A Study on Recognizing Handwritten Mathematical Expressions

1st Shrinivedhaa R
dept. Computer Science Engineering
PES University
Bangalore, India
shrinivedhaa@gmail.com

2nd Ashwin Ragupathy
dept. Computer Science Engineering
PES University
Bangalore, India
ashwin.ragupathy@gmail.com

3rd Gariman Gangwani
dept. Computer Science Engineering
PES University
Bangalore, India
garimangangwani@gmail.com

4th Uma D
dept. Computer Science Engineering
PES University
Bangalore, India
umaprabha@pesu.edu

***Abstract*—Handwritten mathematical expressions play a critical role in numerous domains, encompassing education, engineering, and science. The advent of deep neural networks, specifically those employing attention mechanisms and encoder-decoder architectures, has led to the widespread utilization of online recognition for handwritten mathematical expressions.In this research, Convolutional Neural Networks (CNNs) are extensively studied for their efficacy in transforming hand-drawn mathematical expressions into the corresponding LATEX representations. To further improve handwritten expression recognition, we investigate deep neural network models, such as VGG-16 and analyse its performance . We also studied the integration of transformer-based decoders in the recognition process, which offers concise model architectures and enhanced attention to image features like, Attention Aggregation based Bi-directional Mutual Learning Network (ABM) to leverage complementary information from both left-to-right (L2R) and right-to-left (R2L) directions. Moreover, we also studied and implemented handwritten expression recognition, incorporating syntax information into an encoder-decoder network. The proposed method utilizes grammar rules for converting LATEX markup sequences into parsing trees, significantly improving recognition accuracy. The approach outperforms previous methods on benchmark data sets . We studied the behaviour of Syntax Aware network for multiple different classes of mathematical expression**

***Index Terms*—Encoder-Decoder, CROHME, HMER , Transformers, HMER**

I. Introduction

Handwriting recognition has gained grounds with the widespread adoption of interactive devices and distance learning requirements triggered by the COVID-19 pandemic. Advances in sequence recognition models, particularly convolutional neural networks (CNN), have enabled the processing of handwritten documents, including complex mathematical expressions. However, recognizing mathematical expressions presents unique challenges due to a vast collection of symbols and their similarity in handwritten form. This can further be enhanced to mobile application that can efficiently capture and recognize handwritten mathematical expressions and provide solutions. The focus on user interface and experience will ensure seamless integration and enhanced interactivity, we could also automate handwritten examination correction because of these recognition models. This paper presents a comprehensive survey of handwritten mathematical expression recognition and outlines the behaviour of encoder decoder models , towards various types of mathematical expressions.

II. Related Work

OCR for mathematical expressions has been eagerly pursued amongst researchers. Mathematical expressions with sub/superscript, operator symbols of different sizes, and nested fractions have made mathematical expression recognition an atypical OCR task [1]. Approaches include an attention-based neural encoder-decoder model [2]. There also has been multiple efforts in the recognition of online and off-line handwriting of alphabets and numbers [5]. The CROHME competitions, first held in 2011, were intended to facilitate study of math recognition amongst researchers [3]. One interesting research paper we came across attempted to classify over 300 mathematical symbols in handwritten text [4]. Their model used a Simple Linear Iterative Clustering (SLIC) method to group connected pixels and isolate individual characters in the handwritten text. The researchers then used a pretrained Squeeze Net CNN architecture for classifying these identified regions. The Squeeze Net architecture consists of a combination of 1x1 and 3x3 convolutional filters, alternating between squeezing and expanding. This has the effect of both reducing the total number of parameters in the model and to add an extra learning step between these so called ”fire modules.” A Fire module is comprised of: a squeeze convolution layer (which has only 1×1 filters), feeding into an expand layer that has a mix of 1×1 and 3×3 convolution filters. With SqueezeNet, we achieve a 50× reduction in model size compared to AlexNet, while meeting or exceeding the top-1 and top-5 accuracy of AlexNet.

III. LITERATURE SURVEY

TABLE I
LITERATURE SURVEY

Title	Journal/ Conference Name	Methodology Used	Results	Metric
Handwritten Mathematical Expression Recognition with Bidirectionally Trained Transformer [7]	IEEE Access Journal	An encoder decoder model is used but instead of a decoder using RNN's we use a bi-directional transformer as the decoder which solves the lack of coverage problem using positional encodings	2014-57.91% 2016-54.49% 2019-56.88%	Accuracy (%)
Syntax-Aware Network for Handwritten Mathematical Expression Recognition [8]	CVPR	Establish an encoder -decoder network named syntax-aware network (SAN) , which incorporates grammatical constraints and feature learning in a unified framework	56.4%	Expression Recognition Rate
Handwritten Mathematical Expression Recognition via Attention Aggregation based Bi-directional Mutual Learning. [9]	arXiv	Attention aggregation based Bi-directional Mutual learning Network (ABM) which consists of one shared encoder and two parallel inverse decoders	52.92%	Accuracy (%)
HWR For Math	Stanford 230n Project	pre-trained VGG model for feature extraction	Avg Bleu Score: 0.143	BLEU Score

IV. METHODOLOGY

A. *Data Sets*

Our models are trained and tested based on the CROHME 2014 competition dataset with 111 classes of mathematical symbols and 8836 handwritten mathematical expressions, and test our model on three public test datasets: CROHME 2014, 2016, and 2019, with 986, 1147 and 1199 expressions, respectively. [9]

B. *Pickling the datasets*

- **1. Preserving Image and Label Associations:** Pickling datasets allows you to maintain the association between handwritten images and their corresponding labels. This ensures that the integrity of the dataset, including image-label relationships, is preserved throughout the project.
- **2. Fast Loading and Processing:** Loading pickled datasets is faster compared to reading images individually from disk. Once the dataset is pickled, it can be loaded directly into memory, enabling quicker access and processing of images during training or inference stages.
- **3. Sharing and Collaboration:** Pickling datasets simplifies sharing and collaboration among team members. The pickled format provides a convenient way to distribute and share the dataset, ensuring consistent results across different environments.

C. *Expression-Type Categorization*

We categorized our dataset on the basis of expression type We used the images and labels from the CROHME-14 Dataset itself.

- Trigonometric Expressions - 93 Images
- Basic Arithmetic Expressions - 192 Images
- Basic Algebraic Expression - 514 Images
- Expressions with Subscript and Superscript - 469 Images
- Calculus Expressions - 165 Images

D. *Convolutional Neural Network Model (CNN)*

CNN , a common deep neural network architecture, could be used to accomplish HME Recognition. Traditional CNN classifiers are capable of learning and classifying essential 2D characteristics in pictures, the classification is accomplished using the **soft-max layer**. [10]

*1) **Input Layer**:* The Input Layer serves as a buffer for the input before it is sent on to the following layer. It is responsible for receiving and passing the initial data, such as images, through a series of convolutional and pooling layers to capture hierarchical patterns and representations necessary for the network's learning process.

*2) **Convolutional Neural Layer**:* The pivotal task of extracting features in a Convolutional Layer involves utilizing the input data through a Convolution process. This process entails sliding a kernel across the input, computing the sum of products at each position. The size of the stride determines the step size of the kernel's movement. Additionally, the number of feature maps generated in a convolutional layer, also referred to as the depth of the layer, results from performing multiple convolution operations on the input with different kernels, yielding diverse feature maps.

*3) **RELU - Rectified Linear Unit**:* The activation function introduces nonlinearity into the system and plays a key role in neural networks. It effectively replaces negative values with zeros, leading to accelerated learning. In the architecture

of a convolutional layer, every output is passed through the activation function, ensuring nonlinearity is introduced at each step of the network.

*4) **Pooling Layer**:* Pooling utilizes a sliding window that moves across the feature map in synchronization, aggregating representative values to effectively downsample the data. This downsizing of each feature map not only reduces computational load but also preserves important information. The most common types of pooling are **"minimum pooling," "average pooling," and "max pooling,"** each contributing to the extraction of distinct features from the data.Lp-norm pooling, where instead of using the traditional max, min, or average functions, it employs the Lp-norm to aggregate values, offering a more flexible and customizable way of downsampling the data.

*5) **Fully Connected Layer**:* In a fully connected layer, each neuron establishes connections with all neurons from the preceding layer, enabling it to learn a diverse and non-linear combination of features that helps in classification or prediction tasks. Typically, for classification tasks, a soft-max layer follows the fully connected layer to compute the probabilities of each class based on the input.

*6) **Fire Module**: These methods are used to scale down the size of CNN, while maintaining the desired accuracy levels*

A Fire module is composed of two main components: a squeeze convolution layer with 1×1 filters, followed by an expand layer that combines 1×1 and 3×3 convolution filters. Within a Fire module, there are three adjustable hyperparameters to consider: s1×1, e1×1, and e3×3. The s1×1 parameter represents the number of 1×1 filters in the squeeze layer, while e1×1 and e3×3 represent the number of 1×1 and 3×3 filters in the expand layer, respectively. To ensure effective utilization of Fire modules, the value of s1×1 is set to be less than the sum of (e1×1 + e3×3), which enables the squeeze layer to restrict the number of input channels to the 3×3 filters.

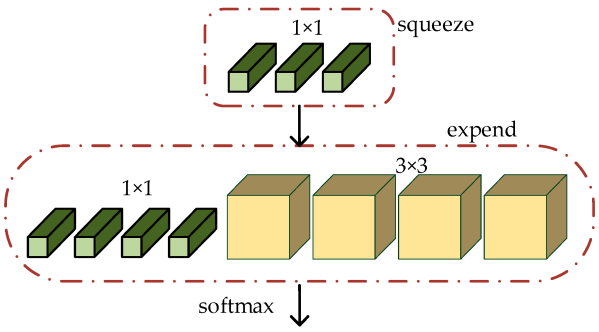


Fig. 1. Fire Module

*7) **Region CNN**:* [11]

- Slow R-CNN
- Fast R-CNN
- Faster R-CNN

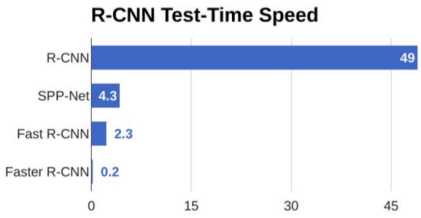


Fig. 2. Time Comparisons

E. Using Pre-Trained VGG Model

This approach involves building a character-level prediction model for generating sequences of characters from LaTeX images. For the convolution part of the model, a pre-trained VGG model is utilized to extract image features. After resizing the images to (224, 224), the last layer of the VGG model used for image classificationis removed , as the interest lies in the internal features rather than image classification. These parameters and weights are pre-trained and loaded from the internet to convert images into a vector of size (4096,) containing image features.

The second part of the model consists of a sequence model serving as a feature extractor, followed by a decoder model. The generation process involves predicting one character at a time, where the previously generated sequence becomes the input for generating the next character. All characters are encoded as integers for representation, and the categorical cross-entropy is used as the final loss function.

F. Attention Aggregation based Bi-directional Mutual Learning

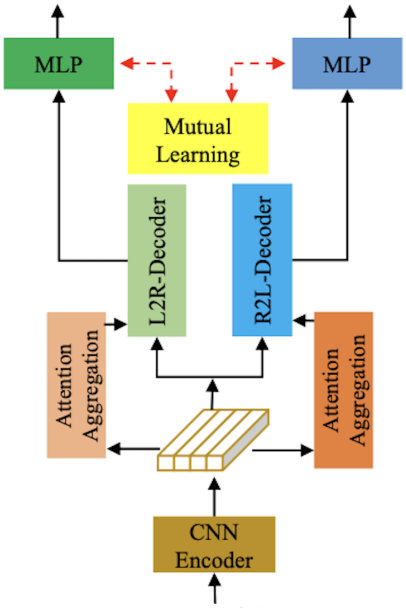


Fig. 3. ABM Model

*1) **Brief Note on The working of ABM Model**:* This model utilizes a **DenseNet as the encoder** to extract features from input images and an Attention Aggregation Module (AAM) to prompt the decoder to focus on specific areas. The AAM aggregates different receptive fields on coverage attention, capturing both detailed features of local areas and global information. The model employs **dual-stream decoders**, translating input images to LaTeX sequences in two directions (L2R and R2L). It then leverages **bi-directional mutual learning** as shown in Fig 3., allowing the decoders to learn from each other's decoding information. KL loss is used to measure the difference in prediction distributions between the two branches during training. The soft probabilities generated by the model provide additional information for better training.

G. Usage of Transformers in OCR

The Transformer is a neural network architecture that relies solely on attention mechanisms. Its revolutionary **self-attention mechanism** sets it apart from RNN in two key aspects. Firstly, unlike RNN, the Transformer does not require dependence on the previous step's state. This design enables efficient parallelization during training, significantly reducing training time. Secondly, the self-attention mechanism establishes direct one-to-one connections among tokens in the same sequence. This fundamental feature effectively addresses the **gradient vanishing problem** of RNN, making the Transformer more suitable for processing long sequences. As a result, in recent years, the Transformer has replaced RNN in various tasks within computer vision and natural language processing.

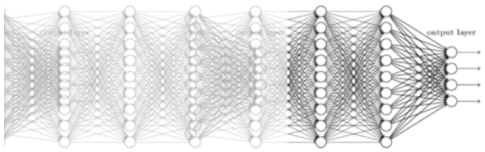


Fig. 4. Vanishing Gradients

H. Implementation and Study of Syntax Aware Network

The syntax aware network is an innovative architecture that leverages **grammar rules and syntax information** to enhance the understanding and prediction capabilities of neural networks. By incorporating linguistic structures into the learning process, this approach shows promise in various natural language processing tasks, leading to more accurate and contextually informed results.

With the development of deep learning methods, the ex- isting text recognition approaches are good at handling text lines in an image-to-sequence manner. However, they may fail to deal with complicated structures such as mathematical expressions (ME). [8]

*1) **Syntax Tree for Structure Identification**:* A syntax tree is used here for effective recognition of complicated mathematical expressions.Like expressions involving fractions , subscripts, superscrits, summations limits etc.

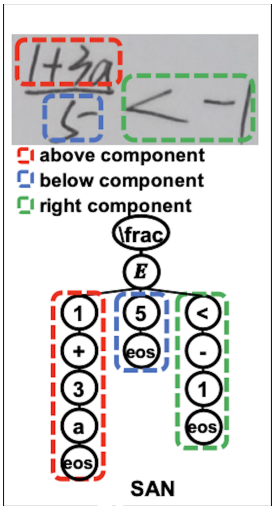


Fig. 5. Syntax Tree

*2) **Heat Maps of Image Recognition Using SAN**:* As seen in Fig 6. We can notice that the model recognises complicated structures efficiently.

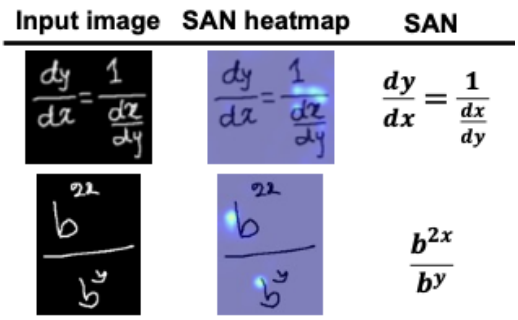


Fig. 6. Image Heat Map

*3) **Densenet CNN Encoder**:* The DenseNet is utilized as the encoder in our approach, serving as a densely connected convolutional network that ensures maximum information flow between layers, promoting feature propagation and reuse. This fully convolutional network is ideal for Handwritten Mathematical Expression Recognition (HMER) since it can handle images of varying sizes. The encoder takes a grayscale image X and returns a matrix of reduced dimensions, where each element corresponds to a local region of the image. The configuration chosen for our implementation sets C (the channel number) to 684 and the down-sampling factor to 16. DenseNet tackles the vanishing gradient problem like in Fig 4., by interconnecting the convolutional layers. The transition layers are used for downsampling. There are total 16 bottleneck units in each of the denseblocks.

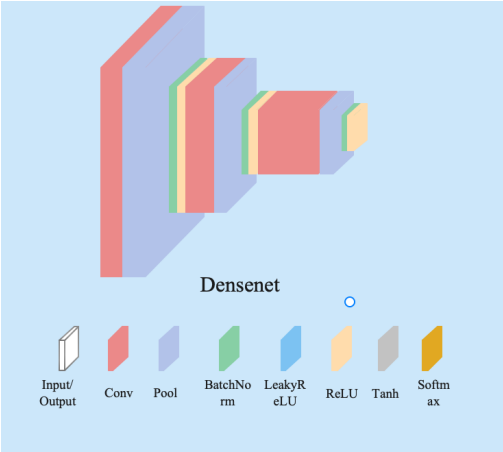


Fig. 7. DenseNet CNN Encoder

Architecture Summary:

- **Input layer:** Convolutional layer with 64 output channels, kernel size of 7x7, and stride of 2.
- **Dense Block 1:** Consisting of 16 Bottleneck units.
- **Transition Layer 1:** Reducing the number of channels and halving the spatial resolution using average pooling.
- **Dense Block 2:** Consisting of 16 Bottleneck units.
- **Transition Layer 2:** Reducing the number of channels and halving the spatial resolution using average pooling.
- **Dense Block 3:** Consisting of 16 Bottleneck units.
- So, there are 3 dense blocks and 2 transition layers in this DenseNet architecture.

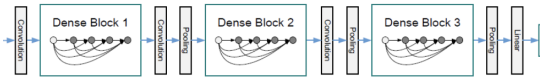


Fig. 8. 3 Dense Blocks 2 Transition Layers

*4) **SAN Decoder**:* The Syntax-Aware Attention Network (SAN) Decoder plays a crucial role in parsing natural language sentences. It comprises two Gated Recurrent Units (GRU) cells, referred to as GRU-alpha and GRU-beta, along with a syntax-aware attention module. The decoder predicts the probabilities of production rules for a given non-terminal symbol alpha, using its context state and the encoding vector of the input sentence. The context state includes a historical state that keeps track of symbol production and a "partner state" that captures short-term contextual information. GRU-alpha takes the partner state as input and the historical state as a hidden vector, generating a new context state (alpha-beta context). The decoder constructs a probability vector for production rules, encompassing predictions for terminal symbols, E predictions, and empty string predictions. This mechanism facilitates an effective construction of the parse tree.

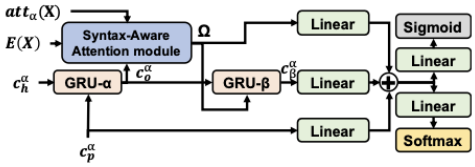


Fig. 9. SAN Decoder

*5) **Implementation Details**:* **Training:**

- We ran our model on batch size 1.
- The number of epochs were 250.
- The model is optimised with **Adadelta Optimiser**.

Training/Testing Platform: All training and testing were carried out on **NVIDIA 2060 Ti 8GB GPU**. The computer had a main memory of 32 GB and had Ubuntu 22.04 Operating System.

*6) **Output Format**:* The Output is a pdf generated by compiling the Latex code by using the mactex compiler. The output consists of :

- Image Filename
- Image
- Predicted Latex
- Ground Truth Latex
- Prediction boolean.

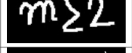
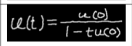
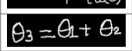
Serial	Image Filename	Image	Predicted LaTeX	Ground Truth LaTeX	Prediction
1	18.em.15		$k = 1000000000$	$k = 1000000000$	Successful Prediction
2	18.em.16		$mz2$	$m \geq 2$	Failed Prediction
3	18.em.17		$u(t) = \frac{u(0)}{1-t(0)}$	$u(t) = \frac{u(0)}{1-tu(0)}$	Failed Prediction
4	18.em.18		$\theta_3 = \theta_1 + \theta_2$	$\theta_3 = \theta_1 + \theta_2$	Successful Prediction
5	18.em.19		$y \neq x$	$y \neq x$	Successful Prediction

Fig. 10. Output Format

*7) **Performance of Different types of Mathematical Expressions**:* We categorized the CHROME-14 test dataset. To study the model in more detail.

TABLE II
ACCURACY BASED EXPRESSION TYPE

Type	Accuracy	No of Images
Calculus	0.49696969696969695	165
Trigonometry	0.5483870967741935	93
Super/Sub Script	0.47761194029850745	469
Algebra	0.6011673151750972	514
Arithmetic	0.6938775510204082	196

We have plotted our results on a bar chart, the arithmetic expressions had the best accuracy of nearly 70 percent.

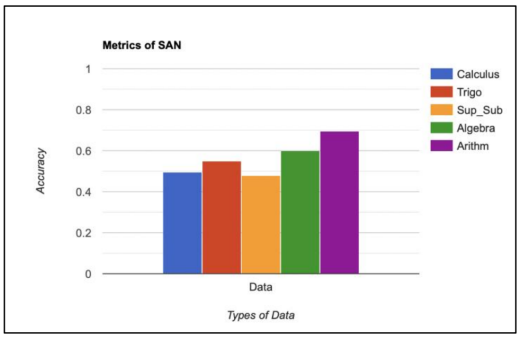


Fig. 11. Graph

*8) **Outputs of diff types of expressions**: .*

Calculus

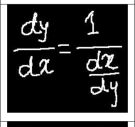
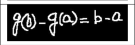
Serial	Image Filename	Image	Predicted LaTeX	Ground Truth LaTeX	Prediction
1	18.em.14		$\frac{dy}{dx} = \frac{1}{\frac{dx}{dy}}$	$\frac{dy}{dx} = \frac{1}{\frac{dx}{dy}}$	Successful Prediction
2	18.em.5		$\int g - \lim_{n\to\infty} \int g$	$\int g = \lim_{n\to\infty} \int g_n$	Failed Prediction
3	18.em.6		$g(0) - g(a) = b - a$	$g(b) - g(a) = b - a$	Failed Prediction
4	20.em.29		$(-\infty, \infty)$	$(-\infty, \infty)$	Successful Prediction

Fig. 12. Calculus

Trigonometry

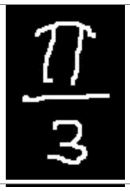
Serial	Image Filename	Image	Predicted LaTeX	Ground Truth LaTeX	Prediction
1	18.em.18		$\theta_3 = \theta_1 + \theta_2$	$\theta_3 = \theta_1 + \theta_2$	Successful Prediction
2	18.em.7		$\sin^2\theta$	$\sin^2\theta$	Successful Prediction
3	20.em.25		$\sin(x + y) = \sin x \cos y + \cos x \sin y$	$\sin(x + y) = \sin x \cos y + \cos x \sin y$	Successful Prediction
4	20.em.32		$\frac{\pi}{3}$	$\frac{\pi}{3}$	Successful Prediction

Fig. 13. Trigonometry

Super/Sub Script

Algebra

Arithmetic

*9) **Region CNN**:*

*10) **Region CNN**:*

ACKNOWLEDGMENT

This work was supported by Centre for Data Science and Applied Machine Learning (CDSAML), PES University, Bangalore , India Pin: 560085.

.

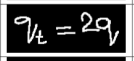
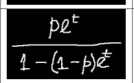
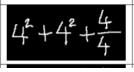
Serial	Image Filename	Image	Predicted LaTeX	Ground Truth LaTeX	Prediction
1	18.em.0		$x_x x x_k + y_k y x_k$	$x_k x x_k + y_k y x_k$	Failed Prediction
2	18.em.11		$q_t = 2q$	$q_t = 2q$	Successful Prediction
3	18.em.12		$\frac{pe^t}{1-(1-p)e^t}$	$\frac{pe^t}{1-(1-p)e^t}$	Failed Prediction
4	18.em.13		$4^2 + 4^2 + \frac{4}{4}$	$4^2 + 4^2 + \frac{4}{4}$	Successful Prediction

Fig. 14. Superscript Subscript

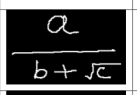
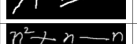
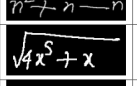
Serial	Image Filename	Image	Predicted LaTeX	Ground Truth LaTeX	Prediction
8	18.em.9		$\frac{a}{b+\sqrt{c}}$	$\frac{a}{b+\sqrt{c}}$	Successful Prediction
9	20.em.35		$M = 0$	$\mu \geq 0$	Failed Prediction
10	20.em.39		$n^2 + n - n$	$n^2 + n - n$	Successful Prediction
11	20.em.40		$\sqrt{4x^5 + x}$	$\sqrt{4x^5 + x}$	Successful Prediction

Fig. 15. Algebra

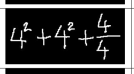
Serial	Image Filename	Image	Predicted LaTeX	Ground Truth LaTeX	Prediction
1	18.em.10		26	26	Successful Prediction
2	18.em.13		$4^2 + 4^2 + \frac{4}{4}$	$4^2 + 4^2 + \frac{4}{4}$	Successful Prediction
3	18.em.1		$\sqrt{48}$	$\sqrt{48}$	Successful Prediction

Fig. 16. Arithmetic

REFERENCES

[1] Kam-fai Chan and Dit-Yan Yeung. Mathematical expression recognition: A survey. International Journal on Document Analysis and Recognition, 3, 02 2001. doi:10.1007/PL00013549.

[2] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. Image-to-markup generation with coarse-to-fine attention, 2016. arXiv:1609.04938.

[3] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, Utpal Garain, Dae Kim, and Jin Kim. Icdar 2013 crohme: Third international competition on recognition of online handwritten mathematical expressions. 08 2013. doi:10.1109/ICDAR.2013.288.

[4] Tavakolian N. Fitzpatrick D. Fernando C. Suen C. Y. Nazemi, A. Offline handwritten mathematical symbol recognition utilising deep learning. 10 2019.

[5] Plamondon and S. N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1):63–84, 2000.

[6] D. Zhelezniakov, V. Zaytsev and O. Radyvonenko, "Online Handwritten Mathematical Expression Recognition and Applications: A Survey," in IEEE Access, vol. 9, pp. 38352-38373, 2021, doi: 10.1109/ACCESS.2021.3063413.

[7] Zhao W, Gao L, Yan Z, Peng S, Du L (2021) Handwritten mathematical expression recognition with Bidirectionally trained transformer, Handwritten Mathematical Expression Recognition with Bidirectionally Trained Transformer

[8] arXiv:2203.01601

[9] arXiv:2112.03603

[10] Recognition of Handwritten Mathematical Expression and Using Machine Learning Approach Prathamesh Tope1, Sarvesh Ransubhe2, Mo-

hammad Abdul Mughni3, Chinmay Shiralkar4, Mrs. Bhakti Ratnaparkhi5
[11] Tesi di Laurea Triennale , Automatic recognition of handwritten mathematical expressions