressions, thereby reducing the workload for users.

Recent progress in encoder-decoder architectures has greatly improved the recognition of both handwritten and printed mathematical expressions (Deng et al., 2016; Yan et al., 2020). Generally, the encoder utilizes a convolutional neural network (CNN) to capture key visual features from the input image, while generating the LaTeX output, the decoder typically relies on a recurrent neural network (RNN) or working with a transformer (Wang & Liu, 2021). Despite these advancements, current methods may still fall short in fully addressing the intricacies of printed mathematical expression recognition due to several unresolved challenges.

- Sequential dependencies in RNN-based models. RNN-based encoder-decoder models process sequences incrementally, which poses challenges in capturing the spatial relationships between symbols in mathematical expressions. In contrast, Transformer models can process entire sequences simultaneously, allowing them to handle long and complex mathematical equations more effectively. As a result, RNNs may produce inaccurate LaTeX outputs and struggle with tasks requiring a global understanding, such as PME recognition tasks (Illium et al., 2022).

- Difficulty handling long-range dependencies. RNNs often face challenges in managing long-range dependencies due to the vanishing gradient problem. In mathematical expressions, symbols that are far apart can still be related, but RNNs may not capture these dependencies effectively (Liu et al., 2023). This can lead to errors in the structure of the generated LaTeX code, as demonstrated by models that fail to maintain consistency across longer sequences (Yang et al., 2020).

- Higher computational requirements in traditional architectures. Encoder-decoder models that do not leverage Inception-based feature extraction or Transformer architectures often require more parameters and longer training times. The increased complexity of these models can lead to higher computational costs, making them less efficient for large-scale mathematical image processing tasks (Sharma & Guleria, 2022). Comparatively, models utilizing efficient feature extractors and parallel processing techniques have been shown to be more cost-effective (Shen et al., 2018).

- Slower inference in RNN-based models. RNN-based encoder-decoder models rely on sequential processing, which can lead to slower inference times. This is particularly problematic in real-time applications, such as converting mathematical images to LaTeX on the go (Lalapura et al., 2024). Transformer-based models, on the other hand, process sequences in parallel, providing faster and more efficient inference, making them preferable for tasks that demand quick results (N. Zhang & Kim, 2023).

Transformer-based printed MEs models, as modeled by (Fu et al., 2021; Zhou et al., 2023), have recently made substantial efforts to solve some of these issues. While the use of vision transformers (ViTs) is powerful in certain scenarios (Dosovitskiy et al., 2020), it faces challenges such as: They don't have specialized feature extraction, which can make it harder to focus on key features. They also have problems with training and resource optimization, which makes it harder to fine-tune efficiently. Additionally, their training process necessitates a substantial amount of data, thereby consuming significant resources. Finally, they have a more complicated architecture for simple tasks, which might be too much for simpler uses.

In this study, we introduce a hybrid model that incorporates CNNs, built on an Inception network, for feature extraction with a Transformer-based encoder-decoder for sequence learning (Ashish et al., 2017; Szegedy et al., 2015). This approach enhances spatial localization, hierarchical feature extraction, data efficiency, and the handling of complex structures, all essential for accurately converting printed mathematical images into LaTeX. Significant contributions made by this research are as follows:

1. Model Development: We designed an encoder-decoder model that enhances feature extraction and recognition accuracy by combining an Inception-based CNN for robust feature extraction with a Transformer-based encoder-decoder for efficient sequence processing.

2. Computational Efficiency: Our lightweight models offer high accuracy while being optimized for use in resource-constrained environments, striking a balance between performance and computational demands.

3. Evaluation: Using the IM2LATEX-100K dataset, we thoroughly tested and compared our model to the best models for converting images to LaTeX. Our model showed better performance, generalizability, and robustness.

4. Real-world Application: Our models are designed for practical deployment, capable of handling diverse input conditions, and are suitable for academic, educational, and document processing purposes.

This paper is structured to provide a comprehensive analysis of the proposed methodologies and their relevance. Section 2 Related work reviews advancements in ME recognition, attention-based encoder-decoder models, and the Inception model's application for text recognition. The section 3 Proposed Method describes a hybrid model that uses Inception-based CNNs for feature extraction and a Transformer-based encoder-decoder for sequence learning. Sections 4 and 5 on Experiments and Performance evaluation and discussion outlines the datasets, training methods, and comparison with leading models. Lastly, Section 6, dedicated to Conclusions and Future Work, highlights the essential outcomes and offers possible directions for advancing research.

2. Related work

Across various AI domains, recent breakthroughs include advancements related to analyzing images, processing text, and recognizing voices. These breakthroughs have been powered by sophisticated neural models that leverage attention mechanisms to handle tasks, including translation, text extraction, and summarization (Alexander et al., 2015; Bahdanau et al., 2014; Jan et al., 2015; Luong et al., 2015; Vinyals et al., 2014). Earlier approaches to converting images into LaTeX format employed CNNs for identifying visual features in the encoder and RNNs for generating output sequences in the decoder. This streamlined the conversion process without needing manual intervention in LaTeX syntax (Deng et al., 2016). Recent research has refined this process, introducing attention-based methods to address computational inefficiencies and enhance both speed and accuracy in handling complex LaTeX structures (Deng et al., 2017).

Similarly, the WAP (Watch, Attend, and Parse) model addressed the difficulty of recognizing offline handwritten mathematical formulas by integrating a coverage-based attention mechanism to enhance formula parsing (J. Zhang et al., 2017). The WAP model also integrated DenseNet within its encoding framework and utilized a hierarchical attention mechanism to address parsing errors, significantly enhancing the detection and interpretation of symbols and structures in mathematical formulas (Gao et al., 2016; Jianshu et al., 2018). Additionally, further research proposed multi-scale attention models to improve the parsing of complex expressions through fine-grained parsing techniques (Bender et al., 2019). Moreover, advancements in attention mechanisms, such as combining spatial and channel attention through joint attention, have significantly enhanced the accuracy and robustness of modern image-to-latex systems (Wang & Liu, 2019).

Attention mechanisms, including self-attention mechanism found in Transformers, have substantially improved the performance of image-to-latex models by enhancing their ability to accurately parse symbols, recognize them, and understand structural details (Ashish et al., 2017; Vu et al., 2023). These advancements enable models to handle long-range dependencies more efficiently, capturing important contextual relationships between symbols. Furthermore, the introduction of multi-scale attention has allowed models to adapt to variations in symbol size, thus improving robustness across a range of simple and complex mathematical expressions. Continued improvement of these attention-based methods, including hybrid approaches that integrate spatial and channel attention, has significantly increased the reliability and effectiveness of modern image-to-latex systems in rendering high-quality mathematical representations.

The utilization of the Inception model as a feature extractor has further propelled advancements in text recognition, especially for comprehending intricate documents. The Inception model leverages deep, multi-level convolutional layers to capture a broad spectrum of spatial features, resulting in enhanced feature maps for subsequent tasks (Szegedy et al., 2015). This ability to process different scales and detailed textures make the Inception model particularly appropriate for text recognition, where symbols and characters can appear in varying sizes and exhibit complex details. For example, research has shown that implementing an Inception-based encoder alongside an attention-based transformer model can significantly enhance outcomes in tasks such as handwriting recognition, mathematical formula parsing, and structured document analysis (Si et al., 2022). By combining robust feature extraction with effective sequence learning, this approach evidently improves the model's accuracy and robustness in understanding complex visual patterns. The synergy between the Inception model's feature extraction capabilities and the transformer's attention mechanisms effectively addresses spatial and contextual nuances in text and images, leading to notable enhancements in recognition performance (Devlin et al., 2019).

3. Proposed method

We introduce a hybrid model consisting of three primary elements: a feature extractor, an encoder, and a decoder as shown in Fig. 1. First, the model incorporates an Inception-based CNN for feature extraction, utilizing convolutional and pooling layers to achieve efficient multi-scale feature detection. The features extracted by CNN are then passed into a Transformer-based encoder, which processes input sequences using multiple attention heads along with feed-forward layers. Finally, the Transformer-based decoder applies masked multi-head attention to accurately generate LaTeX code from the mathematical images.

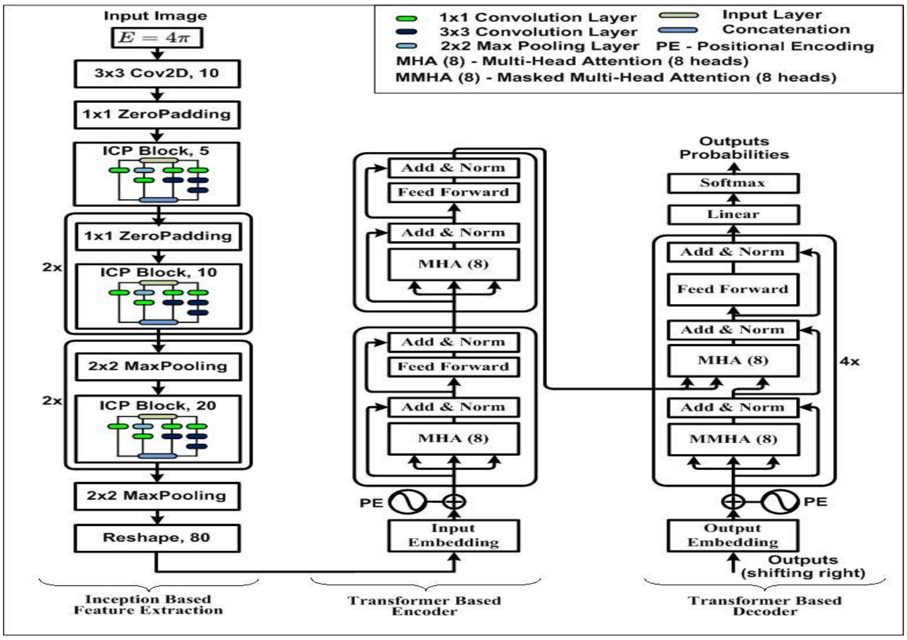


Fig. 1. Architecture of the proposed hybrid model with inception-based feature extraction

and transformer-based encoder-decoder for image-to-LaTeX conversion.

3.1. Feature extraction

Our model incorporates a unique pre-encoder layer serving as a feature extractor as detailed in Table 1. This layer processes raw preprocessed grayscale mathematical images through a sequence of convolutional and pooling operations, reducing the image dimensions (60 x 500) while enhancing feature representation. The inception blocks (5) perform parallel convolutions with various filter sizes (1x1 and 3x3) and then merge the results to facilitate multi-scale feature extraction as detailed in Section 3.1.1. Finally, a Reshape layer adjusts the tensor dimensions, preparing it for subsequent processing in the Transformer encoder layers.

Table1: Architectural details of our feature extraction block

Layer	Kernel Size	Stride	Output Size	1x1 Filters	3x3 Filters
Conv	3x3	1	58 X 498 X 10	-	-
ZeroPadding	-	-	60 X 500 X 10	-	-
IncepBlock_1	-	-	60 X 500 X 20	4	3
ZeroPadding	-	-	62 X 502 X 20	-	-
IncepBlock_2	-	-	62 X 502 X 40	4	3
ZeroPadding	-	-	64 X 504 X 40	-	-
IncepBlock_3	-	-	64 X 504 X 40	4	3
MaxPooling	2x2	2	32 X 252 X 40	-	-
IncepBlock_4	-	-	32 X 252 X 80	4	3
MaxPooling	2x2	2	16 X 126 X 80	-	-
IncepBlock_5	-	-	16 X 126 X 80	4	3
MaxPooling	2x2	2	8 X 63 X 80	-	-
Reshape	-	-	504 X 80	-	-

3.1.1. Inception block

Drawing inspiration from the GoogleNet architecture (Szegedy et al., 2015), the inception block is designed for multi-scale feature extraction. This custom layer utilizes parallel convolutional layers with varying filter sizes to capture different aspects of the input features. Specifically, it combines 1x1 and 3x3 convolutions, as depicted in Fig. 2, along with a max-pooling layer to process input data at multiple scales. The output from these parallel operations are then concatenated to form a rich and comprehensive feature representation. This technique enables the model to learn complex patterns and hierarchical structures characteristic of mathematical input images.

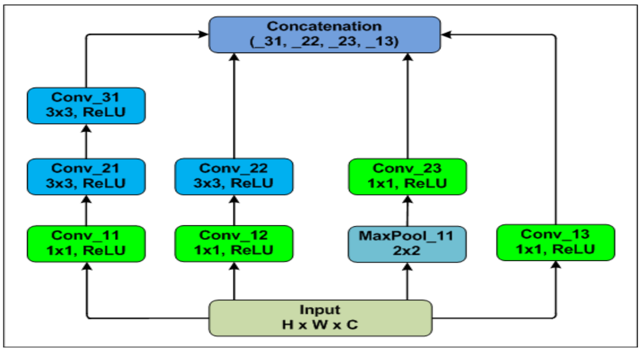


Fig. 2. Inception block architectural

3.2. Transformer-based encoder and decoder

Our model's encoder includes two identical layers, each consisting of two main sub-components. The first sub-component is a multi-head self-attention mechanism (MHA), while the second is a position-wise feed-forward neural network (FFN). A residual connection encloses each of these sub-components, and layer normalization (LN) follows them to stabilize and optimize the training process. In contrast, the decoder in our model features four identical layers. Each layer of the decoder includes the same two sub-components as the encoder, along with an additional MHA mechanism directed at the encoder's output sequence. This additional mechanism enhances the decoder's ability to direct its attention to important parts of the encoded sequence. the decoder's self-attention mechanism uses masking to ensure the model focuses only on previous positions when predicting the next token. Detailed descriptions of each component of the transformer-based encoder and decoder are provided:

1) Input embeddings: These play a crucial role in transformer encoder models by converting input tokens into dense vectors for model processing. These embeddings encapsulate the semantic information and relationships between tokens, enabling the transformer to understand context and meaning. Each input token is transformed into a fixed-size dense vector using an embedding matrix (E) (Eq. 1). For a vocabulary size (V) and embedding dimension (d_{model}), the embedding matrix (E) has dimensions ($V \times d_{\text{model}}$). The vector for a token (t_i) is:

$$x_i = E[t_i] \qquad (1)$$

2) Positional encoding (PE): When encoding the sequential position of each token, both the encoder and decoder incorporate positional encoding into their input and output embeddings. Transformers, by themselves, lack an inherent understanding of the order of tokens. Positional encodings, which utilize sine and cosine functions at varying frequencies (Eqs. 1 and 2), provide this essential sequential information. This mechanism helps the model to recognize and utilize the order of the tokens, important for translation and text generation tasks.

$$PE_{(pos,2i)} = sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \qquad (2)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \qquad (3)$$

Here, pos denotes the position, i the dimension index, and d_{model} the model dimension. This approach enables the model to distinguish between different positions within a sequence.

3) Scaled dot-product attention: This mechanism computes interaction between the query (Q) and key (K) matrices, normalizes it by ($\sqrt{d_k}$), and applies a softmax function to obtain attention weights. The final weighted sum is combined with the value (V) matrix, capturing the relevance of each element in the sequence (Eq. 4):

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (4)$$

When masking is necessary, often in the decoder to prevent attending to future tokens, a mask (M) is added prior to applying softmax (Eq. 5):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V \qquad (5)$$

Here, the mask (M) function adds negative infinity $(-\infty)$ to the attention scores of the positions that should be ignored, ensuring they do not affect the softmax output.

4) Multi-Head Attention (MHA): In both the encoder and decoder, we employed eight distinct MHAs, enabling our model to simultaneously attend to various parts of the input sequence. This configuration allows the model to understand different relationships within the sequence (in the encoder) and between the input and output sequences (in the decoder). We concatenate and linearly transform the outputs from these heads (Eq. 6). The MHA mechanism is represented mathematically as follows:

$$\text{MHA}(Q, K, V) = \text{Concatenate }(\text{head}_1, \dots, \text{head}_8)W^O \qquad (6)$$

where first head is calculated as:

$$\text{head}_1 = \text{Attention}\left(QW_1^Q, KW_1^K, VW_1^V\right) \qquad (7)$$

5) Add and Norm: In transformer models, the Add and Norm operation takes place in two stages within both the encoder as well as the decoder: Add (residual connection): The sub-layer's output (such as from Multi-Head Attention) is added to its input to preserve the original information (Eq. 8):

$$\text{Output} = \text{Input} + \text{Sub-layer Output} \qquad (8)$$

Norm (layer normalization): The result is then normalized to stabilize training (Eq. 9):

$$\text{Normalized Output} = \frac{\text{Output} - \mu}{\sigma + \epsilon} \qquad (9)$$

where the features' mean and standard deviation are represented by (μ) and (σ) respectively, while (ϵ) is a tiny constant introduced to avoid dividing by zero.

6) Position-Wise FFN: The FFN is applied after the attention layers in both the encoder and decoder stages. It consists of two sequential linear transformations with a ReLU activation in between, enhancing the model's ability to learn and capture intricate patterns. The mathematical formulation is provided in Eq. 10.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \qquad (10)$$

This design helps the model process and transform input data effectively across all positions in the sequence.

7) Output embeddings: These play a vital role in transformer decoder models as they convert

dense vectors back into sequences of tokens that can be interpreted as the final output. These embeddings take the encoded information and help in decoding it into meaningful sequences by capturing the semantic context provided by the transformer. Each output token is represented as a dense vector through an embedding matrix O (Eq. 11). For a vocabulary size V and embedding dimension d_{model}, the embedding matrix O has dimensions $V \times d_{\text{model}}$. The vector for a token $\widehat{t_i}$ in the output sequence is:

$$y_i = O[\widehat{t_i}] \qquad (11)$$

8) A Linear layer: Within Transformer architecture, the linear layer within the decoder specifically transforms the concatenated output of the attention heads into the required dimensions for subsequent processing. This operation is represented as (Eq. 12):

$$\text{Linear}(x) = xW + b \qquad (12)$$

Here, (x) represents the input from the attention heads, (W) denotes the weight matrix, and (b) refers to the bias term. This transformation is crucial for refining the attention output before they are passed to the final softmax layer for sequence generation.

9) The Softmax layer: In the Transformer's decoder, the softmax layer is applied after the final linear transformation to convert the output logits into a probability over the target vocabulary (Eq.13). This step is essential for selecting the next token in the sequence. The softmax function normalizes the logits z_i into probabilities p_i as follows:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \qquad (13)$$

This normalization ensures that the probability values for all possible tokens sum to 1, enabling the model to predict the most probable subsequent token in the sequence. The softmax layer thus plays a vital role in generating the final output sequence during decoding.

10) Output probabilities: In the transformer decoder, output probabilities are produced after the tokens pass through the decoder layers. These probabilities represent the possibility of each token being the correct next token in the prediction sequence.

4. Experiments

4.1. Datasets:

Our primary dataset is the prebuilt IM2LATEX-100K (Deng et al., 2017), which contains a large collection of mathematical expressions in LaTeX, sourced from various academic publications. Specifically, the author gathered a total of 103,556 distinct LaTeX equations paired with their rendered images from arXiv papers (Kanervisto, 2016) and the 2003 KDD Cup datasets (GehrkeJohannes et al., 2003). These collections collectively contains over 60,000 papers. The extraction process utilized regular expressions to parse LaTeX sources, focusing on sequences framed with standard LaTeX delimiters such as `\begin{equation}...\end{equation}` and similar structures. They retained only formulae within the character length range of 40 to 1024 to avoid single symbols and overly complex structures, ensuring a rich and valid dataset. It was split into training (~84k equations), validation (~9k equations), and test sets (~10k equations) to maintain a consistent experimental framework. Fig. 3 illustrates the distribution of token lengths in the test set, emphasizing the variability in LaTeX formula lengths. In the dataset, LaTeX formulas vary from 38 to 997 characters in length, an average length of 118 and a median length of 98.

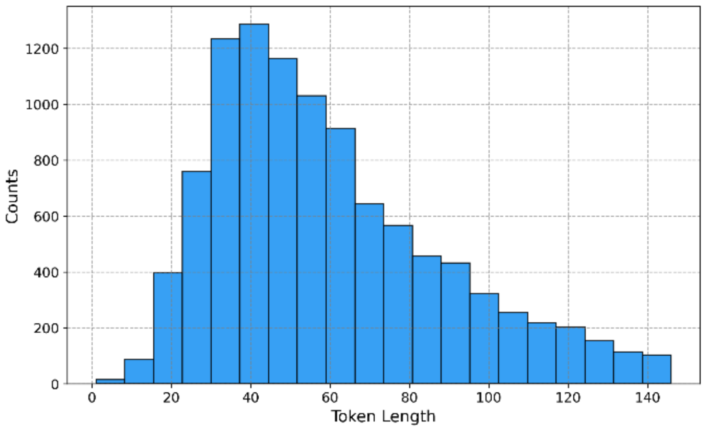


Fig. 3. Test set token length frequency distribution

Tokenization and normalization:

We tokenized the formulas in our dataset since token-based methods have proven to be more efficient than character-based methods. We utilize preprocessing techniques from (Deng et al., 2016), leveraging the Katex library, a LaTeX parser. After tokenizing, we create a vocabulary of 502 unique tokens, including " ," "START," and "END," for padding, start, and the end of sentences, tokens, respectively. We pad the formulas with " " or "PAD" tokens to ensure a uniform length (up to 150 tokens) for input into our transformer decoder. Naturally, LaTeX often contains multiple expressions that yield identical output. So, as shown in Table 2, we added normalization preprocessing to get rid of any uncertainty before training (Deng et al., 2016).

Table 2: LaTeX normalizing transformations

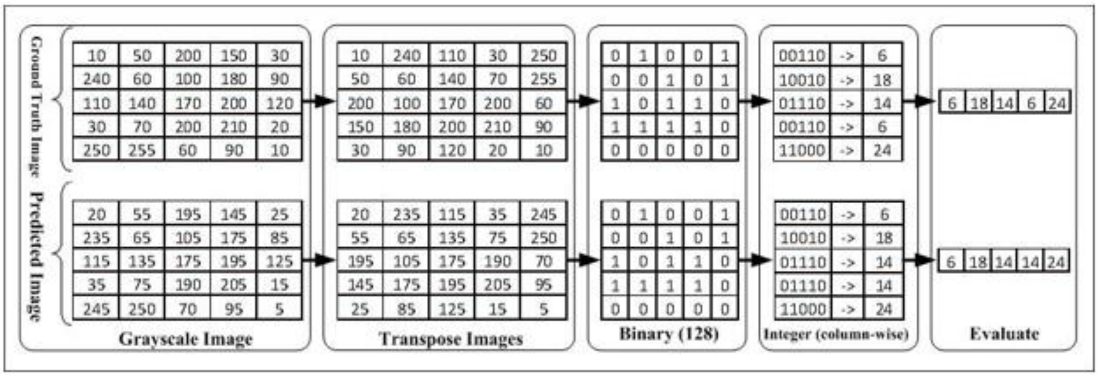
Transform	Original	Normalized
Remove	\label{…}	-
Remove	\$	-
Remove	\\\>	-
Remove	\\\~	-
Normalize	\rm{…}	\mathrm{…}
Normalize	SSSSSS	$
Normalize	S S S S S S	$

4.2. Evaluation metrics

For evaluating our model's performance, understanding that LaTeX is not a normalized language is essential; different codes can still generate the same output when compiled. For instance, the inclusion or exclusion of brackets, such as " \sqrt{x}" and " \sqrt x," will both render the same formula in PDF. We utilize two types of evaluation metrics: the one that looks at the accuracy of the LaTeX code and another that measures how closely the reconstructed image resembles the original.

1) Text-based evaluation: We use commonly adopted Natural Language Processing (NLP) metrics, including BLEU score (BLEU), Edit Distance, and Exact Match. The BLEU score evaluates the n-gram accuracy by comparing the predicted outputs with the reference outputs, with a perfect score of 1 (or 100%) indicating an exact match between the model's output and the reference text (Papineni et al., 2002). The Exact Match (EM) metric assesses the percentage of predictions that precisely replicate the reference text, character by character. An EM score of 100% implies that the model's predictions exactly match the reference text in all cases. Edit Distance (ED) or Levenshtein distance, determines the minimal number of operations (insertion, deletion, substitution) needed to convert the predicted LaTeX string into the ground truth LaTeX string (Levenshtein, 1965). A lower ED score signifies a closer match between the predicted and reference texts.

2) Image-based evaluation: While text-based metrics provide valuable insights, they do not account for the visual representation of LaTeX. Hence, we also use image-based metrics, such as edit distance (Levenshtein distance) and exact match for images, to evaluate visual similarity (Guillaume Genthial, 2017). The evaluation process in Fig. 4 involves the following steps:



Ground Truth Image — Grayscale Image

10	50	200	150	30
240	60	100	180	90
110	140	170	200	120
30	70	200	210	20
250	255	60	90	10

Ground Truth Image — Transpose Images

10	240	110	30	250
50	60	140	70	255
200	100	170	200	60
150	180	200	210	90
30	90	120	20	10

Ground Truth Image — Binary (128)

0	1	0	0	1
0	0	1	0	1
1	0	1	1	0
1	1	1	1	0
0	0	0	0	0

Ground Truth Image — Integer (column-wise)

00110	->	6
10010	->	18
01110	->	14
00110	->	6
11000	->	24

Ground Truth Image — Evaluate

6	18	14	6	24

Predicted Image — Grayscale Image

20	55	195	145	25
235	65	105	175	85
115	135	175	195	125
35	75	190	205	15
245	250	70	95	5

Predicted Image — Transpose Images

20	235	115	35	245
55	65	135	75	250
195	105	175	190	70
145	175	195	205	95
25	85	125	15	5

Predicted Image — Binary (128)

0	1	0	0	1
0	0	1	0	1
1	0	1	1	0
1	1	1	1	0
0	0	0	0	0

Predicted Image — Integer (column-wise)

00110	->	6
10010	->	18
01110	->	14
01110	->	14
11000	->	24

Predicted Image — Evaluate

6	18	14	14	24

Fig. 4. Image-based evaluation process

(a) First, convert both the ground truth and predicted images to grayscale to standardize them for further processing. (b) Next, transpose the grayscale images to properly align the columns for evaluation. (c) Then, convert the transposed images to binary format using a threshold value of 128, setting pixels above the threshold to 1 and those below to 0. (d) Treat each binary column as a token and convert it into an integer for simpler comparison, so that ('00110', '10010') becomes (6, 18). (e) Finally, compare the integer sequences corresponding to the ground truth and predicted images to quantify visual similarity, calculating metrics like image edit distance and exact match.

By integrating these diverse metrics, we deliver a comprehensive evaluation of our model's effectiveness, ensuring that both visual and textual precision is accurately captured. Each metric highlights distinct aspects of the model's output, contributing to a strong and reliable evaluation framework designed to enhance model precision and reliability.

4.3. Implementation details

The feature extraction process employs CNNs inspired by GoogleNet, specifically leveraging Inception blocks as described in Table 1. The architecture of the Inception-based block includes several convolutional and pooling operations, enabling the network to capture rich and diverse feature representations. The initial convolution layer uses a 3x3 kernel and a stride of 1. The final convolutional layer's output is rescaled to dimensions of 504 x 80. For the transformer-based encoder, two encoder layers are used, each with 8 attention heads, following the architecture of the standard Transformer model (Ashish et al., 2017). The embedding size is set to 256, while the output embedding dimensions are 80. Each encoder layer's feed-forward neural network has input and output dimensions of 512 and 160 respectively, assisting in capturing long-range dependencies in the sequence data. To mitigate overfitting, a dropout rate of 0.3 is applied. In the decoder, four stacked decoder layers are employed, each applying multi-head masked attention to the encoder's output.

The model is implemented in TensorFlow, using the Keras API, and training was conducted in a high-performance environment provided by Paperspace, using NVIDIA GTX 1080Ti GPUs. The Adam optimizer, with an initial learning rate set to (3×10^{-4}) is utilized (Kingma & Ba, 2014). Custom loss and accuracy functions are incorporated, specifically designed to handle translation tasks and exclude padding tokens during computations. The training process lasted for 20 epochs, with a custom learning rate schedule applied to dynamically adjust the rate for quicker convergence.

5. Performance evaluation and discussion

1) Comparison with leading models: Table 3 provides a comparison a performance comparison of our model against other models using the IM2LATEX-100k test set. Since identical visual output can be generated by different LaTeX markups, relying solely on text-based metrics may not fully reflect our model's accuracy in the image-to-LaTeX task. Therefore, we utilize a mixture of text-oriented metrics (BLEU) and visual-oriented metrics (IED and IEM) to evaluate the models.

Table 3: Performance evaluation using the IM2LATEX-100k test set

Model	BLEU	IED	IEM
Ours	92.76	95.09	79.72
(Deng et al., 2016)	87.73	87.60	79.88
(Pang et al., 2021)	89.72	90.07	82.13
(Noya et al., 2023)	83.80	11.40	40.54
(Genthial & Sauvestre, 2017)	78	62	35
(Le et al., 2022)	89.94	92.23	86.48

Our model outperforms those in (Deng et al., 2016; Genthial & Sauvestre, 2017; Le et al., 2022; Noya et al., 2023; Pang et al., 2021) across two main metrics, achieving the highest BLEU score of 92.76% and an image edit distance with space (IED) accuracy of 95.09%. While (Le et al., 2022)exceeds our model in image exact match (IEM) accuracy by 8.46%, our model still demonstrates superior overall performance when considering all metrics.

2) A complex structure conversion: The Table 4 presents the evaluation of our model's performance on various complex mathematical structures, including arrays and multi-line

formulas with different types of parentheses and brackets. These structures are particularly challenging due to their intricate formatting and nested components.

Table 4: Array, brackets, and multiline formula

Type	Image	LaTeX markup
Array (square bracket)	$\begin{bmatrix} 4 & 2 & 1 \\ 3 & 6 & 4 \\ 7 & 8 & 9 \end{bmatrix}$	\left [\begin {array} { l l l } { 4 } & { 2 } & { 1 } \\ { 3 } & { 6 } & 4 } \\ { 7 } & { 8 } & { 9 } \end {array} \right]
Array (round bracket)	$x = \begin{pmatrix} 0 & 0 & 0 & p_4 \\ p_1 & 0 & 0 & 0 \\ 0 & p_2 & 0 & 0 \\ 0 & 0 & p_3 & 0 \end{pmatrix}$	x = \left (\begin {array} { c c c c } { { 0 } } & { { 0 } } & { { 0 } } & { { p _ { 4 } } } \\ { { p _ { 1 } } } & { { 0 } } & { { 0 } } & { { 0 } } \\ { { 0 } } & { { p _ { 2 } } } & { { 0 } } & { { 0 } } \\ { { 0 } } & { { 0 } } & { { p _ { 3 } } } & { { 0 } } \end {array} \right)
Multi-line (angle bracket)	$\mathcal{P}_{i+1,j+1} = \left\langle \begin{array}{l} y^0, y^1, \ldots, y^j \\ y^0, y^1, \ldots, y^i \end{array} \right\rangle$	{ \mathcal P } _ { i + 1 , j + 1 } \: = \: \left \langle \begin {array} { l } { { y ^ { 0 } , y ^ { 1 } , \ldots , y ^ { j } } } \\ { { y ^ { 0 } , y ^ { 1 } , \ldots , y ^ { i } } } \end {array} \right \rangle
Multi-line (curly bracket)	$\left\{ \begin{array}{cc} M - \tilde{c}_{\tilde{m}_1+1-1}, & 1 \leq \tilde{m}_1 \\ c_{i-\tilde{m}_1} & i > \tilde{m}_1 \end{array} \right\}$	\left \{ \begin {array} { c c } { { M - \tilde { c } _ { \bar { m } _ { 1 } + 1 - 1 } , } } & { { 1 \leq \tilde { m } _ { 1 } } } \\ { { c _ { i - \tilde { m } _ { 1 } } } } & { { i > \tilde { m } _ { 1 } } } \end {array} \right \}

Table 4 shows that our model successfully converts arrays with square and round brackets, as well as multi-line formulas enclosed in angle and curly brackets in the second column (Image), into accurate LaTeX markup in the third column. Notably, the LaTeX markup generated for complex structures, such as nested arrays and multiline expressions, closely matches the required syntax, demonstrating the model's robustness and precision.

3) The impact of token length: The evaluation in Fig. 5 demonstrates that an increase in the length of mathematical expression tokens correlates with a decline in performance, as indicated by higher image edit distances. This trend is observed in both image edit distance "With Space" and "No Space" scenarios.

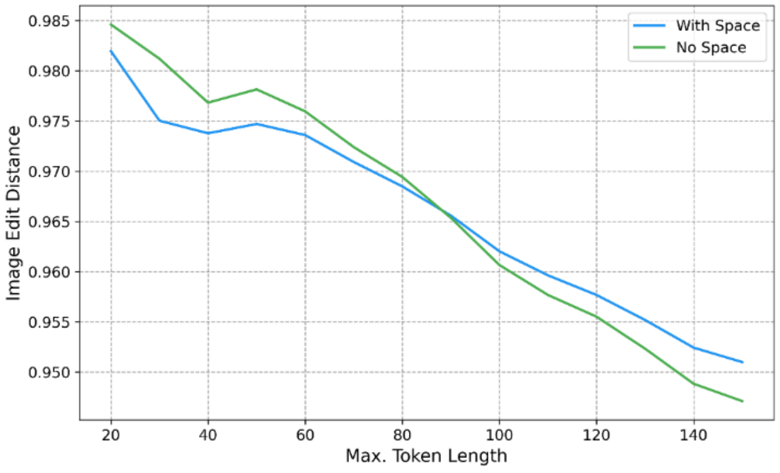


Fig. 5. Our model performance as a function of token length

Longer expressions pose a challenge to the accuracy of the model. However, the performance decline is also influenced by the fewer mathematical expressions with long tokens present in the test set, as depicted in Fig. 3. This suggests that the model's accuracy may be affected not only by the complexity of longer expressions but also by the limited representation of such expressions in the test data.

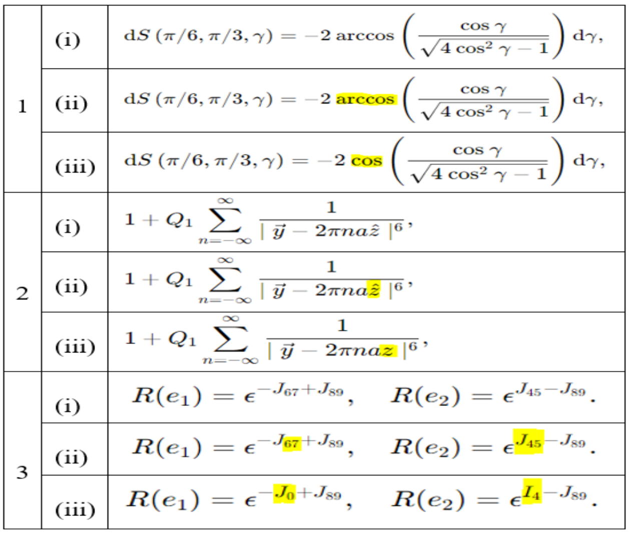
4) Parameter efficiency comparison: Table 5 compares the trainable parameters of our model with various other models, highlighting how effective our approach is in terms of parameter count. This analysis highlights how our model achieves effective performance while maintaining a lower number of trainable parameters.

Table 5: Comparison of trainable parameters across models

Model	Number of Trainable Parameters
Ours	8.58 M
(Deng et al., 2016)	9.48 M
(Yan et al., 2020)	13.01 M
(Zhou et al., 2023)	13.10 M
(Wang & Liu, 2021)	10.87 M
(Deng et al., 2017)	15.85 M

Our model, with 8.58 million trainable parameters, has the fewest parameters among the models evaluated, attributed to the use of the Inception model as our feature extraction approach. The choice of the Inception model enables efficient multi-scale feature extraction while keeping the parameter count low, enhancing our model's efficiency without compromising performance. Compared to other models, such as the one with 15.85 million parameters (Deng et al., 2017), our solution strikes an effective balance between complexity and performance, validating our design choice.

5) Visual comparative analysis of model predictions: Fig. 6 illustrates examples where our model's predictions correctly match the ground truth, unlike the results from other advanced models. Rows labeled (iii) in row 1, 2, and 3 correspond to models (Vu et al., 2023), (Pang et al., 2021), and (Yan et al., 2020), respectively, with the yellow highlights indicate areas where these models made incorrect predictions, whereas our model succeeded.



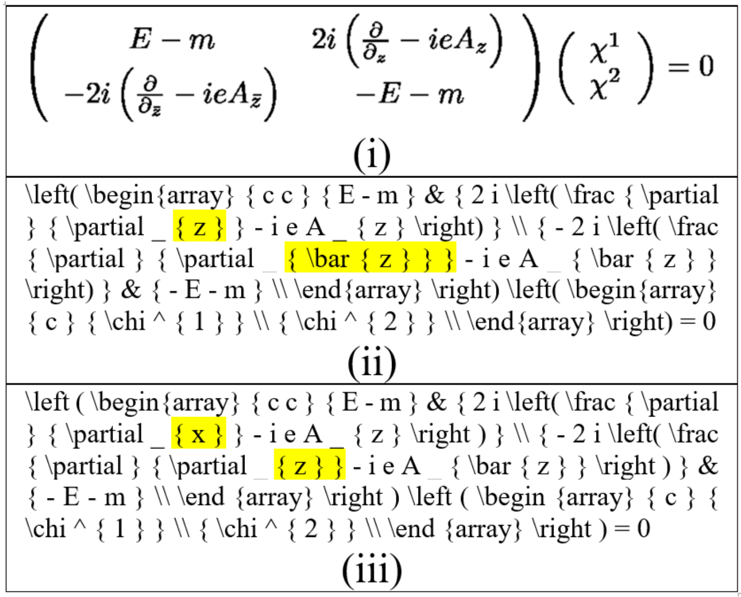
1	(i)	$\mathrm{d}S\,(\pi/6,\pi/3,\gamma) = -2\arccos\left(\frac{\cos\gamma}{\sqrt{4\cos^2\gamma-1}}\right)\mathrm{d}\gamma,$
	(ii)	$\mathrm{d}S\,(\pi/6,\pi/3,\gamma) = -2\arccos\left(\frac{\cos\gamma}{\sqrt{4\cos^2\gamma-1}}\right)\mathrm{d}\gamma,$
	(iii)	$\mathrm{d}S\,(\pi/6,\pi/3,\gamma) = -2\cos\left(\frac{\cos\gamma}{\sqrt{4\cos^2\gamma-1}}\right)\mathrm{d}\gamma,$
2	(i)	$1+Q_1\sum_{n=-\infty}^{\infty}\frac{1}{\mid\vec{y}-2\pi n a\hat{z}\mid^6},$
	(ii)	$1+Q_1\sum_{n=-\infty}^{\infty}\frac{1}{\mid\vec{y}-2\pi n a\hat{z}\mid^6},$
	(iii)	$1+Q_1\sum_{n=-\infty}^{\infty}\frac{1}{\mid\vec{y}-2\pi n a z\mid^6},$
3	(i)	$R(e_1)=\epsilon^{-J_{67}+J_{89}},\quad R(e_2)=\epsilon^{J_{45}-J_{89}}.$
	(ii)	$R(e_1)=\epsilon^{-J_{67}+J_{89}},\quad R(e_2)=\epsilon^{J_{45}-J_{89}}.$
	(iii)	$R(e_1)=\epsilon^{-J_0+J_{89}},\quad R(e_2)=\epsilon^{I_4-J_{89}}.$

Fig. 6. Examples of correct prediction by our model. (i) grand truth images, (ii) our model predictions, and (iii) (Vu et al., 2023), (Pang et al., 2021), and (Yan et al., 2020) predictions

in rows 1, 2, and 3, respectively.

Our model's predictions in (ii) consistently align with ground truth images in (i), showcasing its ability to correctly predict complex mathematical expressions. Notably, in all (iii), the highlighted yellow areas in rows 1, 2, and 3 reveal the mistakes made by models (Vu et al., 2023), (Pang et al., 2021), and (Yan et al., 2020),, respectively, where our model correctly predict the LaTeX sequences. Our model showcases strong resilience and excellent accuracy in dealing with challenging mathematical notations that other models have found difficult.

6) Incorrect predictions: Fig. 7 highlights instances where our model made incorrect predictions when converting images to LaTeX. Despite its strong overall performance, our model is not entirely error-free, particularly with long and complex mathematical expressions as shown.



$$\left(\begin{array}{cc} E-m & 2i\left(\frac{\partial}{\partial_z} - ieA_z\right) \\ -2i\left(\frac{\partial}{\partial_{\bar{z}}} - ieA_{\bar{z}}\right) & -E-m \end{array}\right)\left(\begin{array}{c}\chi^1\\ \chi^2\end{array}\right)=0$$

(i)

```
\left( \begin{array} { c c } { E - m } & { 2 i \left( \frac { \partial
} { \partial _ { z } } - i e A _ { z } \right) } \\ { - 2 i \left( \frac
{ \partial } { \partial _ { \bar { z } } } - i e A _ { \bar { z } }
\right) } & { - E - m } \\ \end{array} \right) \left( \begin{array}
{ c } { \chi ^ { 1 } } \\ { \chi ^ { 2 } } \\ \end{array} \right) = 0
```

(ii)

```
\left ( \begin{array} { c c } { E - m } & { 2 i \left( \frac { \partial
} { \partial _ { x } } - i e A _ { z } \right ) } \\ { - 2 i \left( \frac
{ \partial } { \partial _ { z } } - i e A _ { \bar { z } } \right ) } &
{ - E - m } \\ \end {array} \right ) \left ( \begin {array} { c } {
\chi ^ { 1 } } \\ { \chi ^ { 2 } } \\ \end {array} \right ) = 0
```

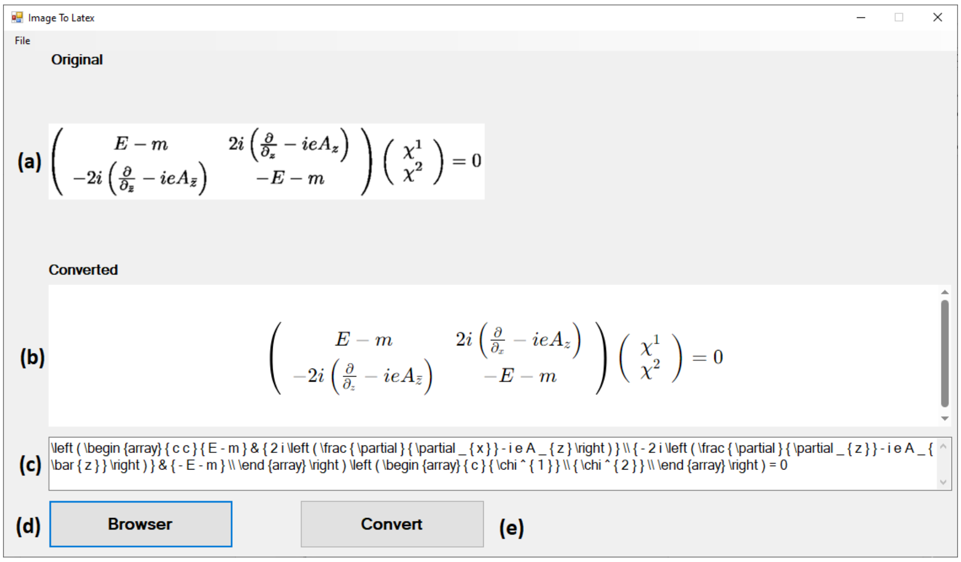
(iii)

Fig. 7. Incorrect prediction by our model. (i) grand truth image, (ii) grand truth LaTeX, and (iii) our model predicted LaTeX

In this evaluation, we observe that our model can misinterpret certain components of intricate expressions. The highlighted areas in the row (iii) illustrate minor mistakes made by our model, which differ from the ground truth in the row (ii). However, even with such complex inputs, our model's errors remain minor, demonstrating its robustness and capability to handle challenging scenarios effectively.

5.1. Deployment and user interface

The user interface of our model, shown in Fig. 8, facilitates real-world deployment by allowing users to easily convert images of mathematical expressions to LaTeX code. The interface displays the original image (a) alongside the predicted image (b) and the generated LaTeX code (c) for users to review. Users can copy and paste images directly or use the browser button (d) to select images, then initiate the conversion process with the convert button (e), making the tool efficient and user-friendly.



```
\left ( \begin {array} { c c } { E - m } & { 2 i \left ( \frac { \partial } { \partial _ { x } } - i e A _ { z } \right ) } \\ { - 2 i \left ( \frac { \partial } { \partial _ { z } } - i e A _ {
\bar { z } } \right ) } & { - E - m } \\ \end {array} \right ) \left ( \begin {array} { c } { \chi ^ { 1 } } \\ { \chi ^ { 2 } } \\ \end {array} \right ) = 0
```

Fig. 8. Model user interface. (a) grand truth image (b) predicted image (c) predicted LaTeX (d) the browser button (e) the convert button.

6. Conclusions and future works

This research introduces a hybrid model designed to recognize printed mathematical expressions and convert them to LaTeX by combining an Inception-based CNN for feature extraction with a Transformer-based encoder-decoder. This method addresses several challenges inherent in traditional RNN-based models, such as the difficulty in capturing long-range dependencies and their slow inference times. Our model demonstrated exceptional performance on the IM2LATEX-100k dataset, achieving a BLEU score of 92.76% and an image edit accuracy of 95.09%, outperforming many existing state-of-the-art models. Additionally, it effectively handled complex mathematical structures, using only 8.58 million trainable parameters, thereby showcasing its computational efficiency.

Even with these promising results, there remains a potential for further enhancement. Future research could explore optimizing the hybrid model further for real-time applications, focusing on reducing inference times even more.

Another approach could involve improving the model's flexibility in applying to different types of mathematical notation and styles, not just those found in the IM2LATEX-100K dataset. Incorporating techniques to handle noisy or low-quality images could enhance its functionality in actual use cases. Expanding the scope to include the recognition of handwritten formulas could significantly increase its versatility. Moreover, adapting this method for recognizing other images that contain structural information, such as musical notation, physics and chemical equations, and chemical molecular formulas, would also be beneficial. Extending the model's support to these diverse applications can enrich its utility across different scientific and academic domains. Overall, our study suggests that leveraging hybrid models can offer significant advancements in mathematical expression recognition, but continued research and optimization are necessary to fully realize their potential.

References

1. Alexander, R. M., Chopra, S., & Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 379–389. https://doi.org/10.18653/V1/D15-1044
2. Anderson, R. H. (1967). Syntax-directed recognition of hand-printed two-dimensional mathematics. Symposium on Interactive Systems for Experimental Applied Mathematics on Proceedings of the Association for Computing Machinery Inc. Symposium -, 436–459. https://doi.org/10.1145/2402536.2402585
3. Ashish, V., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (Vols. 2017-Decem).
4. Bahdanau, D., Cho, K. H., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. https://doi.org/10.48550/arxiv.1409.0473
5. Bender, S., Haurilet, M., Roitberg, A., & Stiefelhagen, R. (2019). Learning fine-grained image representations for mathematical expression recognition. 2019 International Conference on Document Analysis and Recognition Workshops, ICDARW 2019, 1, 56–61. https://doi.org/10.1109/ICDARW.2019.00015
6. Deng, Y., Kanervisto, A., Ling, J., & Rush, A. M. (2017). Image-to-markup generation with coarse-to-fine attention. 34th International Conference on Machine Learning, ICML 2017, 3, 1631–1640. http://arxiv.org/abs/1609.04938
7. Deng, Y., Kanervisto, A., & Rush, A. M. (2016). What You Get Is What You See: A Visual Markup Decompiler. ArXiv.Org.
8. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. North American Chapter of the Association for Computational Linguistics. https://api.semanticscholar.org/CorpusID:52967399
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021 - 9th International Conference on Learning Representations. https://arxiv.org/abs/2010.11929v2
10. Fu, Y., Liu, T., Gao, M., & Zhou, A. (2021). EDSL: An Encoder-Decoder Architecture with Symbol-Level Features for Printed Mathematical Expression Recognition. ArXiv, abs/2007.02517. https://api.semanticscholar.org/CorpusID:220364249
11. Gao, H., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January, 2261–2269. https://doi.org/10.1109/CVPR.2017.243
12. GehrkeJohannes, GinspargPaul, & KleinbergJon. (2003). Overview of the 2003 KDD Cup. ACM SIGKDD Explorations Newsletter, 5(2), 149–151. https://doi.org/10.1145/980972.980992
13. Genthial, G., & Sauvestre, R. (2017). Image to Latex. https://openai.com/
14. Guillaume Genthial, R. S. (2017). Image to Latex.
15. Illium, S., Schillman, T., Müller, R., Gabor, T., & Linnhoff-Popien, C. (2022). Empirical Analysis of Limits for Memory Distance in Recurrent Neural Networks. International

Conference on Agents and Artificial Intelligence, 3, 308–315. https://doi.org/10.5220/0010818500003116
16. Jan, C., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. Advances in Neural Information Processing Systems, 2015-January, 577–585.
17. Jianshu, Z., Du, J., & Dai, L. (2018). Multi-Scale Attention with Dense Encoder for Handwritten Mathematical Expression Recognition. 2018 24th International Conference on Pattern Recognition (ICPR), 2245–2250. https://api.semanticscholar.org/CorpusID:6549978
18. Kanervisto, A. (2016). im2latex-100k , arXiv:1609.04938. https://doi.org/10.5281/ZENODO.56198
19. Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.
20. Lalapura, V. S., Bhimavarapu, V. R., Amudha, J., & Satheesh, H. S. (2024). A Systematic Evaluation of Recurrent Neural Network Models for Edge Intelligence and Human Activity Recognition Applications. Algorithms 2024, Vol. 17, Page 104, 17(3), 104. https://doi.org/10.3390/A17030104
21. Le, A. D., Pham, V. L., Ly, V. L., Nguyen, N. Q., Nguyen, H. T., & Tran, T. A. (2022). A Hybrid Vision Transformer Approach for Mathematical Expression Recognition. 2022 International Conference on Digital Image Computing: Techniques and Applications (DICTA), 1–7. https://doi.org/10.1109/DICTA56598.2022.10034626
22. Levenshtein, V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics. Doklady, 10, 707–710. https://api.semanticscholar.org/CorpusID:60827152
23. Liu, Y., Young, R., & Jafarpour, B. (2023). Long–short-term memory encoder–decoder with regularized hidden dynamics for fault detection in industrial processes. Journal of Process Control, 124, 166–178. https://doi.org/10.1016/j.jprocont.2023.01.015
24. Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing, 1412–1421. https://doi.org/10.18653/v1/d15-1166
25. Mouchère, H., Viard-Gaudin, C., Zanibbi, R., Garain, U., Mouchère, H., Viard-Gaudin, C., Zanibbi, R., & Garain, U. (2014). ICFHR 2014 Competition on Recognition of On-line Handwritten Mathematical Expressions (CROHME 2014). 6 p. https://hal.science/hal-01070712
26. Mouchère, H., Viard-Gaudin, C., Zanibbi, R., Garain, U., Mouchère, H., Viard-Gaudin, C., Zanibbi, R., & Garain, U. (2016). ICFHR 2016 CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions. https://hal.science/hal-01374346
27. Noya, E., Benedí, J. M., Sánchez, J. A., & Anitei, D. (2023). Discriminative estimation of probabilistic context-free grammars for mathematical expression recognition and retrieval. Pattern Analysis and Applications, 26(4), 1571–1584. https://doi.org/10.1007/S10044-023-01158-8/FIGURES/10
28. Oral, B., Emekligil, E., Arslan, S., & Eryiğit, G. (2020). Information Extraction from Text Intensive and Visually Rich Banking Documents. Information Processing and Management, 57(6), 6. https://doi.org/10.1016/j.ipm.2020.102361
29. Orji, E. Z., Haydar, A., Erşan, İ., & Mwambe, O. O. (2023). Advancing OCR Accuracy in Image-to-LaTeX Conversion—A Critical and Creative Exploration. Applied Sciences 2023, Vol. 13, Page 12503, 13(22), 12503. https://doi.org/10.3390/APP132212503
30. Pang, N., Yang, C., Zhu, X., Li, J., & Yin, X. C. (2021). Global context-based network with transformer for Image2latex. Proceedings - International Conference on Pattern Recognition, 4650–4656. https://doi.org/10.1109/ICPR48806.2021.9412072
31. Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). BLEU: A method for automatic

evaluation of machine translation. Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2002-July, 311–318.

32. Sharma, S., & Guleria, K. (2022). Deep Learning Models for Image Classification: Comparison and Applications. 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 1733–1738. https://doi.org/10.1109/ICACITE53722.2022.9823516
33. Shen, Z., Zhang, M., Zhao, H., Yi, S., & Li, H. (2018). Efficient Attention: Attention with Linear Complexities. 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), 3530–3538. https://api.semanticscholar.org/CorpusID:215999966
34. Si, C., Yu, W., Zhou, P., Zhou, Y., Wang, X., & Yan, S. (2022). Inception Transformer. ArXiv, abs/2205.12956. https://api.semanticscholar.org/CorpusID:249062606
35. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 07-12-June-2015, 1–9. https://doi.org/10.1109/CVPR.2015.7298594
36. Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2014). Show and tell: A neural image caption generator. Computer Vision and Pattern Recognition, 07-12-June-2015, 3156–3164. https://doi.org/10.1109/CVPR.2015.7298935
37. Vu, V.-X., Bui, T.-N., Le, T.-L., Phong, B. H., & Hoang, M.-T. (2023). Transformer-based method for mathematical expression recognition in document images. 2022 International Conference on Multimedia Analysis and Pattern Recognition (MAPR), 1–6. https://doi.org/10.1109/MAPR56351.2022.9924740
38. Wang, Z., & Liu, J.-C. S. (2019). Translating Mathematical Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training. ArXiv, abs/1908.11415. https://api.semanticscholar.org/CorpusID:201698307
39. Wang, Z., & 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, 24(1–2), 63–75. https://doi.org/10.1007/s10032-020-00360-2
40. Yan, Z., Zhang, X., Gao, L., Yuan, K., & Tang, Z. (2020). ConvMath: A convolutional sequence network for mathematical expression recognition. Proceedings - International Conference on Pattern Recognition, 4566–4572. https://doi.org/10.1109/ICPR48806.2021.9412913
41. Yang, S., Wang, Y., & Chu, X. (2020). A Survey of Deep Learning Techniques for Neural Machine Translation. ArXiv, abs/2002.07526. https://api.semanticscholar.org/CorpusID:211146401
42. Zhang, J., Du, J., Zhang, S., Liu, D., Hu, Y., Hu, J., Wei, S., & 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. https://doi.org/10.1016/j.patcog.2017.06.017
43. Zhang, N., & Kim, J. (2023). A Survey on Attention mechanism in NLP. 2023 International Conference on Electronics, Information, and Communication (ICEIC), 1–4. https://doi.org/10.1109/ICEIC57457.2023.10049971
44. Zhou, Z., Ji, S., Wang, Y., Weng, Z., & Zhu, Y. (2023). TRMER: Transformer-Based End to End Printed Mathematical Expression Recognition. 2023 International Joint Conference on Neural Networks (IJCNN), 1–6. https://doi.org/10.1109/IJCNN54540.2023.10191139