ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

# Improved Natural Language Processing-Based Optical Character Recognition Model for Image-to-LaTeX Conversion

**Francis Amaechi Okoye[1], Everistus Zeluwa Orji,[2] Godwin Odozo Ozor[3]**

**[1,3] Computer Engineering , Enugu State University of Science and Technology**

**[2] Software Engineering Department, Cyprus Aydin University**

**DOI: https://doi.org/10.51583/IJLTEMAS.2026.151000126**

**Received: 27 October 2026; Accepted: 02 October 2026; Published: 19 November 2026**

## ABSTRACT

Recognition of symbols in mathematical expressions and systematic analysis involving the recognition of basic mathematical expressions (BME) from images are difficult tasks. Many researchers have used optical character recognition (OCR) to support such recognition. However, recognizing such unique mathematical symbols and figuring out where they belong in the formula is still challenging due to the limitations of the existing encoder-decoder architecture that fails to capture the precise position of such symbols and a poor data set. In order to address such a challenge, this study proposes a natural language processing (NLP)-based optical character recognition (OCR) model that converts images of BME to LaTeX markup by using an encoder based on a convolutional neural network (CNN) and a decoder based on a recurrent neural network (RNN). The proposed approach includes an attention mechanism and a novel dataset that enable the encoder-decoder architecture to find the position of the math symbol correctly. The proposed NLP-based OCR model was evaluated using the BLEU score, edit distance, and exact match metrics by compiling the predicted LaTeX markup to produce new math formulas and contrasting them against the ground truth. The NLP-based OCR model achieved a good performance, having achieved 93.22% on the BLEU score, 93.49% on the edit distance, and 78.21 on the exact match.

**Keywords:** Image; LaTeX; encoder-decoder; attention mechanism; GRU; mathematical expression recognition; natural language processing; optical character recognition; recurrent neural network.

## INTRODUCTION

Researchers have tried turning images into labeled categories or structured languages since 1967 (Anderson, 1967). According to (J. Wang et al., 2019), there is currently an increase in research efforts focused on the recognition of complex markup languages like HTML (Hypertext Markup Language), XML (Extensible Markup Language), and LaTeX. This research has significant conceptual significance for the advancement of contemporary OCR systems. In fact, OpenAi has identified "image-to-latex" as a request for research that addresses LaTeX markup language recognition (Ilya et al., 2018; Orji et al., 2023, 2024).

Recent years have seen some improvements in image caption generation and OCR (Jaderberg et al., 2014b). Most of the time, researchers use the sequence-to-sequence approach to complete the task of creating image captions (Ahmad et al., 2023; Martinez et al., 2023). In this process, the encoder creates feature maps from the source sequence (an image), which it then uses to create the target sequence (a caption). An attention model has been added to this architecture (Lin et al., 2023; Vaswani et al., 2017), which chooses a portion of the feature maps to be decoded. Its widespread use encourages researchers to consider mathematical expression recognition as another appropriate application (Bahdanau et al., 2014). These studies, which are based on the architecture of the encoder-decoder (Liu et al., 2023; Sutskever et al., 2014), considered the

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

image of a mathematical expression as the input sequence ("A Hybrid Vision Transformer Approach for Mathematical Expression Recognition," 2022).

The recognition of basic mathematical expressions with different handwriting font styles is regarded as an image-to-sequence task. However, only one font was used in the previous research studies, which makes it hard to recognize with different fonts (Deng et al., 2016; B. Li et al., 2022). But each handwriting-style font for a mathematical expression has its own unique, complex 2-dimensional structure with numbers, letters, and symbols of different shapes and sizes. Thus, it is difficult to train a model with just one font style. In order to address such a difficult problem (Teslya & Mohammed, 2022), the need to create a brand-new dataset including six different handwriting fonts with an improved training model architecture is inevitable (Ilya et al., 2018).

In an attempt to address the challenge, a natural language processing (NLP)-based optical character recognition (NLP-based OCR) has been proposed in this study. NLP-based OCR utilizes OCR techniques to recognize and extract text from images and then uses algorithms to identify the language of the text. The proposed approach uses a convolutional sequence-to-sequence network to convert images of basic mathematical expressions into LaTeX format and adopts a recurrent neural network (RNN) decoder to enhance the recognition effect. Hence, the proposed approach aims at adapting a convolutional sequence-to-sequence network to convert images of basic mathematical expressions into LaTeX format and adopting a recurrent neural network (RNN) decoder to enhance the recognition effect. The following are the major contributions:

- Developed a model that enables the conversion of images of basic mathematical expressions written in six different handwritten font styles into LaTeX markup.
- Unlike previous studies, the proposed NLP-based OCR approach replaces the CNN encoder with a ResNet encoder and adopts an RNN-based decoder with soft attention in between for improved performance.
- Also, the proposed NLP-based OCR model was trained for broad recognition with a newly created dataset containing six different handwritten-style fonts.

The remaining part of this paper is organized as follows: The related works of this study will be stated in Section 2, and the proposed approach will be presented in Section 3. Data set, preprocessing, and experiment set-up evaluation will be described in Section 4. The experimental results will be presented in Section 5 and discussed in Section 6. Finally, Section 7 will conclude this study with future work.

## Related Work

In the past few years, both natural language processing (NLP) and computer vision systems have made a lot of progress. Researchers have made numerous attempts to leverage these advancements in mathematical expression recognition tasks (Lin et al., 2023)(B. Li et al., 2022)(Thakur & Sharma, 2022). The study by (Jaderberg et al., 2014a) demonstrated that a combination of NLP techniques like conditional random fields (CRFs) and neural networks is quite efficient at recognizing words in images. However, (T. Wang et al., 2012) utilized convolutional neural networks (CNNs) for recognizing words in images.

The use of an encoder-decoder model built on RNNs is common in speech recognition, language translation, and other related areas. In (Luong et al., 2015), the author proposed the use of an attention mechanism within the encoder-decoder model to improve accuracy, and a new benchmark for machine translation (MT)

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

or automated translation systems was eventually established. The idea of the encoder-decoder model inspired (Lee & Osindero, 2016) to construct their model in a holistic manner, incorporating RNNs and attention mechanisms, which is typical for an object character recognition task. The same encoder-decoder model used in the image captioning task was used in the image-to-latex task.

The image-to-latex model is capable of generating LaTeX markup corresponding to a given input image of a mathematical expression. Based on the latest work in image captioning and OCR, the same end-to-end model is utilized in mathematical expression recognition in an image. Also, (Deng et al., 2016) used the same method to generate a LaTeX markup of a mathematical expression image as well as an HTML markup of an image. Again, (Deng et al., 2017) also suggested another neural network with the same architecture that could turn images of mathematical expressions into LaTeX alone, this time relying on a new attention mechanism approach known as "coarse-to-fine attention." In the study (Zhang et al., 2017), the author developed the WAP model, also called the watch, attend, and parse model. This model used an encoder-decoder architecture to encode math images and decode LaTeX markup.

The residual network (ResNet) recently performed very well on object classification tasks because feature extraction from the image was improved, and gradient propagation was easier. In the research work of (Z. Li et al., 2020), they improve their encoder-decoder model by encoding the input with ResNet and applying scale augmentation alongside a drop attention mechanism to solve the task of mathematical expression recognition. The problem with this model is that it was trained on a dataset containing only one font, so it cannot recognize other font types when encountered.

Relying on the encoder-decoder baseline architecture, we replaced the CNN encoder with a ResNet encoder and the RNN-based decoder with soft attention in between for improved performance. Our model was trained for broad recognition with a dataset containing six different handwritten-style fonts. The problem with this model, as well as with the previously mentioned related studies, is that they are all trained on a dataset that only contained one font, so they cannot recognize other font types when they are encountered. Also, there is a need for an improved encoder to increase prediction accuracy.

**Proposed NLP-based OCR model**

Taking into account the current encoder-decoder model architecture limitations. The proposed NLP-based OCR model is a hybrid approach that incorporates natural language processing and computer vision components. The NLP-based OCR model adapts the ResNet (He et al., 2016) image encoder to extract features, and to produce LaTeX sequences, the RNN decoder based on gated recurrent units was adopted.

Features were extracted from the images using CNN (Chen et al., 2023), and the extracted feature maps were arranged in grids. After that, a GRU-based RNN decoder equipped with a soft-attention mechanism uses the encoded features to predict the output. The entire model was trained for ten epochs to increase the probability of the observed LaTeX markup.

**Design Requirements**

The NLP-based OCR model is designed to support the conversion of images of basic mathematical expressions into LaTeX format using a sequence-to-sequence network with the aid of a recurrent neural network (RNN) decoder. The proposed NLP-based OCR model has to meet the following functional requirements:

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

- Enable the conversion of images of basic mathematical expressions written in six different handwritten font styles into LaTeX markup.
- Encode images quickly and effectively by using an advanced CNN encoder.
- Decode images quickly and effectively using an RNN decoder.

**Problem Formulation**

Our definition of the image-to-LaTeX problem is the process of converting a rendered source image of a basic mathematical formula into a target presentation of LaTeX markup that accurately describes both the image's content and layout. Let an image-to-LaTeX sequence pair be $(x, y)$. Where $x \in \mathbb{R}^{HxW}$ (source) is a grayscale image with the height $(H)$ and width$(W)$, and $y = [y_1, y_2, y_3, \dots, y_t,]$ (target).

The mathematical formula in the source image is marked up by the actual LaTeX sequence, composed of $t$ tokens. Such that the mathematical expression in source image $x$ is rendered by the LaTeX compiler using target $y$.

**The Encoder**

The goal of the CNN encoder is to make a feature representation of the images it receives that captures their most important features. We have two different types of encoders: one is the standard CNN encoder, and the other is the ResNet-based encoder.

Convolutional Neural Network:The features of the input images were extracted using a CNN, which is made up of the convolution, pooling, and activation layers. A group of kernels that serve as an image filter are convolved with an input image at each convolution layer. Because the kernel values can be trained, the extracted features from the image are not handcrafted but rather data-driven.

The pooling layer uses an average or max-pooling function to down-sample the input image, reducing its spatial dimensions while retaining important information. The activation layer, on the other hand, adds nonlinearity to the network. Most of the time, Rectified Linear Units (ReLU) are used to change negative inputs to zero while leaving positive inputs the same.

**Baseline CNN Encoder**

We utilized a CNN to encode our input images for our baseline model. It contains six CNN layers, with kernels of size$(3 \text{ x } 3)$, padding$(1 \text{ x } 1)$, or 'same', and stride$(1 \text{ x } 1)$ as well. We also include max-pooling layers before each of the last two convolution layers in the model. The function of these max-pooling layers is to decrease the image size while increasing the receptive field size. Table 1 provides more details about the baseline CNN encoder architecture.

**Table 1.** Baseline CNN encoder configuration "Pool": max-pooling layer; "Conv2D": convolution layer; "p": padding; "s": stride size; "k": kernel size; "act": activation; "ReLU": Rectified Linear Units, sizes arranged in (height, width).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

| Layer | Output-shape | Channels | Parameters |
|---|---|---|---|
| Input (image) | 60x200 | 1 | - |
| Conv2D | 60x200 | 16 | k:(3,3), p:(same), s:(1,1), act:ReLU |
| Conv2D | 60x200 | 32 | k:(3,3), p:(same), s:(1,1), act:ReLU |
| Conv2D | 60x200 | 64 | k:(3,3), p:(same), s:(1,1), act:ReLU |
| Conv2D | 60x200 | 64 | k:(3,3), p:(same), s:(1,1), act:ReLU |
| Pool | 30x200 | 64 | k:(2,1), p:(valid), s:(2,1) |
| Conv2D | 30x200 | 128 | k:(3,3), p:(same), s:(1,1), act:ReLU |
| Pool | 30x100 | 128 | k:(1,2), p:(valid), s:(1,2) |
| Conv2D | 28x98 | 128 | k:(3,3), p:(valid), s:(1,1), act:ReLU |

Our baseline CNN model processes an image that is $(H \times W)$ in size and generates a feature map that is $(\acute{H} \times \acute{W} \times C)$ in dimensions.$\acute{H}$stands for the feature map's height,$\acute{W}$its width, and the number of channels in the final convolution layer is represented by C.

The vector $v_i \in \mathbb{R}^C$ is defined by the encoder for each of $(\acute{H} \times \acute{W})$ areas. It $(v_i)$ could be information such as the presence of a shape, token, or operation.

**ResNet Encoder**

In this proposed residual network (ResNet) model of our encoder, the ResNet of (He et al., 2016) is modified to suit our image encoder. The encoder is primarily made up of four residual blocks, as seen in Figure 1, where "Conv2D": convolutional layer, "c": number of channels, "p": padding, "s": stride size, and "ReLU": rectified linear units, while sizes are arranged in (height, width).

With a $3 \times 3$ kernel size, each residual block is made up of 2 layers of convolutions stacked together. A $5 \times 5$ convolution with a stride of $1 \times 1$ is employed in the skip connection to match the output dimension of each block of residual connections' output.

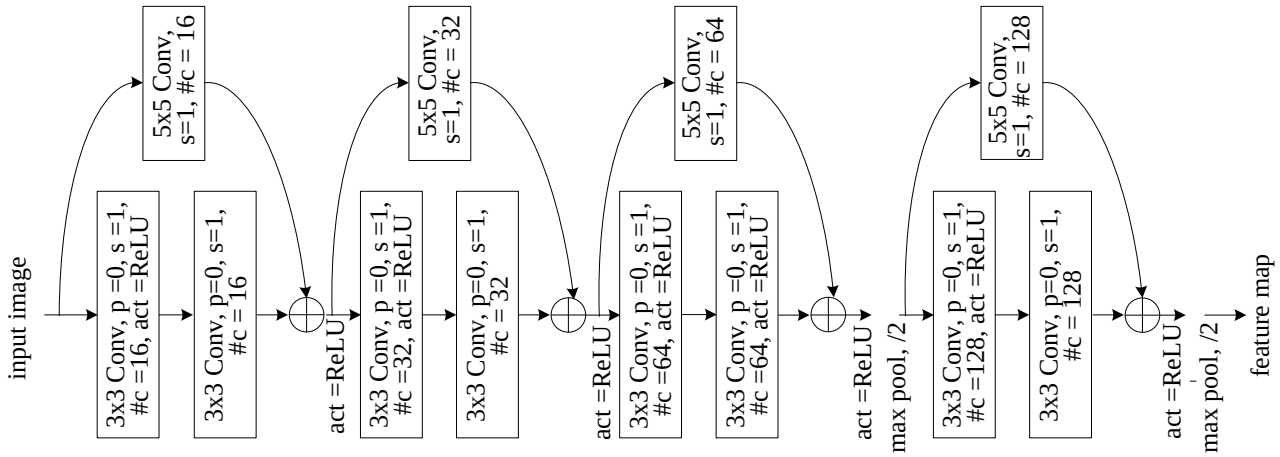



**Figure 1. ResNet image Encoder Configuration.**

The ReLu activation function is applied to some CNN layers in the residual network block but not in the skip connection. After concatenating each residual block with the skip block, an activation function is also

applied, as shown in Figure 1.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

**The Decoder**

This section describes how the recurrent neural network (RNN) decoder uses the encoder output feature maps $V = \{v_1, v_2, v_3, \ldots, v_{\acute{H} \times \acute{W}}\}$[see Figure 2] as input to predict the LaTeX sequence $Y = \{y_1, y_2, y_3, \ldots, y_t\}$ under ground truth supervision of $W = \{w_1, w_2, w_3, \ldots, w_n\}$. Our model decoder makes use of a GRU-based recurrent neural network (RNN) network (Cho, van Merrienboer, Gulcehre, et al., 2014). GRUs and LSTMs have both been shown to be highly effective at managing long-term dependencies, in contrast to the vanilla RNN, which experiences difficulties with the vanishing or exploding gradient problem as the sequence size increases (Niu et al., 2023; Schaefer et al., 2008).

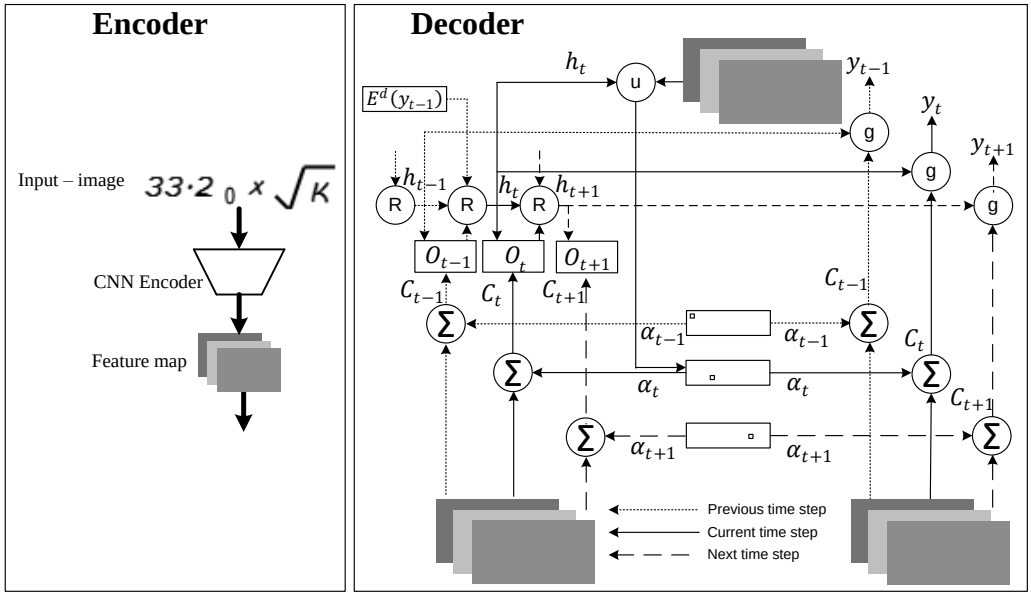



**Figure 2. Propoased NLP-based OCR model for BMER.**

The GRU at each time step $t$ in general takes in both $(x_t)$ and the hidden state from the preceding GRU time step $(h_{t-1})$ as inputs and outputs a new hidden state $(h_t)$ [Equation 4] that serves as input for the next GRU unit. The core of the GRU is the current memory content, which keeps track of information that was seen in a previous time step. The GRU uses the reset gate$(r_t)$ [Equation 2] to control how much information from the past is discarded, and it uses the update gate $(z_t)$ [Equation 1] to control how much previous information is added to the current memory. Each gate is controlled by the sigmoid neural layer. The following expressions show how this works.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (1)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (2)$$

$$h'_t = tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \quad (3)$$

$$h_t = z_t \odot h'_t + (1 - z_t) \odot h_{t-1} \quad (4)$$

where$h'_t$ is GRU's candidate activation vector [Equation 3], the sigmoid and hyperbolic activation functions are represented by "$\sigma$" and "$tanh$" respectively, and the weight matrix is represented by $W$.

It is common for the initial cell state $(c_0)$ and hidden state $(h_0)$ of an RNN decoder in NLP applications (Bitterman et al., 2023; Khurana et al., 2022) to originate from an RNN encoder output. Unlike the RNN

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

encoder, the output required for initial states is not produced by our model's encoder, a CNN. To fix this, extra layers were added to train the initial cell state and hidden, as shown in equations 5 and 6, respectively, based on the output of the CNN encoder.

$$c_0 = tanh\left(W_c * \left(\frac{1}{Q}\sum_{i=1}^{Q} v_i\right) + b_c\right) \quad (5)$$

$$h_0 = tanh\left(W_h * \left(\frac{1}{Q}\sum_{i=1}^{Q} v_i\right) + b_h\right) \quad (6)$$

where $Q = \left(\acute{H}\ x\ \acute{W}\right)$.

**Attention Model**

The author (Vaswani et al., 2017) says that the attention mechanism makes it possible for the decoder to give the most weight to the most important parts of the feature vectors when making LaTeX tokens. According to (J. Wang et al., 2019), we used the soft attention approach to make our RNN decoder work better.

For every time step $t$, the attention mechanism produces a unique context vector $c_t$ which is a weighted sum of the feature vectors [Equation 7].

$$c_t = \sum_{i=1}^{Q} \alpha_{it} v_i \quad (7)$$

The context vector $c_t$ tells the model which parts of the feature vector to pay attention to at each time step. This provides the model with the necessary information it needs from the image input. An extra layer is used to compute the attention weights that utilize the feature vector $V$ and the GRU's hidden state $h_t$ as inputs.

$$e_{it} = \beta^T tanh(W_h h_{t-1} + W_v v_i) \quad (8)$$

$$\alpha_{it} = softmax(e_{it}) \quad (9)$$

The element in vector alpha denotes where the attention model should pay attention [Equation 9], while the softmax function produces a probability distribution that sums to one.

For the context vector$c_t$ information in [Equation 7] to be incorporated into the RNN, based on current GRU hidden vector$h_t$ and the context vector $c_t$, an output vector $o_t$ was computed [Equation 10]. It is also used in calculating the next token's probability distribution.

$$o_t = tanh(W^c[h_t, c_t]) \quad (10)$$

As an extra step, we used the (Luong et al., 2015) input-feeding method, which combines the word embedding of $Ey_{t-1}$ and the output vector $o_{t-1}$ to serve as the RNN input [Equation 11].

$$h_t = GRU(h_{t-1}, [Ey_{t-1}, o_{t-1}]) \quad (11)$$

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

As soon as the decoder has calculated an attention vector for each time step, it then uses a recursive formula to generate a probability distribution over the subsequent token, as shown in [Equation 12].

$$p(y_t) = f(y_{t-1}, h_{t-1}, c_t) \quad (12)$$

where $p$ represents the probability distribution, $y_{t-1}$ represents the previous token, $h_{t-1}$ represents the GRU hidden state, and $c_t$ is an attention context vector.

**Decoding approaches**

The GRU generates a vector $y_t$ with dimension $V$, where $V$ denotes the vocabulary size, at every time step $t$. This vector $y_t$ indicates the distribution probability for the next word over the vocabulary. From the information included in the vector $y_t$, we need to select one of the tokens for output and use it as an input for the subsequent step. We tried to solve this problem in two different ways.

**Greedy search approach**

One common approximate method is to use greedy search prediction, which means that at every time step, one token whose probability score is the highest is selected. This method has the advantage of being very fast, but the final sequences might be suboptimal (Goldberg, 2017). In fact, as an approximation search method, the beam search method works much better than the greedy method.

**Beam search approach**

We used a beam search strategy to identify the best output sentence in an effort to close the gap between the results of a greedy search and the best possible solution. Now, we retain in the memory at each time step $k$ hypotheses (i.e., sequences of tokens) that have the highest probability, where $k$ stands for beam size $(for\ example, \mathrm{k} = 5)$. At each time step t, the k tokens are inputted into the decoder to compute the k vector $\mathrm{y_t}$ and then update the token list that we will retain while maintaining a record of the parents of the k tokens that have the highest probability among the k vectors $\mathrm{y_t}$. Over the decoding time steps, what we aim to maximize is the product of each token's probabilities within a sentence given its parent. So, when the decoding phase is complete, an optimal sentence can be reconstructed and output. Note that the greedy approach is the same as having a beam search of 1 (i.e., k=1).

**Dataset and Preprocessing**

**Dataset (Im2latex-basicmath)**

We developed a C# (C-Sharp) program to generate our image to LaTeX for basic math equations (im2latex-basicmath) dataset that comprises a total of 210k basic mathematical images split into two: train sets (180k) and test sets (30k) (Orji & Alienyi, n.d.), as shown in Table 3.

The dataset was generated using six different handwriting font styles, as shown in Table 3, which gives it an edge over the publicly accessible im2latex-100k (Kanervisto, 2016; Yuan et al., 2022) dataset that used only one font. The dataset also contains images of size 60px in height and 200 px in width with white equations on black backgrounds and font sizes of 12px, as well as their respective LaTeX markups.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

**Table 3. Summary of image to latex for basic math (Im2latex-bacismath) dataset.**

| Font Type | Training Set | Test Set |
|---|---|---|
| Cambria | 30K | 5K |
| Comic Sans MS | 30K | 5K |
| Ink Free | 30K | 5K |
| MV Boli | 30K | 5K |
| Segeo Print | 30K | 5K |
| Segoe Script | 30K | 5K |
| **Total** | **180K** | **30K** |

**Tokenization**

The first step in training our model is to construct a "token vocabulary," or $\overline{V}$. This can be achieved by making tokens out of our LaTeX markup in the im2latex-basicmath dataset, as shown in Table 4. Treating each character as a single token is the simplest way to tokenize the LaTeX sources. While parsing the smallest reserved LaTeX words from the LaTeX sources is a more advanced method. For instance, "\sqrt" or "√" in LaTeX will be rendered as one token instead of five individual tokens: "\" "s" "q" "r" "t." The second method offers the obvious benefits of shortening the sequence and minimizing pointless computations and prediction errors. But this method is hard to use because it requires a complete list of all reserved keywords in LaTeX and a good way to parse our LaTeX sources into pieces. In this case, we used the LaTeX parser created by (Deng et al., 2016). By utilizing KaTex (Emily & Sophie Alpert, n.d.), the parser transforms the LaTeX source into a conceptual syntax tree and tokenizes the tree through repeated iteration.

The vocabulary $\overline{V}$ now includes the two utility tokens "START" and "PAD," which represent, respectively, the start and padding sequences. The "start" token initializes the decoder and will continue to make predictions until all 48 maximum token lengths in a single formula are predicted. To facilitate training, the <PAD> token was used to pad the length of the formulas to a maximum of 48. Table 4 shows our final word map or vocabulary size of 77.

**Table 4. Image to LaTeX for Basic Math (im2latex-bacis math) Word Map (Vocabulary) ID is the token index in the word map, and TKN standz for token.**

| TKN | ID | TKN | ID | TKN | ID | TKN | ID | TKN | ID | TKN | ID |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ! | 0 | 4 | 13 | E | 26 | R | 39 | a | 52 | r | 65 |
| % | 1 | 5 | 14 | F | 27 | S | 40 | b | 53 | s | 66 |
| ( | 2 | 6 | 15 | G | 28 | T | 41 | c | 54 | t | 67 |
| ) | 3 | 7 | 16 | H | 29 | U | 42 | d | 55 | u | 68 |
| * | 4 | 8 | 17 | I | 30 | V | 43 | e | 56 | v | 69 |
| + | 5 | 9 | 18 | J | 31 | W | 44 | f | 57 | w | 70 |
| - | 6 | < | 19 | K | 32 | X | 45 | g | 58 | x | 71 |
| . | 7 | = | 20 | L | 33 | Y | 46 | h | 59 | y | 72 |
| / | 8 | > | 21 | M | 34 | Z | 47 | j | 60 | z | 73 |
| 0 | 9 | A | 22 | N | 35 | \\frac | 48 | k | 61 | { | 74 |
| 1 | 10 | B | 23 | O | 36 | \\sqrt | 49 | m | 62 | } | 75 |
| 2 | 11 | C | 24 | P | 37 | ^ | 50 | n | 63 | PAD | 76 |
| 3 | 12 | D | 25 | Q | 38 | _ | 51 | p | 64 | START | |

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

**Evaluation Criteria and Baseline**

The model evaluation was carried out using the exact match (EM), edit distance (ED), and bilingual evaluation understudy or BLEU score (BS) evaluation matrices. The exact match score is one if all the tokens in the ground truth and predicted formulas are exactly the same. The BLEU methodology (Papineni et al., 2002) considers a translation to be of a high standard if the system's translations are similar to those of machine translation. It captures the n-gram overlap present in machine translations as well as reference translations. The most closely related human translation evaluation matrices have been found to be BLEU and serve as translation benchmark indicators. Additionally, we used ED to measure how much of the predicted text matches the ground truth. The edit distance determines the number of letters we need to add, delete, or modify from one translation to get the other. The edit distance for a perfect match is 1.

Using these evaluation criteria as a basis, our proposed model is compared to InftyReader (Suzuki et al., 2003) and three recent deep learning-based works: DenseNet (J. Wang et al., 2019), WYGIWYS (Deng et al., 2016), and ConvMath (Yan et al., 2020).

**Detail Implementation**

Given that the size of our vocabulary is just 77, as shown in Table 4, we decided on a token embedding size of 20. Both the RNN hidden states and the CNN feature maps have their dimensions set to$\mathrm{dim} = 128$. During the training of our model, we utilized the Adam optimization gradient descent algorithm (Kingma & Ba, 2014) with a 0.1 learning rate. As a result of GPU memory limitations, a batch size of 20 is used.

To decrease the possibility of being confined to undesirable solutions, predictions are made using greedy search and beam search (Graves, 2012) during the testing phase. At each time step, beam search takes from the vocabulary the “k” tokens that have the highest probabilities.

When all “k” predicted tokens become <END>, no new predictions are made by the model. We use in the model a beam size of $\mathrm{k} = 5$. Over the decoding time steps, what we aim to maximize is the product of each token’s probabilities within a sentence given its parent. Note that the greedy approach is the same as having a beam search of 1 (i.e., k=1).

TensorFlow version 2.9.1 is used to implement the entire system (Abadi et al., 2016) and generate a deep learning model. The model was developed using an NVIDIA RTX A4000 GPU, a 16-core CPU, and 90GB of RAM.

**Experimental Results**

Table 5 holds the comprehensive performance results for basic mathematical expressions. The obtained results from the proposed model experiment have been compared with several other models, including baseline CNN, ResNet, WYGIWYS, DenseNet, ConvMath, and InftyReader.

The performance of the proposed model is shown in the top four rows. All five deep learning models, including ours, demonstrated better performance than the InftyReader system [see Table 5]. The best results across two of the three evaluation metrics were: 93.22% BLEU score, 93.49% edit distance, and 78.21% exact match [see Table 5]. As compared to greedy decoding, beam search is somewhat more effective in all performance matrices in both models.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

**Table 5. The Im2latex-bacismath performance evaluation.**

| Model | Decoding | BLEU | Edit Distance | Exact Match |
|---|---|---|---|---|
| Baseline CNN | Greedy search | 84.38 | 85.68 | 57.95 |
| Baseline CNN | Beam search | 86.73 | 87.47 | 60.93 |
| ResNet | Greedy search | 92.02 | 92.55 | 76.41 |
| **ResNet** | **Beam search** | **93.22** | **93.49** | **78.21** |
| | | | | |
| WYGIWYS [11] | - | 87.73 | 87.6 | 77.46 |
| DenseNet [2] | - | 88.25 | 91.57 | 37.09 |
| ConvMath [31] | - | 88.33 | 90.88 | 83.41 |
| InftyReader [30] | - | 66.65 | 53.82 | 15.6 |

**Evaluation metrics distribution:** The BLEU score and edit distance continuous distributions of the mathematical formulas are shown in Figure 3, with roughly 81% of the formulas having a BLEU score of 1 and requiring no edit (i.e., most are rightly rendered). These two evaluation matrices indicate that our model renders mathematical images relatively well.

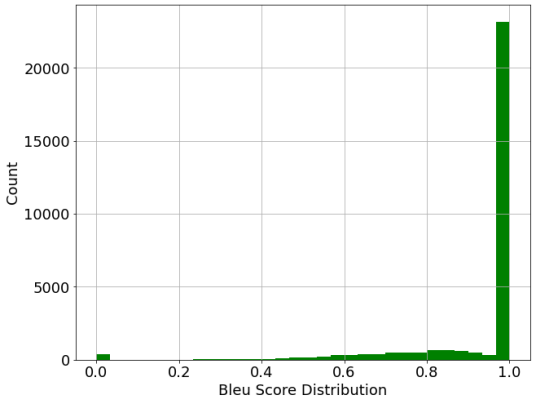



**Figure 3(a). ResNet model with beam search decoding BLEU score distribution.**

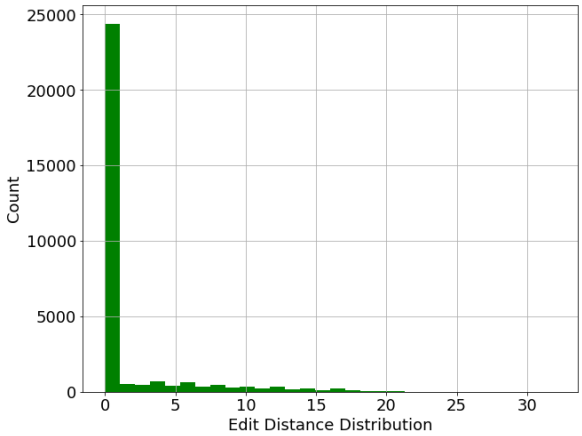



**Figure 3(b). ResNet model with beam search decoding BLEU score distribution.**

Figure 3(a) shows that our model obtained a BLEU score of one in roughly 91% of the 30K test data. While Figure 3(b) indicated that about 81.6% of our model predictions got zero edit distance, meaning that the predicted mathematical expression is exactly the same as the actual expression.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

**Effect of formula length:** Additionally, we looked at how formula length affected the performance of our models. As observed, our model's performance decreased as the length of our sentences increased, as shown in Figure 4. This will impact long mathematical expressions negatively, as they will have a low BLEU score (Figure 4a), an increase in edit distance (Figure 4b), and a low percentage of edit text (Figure 4c).

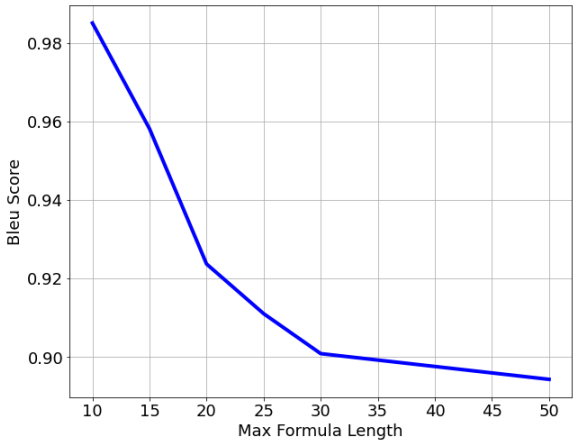



**Figure 4(a). Model performance BLEU score evaluation as a function of formula length.**

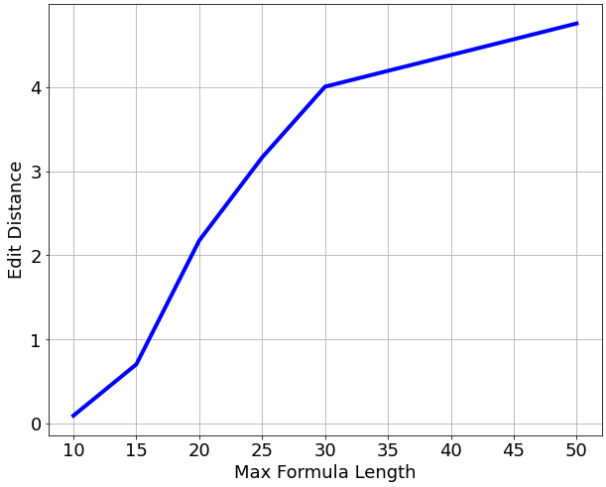



**Figure 4(b). Model performance Edit distance evaluation as a function of formula length.**

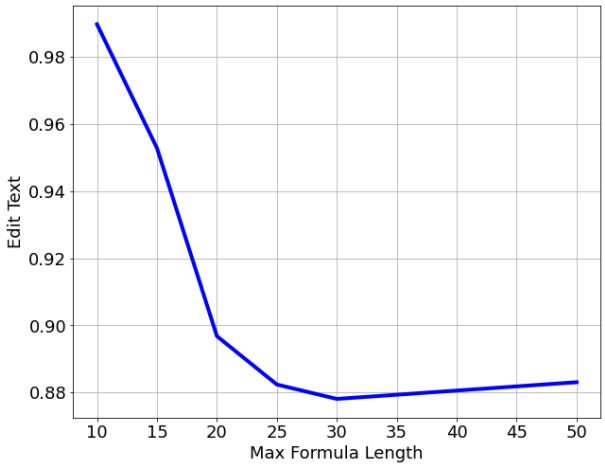



**Figure 4(c). Model performance Edit text evaluation as a function of formula length.**

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

**Performance of individual fonts:** Furthermore, we compared the performance of our model against the six handwritten font styles in the dataset. The results shown in Table 6 are in conformity with our initial results, which show that beam search decoding has a better result when compared to greedy search decoding in all six fonts. Also, our ResNet model has better performance than the baseline CNN model in all three evaluation metrics: BLEU score, Edit Distance (ED), and Exact Match (EM) with respect to all six fonts. With the results obtained from these two analyses, beam search decoding is better than greedy search, and the ResNet model is better than baseline CNN, as supported by the results obtained in our comparisons with other model analyses.

**Table 6. Our models' performance on individual fonts F1: Cambria; F1: Comic Sans MS; F1: Ink Free; F1: MV Boli; F1: Segeo Print; F1: Segoe Script.**

| Greedy search | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Baseline CNN | | | | | | ResNet | | | | | |
| | F1 | F2 | F3 | F4 | F5 | F6 | F1 | F2 | F3 | F4 | F5 | F6 |
| BLEU | 86.99 | 84.66 | 86.93 | 84.96 | 80.65 | 82.01 | 92.56 | 92.19 | 92.36 | 92.10 | 91.22 | 91.69 |
| ED | 88.42 | 86.42 | 88.50 | 86.61 | 81.26 | 82.83 | 93.20 | 92.70 | 92.69 | 92.57 | 91.79 | 92.34 |
| EM | 64.38 | 60.02 | 63.44 | 59.16 | 48.66 | 52.04 | 78.64 | 77.56 | 77.60 | 77.04 | 73.72 | 73.92 |
| **Beam search** | | | | | | | | | | | | |
| | Baseline CNN | | | | | | ResNet | | | | | |
| | F1 | F2 | F3 | F4 | F5 | F6 | F1 | F2 | F3 | F4 | F5 | F6 |
| BLEU | 89.29 | 87.23 | 88.97 | 87.19 | 82.88 | 84.28 | 93.73 | 93.29 | 93.67 | 93.32 | 92.27 | 92.67 |
| ED | 90.21 | 88.32 | 89.96 | 88.14 | 83.38 | 84.80 | 94.05 | 93.80 | 93.85 | 93.44 | 92.64 | 93.18 |
| EM | 67.74 | 62.92 | 66.30 | 61.88 | 51..56 | 55.18 | 80.22 | 79.52 | 79.62 | 78.78 | 75.44 | 75.72 |

Table 6 demonstrates that, on average, our model performed best on F1, then F3, F2, F4, F6, and F5. The following metrics show how well our model worked in each of these fields: BLEU score: F1: 93.73%, F3: 93.67%, F4: 93.32%, F2: 93.29%, F6: 92.67%, and F5: 92.27%. Edit Distance (ED): F1: 94.05%, F3: 93.85%, F2: 93.80%, F4: 93.44%, F6: 93.18%, and F5: 92.64%. Exact match (EM): F1: 80.22%, F3: 79.62%, F2: 79.52%, F4: 78.78%, F6: 75.72%, and F5: 75.44%.

## DISCUSSION

This section discussed how well different deep learning models could recognize basic mathematical expressions. The study found that all five deep learning models, including ours, outperformed the InftyReader system. Specifically, the ResNet model with beam search decoding produced the best results, with a 93.22% BLEU score, 93.49% edit distance, and 78.21% exact match.

However, the study also found that the ResNet model's performance decreased as the length of the mathematical expressions increased, which negatively impacted long mathematical expressions, resulting in a low BLEU score, increased edit distance, and a low percentage of edit text. This decrease in performance is attributed to the decrease in the performance of neural encoder-decoder models with increasing sequence length.

Evaluation metrics distribution using BLEU score and edit distance were also presented. It showed that approximately 81% of the formulas had a BLEU score of 1, meaning that the machine-generated formula

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

was identical to the human-generated formula. While it also suggests that most of the formulas required no editing, indicating that the machine-generated formulas were accurate and required minimal correction. Overall, the continuous distributions of the BLEU score and edit distance indicate that the model performs relatively well in rendering basic mathematical images, with most of the formulas being accurately rendered.

The length of the formulas was also looked at, and the results show that the model's performance got worse as the expressions got longer. This was because the performance of neural encoder-decoder models got worse as the length of the sequence got longer (Cho, van Merrienboer, Bahdanau, et al., 2014). This negatively impacted long mathematical expressions, resulting in a low BLEU score, increased edit distance, and a low percentage of edit text.

The study also evaluated the model's performance in recognizing six different handwritten font styles. The results showed that the ResNet model with beam search decoding outperformed the baseline CNN model and achieved better results than greedy search decoding across all six handwritten font styles. The study discovered that the ResNet model performed best on the F1 font style, then on F3, F2, F4, F6, and F5.

Overall, the study tells us a lot about how well deep learning models work when recognizing mathematical expressions and handwritten font styles.

## CONCLUSIONS

Our research presents a novel and efficient implementation of an object character recognition (OCR) system capable of generating LaTeX from an image of a basic mathematical expression. This is a challenging task, as the OCR system needs to recognize symbols, characters, and their placement in the expression to generate accurate LaTeX code. We improved the OCR system's performance when used for the image2latex task by adding a modified ResNet encoder. The ResNet encoder helps to extract and represent high-level features from the input image, which are then fed to the decoder to generate the LaTeX code.

We made the im2latex-bacismath dataset to test how well our OCR system worked. This dataset is a benchmark for converting images to LaTeX with different font styles. The dataset has a variety of basic mathematical expressions in different font styles and sizes, as well as handwritten-style symbols. We used this dataset to train our OCR system, and it got the best possible score on the im2latex-bacismath benchmark, which shows that our approach works.

Also, this research gives a new way of looking at the object character recognition task. It shows that data-driven models like ours can work even if they don't know the original language, such as LaTeX or HTML. For example, our approach shows that data-driven models can effectively learn the underlying structure of mathematical expressions and generate accurate LaTeX code without the need for explicit rules or knowledge of the LaTeX syntax.

Looking ahead, we suggest several possible directions for future work. First, we could expand the dataset to include more handwritten fonts and symbols, which can further improve the OCR system's performance in recognizing diverse handwriting styles. Second, we could explore the use of more advanced CNNs in the decoder to extract even more high-level features from the input image. Finally, we could explore the potential application of our model in an image caption generator, where it could be used to generate captions from images containing mathematical expressions.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

**Author Contributions:** The manuscript was written through the contributions of all authors.

Conceptualization, E.Z.O., A.H. and İ.E.; methodology, E.Z.O., **A**.H. and İ.E.; software, E.Z.O. and O.O.M.; validation, E.Z.O., A.H. and İ.E.; formal analysis, E.Z.O., **A**.H. and İ.E.; investigation, E.Z.O.; resources, E.Z.O. and O.O.M.; writing—original draft preparation, E.Z.O., O.O.M., A.H. and İ.E.; writing—review and editing, E.Z.O., O.O.M., A.H. and İ.E., supervision, A.H. and İ.E.; All authors have read and agreed to the published version of the manuscript.

## REFERENCES


1. A Hybrid Vision Transformer Approach for Mathematical Expression Recognition. (2022). 2022 International Conference on Digital Image Computing: Techniques and Applications (DICTA). https://doi.org/10.1109/DICTA56598.2022.10034626
2. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). TensorFlow: A System for Large-Scale Machine Learning. Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, 265–283.
3. Ahmad, R. A., Azhar, M., & Sattar, H. (2023). An Image captioning algorithm based on the Hybrid Deep Learning Technique (CNN+GRU). https://doi.org/10.48550/arxiv.2301.02440
4. Anderson, R. H. (1967). Syntax-directed recognition of hand-printed two-dimensional mathematics. Interactive Systems for Experimental Applied Mathematics - Proceedings of the Association for Computing Machinery Inc. Symposium, ACM 1967, 436–459. https://doi.org/10.1145/2402536.2402585
5. 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
6. Bitterman, D. S., Goldner, E., Finan, S., Harris, D., Durbin, E. B., Hochheiser, H., Warner, J. L., Mak, R. H., Miller, T., & Savova, G. K. (2023). An end-to-end natural language processing system for automatically extracting radiotherapy events from clinical texts: NLP to extract radiotherapy events from text. International Journal of Radiation Oncology*Biology*Physics. https://doi.org/10.1016/J.IJROBP.2023.03.055
7. Chen, L., Chen, W., Wang, L., Zhai, C., Hu, X., Sun, L., Tian, Y., Huang, X., & Jiang, L. (2023). Convolutional neural networks (CNNs)-based multi-category damage detection and recognition of high-speed rail (HSR) reinforced concrete (RC) bridges using test images. Engineering Structures, 276, 115306. https://doi.org/10.1016/J.ENGSTRUCT.2022.115306
8. Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.
9. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.
10. Deng, Y., Kanervisto, A., Ling, J., & Rush, A. M. (2017). Image-to-Markup Generation with Coarse-to-Fine Attention.
11. Deng, Y., Kanervisto, A., & Rush, A. M. (2016). What You Get Is What You See: A Visual Markup Decompiler. CoRR, abs/1609.04938.
12. Emily, E., & Sophie Alpert. (n.d.). KaTeX – The fastest math typesetting library for the web. Retrieved December 21, 2022, from https://katex.org/

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026

13. Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. Synthesis Lectures on Human Language Technologies. https://doi.org/10.1007/978-3-031-02165-7
14. Graves, A. (2012). Sequence Transduction with Recurrent Neural Networks.
15. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December, 770–778. https://doi.org/10.1109/CVPR.2016.90
16. Ilya, S., SchulmanTim, S., & Durk, K. (2018, January 31). Requests for Research 2.0. https://openai.com/blog/requests-for-research-2/#im2latex
17. Jaderberg, M., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014a). Deep Structured Output Learning for Unconstrained Text Recognition. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. https://doi.org/10.48550/arxiv.1412.5903
18. Jaderberg, M., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014b). Reading Text in the Wild with Convolutional Neural Networks. International Journal of Computer Vision, 116(1), 1–20. https://doi.org/10.48550/arxiv.1412.1842
19. Kanervisto, A. (2016). im2latex-100k , arXiv:1609.04938. https://doi.org/10.5281/ZENODO.56198
20. Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022). Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications, 1–32. https://doi.org/10.1007/S11042-022-13428-4/FIGURES/3
21. Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization.
22. Lee, C.-Y., & Osindero, S. (2016). Recursive Recurrent Nets with Attention Modeling for OCR in the Wild.
23. Li, B., Yuan, Y., Liang, D., Liu, X., Ji, Z., Bai, J., Liu, W., & Bai, X. (2022). When Counting Meets HMER: Counting-Aware Network for Handwritten Mathematical Expression Recognition.
24. Li, Z., Jin, L., Lai, S., & Zhu, Y. (2020). Improving Attention-Based Handwritten Mathematical Expression Recognition with Scale Augmentation and Drop Attention.
25. Lin, Z., Li, J., Yang, F., Huang, S., Yang, X., Lin, J., & Yang, M. (2023). Spatial Attention and Syntax Rule Enhanced Tree Decoder for Offline Handwritten Mathematical Expression Recognition. ArXiv, 13639 LNCS, 213–227. https://doi.org/10.1007/978-3-031-21648-0_15
26. 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/https://doi.org/10.1016/j.jprocont.2023.01.015
27. Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation.
28. Martinez, Q., Chen, C., Xia, J., & Bahai, H. (2023). Sequence-to-Sequence Change-Point Detection in Single-Particle Trajectories via Recurrent Neural Network for Measuring Self-Diffusion. Transport in Porous Media, 147(3), 679–701. https://doi.org/10.1007/S11242-023-01923-7/FIGURES/21
29. Niu, Z., Zhong, G., Yue, G., Wang, L.-N., Yu, H., Ling, X., & Dong, J. (2023). Recurrent attention unit: A new gated recurrent unit for long-term memory of important parts in sequential data. Neurocomputing, 517, 1–9. https://doi.org/https://doi.org/10.1016/j.neucom.2022.10.050
30. Orji, E. Z., & Alienyi, D. C. (n.d.). Image-to-LaTeX Basic Mathematical Equation. Retrieved December 21, 2022, from https://github.com/everistus-orji/image-to-latex-basic-math-equation/blob/main/dataset.zip

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue X, October 2026


31. Orji, E. Z., Haydar, A., & Erşan, İ. (2024). Inception-Transformer: End-To-End Printed Mathematical Expression Recognition Using Convolutional and Transformer Networks. Nanotechnology Perceptions, 20(6), 2672–2690. https://doi.org/10.62441/NANO-NTP.VI.3274
32. 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
33. Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. Annual Meeting of the Association for Computational Linguistics.
34. Schaefer, A. M., Udluft, S., & Zimmermann, H. G. (2008). Learning long-term dependencies with recurrent neural networks. Neurocomputing, 71(13–15), 2481–2488. https://doi.org/10.1016/J.NEUCOM.2007.12.036
35. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. Advances in Neural Information Processing Systems, 4(January), 3104–3112. https://doi.org/10.48550/arxiv.1409.3215
36. Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., & Kanahori, T. (2003). INFTY - An integrated OCR system for mathematical documents. In C. Vanoirbeek, C. Roisin, & E. Munson (Eds.), Proceedings of the 2003 ACM Symposium on Document Engineering (pp. 95–104).
37. Teslya, N., & Mohammed, S. (2022). Deep Learning for Handwriting Text Recognition: Existing Approaches and Challenges. 2022 31st Conference of Open Innovations Association (FRUCT), 2022-April, 339–346. https://doi.org/10.23919/FRUCT54823.2022.9770912
38. Thakur, U., & Sharma, A. (2022). Offline Handwritten Mathematical Recognition using Adversarial Learning and Transformers. ArXiv. https://doi.org/10.48550/ARXIV.2208.09662
39. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need.
40. Wang, J., Sun, Y., & Wang, S. (2019). Image To Latex with DenseNet Encoder and Joint Attention. Procedia Computer Science, 147, 374–380. https://doi.org/10.1016/J.PROCS.2019.01.246
41. Wang, T., Wu, D. J., Coates, A., & Ng, A. (2012). End-to-end text recognition with convolutional neural networks. Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012).
42. Yan, Z., Zhang, X., Gao, L., Yuan, K., & Tang, Z. (2020). ConvMath: A Convolutional Sequence Network for Mathematical Expression Recognition. 2020 25th International Conference on Pattern Recognition (ICPR), 4566–4572.
43. Yuan, Y., Liu, X., Dikubab, W., Liu, H., Ji, Z., Wu, Z., & Bai, X. (2022). Syntax-Aware Network for Handwritten Mathematical Expression Recognition. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022-June, 4543–4552. https://doi.org/10.1109/CVPR52688.2022.00451
44. 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/https://doi.org/10.1016/j.patcog.2017.06.017