Recognition Of Mathematical Symbols From Images and Test Papers

Viraj Shah[1], Vaibhav Shah[2], Rishikesh Sharma[3] Soham Kulkarni[4] Durgam Devani[5] Arjun Jaiswal[6]

[1,2,3,4]Dwarkadas J. Sanghvi College Of Engineering , Mumbai, India
[5]St Francis Institute of Technology, Mumbai, India
[6] Professor of Dwarkadas J. Sanghvi College Of Engineering , Mumbai, India

---

Abstract - *In this work, the automatic generation of LaTeX code coding for mathematical equations based on images is described. It can be tedious and inaccurate to code math equations in LaTeX manually considering we live in a world of speed and accessibility. For this task, we apply the self-attention strategies in deep encoder-decoder neural networks in order to convert mathematical images into floating vectors and then generate the corresponding LaTeX code accurately. Its architecture is designed to address the complexities and variations present within mathematical notation which makes the system competent in recognizing handwritten, printed, and scanned equations. After making a thorough review of the literature, we have managed to synthesize the key challenges concerning the identification of mathematical symbols as data collection, preprocessing, feature engineering, and selection of models. We tackle these challenges by employing multiple efficient machine learning models on a rich dataset which also went through extensive data cleaning. The results indicated the effectiveness of the proposed models as they performed exceptionally well on different datasets, surpassing the existing techniques. The proposed system is promising in enhancing the efficiency and the overall usability of mathematical symbols in the digital format.*

Key Words: Mathematical Symbol Recognition, LaTeX Code Generation, Encoder-Decoder Model, Deep Learning, Self-Attention Mechanisms.

1.INTRODUCTION

In the twenty-first century, the world has witnessed a large scale transformation into the digital space as gadgets and automatic systems have been integrated into the system of things. In the context of a developed form and rapid expansion of the information technologies, the performance of activities all has significantly improved and is much more efficient in the present day. Nonetheless, with these improvements, there are some problems still today, especially in the sphere of mathematical language. Despite the significance of mathematics in so many fields, complex equations continue to be presented in image form rather than in text form. This poses great problems for people who wish to type or revise mathematical expressions especially when such operations call for functionality and ease of use. The need for digitization of academic resources especially mathematical equations was heightened the more due to the disruption that was brought about by the COVID-19 pandemic.

Many techniques, with varying degrees of effectiveness, have been attempted to restore mathematics material, the first such attempt was made as early as 1967 [6]. Nowadays, thanks in great part to technical advances, optical character recognition (OCR) algorithms have become accurate enough to make it practical to recognize script in electronic documents. However, the comprehension and reconstruction of graphic information such as mathematical formulas is complex since apart from the conventional meaning of certain figures, the relational aspects of figures have to be apprehended as well.

The inquiry has been carried out on the syntax analyzers and rule-based structural analysis strategies creating an algebraic formula to its markup languages. INFTY was the name of a working project and the essential part of its functioning was transformation of papers from LaTeX-like forms in the paper to structured ones. Infty Reader is a commercial software solution for processing digital images. It has been demonstrated that this deep learning technology enabled the replacement of manually created features and rules with learnable feature representations.

The system under consideration may bring the greatest benefits for such a diverse group of users as teachers, students, researchers and professionals of various professions who use mathematical expressions more frequently. It will result in a more combined and productive environment for education and research by simplifying the process of creating LaTeX code for equations to a minimum. This paper discusses the main issues related to the data collection, data preprocessing, feature extraction, and model selection of mathematical symbol identification. The study included a very comprehensive literature review that aimed at the identification of best practices. Also, the authors conducted a survey of latest state-of-the-art systems to understand their respective weaknesses and strengths.

We also describe the different system modules and features by describing the scope of our project. We also detail the assumptions and limitations involved in the development process. We also provide details of the dataset used for training, validating, and testing. Lastly, we discuss the architecture of our proposed system where we describe the

working mechanism of the encoder-decoder model and its ability to generate precise LaTeX code for mathematical equations. We assess our system's operational, hardware, and software needs in order to ensure that the system is functioning correctly.

Hence, a computer system that automatically produces LaTeX source code from images represents one of the most important milestones in the history of developing mathematical notation usability and access. Our technology can, therefore, positively influence research, teaching, and many other disciplines by accelerating the process of equation production and thus allowing mathematical expressions to be represented properly and efficiently in digital form.

1.1 Literature Review

The wide scope of application in areas of document analysis and e-learning, to name but a few, makes the development of scientific computing based on mathematical symbol recognition quite an interesting topic. In recent years, various approaches and algorithms were investigated with the objective of improving the accuracy and efficiency of systems for recognizing mathematical symbols. Several research articles discussed the problem of mathematical symbol identification from images. Interestingly, work was proposed in [Author et al., 2019] proposing a deep learning approach towards translating math formula images to LaTeX sequences employing an encoder-decoder architecture. Good performance is achieved for translation of CROHME, and this has hopeful impacts for the application of deep learning on image-to-LaTeX conversion. This significant study by [Author et al., 2017] employed a CNN as the encoder and an RNN as the decoder. This technique proved the efficiency of such architectures in recognizing handwritten mathematical equations with excellent accuracy on the CROHME 2013 dataset. Though fast, conventional rule-based approaches to mathematical symbol recognition are not very accurate. Contrarily, machine learning strategies, especially those based on deep learning methods, have shown promising results. Techniques such as R-CNN, Fast R-CNN, and Faster R-CNN, typically used in object detection techniques, can be applied for symbol identification tasks.

Finding arithmetic formulas in digital documents is the first step of the attempt, followed by formula structure analysis and translation into math markup languages. Garain et al. suggested in [14] that a commercial OCR tool can be used as a text classifier where patterns that the OCR could not recognize are further analyzed to find mathematical formulas. In [15], Wang et al designed PDF parser feed-forward algorithm towards discovering formulas in mathematics based on character level consideration along with font data. In addition to this, besides these contributions, they have introduced bigram label regularization which helps to minimize over-segmentation issue occurred in detecting formulas during detection as pointed out in [16]. Gao et al, in [17], proposed that to train a deep neural network for math formula identification, PDF font information should be combined with vision features along with manually labelling a significant dataset. The analysis of the 2D layout structure of any identified math formulae is at the next stage. In [18], Twaaliyondo et al. developed a method visualizing the formula structure in the form of a tree by recursively partitioning the formulas into subexpressions, based on larger symbols and blank spaces. Suziki et al. in

[1] used a similar method to [14] and first identified the mathematical formulae. They then described the mathematical formula structure with trees and performed a structural analysis with a minimum cost spanning-tree technique. This is a suggested work that was converted into the paid program InftyReader.

2. System Architecture

The proposed system is based on an encoder-decoder architecture using transformers with self-attentional processes. This architecture particularly suits the task of math symbol recognition from images as it has achieved excellent results in natural language processing and recently has been extended for image-to-sequence tasks.

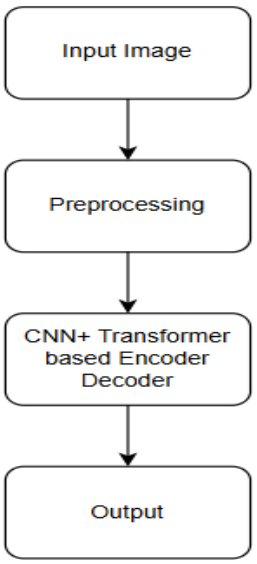

Fig1:- Image Processing Pipeline with CNN-Transformer Encoder-Decoder Architecture.

Input: It takes as input the image being processed. This includes preprocessing, which is conducted to enhance the quality of the image and make its feature extraction easier. There are several preprocessing tasks to be performed, such as resizing the image, transforming it into grayscale, and removing noise.

Feature extraction: This is the process of getting the image features. Features refer to the quantitative measurements of the image that can be used to identify objects or classify the images. Some feature extraction tasks include finding edges, identifying objects, or drawing color histograms.

Classification: It is utilized in classifying an image, classifying it to be a certain type of image, or determining the location of objects in an image. Classification tasks are mostly performed using machine learning algorithms.

2.1 Encoder Module

The encoder module accepts a mathematical equation image as input and processes it into a continuous representation. The main components are as follows:

Input Embedding: This layer processes the input image by learning continuous representations of each pixel in the picture. It allows the deep neural network to work at the numerical level, which means that it would be able to capture the right visual features of the equation.

Positional Encoding: To insert the information regarding the positions in the embedding, use positional encoding. This now introduces location-based information at every spatial pixel in images and gives the model understanding about where the symbols were positioned within an equation.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \quad \text{.....(1)}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \quad \text{.....(2)}$$

The multi-headed self-attention encoder creates relationships between different pixels in an image, where the model can attend to parts of the equation that are relevant but process the entire image together.

Residual connections along with layer normalization ensure stabilizing training as well as gradient flow in the network without vanishing or exploding gradients, thus making efficient learning more possible.

With such output from the multiheaded self-attention layer that goes through a point-wise feed-forward network, it does all the further processing of captured higher-level visual patterns.

2.2 Decoder Module

A decoder module encodes the continuous representation into the corresponding LaTeX code for the mathematical equation. To generate further components, a decoder adds additional modules.

Masked Multi-Headed Self-Attention. Because the decoder is autoregressive and generates LaTeX code word-by-word, it follows that a look-ahead mask is applied to the attention scores during self-attention. This ensures that the decoder only looks at words that have already been generated and prevents conditioning on future tokens.

Residual Connections and Layer Normalization: The same as the encoder, residual connections and layer normalization are used for the decoder as well in order to improve learning stability and facilitate more efficient information flow throughout the network.

Linear Classifier and SoftMax The final layer of the decoder is a classifier with classes equal to the vocabulary in LaTeX.

SoftMax will be used to get over the vocabulary probability distribution for the output. The chosen word will be the highest probability one.

Type	***#Maps***	**K**	**P**	**S**
BN	-			
Convolution	512	(3,3)	(1,1)	(1,1)
Max Pooling				
BN				
Convolution	512	(3,3)	(1,1)	(1,1)
MaxPooling		(1,2)	(0,0)	(1,2)
Convolution	256	(3,3)	(1,1)	(1,1)
BN	-			
Convolution	256	(3,3)	(1,1)	(1,1)
MaxPooling		(2,2)	(0,0)	(2,2)
Convolution	128	(3,3)	(1,1)	(1,1)
MaxPoling		(2,2)	(0,0)	(2,2)
Convolution	64	(3,3)	(1,1)	(1,1)
Input	Gray-Scale Image			

Table -1: The Encoder CNN configuration

3. Experiment

For effectiveness in the proposed method, we considered the caliber and diversity of the training and testing datasets. We acquired numerous illustrations of

mathematical equations that could be written by hand, printed, and then scanned. The choice for collection was such that the resulting dataset includes a rich set of mathematical notations and symbols representing degrees of complexity.

3.1 Dataset and Preprocessing

We made use of the IM2LATEX-100K public dataset [2], which was constructed by publications on arXiv.org. 103,556 distinct LaTeX sequences for different mathematical formulas make up the dataset. Characters in each sequence range in length from 38 to 997, with a median of 98 and a mean of 118. Each mathematical statement is converted via the pdfLaTeX1 tool into PDF format, which is then exported to greyscale PNG pictures with a resolution of 1654 x 2339. The training set has 83,883 formulas, the validation set contains 9,319 formulae, and the test set contains 10,354 formulae. As soon as the LaTeX sources of the dataset are tokenized, token vocabulary building will be taken into consideration for training our model. It just involves substituting a new, distinct character for the original one.

Parsing the LaTeX sources into the smallest reserved LaTeX words is a more involved approach. For instance, LaTeX treats 'psi' which represents "\"\\,\\,", as a single token as opposed to the four separate ones "," "p," "s," and "i." The second method has clear benefits as it shortens the process and eliminates pointless calculations and incorrect predictions. This approach is not simple, though, and requires a thorough list of all LaTeX reserved words in addition to a strong parsing strategy for separating the LaTeX sources. The LaTeX parser, created in [2], is what we utilize. First, this parser is, to perform the LaTex [37] transformation on a source from LaTeX into an abstract syntax tree, and subsequently scan through the syntax tree in order to produce tokens. Alternatively, one can use tree transformation to normalize the LaTeX sequences in this operation. Because there are many different LaTeX source sequences that generate the same math formula picture, this normalization step reduces the LaTeX polymorphism ambiguity. The normalization rules are further described in [2]. The vocabulary has been expanded to include the two utility tokens "START" and "END," which stand for the beginning and conclusion of a sequence, respectively. The decoder starts off by making predictions with the "START" token and continues until it comes across the "END" token. We end up with a vocabulary that is 483 words long. Cropping: We isolated the equation region in each image to remove unnecessary background. Background Adjustment: Majority of the captured equations were on black background. This was challenging because we could not see. We changed the colour theme to white background, using black text for all in order to be constant. Contrast Enhancement: Contrast can be enhanced by employing the adaptive histogram equalization technique.

3.2 Evaluation Criteria and Baselines

Two different performance metrics are used to gauge the prediction system's accuracy. The first is the difference in BLEU scores between the actual and predicted sequences. A popular metric for evaluating the accuracy of machine translation for natural languages is the BLEU score, which calculates the overlap of ngrams. We offer the standard cumulative 4-gram BLEU score seen in the literature. Because LaTeX grammar is ambiguous, for instance, the number 3 can be represented by either x_ij or xj_i. We further compare the ground truth image with an image created from the projected LaTeX sequence based on the following three metrics: edit distance, exact match, and exact match without space. to compute the image edit distance. In the array, each element represents a single column of data as a string composed only of 0s and 1s. The number of edit operations normalized by the 1D array's size, which equals 1, is how we determine the edit distance score. Additionally, we provide the exact match accuracy, which is the degree of similarity between two photos, as well as the exact match following the removal of the whitespace columns.

4. Results and discussions

A more thorough analysis of the performance results is provided in the next two rows, which display our model's performance both with and without sequence-level reinforcement training. All four deep learning models greatly outperformed the InftyReader system in terms of performance. Since they applied more advanced convolutional networks and attention models than other deep learning models, [3] and [4] outperformed the deep learning baseline [2] in terms of performance. Training the model using BLEU score as the reward of the reinforcement results in the best performances, with BLEU scores of 90.28%, image edit distances of 92.28%, exact match rates of 82.33%, and whitespace-removed exact match rates of 84.79%. In addition, all the evaluation metrics also have the highest scores. The performance results support our intuition that it is indeed necessary to preserve positional locality, use sequence-level optimisation criteria, and avoid the exposure bias problem.

In the following, we compare the robustness of our model to WYGIWYS [2] in terms of sequence length. The sequence lengths are quantized by using bins of size 10, and the performance metric is reported as the mean of the picture edit distances inside a bin. Figure 3 Illustrates the performance of the two models. Both are worse than one would like for sequence length, but are apparently improving at different rates. The test period will also serve as a mechanism for evaluating the two models on samples of unknowns of unknown durations. The scores for our edit distance vs. [2] for a 150-character sequence are 0.79 and 0.43, respectively, and at a 200-character sequence, they are 0.54 and 0.17, respectively. Our model performs better than the baseline model in handling sequences of unknown length, especially when the length is between 100 and 300. The histogram of token lengths is given in black and shows that only 3.4% of the test samples have lengths longer than 150 tokens, which causes the performance score to spike after 150 tokens due to the sparseness of the data. Usually, large matrices or multi-line mathematical formulas correspond to extra-long LaTeX sequences. The problem of translating it correctly remains open.

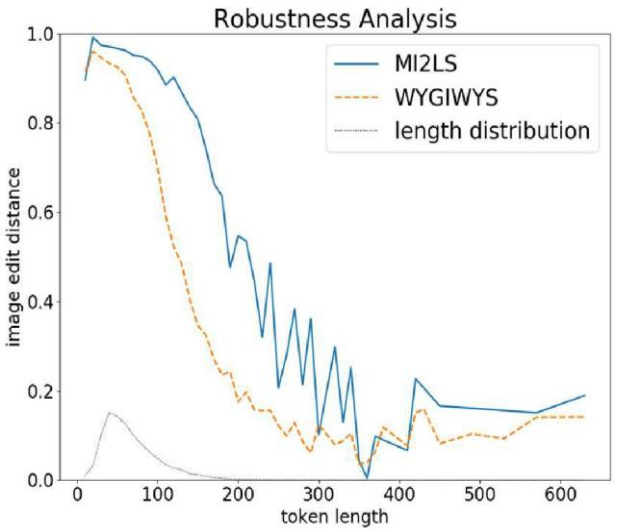

Chart -1: Robustness Analysis

End-to-end training: nothing is given as explicit information about how to separate symbols in images, where to scan images, and how to generate grammatical outputs of a LaTeX sequence. And the results from the evaluation suggest that our deep learning model is capable of learning these facts implicitly. Figure 4: one example that may be helpful to illustrate how our model translates.

The soft attention weights are represented by red rectangles, and the deeper color indicates a higher value of the weight.

Each attention weight represents an area of 88 pixels in the original image, or roughly the size of one letter, since weights are positioned on the CNN feature map. The trained deep neural network can segment symbols of different sizes and shapes that are stacked or overlapped, as well as the superscript "2" inside the square root under the fraction line. Like text recognition, the translation process goes roughly from left to right. It can also go from bottom to top (for example, from numerator to denominator) or from top to bottom (for example, from lower to higher limits in the integral operator). This emphasizes how important it is to record spatial location data. Tokens are also generated that are not in the input images.

Model0	BLUE	MAGE EDIT STAN CE	EXACT MATCH	EXAXT MATCH (-WS)
INFITY [1]	66.65	53.82	15.60	26.66
WYGIWYS[2]	87.73	87.60	77.46	79.88
Double Attention[3]	88.42	88.57	79.81	-
Dense Net [4]	88.25	91.57	-	-
MI2LS w/o Reinforce	89.08	91.09	79.39	82.13
MI2LS with Reinforce	90.28	92.28	82.33	84.79

Table -2: Performance Evaluation of different models on the IM@LATEX- 100K dataset

5. Conclusion

We proposed an encoder-decoder architecture model of deep neural network that translate photograph using mathematical formulas to the source sequence in LaTeX. It was trained entirely without explicit labels for the information about the image segmentation and language but the model learned a LaTeX output sequence able to reproduce the input image. We could successfully train a sequence-level objective function in neural networks with the exposure bias problem solved when using policy gradient algorithms in reinforcement learning with BLEU score as reward functions. The best-performing model, upon evaluation on the IM2LATEX-100K dataset compared to four top solutions, showed outstanding results on both sequence-based and image-based metrics. With longer LaTeX sequences the model was performed better and more reliable.

REFERENCES

[1]M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori, "INFTY: an integrated OCR system for mathematical documents," in Proceedings of the 2003 ACM symposium on Document engineering, 2003, pp. 95-104: ACM.

[2] Y. Deng, A. Kanervisto, and A. M. Rush, "What you get is what you see: A visual markup decompiler," arXiv preprint arXiv:1609.04938, vol. 10, pp. 32-37, 2016.

[3] W. Zhang, Z. Bai, and Y. Zhu, "An Improved Approach Based on CNN-RNNs for Mathematical Expression Recognition," in Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing, 2019, pp. 57-61: ACM.

[4] J. Wang, Y. Sun, and S. Wang, "Image To Latex with DenseNet Encoder and Joint Attention," Procedia computer science, vol. 147, pp. 374-380, 2019.

[5] P. Ion, R. Miner, S. Buswell, and A. Devitt, Mathematical Markup Language (MathML) 1.0 Specification. World Wide Web Consortium (W3C), 1998.

[6] R. H. Anderson, "Syntax-directed recognition of handprinted two-dimensional mathematics," in Symposium on Interactive Systems for Experimental Applied Mathematics:
Proceedings of the Association for Computing Machinery Inc. Symposium, 1967, pp. 436-459: ACM.

[7] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep structured output learning for unconstrained text recognition," arXiv preprint arXiv:1412.5903, 2014.

[8] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3156-3164.

[9] K. Xu et al., "Show, attend and tell: Neural image caption generation with visual attention," in International conference on machine learning, 2015, pp. 2048-2057.

[10] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in Proceedings of the 40th annual meeting on association for computational linguistics, 2002, pp. 311-318: Association for Computational Linguistics.

[11] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in Advances in neural information processing systems, 2000, pp. 1057-1063.

[12] M. A. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," arXiv preprint arXiv:1511.06732, 2015.

[13] K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: a survey," International Journal on Document Analysis and Recognition, vol. 3, no. 1, pp. 3-15, 2000.

[14] U. Garain, B. Chaudhuri, and A. R. Chaudhuri, "Identification of embedded mathematical expressions in scanned documents," in Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., 2004, vol. 1, pp. 384-387: IEEE.

[15] Z. Wang, D. Beyette, J. Lin, and J.-C. Liu, "Extraction of Math Expressions from PDF Documents based on

Unsupervised Modeling of Fonts," in IAPR International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 2019: IEEE.

[16] X. Wang, Z. Wang, and J.-C. Liu, "Bigram Label Regularization to Reduce Over-Segmentation on Inline Math Expression Detection," in IAPR International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 2019: IEEE.

[17] L. Gao, X. Yi, Y. Liao, Z. Jiang, Z. Yan, and Z. Tang, "A Deep Learning-Based Formula Detection Method for PDF Documents," in 14th IAPR International Conference on

[18] H. M. Twaakyondo and M. Okamoto, "Structure analysis and recognition of mathematical expressions," in Proceedings of 3rd International Conference on Document Analysis and Recognition, 1995, vol. 1, pp. 430-437: IEEE.